2013 IEEE 13th International Conference on Data Mining Workshops

Select-Organize-Anonymize: A framework for trajectory data anonymization

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*—Advances in positioning technologies together with the wide adoption of GPS-enabled smartphones enable accurate and low-cost tracking of user location. This allows the collection of large amounts of person-specific mobility data that offer remarkable opportunities for data analysis. Yet, the sharing of such data poses significant privacy risks. This enunciates the need for privacy-preserving, trajectory data publishing methods. Existing approaches are either limited in their privacy specification component or they incur significant, and often unnecessary, data distortion. In response, we propose a novel framework for anonymizing trajectory data that prevents the disclosure of both identity and sensitive location information, while retaining data utility. Our framework involves: (i) selecting similar trajectories, by employing Z-ordering or data projections on frequent subtrajectories, (ii) organizing the selected trajectories into carefully constructed clusters, and (ii) anonymizing each cluster separately. We develop algorithms to realize our framework, which are effective and efficient, as verified by extensive experiments.**

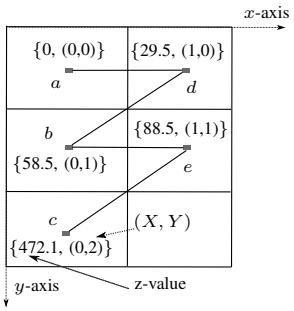
I. Introduction

The enormous advances in positioning technologies, such as GPS, GSM, UMTS and RFID, along with the rapid developments in the wireless communications industry, have made possible the accurate tracking of user location, at a low cost [7]. This, together with the wide adoption of GPS-enabled smartphones, gave rise to novel applications, which are based on user location. At the same time, the mobility data that are collected from these applications provide a valuable resource for understanding human behavior, as well as for supporting various processes. The sharing, however, of person-specific movement data, for research or other purposes, poses significant challenges, as it may threaten individuals' privacy.

Specifically, the publishing of person-specific trajectories (i.e., sequences of locations visited by individuals) can lead to *identity disclosure*, even when they are devoid of identifying information [3], [21]. Identity disclosure attacks are possible, when an individual can be associated with a sequence of locations in the published data. Consider, for example, the dataset in Fig. 1a, where each trajectory t_i corresponds to an individual and contains an ordered list of locations visited by them. Observe that identity disclosure is possible, based on the sequence of locations d and a, because this sequence appears only in t_1. A sequence of locations that may lead to identity disclosure, is called *quasi-identifier* (QID) [21]. Existing approaches prevent identity disclosure by either anonymizing: (i) each trajectory as a whole (e.g., by producing cylindrical tubes that contain many trajectories [1]), or (ii) parts

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

(a)



(b)

Fig. 1: (a) The original database $\mathcal{T}$ (b) Z-ordering

of trajectories, based on specific QIDs [21], [23]. The first category of approaches, termed *QID-blind*, employ *clustering and perturbation* [1], [16], while approaches that fall into the second category, termed *QID-aware*, use *generalization* and *suppression* [15], [21], [23].

QID-blind approaches are limited into their privacy specification component and may harm the utility of the published data, unnecessarily. This is because: (i) they assume that an attacker may know all locations visited by an individual, which is extremely difficult, due to the high-dimensionality and sparsity of trajectory data [21], (ii) clustering-based methods may lose information about the direction of movement of co-clustered trajectories, and (iii) perturbation-based approaches may generate false associations, harming data *truthfulness*. On the other hand, QID-aware approaches assume that data owners possess (i) detailed knowledge of QIDs (e.g., an attacker knows a certain sequence of locations about an individual [21]), which is unlikely in the context of trajectory data publishing [3], and (ii) taxonomies that organize all locations in terms of semantic similarity, which is restrictive for real-world trajectory data publishing applications [18].

Adapting k^m-anonymity [22] to trajectory data and employing a *distance-based* generalization model has been proposed recently [18] to address some of these limitations. However, the approach of [18] may risk the disclosure of *sensitive* location information (e.g., the fact that an individual visited a psychiatric clinic) and incur excessive distortion, as it applies distance-based generalization to the entire dataset.

In this paper we propose a novel anonymization framework, which allows preventing the disclosure of both identity and

978-0-7695-5109-8/13 $31.00 © 2013 IEEE
DOI 10.1109/ICDMW.2013.136

sensitive location information, while publishing minimally distorted data. This is achieved by performing a series of operations: (i) *Select*, which identifies similar trajectories based on summary information about them, (ii) *Organize*, which sorts the selected trajectories with respect to similarity and groups them into clusters, and (iii) *Anonymize*, which constructs clusters that prevent both types of disclosure.

Our work makes the following specific contributions:

- We develop two algorithms, called ZGA and SGA, to realize our anonymization framework. ZGA performs *Select* by capturing location similarity, based on *Z-ordering* [20], whereas SGA by measuring trajectory similarity using projections of trajectories on *frequent subtrajectories* [2]. These algorithms subsequently organize similar trajectories, using *Gray order* [19], and form clusters which are anonymized independently.
- We design ℓ^m-ANON, an algorithm for anonymizing the clusters created by ZGA or SGA. This algorithm employs distance-based generalization to preserve privacy and works in an apriori-like fashion.
- We experimentally demonstrate that our approach is effective at preserving data utility and very efficient.

The rest of the paper is organized as follows. Section II formulates the problem and Section III presents our framework. Related work and experiments are presented in Sections IV and V, respectively. Section VI concludes the paper.

II. Problem formulation

Let $\mathcal{L}$ be a set of locations visited by individuals. Each location in $\mathcal{L}$ corresponds to a different region (cell) of a two-dimensional, uniform grid, and we may use the (X, Y)-coordinates of the cell to refer to its associated location. Some locations in $\mathcal{L}$ are *sensitive*, as they represent places that individuals are not willing to be associated with, such as an oncology clinic. Following [14], we assume that sensitive locations are specified by data owners.

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., $|t| = n$. A trajectory s is a *subtrajectory of*, or is *contained in*, a trajectory $t = (l_1, \ldots, l_n)$, when: (i) $|s| \leq |t|$, and (ii) there is a mapping f such that $\lambda_1 = l_{f(1)}, \ldots, \lambda_\nu = l_{f(\nu)}$ and $f(1) < \cdots < f(\nu)$. Thus, s is formed by removing some locations from t, while maintaining the order of all other locations.

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. These trajectories are referred to as the *supporting* trajectories of s, and their set is denoted with $\mathcal{T}_s$. Given a set of trajectories $\mathcal{T}$ and a minimum support threshold $minSup$, specified by data owners, the set of *frequent* subtrajectories contains a subtrajectory s, if and only if $\sup(s, \mathcal{T}) \geq minSup$ [2].

Example 1: A trajectory dataset is shown in Fig. 1a and is comprised of trajectories t_1 to t_6 (id is shown for reference purposes only). Each trajectory contains some of the locations in the set $\mathcal{L} = \{a, b, ..., g\}$, and the locations f and g are sensitive. A subtrajectory of the trajectory $t_1 = (d, a, c, e)$ is $s = (a, e)$, which has a support of 3. Assuming that $minSup = 3$, the subtrajectories (d, c) and (d, e) are both *frequent*, unlike (d, c, e) and (d, e, c).

In the following, we define the concept of *trajectory key*, which is essentially a projection of the trajectory on the feature space of an ordered set of subtrajectories.

Definition 1 (Trajectory key): Given a trajectory t in $\mathcal{T}$ and an ordered set of subtrajectories $S = (s_1, \ldots, s_l)$, where $s_i \in \mathcal{T}$, $1 \leq i \leq l$, the key of t, denoted as $\mathcal{K}_t^S$ is a set of size $|S|$, whose i-th element is 1, if s_i is contained in t, and 0 if not.

Example 2: Consider the trajectory t_1 in Fig. 1a and assume the set $\big((d, e), e, d\big)$, which contains the three most frequent subtrajectories in $\mathcal{T}$ ordered in decreasing support; the key for t_1 is (111), as all three of them are subtrajectories of t_1.

As mentioned before, a location in $\mathcal{L}$ is associated with a distinct pair of (X, Y)-coordinates. To measure similarity between locations, we use the *Z-order* (or *Morton order*), a function that maps locations to numbers, called *z-values*. The z-value for a location is calculated by interleaving the binary representations of the location's (X, Y) coordinates [20], and the mapping preserves the locality of locations. That is, locations with "similar" coordinates are mapped to numbers whose difference is "small". Based on this function, we obtain the Z-ordering of locations, as explained below.

Definition 2 (Z-ordering of locations): Given a set of locations in $\mathcal{L}$ and their corresponding z-values, Z-ordering is defined as the ordering obtained by sorting the locations in ascending order of their z-values.

The Z-ordering of all nonsensitive locations in $\mathcal{T}$, is shown in Fig. 1b. To publish $\mathcal{T}$ in a way that prevents both the disclosure of individuals' identity and sensitive location information, we use the k^m-anonymity and l^m-diversity privacy principles, which were originally developed for transaction data [22]. The following definition explains how these principles can be adapted to trajectory data and combined together, in order to obtain the principle of $(k, \ell)^m$-anonymity.

Definition 3 ($(k, \ell)^m$-anonymity): Given parameters k and ℓ, which are specified by data owners, a trajectory dataset $\mathcal{T}$ satisfies $(k, \ell)^m$-anonymity, if and only if: (i) $\sup(s, \mathcal{T}) \geq k$, for any subtrajectory s, comprised of m nonsensitive locations in $\mathcal{L}$, and (ii) $\sup(s', \mathcal{T}_s) \leq \ell$, where s' is any subtrajectory of sensitive locations that are contained in $\mathcal{T}_s$.

Example 3: The dataset in Fig. 1a is $(2, 2)^1$-anonymous, because each nonsensitive location a to e has support of at least 2, and none of the sensitive locations f and g co-occurs with all the supporting trajectories of a or e. However, this dataset is not $(2, 2)^2$-anonymous, as the subtrajectory (a, d) of t_3 co-occurs with f, in no other trajectory.

$(k, \ell)^m$-anonymity guarantees that an attacker, who knows any subtrajectory s of m nonsensitive locations about an individual, cannot link the individual to fewer than k trajectories

in the published dataset, nor can associate the individual with a sensitive location with a probability that exceeds $1/\ell$.

To enforce $(k,\ell)^m$-anonymity, we employ the distance-based generalization model [18], which replaces nonsensitive locations with *generalized locations*. A *generalized location* $\{l_1,\ldots,l_v\}$, is a set of at least two nonsensitive locations $l_1,\ldots,l_v$ in $\mathcal{L}$, which is interpreted as any of the locations $l_1,...,l_v$. Sensitive locations are not generalized, because they typically need to be retained intact in applications. Thus, given a trajectory t in $\mathcal{T}$, we may replace a nonsensitive location l_i in t, where $1 \leq i \leq u$, with a generalized location $\{l_1,...,l_v\}$.

Clearly, generalization must avoid unnecessarily distorting the original dataset $\mathcal{T}$, thus we need to quantify distortion. To achieve this, we adopt the *location*, *trajectory*, and *trajectory dataset* distance measures [18], which are applicable to distance-based generalization and are defined as follows.

Definition 4 (Distance): Given a nonsensitive location l that is generalized to $\{l_1,\ldots,l_v\}$, the *location distance* between l and $\{l_1,\ldots,l_v\}$, is defined as:

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

where *EuclDist* is the Euclidean distance. The *trajectory distance* between $t=(l_1,\ldots,l_n)$ and its generalized counterpart $t'=(l'_1,\ldots,l'_n)$ is defined as:

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

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, $1 \leq i \leq u$, is defined as:

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

These measures quantify distortion, based on the distance between the original and generalized data and apply average to combine specific distances, as illustrated below. Normalizing these measures is possible by dividing each of them with the maximum distance between locations in $\mathcal{T}$.

Example 4: Consider trajectory t_1 of Fig. 1a and let $\mathit{EuclDist}(a,b)=1$, $\mathit{EuclDist}(a,c)=1$ and $\mathit{EuclDist}(b,c)=2$. If locations a and c of t_1 are generalized to location $\{a,b,c\}$, then the location distances are $\mathcal{D}_{loc}(a,\{a,b,c\}) = (0+1+1)/3 = 2/3$ and $\mathcal{D}_{loc}(c,\{a,b,c\}) = 1$. Also, if trajectory $t_1=(d,a,c,e)$ is generalized to $(d,\{a,b,c\},\{a,b,c\},e)$ the trajectory distance $\mathcal{D}_{t_1} = (0+2/3+1+0)/4 \approx 0.42$.

The problem we consider is formulated as follows.
Problem: Given a trajectory dataset $\mathcal{T}$, and parameters k, ℓ, and m, construct a $(k,\ell)^m$-anonymous version $\mathcal{T}'$ of $\mathcal{T}$, such that $\mathcal{D}(\mathcal{T},\mathcal{T}')$ is minimized.

III. Select-Organize-Anonymize framework

This section presents our framework for enforcing $(k,\ell)^m$-anonymity to a trajectory dataset $\mathcal{T}$. Our framework performs the following operations:

1) *Select*: This operation selects similar trajectories, based on different properties of their keys.
2) *Organize*: In this operation, the selected trajectories are sorted, based on their *Gray order* [19], and grouped into carefully constructed clusters.
3) *Anonymize*: This operation constructs a minimally distorted $(k,\ell)^m$-anonymous dataset, by anonymizing the trajectories of each cluster separately.

To realize this framework, we develop two algorithms, henceforth called ZGA (for Z-ordering, Gray-code ordering, and Anonymize) and SGA (for Subtrajectory selection, Gray-code ordering, and Anonymize). As is evident from their names, these algorithms perform *Select*, based on different notions of trajectory key similarity. In addition, we introduce ℓ^m-Anon, an algorithm that enforces $(k,\ell)^m$-anonymity, using distance-based generalization. In what follows, we present the details of the ZGA, SGA, and ℓ^m-Anon algorithms.

ZGA algorithm. This algorithm performs the *Select* operation, based on the similarity of nonsensitive locations. To achieve this, it constructs the Z-ordering of the locations, which preserves the locality of locations, as discussed in Section II. Then, ZGA performs *Organization* by: (i) creating a key, for each trajectory in the dataset, based on Z-ordering, (ii) sorting trajectories, based on the Gray order of their keys, and (iii) formulating clusters, based on the latter sorting order. Thus, trajectories that contain many similar, nonsensitive locations are organized together.

Consider, for instance, three trajectories with keys $t=(001)$, $t'=(011)$, and $t''=(101)$, which are sorted in Gray order. Note that t is more similar to t' than to t'', because: (i) t and t' differ by only one bit (the 2nd bit from left to right), and (ii) the bit in which they differ is adjacent to the bit they share (the 3rd bit from left to right). Thus, ZGA organizes trajectories that share many neighboring locations, which can be subsequently generalized with low information loss. Due to its efficient computation and effectiveness, Gray order has been employed as an alternative to more computationally demanding clustering algorithms (e.g., in [10], [22]). However, the way that Gray order is combined with Z-ordering is new, to the best of our knowledge. Subsequently, the SGA algorithm performs the *Anonymize* operation, by applying ℓ^m-Anon to each cluster separately.

Algorithm ZGA takes as input a trajectory dataset $\mathcal{T}$, as well as parameters k, ℓ, m, and $\mathcal{C}$, and it constructs the $(k,\ell)^m$-anonymous counterpart $\mathcal{T}'$ of $\mathcal{T}$. The parameter $\mathcal{C}$ specifies the number of clusters, which will be created in the *Organization* operation, and is set by data owners. Automated specification of $\mathcal{C}$ is possible [13] but left as future work. After initialization, ZGA constructs the Z-ordering of all nonsensitive locations in $\mathcal{L}$ and stores it in a set S (Steps 1-3). Next, the algorithm stores each trajectory $t \in \mathcal{T}$ and its key $\mathcal{K}_t^S$ in a 2D array D, which is then sorted according to the Gray order of the stored trajectory keys (Steps 4-7). After that, ZGA assigns the ordered trajectories of D into clusters (Step 8). Each cluster is then anonymized using ℓ^m-Anon (to be discussed later) and added into $\mathcal{T}'$ (Steps 9-9). Last, $\mathcal{T}'$ is returned (Step 12).

Example 5: Consider applying ZGA to the dataset in Fig. 1a,

Algorithm: ZGA

Input: A dataset $\mathcal{T}$, parameters k, ℓ, m, and $\mathcal{C}$
Output: A $(k,\ell)^m$-anonymous dataset $\mathcal{T}'$

1 $\mathcal{T}' := \varnothing$ // Initialize output
2 $D := S := \varnothing$ // Initialize D and S
3 $S :=$ Z-ordering of nonsensitive locations in $\mathcal{L}$
4 **for** *every trajectory* $t \in \mathcal{T}$ **do**
5 $\quad D(t).traj := t$
6 $\quad D(t).key := \mathcal{K}_t^S$
7 sort D according to $D.key$ in Gray order
8 $G :=$ set of clusters, each containing $|D|/\mathcal{C}$ consecutive trajectories from D
9 **for** *every cluster* $C \in G$ **do**
10 $\quad \tilde{T} := \ell^m$-ANON$(C, k, \ell, m)$
11 $\quad \mathcal{T}' := \mathcal{T}' \cup \tilde{T}$
12 **return** $\mathcal{T}'$

Algorithm: SGA

Input: A dataset $\mathcal{T}$, parameters k, ℓ, m, $\mathcal{C}$, and K
Output: A $(k,\ell)^m$-anonymous dataset $\mathcal{T}'$

1 $\mathcal{T}' := \varnothing$ // Initialize output
2 $D := \varnothing$ // Initialize D
3 S:= the set of top K frequent subtrajectories, comprised of nonsensitive locations, in descending order of support
4 **for** *every trajectory* $t \in \mathcal{T}$ **do**
5 $\quad D(t).traj := t$
6 $\quad D(t).key := \mathcal{K}_t^S$
7 sort D according to $D.key$ in Gray order
8 G:= set of $\mathcal{C}$ clusters, each containing $|D|/\mathcal{C}$ consecutive trajectories from D
9 **for** *every cluster* $C \in G$ **do**
10 $\quad \tilde{T} := \ell^m$-ANON$(c, k, \ell, m)$
11 $\quad \mathcal{T}' := \mathcal{T}' \cup \tilde{T}$
12 **return** $\mathcal{T}'$

when all parameters are set to 2. The algorithm constructs the Z-ordering $\{a, b, d, e, c\}$ by sorting these locations in ascending order of their z-values, and populates the 2D-array D with all trajectories and their keys. Then, ZGA sorts D based on the Gray order of trajectory keys (Step 7). The unsorted and sorted keys of these trajectories are shown in Figs. 2a and 2b, respectively. As $\mathcal{C} = 2$, two clusters containing $\{t_6, t_3, t_1\}$ and $\{t_4, t_2, t_5\}$ are created and anonymized separately by ℓ^m-ANON (Steps 9-9). Last, the dataset $\mathcal{T}'$ in Fig. 2c is returned.

SGA algorithm. This algorithm performs the *Select* operation of our framework, based on the notion of frequent subtrajectories, and the *Organization* operation, by creating trajectory keys from the selected subtrajectories. That is, trajectories are projected on distinct sets of nonsensitive locations, which are visited by most individuals in a specific order, and trajectories with "similar" projections are brought together. In this way, ZGA organizes trajectories that share many frequent subtrajectories, which can be subsequently anonymized with low information loss. To see this, observe that a subtrajectory of m nonsensitive locations and a support of at least k, is k^m-anonymous. Then, the sorting, clustering, and anonymization of trajectories are performed, as in the ZGA algorithm.

Algorithm: ℓ^m-ANON

Input: A cluster C, parameters k, ℓ, and m
Output: A $(k,\ell)^m$-anonymous set C'

1 $C' := C$ // Initialize output
2 **for** $i := 1$ **to** m **do**
3 $\quad \mathcal{S} := \varnothing$ // Initialize $\mathcal{S}$
4 $\quad$ **for** *every subtrajectory* $s \in C'$ *with size* i **do**
5 $\quad\quad$ compute $\sup(s', C'_s)$
6 $\quad\quad$ **if** $\sup(s, C') < k$ *or* $\sup(s', C'_s) > \ell$ **then**
7 $\quad\quad\quad \mathcal{S} := \mathcal{S} \cup s$ // Insert s to $\mathcal{S}$
8 $\quad$ sort $\mathcal{S}$ in increasing order of $\sup(s, C')$
9 $\quad$ **for** *every subtrajectory* $s \in \mathcal{S}$ **do**
10 $\quad\quad$ **while** $\sup(s, C') < k$ *or* $\sup(s', C'_s) > \ell$ **do**
11 $\quad\quad\quad$ find the location l_1 in s with the minimum support in C'
12 $\quad\quad\quad$ find the location $l_2 \in C'$ such that $l_2 \neq l_1$ and $\mathcal{D}_{loc}(l_1, l_2)$ is minimum
13 $\quad\quad\quad$ replace each occurrenc of l_1 and l_2 in C' with the generalized location $\{l_1, l_2\}$
14 **return** C'

The SGA algorithm takes as input a trajectory dataset $\mathcal{T}$, as well as parameters k, ℓ, m, $\mathcal{C}$, and K, and it returns $\mathcal{T}'$, the $(k,\ell)^m$-anonymous counterpart of $\mathcal{T}$. The parameter K controls the number of subtrajectories, contained in trajectory keys. Specifically, SGA creates a key with the top K frequent subtrajectories in $\mathcal{T}$ (i.e., those with the largest support). These subtrajectories are comprised of nonsensitive locations only. K is set by data-owners and its impact will be assessed in Section V. After initialization, SGA finds the top K frequent subtrajectories, using the method in [17], which is selected due to its efficiency (Steps 1-3). Then, in Steps 7-9, SGA performs sorting, clustering, and anonymization, and it returns the $(k,\ell)^m$-anonymous dataset $\mathcal{T}'$, in Step 12.

Example 6: Consider applying SGA to the dataset in Fig. 1a, when all parameters are set to 2. The algorithm finds the top 2 frequent subtrajectories, d and e, and stores them in S, in descending order of support (Step 3). Then, it constructs the 2D-array D, using all trajectories in $\mathcal{T}$ and their keys, which is sorted, as in Fig. 2b (Steps 4-7). Next, SGA creates the clusters $\{t_5, t_1, t_3\}$ and $\{t_4, t_6, t_2\}$, which are anonymized, and produces the dataset in Fig. 3d (Steps 8-12). This dataset differs from the output of ZGA (Fig. 2c), due to the different notion of trajectory similarity used by SGA.

ℓ^m-ANON algorithm. This algorithm is used by ZGA and SGA, to enforce $(k,\ell)^m$-anonymity to a cluster produced by these algorithms, with minimal distortion. Given a cluster C, and parameters k, ℓ and m, ℓ^m-ANON constructs the $(k,\ell)^m$-anonymous counterpart C' of C. To achieve this effectively and efficiently, it employs distance-based generalization [18] and the apriori principle [2]. That is, it first considers generalizing subtrajectories, containing one location, and then applies the same procedure to increasingly larger subtrajectories, as long as $(k,\ell)^m$-anonymity is not satisfied.

In more detail, ℓ^m-ANON initializes C' to the input cluster C and iterates over all subtrajectories of size i (Steps 1-4).

	location				
id	c	e	b	d	a
t_1	1	1	0	1	1
t_2	1	1	1	0	1
t_3	0	1	0	1	1
t_4	1	1	1	1	0
t_5	1	0	0	1	0
t_6	0	1	0	1	0

(a)

	location				
id	c	e	b	d	a
t_6	0	1	0	1	0
t_3	0	1	0	1	1
t_1	1	1	0	1	1
t_4	1	1	1	1	0
t_2	1	1	1	0	1
t_5	1	0	0	1	0

(b)

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

(c)

Fig. 2: (a) key of $\mathcal{T}$ using Z-ordered locations (b) Gray ordered key, and (c) output of ZGA

$F.sb$	$F.\sup$
d	5
e	5
(d,e)	4
c	4
(d,c)	3
a	3

(a)

	freq. subtraj.	
id	e	d
t_1	1	1
t_2	1	0
t_3	1	1
t_4	1	1
t_5	0	1
t_6	1	1

(b)

	freq. subtraj.	
id	e	d
t_5	0	1
t_1	1	1
t_3	1	1
t_4	1	1
t_6	1	1
t_2	1	0

(c)

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

(d)

Fig. 3: (a) ordered array F containing subtrajectories and its respective support (presenting the 5 most frequent) (b) key of $\mathcal{T}$ using 2 most frequent subtrajectories (c) gray ordered key, and (d) output of Algorithm SGA

Note that i increases from 1 to m in each iteration. For each subtrajectory of size i, it calculates the support $\sup(s', C'_s)$, where C'_s is the set of supporting trajectories of s in C', and s' is the sensitive location with the largest support in C'_s (Step 5). Then, ℓ^m-ANON populates a set $\mathcal{S}$ with all subtrajectories that require protection, either because they have a lower support than k in C', or because they co-occur with s' in more than ℓ subtrajectories in C'_s (Steps 6-7). After considering all subtrajectories in C', the algorithm sorts $\mathcal{S}$ with respect to the support of its members, in increasing order (Step 8). Next, it considers each subtrajectory in $\mathcal{S}$ and applies distance-based generalization to it, so that: (i) its support is at least k, and (ii) it does not co-occur with s' in more than ℓ subtrajectories in C'_s (Steps 9-13). Note that the generalization aims at minimizing the *trajectory distance* measure (see Definition 4), and it is applied to each subtrajectory in $\mathcal{S}$. After that, the algorithm proceeds to the next iteration, if i does not exceed m. Otherwise, ℓ^m-ANON returns C', which satisfies $(k,\ell)^m$-anonymity (Step 14).

Example 7: Consider applying ℓ^m-ANON to a cluster containing t_4, t_2, and t_5 in Fig. 1a, when all parameters are set to 2. The algorithm starts by considering the subtrajectories of size 1 in Fig. 4a and calculates $\sup(s', C'_s)$, for each of them (Steps 2-5). Then, it adds a into $\mathcal{S}$, since its support is $1<k$ (Steps 6-7). No other subtrajectory satisfies the check in Step 6, so ℓ^m-ANON sorts $\mathcal{S}$ and applies generalization. Thus, the generalized location $\{a,b\}$, which has minimum $\mathcal{D}_{loc}(a,b)$, is constructed and replaces a and/or b, in all trajectories of C' (Steps 8-13). This produces the cluster in Fig. 4b. After that, ℓ^m-ANON considers the subtrajectories of size 2 in Fig. 4c and applies the same procedure. This creates the cluster, shown in Fig. 4d, which contains $\{a,b,d\}$ and satisfies $(2,2)^2$-anonymity. As all subtrajectories of size 2 have been considered, ℓ^m-ANON returns the cluster in Fig. 4d.

IV. Related work

Trajectory data anonymization has attracted significant attention, due to the pervasive use of location-aware devices that led to tremendous increase in the volume of collected spatiotemporal data about individuals. Bonchi et al. [3] surveyed works on trajectory data anonymization and classified them into *motion-pattern based* and *location based*, according to the adversarial model they adopt. Motion-pattern based methods investigate how anonymized data may allow an attacker to predict individuals' locations, based on their mobility patterns [9], [11], whereas the goal of location based methods is to prevent the inference of individuals' identity and/or sensitive location information from anonymized data (e.g., [1], [14], [16], [18], [21], [23]). Our method falls into the latter category, and, more specifically, to the subcategory of *QID-aware* methods, which guard against attackers with specific background knowledge. In what follows, we discuss QID-aware, location based methods that are relevant to the one we propose. We also note that there are *QID-blind* methods [1], [16], which do not consider specific locations that may risk privacy.

Terrovitis et al. [21] considered multiple attackers, each knowing a different set of places of interest (POIs), visited by individuals. To preserve privacy in this setting, the authors proposed limiting the probability of associating these POIs to an individual's record in the published trajectory dataset. This is achieved through a suppression-based method, which aims at removing the least number of POIs from trajectories, so that the remaining trajectories are protected with respect to the knowledge of each adversary. Unlike [21], our method additionally prevents the inference of sensitive location information, and employs generalization, which generally preserves data utility better than suppression.

Yarovoy et al. [23] considered trajectories containing time information, in addition to POIs, and assumed that each individual has a different set of POIs and times that need to be protected. To offer protection, the authors followed a k-anonymity based approach, which protects trajectories based

s	$\sup(s, C')$
a	1
b, d, e	2
c	3

(a)

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

(b)

s	$\sup(s, C')$
$(\{a,b\},\{a,b\})$	1
$(\{a,b\},d)$	1
(d,e)	1

(c)

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

(d)

Fig. 4: (a) Support for subtrajectories of size $i = 1$ (b) transformed set C' after the processing of subtrajectory a, (c) support for subtrajectories of size $i = 2$, and (d) the final $(2, 2)^2$-anonymous result C'

on individuals' privacy requirements. In practice, eliciting privacy requirements from individuals may be challenging [3], so the use of privacy principles that provide more uniform protection, such as $(k, \ell)^m$-anonymity that we adopt, is more feasible. The authors of [23] developed two generalization-based algorithms, which employ Hilbert curves. The algorithms proposed in [23] follow a very different process of generalizing data than that of our algorithms and may create overlapping groups of records. This is mainly because the algorithms in [23] adopt a different privacy model and consider time information.

Recently, a k^m-anonymity-based algorithm for trajectory data has been proposed in [18]. This algorithm, called SEQANON, works in an apriori-like fashion (i.e., it aims at protecting increasingly larger combinations of individuals' locations from identity disclosure) and applies generalization to the entire dataset. However, SEQANON does not provide protection against the inference of individuals' sensitive location information, and may heavily generalize data. This is because it creates a large number of generalized locations compared to our algorithms, as our experiments demonstrate.

Methods that employ differential privacy [8] have also been proposed [5], [6]. These methods focus on specific data analytic tasks, such as query answering or frequent pattern mining [2], and they employ noise addition. Thus, unlike our approach, they harm data truthfulness, which is necessary to preserve in many applications [12].

V. EXPERIMENTAL EVALUATION

This section presents an experimental evaluation of our algorithms, in terms of data utility and efficiency.

Experimental setup. All algorithms were evaluated using a dataset of moving objects in the Oldenburg city. The dataset was constructed using Brinkhoff's data generator [4], which is employed by many related works [1], [16], [21], [23]. The dataset consists of 18,143 trajectories, whose average length is 4.72 and are created as in [18]. We compare our algorithms, referred to as SGA and ZGA, respectively, with the SEQANON algorithm [18], which is the most closely related to them. All algorithms were implemented in C++ and tested on an Intel Core i7 at 2.2 GHz with 6 GB of RAM.

Data utility. In this section, we evaluate the impact of parameters k, m, ℓ, and $\mathcal{C}$ on data utility, whose default values are $k = 5$, $m = 2$, $\ell = 2$, and $\mathcal{C} = 5$. The parameter K in SGA is fixed to 10, because this offered a good trade off between utility and efficiency, as we found empirically (results omitted due to space limitations). To quantify data utility, we measure the number of original and generalized locations in the anonymized datasets. For the generalized locations, we also measure their average size and distance, similarly to [18].

We first considered the effect of k, which varied in [2, 100]. Observe, in Figs. 5a and 5b, that the number of original locations, as well as that of generalized locations, decreases with k. This is because all algorithms create fewer and larger generalized locations (i.e., they generalize more original locations together), as k increases. Both SGA and ZGA retain many more original locations than SEQANON, which helps data utility. Specifically, ZGA and SGA retained 81% and 28% more original locations than SEQANON, on average. This is because they are applied to each cluster separately. This result is particularly encouraging, as our algorithms prevent both types of disclosure, unlike SEQANON. Furthermore, ZGA and SGA created fewer generalized locations than SEQANON, by 88% and 87.8% (on average), respectively.

Fig. 5c reports the average number of locations in a generalized location, and Fig. 5d the average distance of all locations contained in each generalized location, which is computed as a percentage of the distance between the two furthest locations in a generalized location. Both of these statistics increase with k, as the distortion required to preserve privacy increases. However, the algorithms behave differently. That is, SEQANON creates generalized locations that contain a small number of locations that are "close" to one another, whereas the generalized locations constructed by SGA are more, and consist of more distant original locations. This is because, the distribution of clusters created by SGA is rather skewed (i.e., a small number of clusters with dissimilar trajectories influence the average statistics in Figs. 5c and 5d). We also observed that the issue becomes evident for $k > 10$ and can be ameliorated by creating signatures comprised of more subtrajectories. On the other hand, the generalized locations constructed by ZGA are similar in terms of distance to those created by SEQANON.

To study the impact of m on data utility, we varied this parameter in [1, 4]. Since the dataset we use has an average of 4.72 locations per trajectory, using $m = 3$ (respectively $m = 4$) means that we assume that an attacker knows approximately 65% (respectively 85%) of a user's locations. So, for $m = 3$ and $m = 4$, we expect that few locations will be published intact. On the contrary, for $m = 1$, all locations have support at least k, and a negligible amount of generalization is required. This is true for all algorithms, as can be seen in Fig. 6a. For $m = 2$, all algorithms create many generalized locations containing locations that are close to one another, as shown in Figs. 6b and 6c. Moreover, more generalization needs to be applied to satisfy privacy, as m increases. This leads

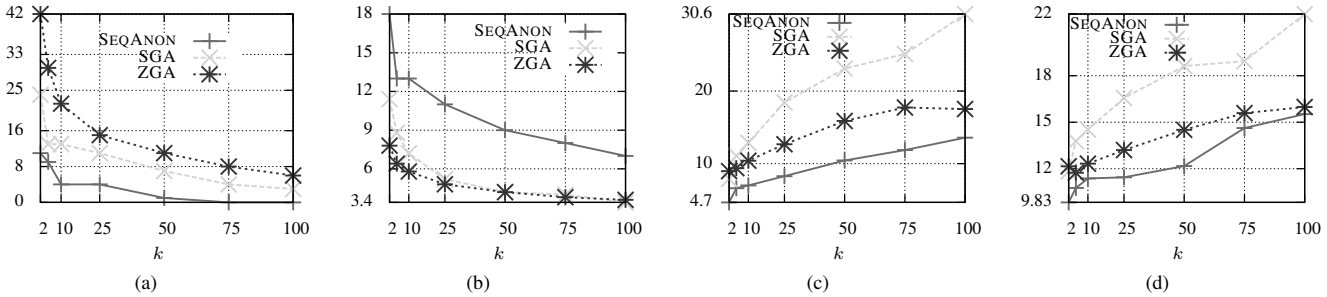


Fig. 5: (a) number of original locations published, (b) number of generalized locations published, (c) avg. number of generalized locations' size, and (d) avg. % of distance in generalized locations for parameter k.

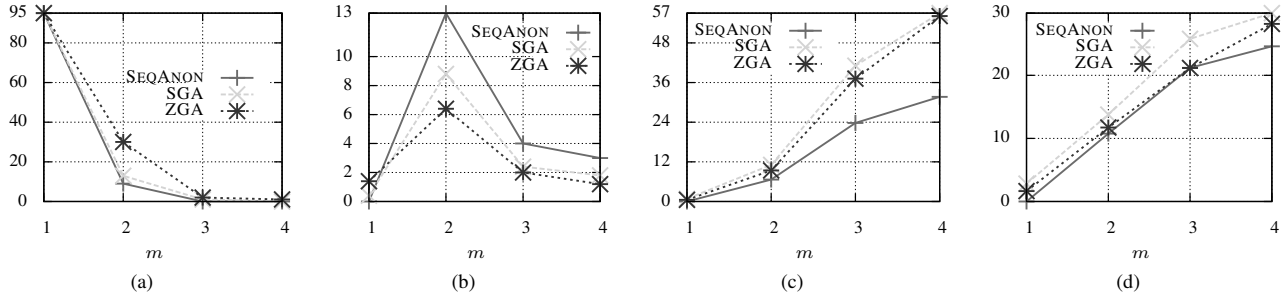


Fig. 6: (a) number of original locations published, (b) number of generalized locations published, (c) avg. number of generalized locations' size, and (d) avg. % of distance in generalized locations for parameter m.

to generalized locations with larger sizes that contain more distant locations, as shown in Figs. 6c-6d. Next, we evaluated

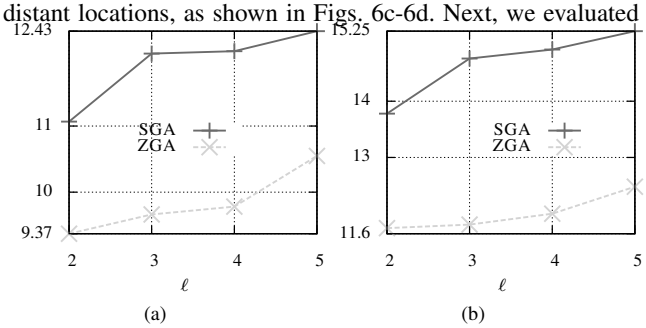


Fig. 7: (a) avg. number of generalized locations' size and (b) avg. % of distance in generalized locations for ℓ.

the impact of parameter ℓ. As SEQANON does not satisfy l^m-diversity, we only report results for ZGA and SGA in Fig. 7. Notice that ZGA created generalized locations that consist of fewer and less distant locations, thereby helping data utility. On average, the generalized locations, created by ZGA consist of 82.5% fewer locations than those constructed by SGA, and their average distance is lower by 81.9%. The superiority of ZGA is attributed to the fact that it captures the distance of original locations more effectively, and it is evident for all tested values of ℓ.

Last, the effect of parameter $\mathcal{C}$, which controls the size of clusters, on data utility was investigated. The results in Fig. 8 demonstrate that ZGA outperforms SGA by a large margin, particularly for larger values of $\mathcal{C}$. Specifically, ZGA permits the publishing of 2.54 times more original locations than SGA, and it creates 78% fewer generalized locations, as shown in Figs. 8a and 8b, respectively. Furthermore, the generalized locations that are constructed by ZGA contain 22.7% fewer original locations, on average, as can be seen in Fig. 8c. The locations in these generalized locations are also "close" to one another, as can be observed from Fig. 8d. In fact, the average percent of distance for the locations of ZGA is smaller than the corresponding percent for SGA, by at least 14.7%. This confirms that taking into account the distance of original locations allows preserving data utility better than doing so based on their frequency, as SGA does through the use of frequent subtrajectories.

Efficiency. We report results for parameters k, m, $\mathcal{C}$, and dataset size, which affect runtime the most. The results for k in Fig. 9a show that ZGA and SGA are more efficient than SEQANON by 48.3% and 13.6%, respectively, when $k > 10$, but need more time, for smaller k values. This is because, they enforce l^m-diversity, which requires applying more generalization, when k is smaller. That is, our algorithms need to progressively increase the support of generalized locations to larger values than k, until l^m-diversity is satisfied, when $k < 25$. Furthermore, ZGA and SGA need less time as k increases, unlike SEQANON. This is because, SEQANON needs to consider a much larger number of potential generalizations to deal with subtrajectories with a lower support than k. Also, ZGA is more efficient than SGA by 72.8%, on average, as it does not require frequent subtrajectory mining.

The impact of m is shown in Fig. 9b. As m increases, all algorithms need more time, but ZGA outperforms SEQANON and SGA by 55% and 72%, on average. However, all algorithms scale well with m, as they employ the apriori principle. Fig. 9c shows the effect of $\mathcal{C}$. As expected, our algorithms are significantly faster as $\mathcal{C}$ increases, because they run on smaller clusters. On the other hand, SEQANON is not affected by this parameter, as it is applied to the entire dataset. In addition,

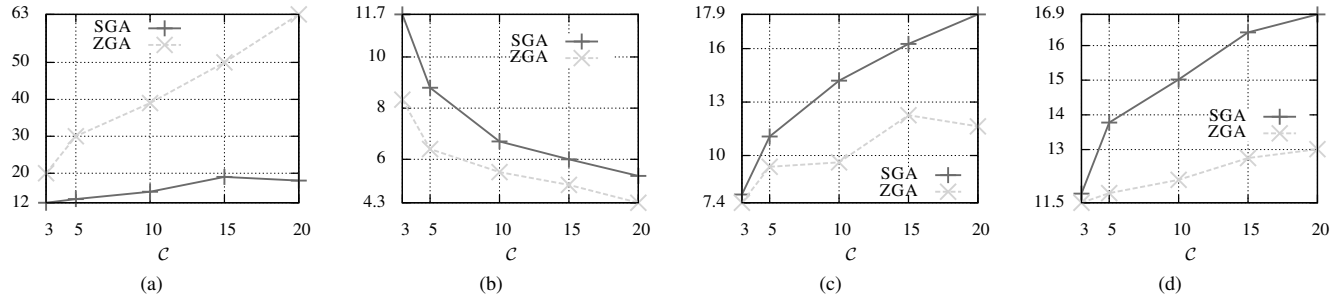


Fig. 8: (a) number of original locations published, (b) number of generalized locations published, (c) avg. number of generalized locations' size, and (d) avg. % of distance in generalized locations for parameter $\mathcal{C}$.

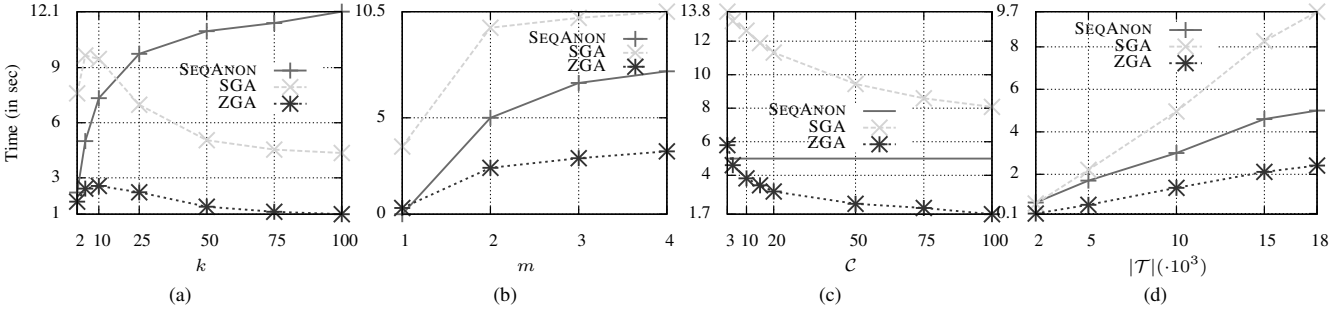


Fig. 9: Runtime with respect to (a) k, (b) m, (c) $\mathcal{C}$, and (e) dataset size

ZGA outperforms SEQANON and SGA by 33% and 71% on average, respectively.

Last, we examined the impact of dataset size and report the results in Fig. 9d. In this experiment, we used increasingly larger subsets of records, which are contained in all larger sets. Observe that all algorithms scale equally well with the dataset size and that ZGA is more efficient than SEQANON and SGA by 61% and 75%, on average.

VI. CONCLUSIONS

In this paper, we proposed a novel framework for anonymizing trajectory data. Our framework enforces $(k, \ell)^m$-anonymity on trajectory data, using two generalization-based algorithms that follow a Select-Organize-Anonymize paradigm. The benefit of our framework is that it enables the generation of truthful data with low distortion, in an efficient and scalable manner.

ACKNOWLEDGEMENTS

G. Poulis is supported by the Research Funding Program: Heraclitus II. S. Skiadopoulos is partially supported by the research program EICOS/Thalis funded by the EU and Greece. G. Loukides is partly 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] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE*, 1995.
[3] F. Bonchi, L. V. S. Lakshmanan, and W. H. Wang. Trajectory anonymity in publishing personal mobility data. *SIGKDD Expl*, 13(1):30–42, 2011.
[4] T. Brinkhoff. A framework for generating network-based moving objects. *GeoInf*, 6(2):153–180, 2002.
[5] R. Chen, G. Acs, and C. Castelluccia. Differentially private sequential data publication via variable-length n-grams. In *CCS*, 2012.
[6] R. Chen, B. C. Fung, B. C. Desai, and N. M. Sossou. Differentially private transit data publication: a case study on the montreal transportation system. In *KDDM*, KDD '12, pages 213–221. ACM, 2012.
[7] R. Clarke. Person location and person tracking - technologies, risks and policy implications. *ITP*, 14(2):206–231, June 2001.
[8] C. Dwork. Differential privacy. In *ICALP (2)*, pages 1–12, 2006.
[9] G. Ghinita, M. L. Damiani, C. Silvestri, and E. Bertino. Preventing velocity-based linkage attacks in location-aware applications. In *ICAGIS*, pages 246–255, 2009.
[10] G. Ghinita, P. Kalnis, and Y. Tao. Anonymous publication of sensitive transactional data. *KDE*, 23(2):161–174, 2011.
[11] W. Jin, K. LeFevre, and J. M. Patel. An online framework for publishing privacy-sensitive location traces. In *DEWMA*, pages 1–8, 2010.
[12] G. Loukides, A. Gkoulalas-Divanis, and B. Malin. COAT: Constraint-based anonymization of transactions. *Knowl. Inf. Systems*, 28(2), 2011.
[13] G. Loukides and J. Shao. An efficient clustering algorithm for k-anonymisation. *CSTJ*, 23(2):188–202, 2008.
[14] A. Monreale, G. L. Andrienko, N. V. Andrienko, F. Giannotti, D. Pedreschi, S. Rinzivillo, and S. Wrobel. Movement data anonymity through generalization. *TDP*, 3(2):91–121, 2010.
[15] A. Monreale, R. Trasarti, D. Pedreschi, C. Renso, and V. Bogorny. C-safety: a framework for the anonymization of semantic trajectories. *TDP*, 4(2):73–101, 2011.
[16] M. E. Nergiz, M. Atzori, and Y. Saygin. Towards trajectory anonymization: a generalization-based approach. In *SPRINGL*, pages 52–61, 2008.
[17] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Mining sequential patterns by pattern-growth: the prefixspan approach. *KDE*, 16(11):1424–1440, 2004.
[18] G. Poulis, S. Skiadopoulos, G. Loukides, and A. Gkoulalas-Divanis. Distance-based k^m-anonymization of trajectory data. In *MDM*, 2013.
[19] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, 1992.
[20] R. Raman and D. Wise. Converting to and from dilated integers. *ToC*, 57(4):567–573, 2008.
[21] M. Terrovitis and N. Mamoulis. Privacy preservation in the publication of trajectories. In *MDM*, pages 65–72, 2008.
[22] M. Terrovitis, N. Mamoulis, and P. Kalnis. Local and global recoding methods for anonymizing set-valued data. *VLDB J.*, 20(1):83–106, 2011.
[23] 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.

Privacy Preservation by Disassociation

Manolis Terrovitis*
IMIS, Research Center 'Athena'
mter@imis.athena-innovation.gr

John Liagouris†
Dept. of Electrical and Comp. Eng., NTUA
liagos@dblab.ece.ntua.gr

Nikos Mamoulis
Dept. of Comp. Sci., Univ. of Hong Kong
nikos@cs.hku.hk

Spiros Skiadopoulos‡
Dept. of Comp. Sci., Univ. of Peloponnese
spiros@uop.gr

ABSTRACT

In this work, we focus on protection against identity disclosure in the publication of sparse multidimensional data. Existing multidimensional anonymization techniques *(a)* protect the privacy of users either by altering the set of quasi-identifiers of the original data (e.g., by generalization or suppression) or by adding noise (e.g., using differential privacy) and/or *(b)* assume a clear distinction between sensitive and non-sensitive information and sever the possible linkage. In many real world applications the above techniques are not applicable. For instance, consider web search query logs. Suppressing or generalizing anonymization methods would remove the most valuable information in the dataset: the original query terms. Additionally, web search query logs contain millions of query terms which cannot be categorized as sensitive or non-sensitive since a term may be sensitive for a user and non-sensitive for another. Motivated by this observation, we propose an anonymization technique termed *disassociation* that preserves the original terms but hides the fact that two or more different terms appear in the same record. We protect the users' privacy by disassociating record terms that participate in identifying combinations. This way the adversary cannot associate with high probability a record with a rare combination of terms. To the best of our knowledge, our proposal is the first to employ such a technique to provide protection against *identity disclosure*. We propose an anonymization algorithm based on our approach and evaluate its performance on real and synthetic datasets, comparing it against other state-of-the-art methods based on generalization and differential privacy.

1. INTRODUCTION

The anonymization of sparse multidimensional data in the form of transactional data (e.g., supermarket sales logs, credit card logs, web search query logs) poses significant challenges. Adversaries that have a part of a record as background knowledge are aided by the dataset's sparsity in identifying the original record. Consider, for example, a dataset D which contains records that trace web search query logs. Even without any direct identifier in the data (user's name or ID) the publication of D might lead to privacy breaches, if an attacker has background knowledge that associates some queries to a known user. Assume that John knows that Jane was interested in buying air tickets to New York, so he has a background knowledge consisting of terms New York and air tickets. If D is published without any modification then John can trace all records that contain both terms New York and air tickets. If only one such record exists, then John can easily infer that this record corresponds to Jane.

To counter such privacy leaks, several anonymization techniques have been proposed in the literature [5, 6, 11, 13, 14, 17, 19, 24, 27]. Most of these methods employ *generalization* [5, 13, 19, 27] to reduce the original term domain and eliminate identifying combinations. For example, they would generalize New York to North America, so that the infrequent combination would be replaced by the more frequent {North America, air tickets}. Alternatively, other methods which are based on suppression, simply remove infrequent terms or terms which participate in infrequent combinations. Generalization and suppression have been mostly used to provide protection against *identity* and *attribute* disclosure. There are few works that rely on adding noise (fake records or terms) to offer differential privacy [6, 14] or to hide the user intent in web search engines [24]. The problem with existing methods is that a large part of the the initial terms are usually missing from the anonymized dataset. There are only a few works [11, 18, 30] that preserve all original terms, without adding noise, based on the *Anatomy* [30] idea of separating quasi identifiers from sensitive values. Still, all these methods can only protect against attribute disclosure.

The main contribution of this work is a novel method called *disassociation* that preserves the original terms but hides identifying combinations. The privacy model is based on k^m-anonymity [27]: an adversary, who knows up to m items from any record, will not be able to match his knowledge to less than k records in the published data. Anonymization is achieved, not by hiding their constituent terms (as done by earlier approaches), but instead by suppressing the fact that some terms (like New York and air tickets) appear together in the same record. The *disassociation* transformation extends the idea of *Anatomy* [30] to provide for the first time protection against identity disclosure by separating terms of the original data. We focus on protection against *identity disclosure* for three reasons: *(a)* it is usually explicitly or implicitly required by law in many countries and applications, *(b)* it is often the case that the sensitivity of items cannot be accurately characterized, so protection against attribute disclosure is not an option, and *(c)* differential pri-

*Supported by the EU/Greece funded COOPERATION: CAP Project.

†Supported by the EU/Greece funded Heracleitus II Program

‡Supported by the EU/Greece funded Thalis Program.

Proceedings of the VLDB Endowment, Vol. 5, No. 10

vacy approaches [6, 14], which offer strong privacy protection, incur a high information loss that is often not an acceptable trade-off. Protecting identities using disassociation has already been identified as a complicated problem even for the case of simple relational data [18], and no previous solution exists for our problem settings. Finally, the proposed anonymization technique is equally capable to existing state-of-the-art methods in providing protection against attribute disclosure if sensitive terms have been identified.

In brief, the main contribution of the paper is the proposal of *disassociation*, a new data transformation for sparse multidimensional data that preserves the original terms of the dataset. We show how this transformation can be used to anonymize a dataset and prove that the resulting data adhere to our privacy guarantee. Moreover, we present an anonymization algorithm that uses disassociation, and we show that it achieves limited information loss, by evaluating it experimentally on real and synthetic datasets.

2. PROBLEM DEFINITION

The proposed anonymization method focuses on sparse multidimensional data and provides protection against identity disclosure. This section formally presents our assumptions about the data and the attack model. In addition, we define the anonymity guarantee our method targets to. Figure 1 summarizes our notation.

Data. We assume a collection D of *records*; each record is a set of *terms* from a huge *domain* T. For example, a term can be a query posed by a user in the context of web search logs, or a product bought by a customer in the context of supermarket logs. As a motivating example, consider a web search query log that traces the queries posed by users over a period of time. Each record models a different user and contains the set of queries posed by the user. Figure 2a presents an exemplary web search query log consisting of 10 records (each being the web search history of a different user). We do not distinguish between sensitive and non-sensitive terms; we consider the general case, where any term might reveal sensitive information for the user and any term can be used as part of a quasi-identifier for a user. As we discuss in Section 5, having a clear distinction between sensitive and identifying terms simplifies the problem and our proposed technique can guarantee, in this case, protection against attribute disclosure.

Attack model. The identification of a user in D is made possible by tracing records that contain unique combinations of terms. For example, if the database of Figure 2a is published and an adversary A knows that a user U has searched for terms madonna and viagra, he can infer that only record r_2 contains both of them; therefore A is certain that r_2 is associated to U. We assume that the adversary A may have background knowledge of up to m terms (i.e., queries) for any record (i.e., user) and that A does not have negative background knowledge (i.e., the adversary does not know whether a user *did not* pose a specific query). Moreover, we assume that the adversary A does not have background knowledge for so many individuals that it will allow her to infer negative knowledge about U (see Section 5 for details).

Anonymity guarantee. The most popular guarantee for protection against identity disclosure is k*-anonymity* [26]. k-anonymity makes each published record indistinguishable from other $k-1$ published records. Applying k-anonymity on sparse multidimensional data can result to a huge information loss, since groups of identical records must be created in a sparse data space [1, 13, 28]. For this reason, we opt for k^m*-anonymity* [27], a conditional form of k-anonymity, which guarantees that an adversary, who has partial knowledge of a record (up to m items, according to our assumption above), will not be able to distinguish any record from other $k-1$ records. More formally:

Symbol	Explanation	Symbol	Explanation
D, D^A	Original, anonymized dataset	T	Domain
$\mathcal{A}$, $\mathcal{I}$	Anonymization, inverse transf.	$s(a)$	Support of a
P / J ...	Clusters / Joint cluster	T^P	cluster domain
$C, C_1, \ldots$	Record Chunks	T^C	Chunk domain
$SC, SC_1, \ldots$	Shared chunks	C_T	Term chunk

Figure 1: Notation

DEFINITION 1. *An anonymized dataset D^A is k^m-anonymous if no adversary that has a background knowledge of up to m terms of a record can use these terms to identify less than k candidate records in D^A.*

For the original dataset D and its anonymized counterpart D^A, we define two transformations $\mathcal{A}$ and $\mathcal{I}$. The anonymization transformation $\mathcal{A}$ takes as input dataset D and results in the anonymized dataset D^A. The inverse transformation $\mathcal{I}$ takes as input the anonymized dataset D^A and outputs all possible (non-anonymized) datasets D' that could produce D^A, i.e., $\mathcal{I}(D^A) = \{D' \mid D^A = \mathcal{A}(D')\}$. Obviously, $D \in \mathcal{I}(\mathcal{A}(D))$. For example, consider the dataset

$$D(age, zip) = \{(32, 14122), (39, 14122)\}$$

and its corresponding anonymized dataset (using generalization)

$$D^A(age, zip) = \{(3x, 14xxx), (3x, 14xxx)\},$$

we have: $\mathcal{A}(D) = D^A$ and

$$\mathcal{I}(D^A) = \left\{ \begin{array}{l} \{(30, 14000), (30, 14000)\}, \ldots \\ \{(30, 14000), (31, 14000)\}, \ldots \\ \{(32, 14122), (39, 14122)\}, \ldots \end{array} \right\}$$

In this paper, to achieve k^m-anonymity (Definition 1), we enforce the following privacy guarantee.

GUARANTEE 1. *Consider an anonymized dataset D^A and a set $\mathcal{S}$ of up to m terms. Applying $\mathcal{I}(D^A)$ will always produce at least one dataset $D' \in \mathcal{I}(D^A)$, for which there are at least k records that contain all terms in $\mathcal{S}$.*

Intuitively, an adversary, who has limited background knowledge (consisting of a set $\mathcal{S}$ of up to m terms) about a person, will have to consider k distinct candidate records in a possible original dataset.

3. ANONYMIZATION BY DISASSOCIATION

In this paper, we propose an anonymization transformation $\mathcal{A}$ that is based on disassociation. The proposed transformation partitions the original records into smaller and disassociated subrecords. The objective is to hide infrequent term combinations in the original records by scattering terms in disassociated subrecords. To illustrate the crux of the disassociation idea, we will use Figure 2. We have already seen that the dataset of Figure 2a is prone to identity attacks (e.g., r_2 can be identified by madonna and viagra). The corresponding disassociated anonymized dataset is illustrated in Figure 2b. Our approach, initially, divides the table into two *clusters* P_1 and P_2 containing records r_1-r_5 and r_6-r_{10} respectively. In each cluster P_i, records are partitioned to smaller subrecords, after dividing the terms in P_i into subsets. For example, in P_1, the terms are divided into subsets $T_1 =$ {itunes, flu, madonna}, $T_2 =$ {audi a4, sony tv}, and $T_T =$ {ikea, viagra, ruby}. Then, each record is split into subrecords according to these subsets. The collection of all subrecords of different records that correspond to the

ID	Records
r_1	{itunes, flu, madonna, ikea, ruby}
r_2	{madonna, flu, viagra, ruby, audi a4, sony tv}
r_3	{itunes, madonna, audi a4, ikea, sony tv}
r_4	{itunes, flu, viagra}
r_5	{itunes, flu, madonna, audi a4, sony tv}
r_6	{madonna, digital camera, panic disorder, playboy}
r_7	{iphone sdk, madonna, ikea, ruby}
r_8	{iphone sdk, digital camera, madonna, playboy}
r_9	{iphone sdk, digital camera, panic disorder}
r_{10}	{iphone sdk, digital camera, madonna, ikea, ruby}

(a) Original dataset D

Cluster P_1 $|P_1| = 5$

	Record chunks		Term chunk
	C_1	C_2	C_T
r_1	{itunes, flu, madonna}		
r_2	{madonna, flu}	{audi a4, sony tv}	ikea, viagra,
r_3	{itunes, madonna}	{audi a4, sony tv}	ruby
r_4	{itunes, flu}		
r_5	{itunes, flu, madonna}	{audi a4, sony tv}	

Cluster P_2 $|P_2| = 5$

	Record chunk	Term chunk
	C_1	C_T
r_6	{madonna, digital camera}	
r_7	{iphone sdk, madonna}	panic disorder,
r_8	{iphone sdk, digital camera, madonna}	playboy, ikea, ruby
r_9	{iphone sdk, digital camera}	
r_{10}	{iphone sdk, digital camera, madonna}	

(b) Anonymized dataset D^A

Figure 2: Disassociation example

same subset of terms is called a *chunk*. For example, r_1 is split into subrecords {itunes, flu, madonna}, which goes into chunk C_1 (corresponding to T_1), {}, which goes into chunk C_2, and {ikea, ruby}, which goes into chunk C_T. C_T is a special, *term chunk*; the subrecords that fall into the last chunk (C_T) are merged to a single set of terms. In our example, C_T contains set {ikea, viagra, ruby}, which represents the subrecords from all r_1−r_5 containing these terms. In addition, the order of the subrecords that fall in a chunk is randomized; i.e., the association between subrecords in different chunks is hidden. According to this representation, the original dataset could contain any record that could be *reconstructed* by a combination of subrecords from the different chunks plus *any subset of terms* from C_T. For example, {itunes, madonna, viagra, ruby} is a reconstructed record, which takes {itunes, madonna} from C_1, {} from C_2, and {viagra, ruby} from C_T. Observe that this record does not appear in the original dataset.

Similarly to the generalization based techniques, the disassociated dataset D^A models a set of possible original datasets $\mathcal{I}(D^A)$. However, in our case the possible datasets are not described in a closed form captured by the generalization ranges, but by the possible combinations of subrecords. In other words, the original dataset is hidden amongst the multiple possible datasets in $\mathcal{I}(D^A)$ that can be reconstructed by combining the subrecords and terms taken from the disassociated dataset.

Overall, the anonymized dataset in Figure 2b satisfies Guarantee 1 for $k = 3$ and $m = 2$. We see in detail how this happens in Section 5, but we can observe that an attacker who knows up to $m = 2$ terms from a record r of the original database is not able to reconstruct less that $k = 3$ records (by combining appropriate subrecords) that might have existed in the original data.

In the following, we present the details of our technique, which performs 3 steps: a horizontal partitioning, a vertical partitioning and a refining. The *horizontal partitioning* brings similar records together into clusters. The heart of the anonymization procedure lies in the *vertical partitioning* which disassociates infrequent combinations of terms. Finally, to reduce information loss and improve the quality of the anonymized dataset a *refining* step is executed.

Horizontal partitioning. Records of the original dataset D are grouped into *clusters* according to the similarity of their contents (e.g., Jaccard similarity). For instance, cluster P_1 is formed by records r_1−r_5 (Figure 2b). Horizontal partitioning reduces the anonymization of the original dataset to the anonymization of multiple small and independent clusters. The benefits of this approach are threefold. First, it limits the scope of the term disassociation to the records that are contained in the cluster; two terms may be disassociated only within the local scope of a partition, limiting this way the negative effect in the information quality of the published dataset. Second, since clustering brings similar records together in the same partition, the anonymity guarantee can be achieved with reduced disassociation. Third, the anonymization process can be done more efficiently and even in parallel.

Vertical partitioning. Intuitively, vertical partitioning leaves term combinations that appear many times intact and disassociates terms that create infrequent and, thus, identifying combinations. The disassociation is achieved by concealing the fact that these terms appear together in a single record. Vertical partitioning applies on each cluster and divides it into *chunks*. There are two types of chunks: record and term chunks. *Record chunks* contain subrecords of the original dataset; i.e., each chunk is a collection (with bag semantics) of sets of terms, and they are k^m-anonymous. That is, every m-sized combination of terms that appears in a chunk, appears at least k times. *Term chunks* do not contain subrecords; they contain the terms that appear in the cluster, but have not been placed to record chunks. A term chunk is a simple collection of terms with set semantics. Each cluster may contain an arbitrary number of record chunks (≥ 0) and exactly one term chunk (which might be empty). In Section 5, we explain how the term chunk can be used to provide l-diversity some terms have been designated as sensitive.

Vertical partitioning is applied to each cluster independently. Let us consider a cluster P and let T^P be the set of terms that appear in P. To partition P into v record chunks $C_1, \ldots, C_v$ and a term chunk C_T, we divide T^P into $v+1$ subsets $T_1, \ldots, T_v, T_T$ that are pairwise disjoint (i.e., $T_i \cap T_j = \emptyset$, $i \neq j$) and jointly exhaustive (i.e., $\bigcup T_i = T^P$). Subsets $T_1, \ldots, T_v$ are used to define record chunks $C_1, \ldots, C_v$ while subset T_T, is used to define term chunk C_T. Specifically, $C_T = T_T$ and record chunks C_i, $1 \leq i \leq v$ are defined as $C_i = \{\!\{ T_i \cap r \mid \text{for every record } r \in P \}\!\}$ where $\{\!\{\cdot\}\!\}$ denotes a collection with bag semantics (i.e., duplicate records are allowed in C_i). Thus, chunks $C_1, \ldots, C_v$ are collections of records while chunk C_T is a set of terms. The partitioning of T^P to $T_1, \ldots, T_v, T_T$ is performed in a way which ensures that all resulting record chunks $C_1, \ldots, C_v$ are k^m-anonymous. In Figure 2b, two 3^2-anonymous record chunks C_1 and C_2 are formed for P_1, by projecting the records of P_1 to sets T_1 = {itunes,flu,madonna} and T_2 = {audi a4, sony tv} respectively; the remaining terms {ikea, viagra,ruby} of P_1 form the term cluster C_T.

Note that, for each published cluster, we explicitly show the number of original records in it. Without this explicit information, a data analyst may only infer that the cluster has at least as many records as the cardinality of the chunk with the greatest number of subrecords. Not knowing the cardinality of a cluster introduces significant information loss; for instance, it is not feasible to estimate the co-existence of terms in different chunks.

Finally, we remark that horizontal and vertical partitioning are applied in reverse order from what is followed by approaches that employ similar data transformations [11, 18, 30]. Thus, since verti-

Record		Term	Shared
P_1 cluster			
{itunes, flu, madonna}			{ikea,ruby}
{madonna, flu}	{audi a4, sony tv}	viagra	{ruby}
{itunes, madonna}	{audi a4, sony tv}		{ikea}
{itunes, flu}			
{itunes, flu, madonna}	{audi a4, sony tv}		
P_2 cluster			
{madonna, digital camera}			
{iphone sdk, madonna}		panic	{ikea,ruby}
{iphone sdk, digital camera, madonna}		disorder,	
{iphone sdk, digital camera}		playboy	
{iphone sdk, digital camera, madonna}			{ikea,ruby}

Figure 3: Disassociation with a shared chunk.

cal partitioning is applied *independently* in each horizontal partition (i.e., cluster), our method follows a *local* anonymization approach. This constitutes a significant difference from previous works that anonymize datasets by performing a global partitioning between terms (usually between sensitive terms and quasi-identifiers).

Refining. At this final step of the method, we focus on improving the quality of the published result while maintaining the anonymization guarantee. To this end, we examine the terms that reside in term chunks. Consider the example of Figure 2b. Terms ikea and ruby are in the term chunk of P_1 because their support in P_1 is low (each term appears in only 2 records). For similar reasons these terms are also in the term chunk of P_2. However, the support of these terms considering both clusters P_1 and P_2 is not small enough to endanger user privacy (ikea and ruby appear as many times as itunes and iphone that are in record chunks).

To address such situations, we introduce the notion of *joint clusters* that offer greater flexibility in our partitioning scheme by allowing different clusters to *share* record chunks. Given a set T^s of *refining terms* (e.g., ikea and ruby), which commonly appear in the term chunks of two or more clusters (e.g., P_1 and P_2), we can define a joint cluster by *(a)* constructing one or more *shared chunks* after projecting the original records of the initial clusters to T^s and *(b)* removing all T^s terms from the term chunks of the initial clusters. Figure 3 shows a joint cluster, created by combining clusters P_1 and P_2 of Figure 2b, based on T^s={ikea,ruby}.

The idea of a joint cluster can be recursively generalized. We may form higher-level joint clusters by combining simple and joint clusters of a lower level (for example see Figure 5). In the general case a joint cluster J, has as children the joint clusters $J_1, \ldots, J_n$ and at its leaves the simple clusters $P_1, \ldots, P_m$. Moreover it contains the k^m-anonymous shared chunks $SC_1, \ldots, SC_w$, which are created over a domain T^s. All terms of T^s come from the term chunks $C_{T_1}, \ldots, C_{T_m}$ of $P_1, \ldots, P_m$. If $T_1, \ldots, T_w$ are the domains of $SC_1, \ldots, SC_w$, $T_1 \cup \cdots \cup T_w = T^s$ and $T_i \cap T_j = \emptyset$ for $i \neq j$, then each shared chunk SC_i is created by projecting the records of every P_j to $C_{T_j} \cap T_i$. Shared chunks are defined in this way, in order to avoid having a record contributing the same projection to shared or simple record chunks more than once.

Reconstruction of datasets. A disassociated dataset D^A has the original records of D partitioned into subrecords (residing in record or shared chunks) and terms (residing in term chunks). An adversary A can combine record, shared and term chunks in an effort to reconstruct the world of all possible original datasets $\mathcal{I}(D^A)$. Possible original datasets may be reconstructed by combining the subrecords of record and term clusters padded with some terms from the term chunks. Such datasets D' are called *reconstructed datasets* and by construction belong to $\mathcal{I}(D^A)$. The adversary A may consider only the reconstructed datasets that abide to his background knowledge. Guarantee 1 requires that for every m terms that exist in a record of D, there will be a D' that contains k records with these terms. Thus, an adversary will always have k candidate records that will match her background knowledge.

Reconstructed datasets are also useful to data analysts, since they have similar statistical properties to the original one. We experimentally evaluate this similarity in Section 7. The benefit of providing the disassociated form, instead of a reconstruction directly, is threefold: *(a)* an analyst can work directly on the disassociated dataset. The disassociated dataset reveals some information, i.e., itemset supports, that is *certain* to exist on the original data, *(b)* the reconstruction procedure is transparent; an adversary cannot draw incorrect conclusions about the identity of a user by considering the reconstructed dataset as original or as ineffectively perturbated and *(c)* an analyst can create an arbitrary set of reconstructed datasets and average query results from all of them.

4. THE ANONYMIZATION ALGORITHM

The proposed algorithm uses heuristics to perform the partitioning (horizontal and vertical) and the refining step of Section 3.

Horizontal partitioning. Horizontal partitioning should bring together similar records that contain many common terms and few uncommon ones. Similarity may be assessed using measures from Information Theory (e.g., Jaccard coefficient). Related clustering algorithms exist in the literature for set-valued data [29], but unfortunately they are not appropriate for our setting since: *(a)* they are not efficient on large datasets and *(b)* they do not explicit control the size of the clusters. We employ Algorithm HORPART, a lightweight heuristic that does not have these problems. The key idea is to split the dataset into two parts: one with the records that contain the most frequent term a in the dataset and another with the remaining records. This procedure is recursively applied to the new datasets until the final datasets are small enough to become clusters. Terms that have been previously used for partitioning are recorded in set *ignore* and are not used in subsequent splitting (Line 3).

Vertical partitioning. To vertically partition the clusters, we follow a *greedy* strategy (Algorithm VERPART), executed independently for each cluster. VERPART takes as input a cluster P and integers k and m; the result is a set of k^m-anonymous record chunks $C_1, \ldots, C_v$ and the term chunk C_T of P.

Let T^P be the set of terms of P. Initially, the algorithm computes the number of appearances (support) $s(t)$ of every term t and orders T^P with decreasing $s(t)$. All terms that appear less than k times are moved from T^P to the term chunk T_T. Since all the remaining terms have support at least k, they will participate in some record chunk. Next, the algorithm computes sets $T_1, \ldots, T_v$ (while loop). To this end, the algorithm uses set T_{remain} that contains the non-assigned terms (ordered by decreasing support s) and T_{cur} (that contains the terms that will be assigned to the current set). To compute T_i ($1 \leq i \leq v$), Algorithm VERPART considers all terms of set T_{remain}. A term t is inserted into T_{cur} only if the C_{test} chunk constructed from $T_{cur} \cup \{t\}$ remains k^m-anonymous (Line 12). Note that the first execution of the for loop (Line 10) will always add a term t to T_{cur} since $C_{test} = \{t\}$ is k^m-anonymous

Algorithm: HORPART
Input : Dataset D, set of terms *ignore* (initially empty)
Output : A HORizontal PARTitioning of D
Param. : The maximum cluster size $maxClusterSize$

1 **if** $|D| < maxClusterSize$ **then return** $\{\!\{D\}\!\}$;
2 Let T be the set of terms of D;
3 Find the most frequent term a in $T - ignore$;
4 $D_1 =$ all records of D having term a;
5 $D_2 = D - D_1$;
6 **return** HORPART$(D_1, ignore \cup \mathsf{a}) \cup$HORPART$(D_2, ignore)$

Algorithm: VERPART
Input : A cluster P, integers k and m
Output : A k^m-anonymous VERtical PARTitioning of P

1 Let T^P be the set of terms of P;
2 **for** *every term* $t \in T^P$ **do**
3 Compute the number of appearances $s(t)$;
4 Sort T^P with decreasing $s(t)$;
5 Move all terms with $s(t) < k$ into T_T; // T_T is finalized
6 $i = 0$;
7 $T_{remain} = T^P - T_T$; // T_{remain} has the ordering of T^P
8 **while** $T_{remain} \neq \emptyset$ **do**
9 $T_{cur} = \emptyset$;
10 **for** *every term* $t \in T_{remain}$ **do**
11 Create a chunk C_{test} by projecting to $T_{cur} \cup \{t\}$;
12 **if** C_{test} *is* k^m*-anonymous* **then** $T_{cur} = T_{cur} \cup \{t\}$;
13 i++;
14 $T_i = T_{cur}$;
15 $T_{remain} = T_{remain} - T_{cur}$;
16 Create record chunks $C_1, \ldots, C_v$ by projecting to $T_1, \ldots, T_v$;
17 Create term chunk C_T using T_T;
18 **return** $C_1, \ldots, C_v, C_T$

($s(t) \geq k$). If the insertion of a term t to T_{cur} renders $T_{cur} \cup \{t\}$ non k^m-anonymous, t is skipped and the algorithm continues with the next term. After having assigned to T_{cur} as many terms from T_{remain} as possible, the algorithm *(a)* assigns T_{cur} to T_i, *(b)* removes the terms of T_{cur} from T_{remain} and *(c)* continues to the next set T_{i+1}. Finally, Algorithm VERPART constructs record chunks $C_1, \ldots, C_v$ using $T_1, \ldots, T_v$ and the term chunk C_T using T_T.

Refining. The result of the vertical partitioning is a set $\mathcal{P}$ of k^m-anonymous clusters. The refining step improves the quality of the anonymized dataset by iteratively creating joint clusters until no further improvement is possible. A naïve method to perform this step consists of computing the information loss (e.g., using a metric of Section 6) for all possible refinement scenarios and choosing the one with the best effect on data quality. Since such an option is very inefficient, we define a refining criterion. Let us consider two clusters J_1 and J_2. These cluster are joined into cluster J_{new} if the following inequality holds:

$$\frac{s(t_1) + \cdots + s(t_n)}{|J_{new}|} \geq \frac{u_1 + \cdots + u_m}{|P_1| + \cdots + |P_m|} \quad (1)$$

where *(a)* $t_1, \ldots, t_n$ are the refining terms T^s (Section 3), *(b)* $s(t_1)$, $\ldots$, $s(t_n)$ are the supports of $t_1, \ldots, t_n$ respectively in the shared chunks of J_{new}, *(c)* $P_1, \ldots, P_m$ are the simple clusters of J_1 and J_2 that contain $t_1, \ldots, t_n$ and *(d)* $v_1, \ldots, v_m$ are the number of terms $t_1, \ldots, t_n$ that appear in the term chunk of each of $P_1, \ldots, P_m$ respectively. For instance, if J_1 and J_2 are clusters P_1 and P_2 of Figure 2b and J_{new} is the joint cluster of Figure 3 then the refining terms are ruby and ikea and we have: $\frac{s(\text{ruby})+s(\text{ikea})}{|J_{new}|} = \frac{4+4}{10} \geq \frac{2+2}{10} = \frac{u_1+u_2}{|P_1|+|P_2|}$. Thus, J_1 and J_2 are replaced by J_{new}.

Note that the left part of Equation 1 estimates the probability of attributing one of $t_1, \ldots, t_n$ to the records of the joint J_{new} while the its right part expresses the probability of attributing one of $t_1, \ldots, t_n$ to the initial records of J_1 and J_2.

Even with the criterion of Equation 1, we still need to exhaustively explore all the combinations of clusters (simple or joint) in order to choose the best ones. This is computationally infeasible. Thus, we have opted for a heuristic that merges each time only two existing clusters (simple or joint) to form a new joint cluster. The sketch of this method is illustrated in Algorithm REFINE. The algorithm takes as input a collection of simple clusters $\mathcal{P}$ and transforms it to a collection of joint clusters. The algorithm orders the

Algorithm: REFINE
Input : A set $\mathcal{P}$ of k^m-anonymous clusters
Output : A refinement of $\mathcal{P}$

1 **repeat**
2 Add to every joint cluster a virtual term chunk as the union of the term chunks of its simple clusters;
3 Order (joint) clusters in $\mathcal{P}$ according to the contents of their (virtual) term chunks;
4 Modify $\mathcal{P}$ by joining adjacent pairs of clusters (simple or joint) based on Equation 1;
5 **until** *there are no modifications in* $\mathcal{P}$;
6 **return** $\mathcal{P}$

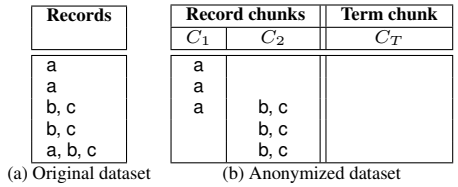
clusters of $\mathcal{P}$ as follows: a) each term t is given a *term chunks support* $tcs(\mathsf{t})$; i.e., the number of term chunks in clusters of $\mathcal{P}$ where t appears; b) the terms in term chunks are ordered in descending order of their tcs; and c) clusters are ordered by comparing lexicographically their term chunks. After the first iteration, joint clusters are introduced in $\mathcal{P}$. To each joint cluster J, we add a *virtual term chunk*, which is the union of the term chunks of its simple clusters, and we use it in the ordering step. REFINE modifies $\mathcal{P}$ by merging adjacent pairs of clusters and repeats the process until $\mathcal{P}$ does not change. The merging is done only if the criterion of Equation 1 is satisfied, and produces a joint cluster as defined in Section 3.

Correctness of the algorithm. Disassociation performs the partitioning (vertical and horizontal) and refining steps detailed in the previous sections. The proposed method is correct; it succeeds for any input and it always produces a disassociated k^m-anonymous dataset. It is not hard to verify that the algorithm terminates and produces a disassociated result. The proof that a disassociated dataset is k^m-anonymous is provided in Section 5. In a nutshell notice that *(a)* horizontal partitioning does not alter the original dataset and always produces clusters, *(b)* vertical partitioning creates k^m-anonymous clusters since Algorithm VERPART will put every term that has support over k to the record chunks (Lines 10-17) and the rest of the terms in the term chunk (Lines 6 and 18) and *(c)* refining has the trivial solution of not merging any clusters and if a joint cluster is created (i.e., if shared chunks are added), then k^m-anonymity is preserved as we prove in Section 5.

Complexity. The most expensive part of disassociation is the horizontal partitioning that has a worst case complexity of $O(|D|^2)$ time. The horizontal partitioning can be seen as a version of quicksort, which instead of a pivot uses the most frequent term to split each partition; in the worst case it will do $|D|$ partitionings and at each partitioning it has to re-order $|D|$ records. The complexity of vertical partitioning depends on the domain T^P of the input cluster P, and not on the characteristics of the complete dataset. The most expensive operation in the vertical partitioning is to establish whether a clunk is k^m-anonymous or not. This task requires examining $\binom{|T^P|}{m}$ combinations, thus it takes $O(|T^P|!)$ time. Since we regulate the size of clusters, the behavior of the overall algorithm, as the dataset size grows, is dominated by the behavior of the horizontal partitioning. Finally, the refining algorithm has again a $O(|D|^2)$ time complexity, since in the worst case it will perform as many passes over the clusters as the number of the clusters. Note that this a worst case analysis; in practice, the behavior of our algorithm is significantly better; this is also verified by the experimental evaluation of Section 7, which shows a linear increase of the computational cost with the input dataset size $|D|$.

5. ANONYMIZATION PROPERTIES

In Section 3, we described our disassociation transformation technique, which is implemented by the algorithm presented in Section



(a) Original dataset (b) Anonymized dataset

Figure 4: Illustration of Example 1, Original cluster size = 5

4. In this section, we prove how the disassociated result can guarantee k^m-anonymity, by showing how the transformed data can be used to reconstruct a possible initial dataset that contains k times any combination of m terms. In this proof we define two additional properties that must be preserved in a disassociated dataset.

Cluster anonymity. First, we prove that each disassociated cluster is k^m-anonymous, by constructing an initial cluster that contains k times any m terms of the disassociated cluster.

Let P be an arbitrary cluster of the anonymized dataset which is vertically partitioned into k^m anonymous record chunks $C_1, \ldots, C_v$ and a term chunk C_T. Then the following Lemma holds:

LEMMA 1. *For any m terms $\mathcal{S} = t_1, \ldots, t_m$ that appear in P, at least k distinct records that contain $\mathcal{S}$ can be reconstructed by combining subrecords from the chunks $C_1, \ldots, C_v$ and terms from C_T, or no record can be reconstructed that contains $\mathcal{S}$.*

PROOF. We first prove that Lemma 1 holds if all m terms fall inside the record chunks. In this case the m terms $\mathcal{S} = t_1, \ldots, t_m$ are scattered in $n, (n \leq m, n \leq v)$ record chunks $C_1, \ldots, C_n$. Let $S_1, \ldots, S_n$ be the subsets of $\mathcal{S}$ that appear in each of $C_1, \ldots, C_n$. Since each chunk is k^m anonymous, S_i will appear in the respective record chunk C_i at least k times together or none at all. The latter case happens if the S_i terms exist in disjoint groups of subrecords inside C_i. If there is even one of $S_1, \ldots, S_n$ whose terms do not appear together at all in the respective chunk, then the $\mathcal{S}$ terms cannot appear together in any reconstructed record. If every set of $S_1, \ldots, S_n$ appears together in the respective chunk, then it has to appear in at least k subrecords in each chunk. Let $SR_1, \ldots, SR_n$ be these sets of subrecords, one from each record chunk. We can then create a record by combining 1 subrecord from each of $SR_1, \ldots, SR_n$, i.e., $r = sr_1 \cup \cdots \cup sr_n$, where sr_i is a subrecord, $sr_i \in SR_i$. Since each SR_i contains at least k subrecords, we remove the used subrecord and repeat the process at least $k-1$ more times. This results to at least k distinct records that contain all $\mathcal{S}$ terms. Assume now that only $g, g < m$ terms fall inside the record chunks and $m - g$ terms fall in the term chunk. The previous proof holds for the g terms too, since $g < m$, thus k records can be reconstructed from the record chunks that contain the g items. We can then directly pad these k records with the rest of $m-g$ terms from the term chunk. We are free to do so, since the multiplicity and the correlations of these terms are not disclosed in the disassociated cluster. □

Lemma 1 shows that k records can be constructed from a disassociated cluster; still, this is not sufficient for providing k^m-anonymity as defined in Guarantee 1 as the following example illustrates.

EXAMPLE 1. *Let us consider the dataset of Figure 4a. Assume that we want to publish it as 3^2-anonymous and that we create two record chunks C_1 and C_2 with domains $T_1 = \{\mathsf{a}\}$, $T_2 = \{\mathsf{b,c}\}$ and $T_T = \emptyset$, as illustrated in Figure 4b. It is not hard to verify that all chunks are 3^2-anonymous and that Lemma 1 holds.*

Let us now consider an adversary A that knows: (a) *the anonymized dataset of Figure 4b,* (b) *that the size of the original cluster is 5 and* (c) *that a user had used terms* a *and* b*, i.e., $\{\mathsf{a,b}\}$ is a subrecord of the original dataset. Adversary A also knows that the original dataset is composed by a combination of the records stored in chunks C_1 and C_2.*

While the subrecords from C_1 and C_2 can be combined to create $k = 3$ records that contain a *and* b*, these records cannot appear in any original dataset, which must contain 5 records. It is not hard to verify that the only combination that results in a dataset with 5 records is the one presented in Figure 4a. Thus, no dataset that contains $\{\mathsf{a,b}\}$ 3 times can be the initial dataset of the example of Figure 4a. This way, the user's record $\{\mathsf{a,b,c}\}$ is revealed.*

Example 1 demonstrates that Lemma 1 is not sufficient to guarantee k^m-anonymity. Lemma 1 guarantees that k records that contain any m terms can be constructed, but it does not guarantee that these records can appear in a valid dataset of a predefined size. The sparsity of the original data, often leads to empty subrecords inside different chunks. Since there cannot be empty records, a record that is created as a result of combining empty subrecords is not valid. An initial dataset that contains an empty record is also not valid, thus and adversary can discard it. To enforce Guarantee 1, we must require not only that Lemma 1 holds, but also that these records can appear in a valid initial dataset. Fortunately, we do no need to reconstruct all possible original datasets to see if this condition is satisfied. It suffices to enforce the condition of the following lemma.

LEMMA 2. *Let $C_1, \ldots, C_v$ be the record chunks that correspond to the anonymization of a cluster P with size s. If* (a) *chunks $C_1, \ldots, C_v$ are k^m-anonymous and* (b) *the total number of subrecords in all chunks $\sum(|C_i|)$ is greater than or equal to $s + k \cdot (h-1)$, $h = min(m, v)$ or the term chunk is not empty, then Guarantee 1 holds.*

PROOF. To prove this lemma, it suffices to show that for every different combination of m items: (a) no record that contains the m terms can be constructed or (b) a valid initial cluster P_r of size s where the m terms appear in at least k records can be reconstructed.

Assume m random terms $t_1, \ldots, t_m$ from T^P. According to Lemma 1, given a disassociated cluster P_a, no record that contains these m terms can be created or at least k records can be reconstructed. In the former case, the k^m anonymity trivially holds (this case corresponds to a combination of m terms that did not appear in the initial dataset[1]) and it covers case (a). In the latter case, to prove (b) we need to show that these k records can appear in at least one valid reconstruction of the disassociated cluster P_a. A valid reconstruction of cluster P_a is a possible initial cluster that has s non-empty records. We construct a cluster that contains s records in total, where at least k of them contain $t_1, \ldots, t_m$ as follows. We first construct the k records, denoted as R_k that contain the m terms as described in Lemma 1. If the m terms are scattered in h chunks, then to construct each of these records we need h subrecords; one form each chunk, thus $k \cdot h$ subrecords. To create a valid initial dataset of size s that contains the R_k records we only need to populate it with $s - k$ additional records R_o that are valid i.e., non-empty. If the term chunk is non-empty then the $s - k$ records can be populated by randomly combining terms from the term chunk. If the term chunk is empty, we can create such records by assigning 1 subrecord that has not been used in the construction of R_k, from any of the $C_1, \ldots, C_v$ chunks,. The total number of subrecords needed is $h \cdot k + s - k = s + k \cdot (h-1)$. The worst

[1]If a combination of terms cannot be created by combining subrecords, it holds that it did not appear in the original data. The reverse is not true; if a combination can be created, it does not mean that it existed in the original data.

case, i.e., the maximum number of subrecords that are required for constructing a valid cluster, is when we need to combine one different subrecord for each of $t_1, \ldots, t_m$ to create a record of R_k. In this case, $h = m$ or if the cluster has less than m record chunks $h = v$. Thus, having $s + k \cdot (h-1)$, $h = min(m, v)$ subrecords is sufficient to create a valid initial cluster. □

Joint cluster anonymity. An example which demonstrates that careless creation of shared chunks can lead to cases where combinations of m terms might not appear k times in any reconstructed dataset is depicted in Figure 5a. Although every chunk (i.e., vertical partition) in the illustrated dataset is 3^2-anonymous, the overall dataset is not. Since each record has set semantics, an adversary can discard initial datasets that contain records with two identical terms. An attacker A knowing that a user U asked for terms x and o can only find one matching record in every possible original dataset using the following reasoning. Term x appears only in the 1^{st} cluster (always together with a) and o appears in the shared chunk. Thus, to construct U's record, A has to combine {a,x} with any of {a,x}, {a} and {o}; but, by the semantics of shared chunks, the only allowed combination is {a,x,o} which appears just once. In order to avoid these conflicts we enforce the following property.

PROPERTY 1. *Let J be a joint cluster and T^r be the set of terms that appear in the record and shared chunks of the clusters (joint or not) forming J. A shared chunk of J that does not contain terms from T^r must be k^m-anonymous; if it does, it must be k-anonymous.*

For example in Figure 5a, Property 1 does not hold since T^r = {a,b,c,d,e,f,x}, term a appears on the shared chunk, a $\in T^r$ and the shared chunk is not k-anonymous. On the other hand, the property holds for Figure 5b. T^r contains all terms that appear in J except those that are placed in term chunks and those that appear only in J's shared chunks (only o in the previous example). Let us now consider the following lemma.

LEMMA 3. *A joint cluster for which Property 1 holds, is k^m-anonymous.*

PROOF. We will prove the Lemma by induction. Lemma 2 shows that simple clusters are k^m-anonymous. It is also easy to see why joint clusters who contain only simple clusters are k^m-anonymous, since no conflicts between the terms of the record and shared chunks appear there. In the following we will prove the inductive step; a joint cluster J that is formed by existing k^m-anonymous joint clusters is k^m-anonymous too.

Let J be a joint cluster with domain T^J, the k^m-anonymous joint clusters $J_1, \ldots, J_q$ be its children and the simple clusters k^m-anonymous $P_1, \ldots, P_w$ be its leaves. Let $\mathcal{SC}$ be the set of the shared chunks of J that all satisfy Property 1. Moreover, let T^r be the set of terms that appear in the record and shared chunks of $J_1, \ldots, J_q$. Since $J_1, \ldots, J_q$ are k^m anonymous we only have to check how the introduction of the shared chunks $\mathcal{SC}$ affects anonymity. Because Lemma 2 holds for each cluster independently, there is no need to set a new bound for the number of subrecords contained in $\mathcal{SC}$. We only have to show that the addition of $\mathcal{SC}$ allows the creation of k records (or no record at all) that contain any m-sized combination of terms from T^J.

Assume a random combination of m terms $t_1, \ldots, t_m$ from T^J where terms $t_1, \ldots, t_i$ appear in $J_1, \ldots, J_q$ (in either record or term chunks) and $t_{i+1}, \ldots, t_m$ appear in the shared chunks $\mathcal{SC}$. If $i = m$, i.e., all terms belong to $J_1, \ldots, J_q$, then k^m anonymity holds since we assumed that $J_1, \ldots, J_q$ are k^m-anonymous. If $i = 0$, i.e., all terms belong to the shared chunks, then by following the same constructive proof as we did in Lemma 1 we can create k records that contain $t_1, \ldots, t_m$. This is sufficient for proving k^m-anonymity since there is no requirement for the number of subrecords in the shared chunks. Finally, if some of the m terms cannot appear together by any combination of subrecords, i.e., they did not appear together in the original data at all, then the k^m-anonymity trivially holds. It remains to prove that J is k^m-anonymous for $0 < i < m$.

Let $SC_1, \ldots, SC_n$, $n \leq m$, with domains $T^1, \ldots, T^n$ be the shared chunks of $\mathcal{SC}$ that contain $t_{i+1}, \ldots, t_m$. Using the reconstructed clusters of $J_1, \ldots, J_q$ we partially reconstruct a joint cluster J_r that contains at least k records with the terms $t_1, \ldots, t_i$. Let PR be the partially reconstructed records of J that contain $t_1, \ldots, t_i$, $|PR| \geq k$. We expand the PR with subrecords from each SC_i of $SC_1, \ldots, SC_n$ to create records that contain all m terms. For each of SC_i with domain T^i we have two cases:

$T^r \cap T^i = \emptyset$ **holds:** In this case, SC_i is k^m-anonymous and no term from SC_i appears in any of the PR records. We can then select k subrecords that contain the terms from $t_{i+1}, \ldots, t_m$ that fall in T^i and concatenate them to k records of PR.

$T^r \cap T^i \neq \emptyset$ **holds:** In this case, SC_i is k-anonymous. Let SR_i be the records of SC_i that contain the terms of $t_1, \ldots, t_m$ that fall in T^i, $|SR_i| \geq k$. We want to append k subrecords from SR_i to k records of PR to create records that contain all m terms. Still, not all combinations of $PR \times SR_i$ are valid due to conflicts. The conflicts are caused by terms that appear both in the subrecords of SC_i and the records of J_r that have partially been constructed until now. Assume that the conflict is based on a term a. a is independent of $t_1, \ldots, t_m$. Assume that a appears in the record or shared chunks of the simple or joint clusters $\mathcal{J}_a$, which are descendants of J. The existence of a in these record chunks means that SR_a did not exist in any of $\mathcal{J}_a$, thus the records of SR_a cannot be combined with any of the records of $\mathcal{J}_a$. Let JR_a be the partially reconstructed records of $\mathcal{J}_a$. Because of the conflict, the adversary knows that if any record of PR belongs to JR_a too, then it cannot be combined with SR_a to create the k records we need. To guarantee k^m anonymity, we must be able to combine at least k records from $PR' = PR \setminus JR_a$ and $SR'_i = SR_i \setminus SR_a$ or none at all. We will prove this by showing that either all records of PR' and all subrecords of SR'_i are disqualified, or that at least k remain in each set. Since each joint cluster is anonymized independently, it contributes at least k records to PR. Any conflict with even one record of a cluster from $\mathcal{J}_a$ disqualifies all the records from the same cluster, thus JR_a will be equal to PR or they will differ at least by k records, i.e., all the records contributed by a cluster that has no conflict. Thus $|PR'| = 0 \vee |PR'| \geq k$. Moreover, since we required that S_i is k-anonymous, there will be at least k duplicates of each record. A conflict over term a will disqualify at least k records, and if records without a exist in SR' there will be at least k of them. So, after eliminating conflicts, $|SR'| = 0 \vee |SR'| \geq k$. Since both PR' and SR' will have either more than k records or none after eliminating conflicting records, we can either create k records that contain all $t_1, \ldots, t_m$ or no such record. □

The proof is similar for conflicts based on more than one item.

Since a disassociated dataset consists of either joint or simple clusters, Lemmas 2 and 3 are sufficient to prove that the whole dataset is k^m-anonymous. We only have to show that the properties required by the previous Lemmas can be guaranteed by the algorithm of Section 4. To guarantee the property required by Lemma 2 we only need to add a check at the end of VERPART that verifies that the cluster contains enough subrecords. If the size limit is not

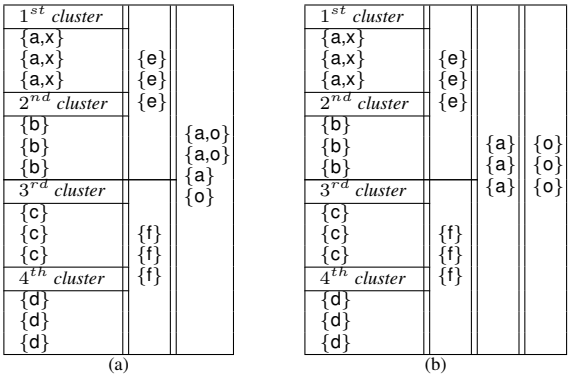


Figure 5: Unsafe (a) vs. safe (b) creation of a shared chunk

met, then by moving the least frequent item of the record chunks to the term chunk, we guarantee that the conditions set by Lemma 2 are satisfied. This solution is always feasible; at least one term will exist to populate the term chunk in each cluster. To satisfy Lemma 3 the refining algorithm has to check in the creation of a shared chunk, if any of its terms appears in the record chunk of any descendant joint or simple cluster. If this holds, then the chunk must be k-anonymous, else it can be k^m-anonymous. Since there is always the trivial solution of a record chunk that contains only 1 term, which is both k-anonymous and k^m-anonymous, the refining algorithm always produces a k^m-anonymous dataset.

Protection against stronger adversaries. k^m-anonymity is a conditional guarantee and the protection it offers is reduced against adversaries with background knowledge that exceeds the attack model assumptions. The most common case is to have adversaries that have more knowledge than m terms about a user or adversaries that have background knowledge about all users that contain certain m terms. In both cases, the adversary's background knowledge consists of enough information to accurately associate some records to a known group of users $\mathcal{U}$. This allows the adversary to remove these records from the groups of candidate records that match her background knowledge for any user who does not belong to $\mathcal{U}$. Still, this attack does not lead automatically to complete re-identification of the additional users, but reduces the number of candidates according to their overlap with the records that are associated with $\mathcal{U}$. This type of attacks has been studied in other contexts [2, 32] and their effect on disassociation and generalization based methods is similar. Disassociation has an additional weakness that is related to Lemma 2; if a record of a user is identified and the remaining terms violate Lemma 2, then the probability of identifying additional records might be reduced to less than k-1.

Diversity. So far we have discussed an anonymization framework offering protection against identity disclosure. In this section, we discuss how the proposed framework may also offer protection against attribute disclosure and achieve l-diversity.

Former works that guarantee l-diversity, separate sensitive attributes from quasi identifiers [11, 18, 30]. Following the same idea, we can enforce l-diversity in our framework by *(a)* ignoring all sensitive values in the horizontal partitioning and *(b)* placing all sensitive values in the term chunk at the vertical partitioning stage. In the resulting data, all sensitive values will reside at the term chunk and no association between them and any other subrecord or value can be done with probability over than $1/|C|$, where $|C|$ is the size of the cluster. By adjusting the size of the clusters, the anonymization method achieves the desired degree of l-diversity.

The proposed anonymization framework offers protection against both identity and attribute disclosure. We focus on the former because to the best of our knowledge there is no other work that employs a similar to disassociation transformation to guarantee protection against identity disclosure (works enforcing l-diversity do not consider re-identification dangers [11, 18, 30]). We expect that in practice both protection against identity and attribute disclosure (for the recognized sensitive values) are needed.

6. INFORMATION LOSS

By definition, disassociation incurs a different information loss compared to classic anonymization methods. The disassociated dataset preserves all the initial terms and many of the initial itemsets. An analyst can work directly on the disassociated dataset or reconstruct a possible initial one. In the former case, the analyst can compute lower bounds of the supports of all terms and itemsets. These bounds can be computed by counting all the appearances of terms and itemsets in the record chunks of the simple and joint clusters and by adding one to the support of each term that appears in a term chunk. Moreover, the analyst can employ models for answering queries in probabilistic databases to directly query the anonymization result [9]. Using such a model, one can assume that the contents of each record chunk are possible assignments to every record of the cluster with probability $(1/|P|)$. Still, existing work on uncertain data management is not tailored to the disassociated dataset and does not take advantage of the constraints in the reconstruction procedure that we detailed in Section 3 to increase the quality of the estimations. Moreover, working directly on the disassociated dataset requires adjusting existing tools and models for analyzing data. Because of this, we believe that it is easier to apply most analysis tasks on a reconstructed dataset. During horizontal partitioning, clusters are created by bringing similar records together; thus, the statistical properties of a reconstructed dataset are quite close to the original one. A way to further increase the accuracy of the analysis on reconstructed data is to create more than one reconstructed datasets and average the query results on them. We experimentally evaluate the similarity between the reconstructed datasets and the original one in Section 7.

Disassociation hides infrequent term combinations, therefore the incurred information loss is related to term combinations that exist in the original dataset but are lost in the disassociated dataset. To assess the impact of the information loss, we examine the behavior of common mining and querying operations on the transformed data. We employ metrics that are generic and can be used as a comparison basis with anonymization methods that employ different data transformations (such as generalization, suppression and differential privacy). More specifically, we examine how many of the frequent itemsets that exist in the original data are preserved in the published data, and we also measure the relative error in the estimation of the supports of pairs of items.

Top-K deviation (tKd). The tKd metric measures how the top-K frequent itemsets of the original dataset change in the published anonymized data. Let FI (respectively, FI') be the top-K frequent itemsets in the original dataset (respectively, the anonymized dataset); tKd is defined as follows:

$$tKd = 1 - \frac{|FI \cap FI'|}{|FI|} \tag{2}$$

Intuitively, tKd expresses the ratio of the top-K frequent itemsets of the original dataset that appear in the top-K frequent itemsets of the anonymized data.

To compare disassociation with generalization-based methods, we define an appropriate version of tKd, called the *top-k multiple level mining loss* tKd-ML^2, which is based on the ML^2 metric

Dataset	$\|D\|$	$\|T\|$	max rec. size	avg rec. size
POS	515,597	1,657	164	6.5
*WV*1	59,602	497	267	2.5
*WV*2	77,512	3,340	161	5.0

Figure 6: Experimental datasets

defined in [27]. Mining a dataset at multiple levels of a generalization hierarchy is an established technique [12], which allows detecting frequent association rules and frequent itemsets that might not appear in the most detailed level of the data. If a generalization hierarchy that allows the anonymization of the data exists, then we can assume that the same hierarchy can be used to mine frequent itemsets from the published (and the original) data at different levels of abstraction. $tKd\text{-}ML^2$ is given again by Equation 2, but in this case FI and FI' are the sets of generalized frequent itemsets that can be traced in the original and anonymized data, respectively. In the case of generalized datasets, a generalized frequent itemset is lost if it contains terms that have been generalized at a higher level during the anonymization process. Reconstructed datasets do not contain any generalized items, but given a generalization hierarchy generalized frequent itemsets can be mined.

Relative error (re). This metric (used also in [6]) is used to measure the relative error in the support of term combinations in the published data. Since there is a huge number of possible combinations, we limit ourselves to combinations of size two as an indication of the dataset quality. Larger combinations are usually infrequent, and the case of very frequent ones is already covered by tKd. The relative error is defined as follows:

$$re = \frac{|s_o(a,b) - s_p(a,b)|}{AVG(s_o(a,b),\ s_p(a,b))}, \quad (3)$$

where $s_o(a,b)$ and $s_p(a,b)$ is the support of the combination of terms (a,b) in the original and in the published data, respectively. Reconstructing anonymized datasets might introduce new item combinations in the published data, which did not exist in the original data. In order to take them into account in the definition of the relative error, we use the average of the two supports as denominator, instead of using the original support $s_o(a,b)$. The average has a smoothing effect on the metric, since it normalizes re to $[0,2]$, and avoids divisions by 0.

7. EXPERIMENTAL EVALUATION

The goal of the experimental evaluation is to demonstrate the advantages that disassociation in preserving data quality and to show that disassociation has a robust behavior in different settings.

7.1 Experimental Settings

Datasets. In the experiments, we use the 3 real datasets described in Figure 6, which were introduced in [33]. Dataset *POS* is a transaction log from an electronics retailer. Datasets *WV*1 and *WV*2 contain click-stream data from two e-commerce web sites, collected over a period of several months. Synthetic datasets were created with IBM's Quest market-basket synthetic data generator (http://www.almaden.ibm.com/cs/quest/syndata.html). Unless otherwise stated, the default characteristics for the synthetic datasets are $1M$ records, $5k$ term domain and 10 average record length.

Evaluation metrics. We measure the information loss incurred by our method with respect to the following: *(a)* the tKd, $tKd\text{-}ML^2$, and re measures defined in Section 6 and *(b)* the percentage of terms $tlost$ that have support more than k in the original dataset D but they are placed in term chunks by our method. We report tKd and re for the disassociated datasets calculated in two different ways: *(a)* one on a single random reconstructed dataset, labeled as tKd and re, and *(b)* one calculated only by taking into account the subrecords that appear inside the record and shared chunks, labeled as $tKd\text{-}a$ and $re\text{-}a$. In the latter case, we do not take into account the probability that an itemset can be created by combining subrecords. $tKd\text{-}a$ and $re\text{-}a$ trace the itemsets that would exist in *any* reconstructed dataset, thus they are based on lower bounds of itemset supports in the original dataset. tKd and $tKd\text{-}ML^2$ are measured for the 1000 most frequent itemsets. Finally, computing an average re on all combinations of size 2 is not very informative in cases of skewed distributions and large domains. The majority of combinations would be rare or would not exist at all in the original data, but they would dominate the result. To avoid this, we ordered the domain of each dataset by descending term support and we used a small range of consecutive terms to trace their re. After some testing we chose the 200th-220th most frequent terms. re in this case is an indicator of how well less frequent but not utterly rare combinations are preserved in the anonymized dataset.

Evaluation parameters. We compared performance by varying the following parameters: *(a)* k, *(b)* the size of the dataset, *(c)* the size of the dataset's domain, *(d)* the average size of the records, *(e)* the terms we use to calculate re and *(f)* the number of reconstructed datasets we use to calculate re and tKd. We do not present a detailed evaluation for m, because in all the available datasets its effect for values $m > 2$ is negligible. The explanation for this is that most record clusters are k^m-anonymous for any m either because they have gathered very frequent terms or because they contain small subrecords. The experiments are all performed with $k = 5, m = 2$ unless explicitly stated otherwise.

Comparison to state-of-the-art. Comparing disassociation to other methods is not straightforward; no other method offers the same privacy guarantee while introducing the same type of information loss. We chose to compare disassociation to the generalization-based *Apriori* approach [27], since it offers the same privacy guarantee and it is the most closely related method. This comparison allows us to see how the different data transformations, generalization and disassociation, affect the quality of the anonymized dataset in a similar privacy framework. Furthermore, we compare disassociation to *DiffPart* [6], which offers differential privacy for set-valued data by suppressing infrequent terms and adding noise. The comparison with *DiffPart* demonstrates the gains disassociation offers in terms of information quality, when a more relaxed guarantee like k^m-anonymity is chosen over differential privacy. All methods were implemented in C++ (g++ 4.3.2).

7.2 Experimental Results

The first experiment (Figures 7a-d) investigates the information loss by our method on the real datasets. In Figure 7a, we see the result of disassociation in the quality of all datasets and in Figures 7b-d just for the *POS* dataset. The $tKd\text{-}a$ in Figure 7a is similar for all datasets, showing that the most frequent combinations are preserved for different data characteristics. Still, when we trace tKd on the reconstructed datasets, the results significantly improve only for the *POS* dataset, which is the largest of the 3 and its records have the longest average length. This reflects the fact that disassociation managed to create multiple record chunks for *POS*. The combinations of their contents results to a significantly better reconstructed dataset. Disassociation produces significantly different results for the 3 datasets, when looking to the re and $re\text{-}a$ metrics. The supports of the combinations traced by re are preserved better when the ratio of the dataset size to the dataset domain is high. This ratio is higher for *POS* and *WV1*, where re has significantly superior results to $re\text{-}a$. This indicates the gains from combining terms from

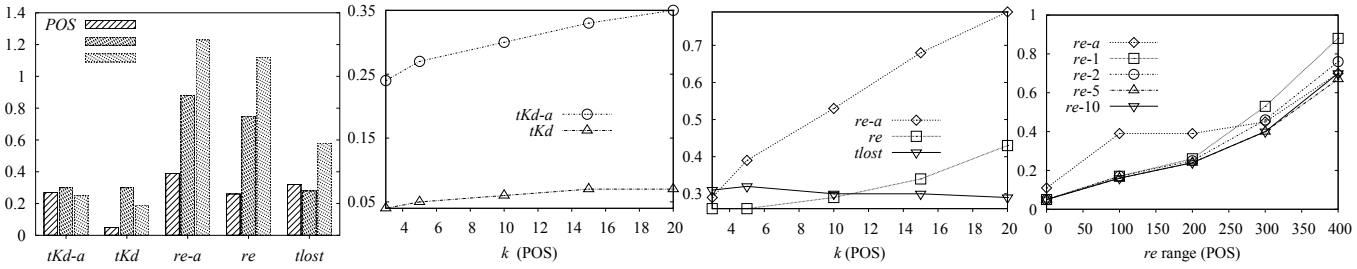


Figure 7: Information loss on real data (a-d)

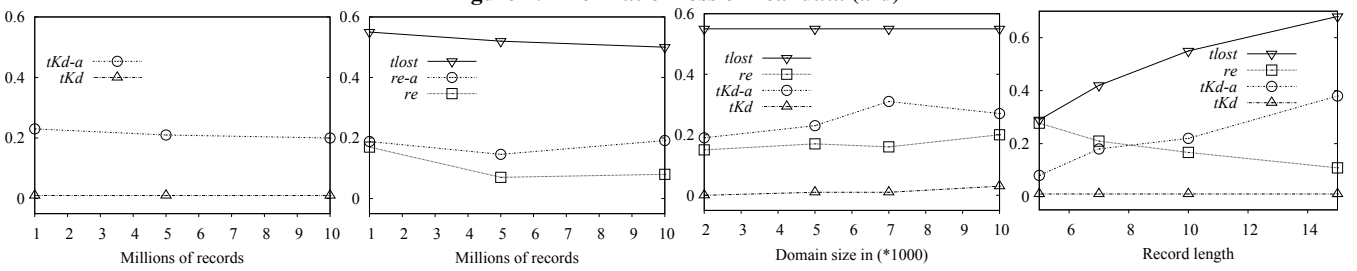


Figure 8: Information loss on synthetic data (a-d)

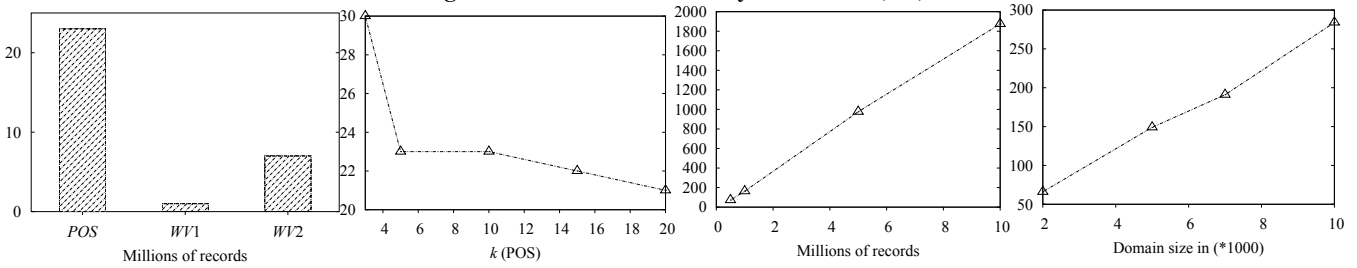


Figure 9: Performance on real data (a-b)

Figure 10: Performance on synthetic data (a-b)

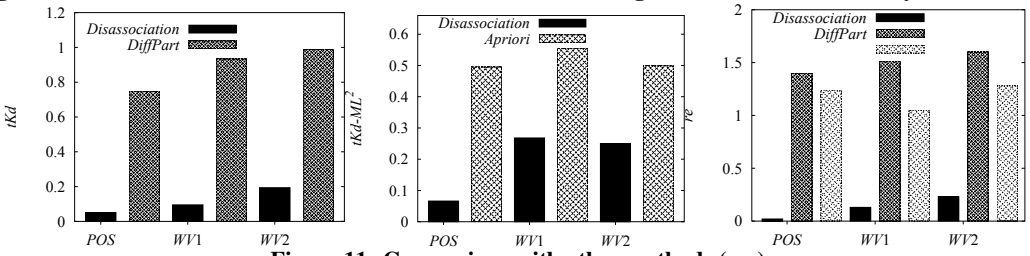


Figure 11: Comparison with other methods (a-c)

different record chunks in the reconstructed datasets. Finally, the same ratio affects how many terms are placed in the record chunk, as reported by $tlost$, but to a lesser degree. In Figures 7b and 7c, we see how information loss escalates as the power of the guarantee, expressed by the k parameter, grows. The measures that depend on the most frequent items and itemsets are only slightly affected (Figure 7b), since the disassociation algorithm preserves them in record chunks. On the other hand, re, which does not depend on the most frequent items, increases linearly with k, but with a low rate (Figure 7c). In Figure 7d we explore the gain in information quality by creating several reconstructed datasets and averaging the itemset supports on them. We created 10 random reconstructions of the anonymized *POS* dataset, and we traced re taking this time into account the average supports of the itemsets in 2 (re-$r2$), 5 (re-$r5$) and 10 (re-$r5$) of the reconstructed datasets. We do not report results for tKd since they were already close to 0 and did not benefit substantially from multiple reconstructions. We measure the re on the combinations of the 0-20, 100-120, 200-220, 300-320 and 400-420 most frequent terms in *POS*. In the x-axis of Figure 7d, we depict the frequency order of the terms; e.g. a point over 100 refers to the re of the combinations of the 100th-120th most frequent terms in *POS*. When the terms are frequent, the support of their combinations is reported accurately in any reconstructed dataset, so taking the average does not provide any benefit. As the combinations become less frequent, using more than one reconstructed datasets allows for more accurate estimations. In the previous experiments we also examined separately how frequent itemsets of size less or equal to m and of size greater than m ($m = 2$) are preserved. We did not notice any systematic behavior; depending on the dataset, any of the aforementioned frequent itemset classes may be preserved better. For example, frequent itemsets smaller than m were preserved better in *POS* and in *WV2* and worse in *WV1*. We do not report detailed results due to space limitations.

In the experiments of Figure 8 we used synthetic data to see how the information loss is affected, when the dataset characteristics variate. Since the anonymization is applied independently on each cluster, the database size does not have a significant effect on the quality of the results as demonstrated in Figures 8a and 8b. Only the re and re-a are positively affected, because the terms it traces become more frequent and they end up in record chunks more of-

ten. Moreover, in Figure 8c we see that increasing the domain when the distribution is skewed, basically affects the tail of the distribution, thus it does not have a significant effect on frequent combinations of terms traced by tKd, whereas re slightly deteriorates. The effect of record length is depicted in Figure 8d. Having larger records results in more record chunks and more rare terms in each cluster, thus tKd-a and $tlost$ increase. On the other hand, when we keep the dataset and domain size constant and we only increase the record size, the support of the terms in the dataset increases and this explains how re benefits from larger records. Finally, tKd remains close to 0 for all record sizes, since the multiple record chunks reconstruct most of the frequent itemsets in the reconstructed dataset.

Figures 9 and 10 illustrate the performance of the proposed algorithm in terms of CPU time (results in seconds). Disassociation is not significantly affected by the value of k, and at the same time it scales linearly to the dataset and the domain size.

Figure 11 shows how disassociation performs compared to *DiffPart* and the *Apriori* algorithm. The graphs illustrate the impact of all algorithms on the quality of the anonymized dataset for $k = 5$ and $m = 2$ (*DiffPart* is unaffected by this parameter). For the *DiffPart* algorithm we used privacy budgets ranging from 0.5 to 1.25, using a step 0.25 with the same parameters as in [6] and we report the best results. In Figure 11a we see how disassociation compares to *DiffPart* in terms of tKd. Since in both cases the anonymized datasets contain only original terms (the differential private one has only a subset of them) tKd is computed in exactly the same way. The trade-off for using a stronger privacy guarantee like differential privacy is quite important; in the best case 75% of the top frequent items have been lost, whereas disassociation loses only 5% in the same experiment. In Figure 11b we see how disassociation compares to *Apriori* in terms of tKd-ML^2, since no original frequent itemset appears in the generalized dataset. Disassociation performs again significantly better than *Apriori* especially for *POS* which is the largest dataset and has more frequent terms than *WV1* and *WV2*. A problem of *Apriori* is that few uncommon terms cause the generalization of several common ones. Finally, Figure 11c shows how all algorithms compare in terms of re. re in the generalized dataset is calculated by uniformly dividing the support of a generalized term to the original terms that map to it. *DiffPart* has suppressed all the 200-220th most frequent terms in *POS* (less that 100 of the original 1657 terms are left), so in order to make the comparison meaningful we report the re for the (0-20th) most frequent terms. The re for both *DiffPart* and *Apriori* is over 1, which indicates that the supports of the term combinations have limited usefulness for analysis, whereas disassociation provides 0.18 re in the worst case.

In summary, the experiments on both real and synthetic datasets demonstrate that disassociation offers an anonymized dataset of significantly superior quality compared to other state-of-the-art methods. Moreover, the information loss does not increase aggressively as k increases. Finally, disassociation is not computationally expensive and it is practical for large datasets.

8. RELATED WORK

Privacy preservation was first studied in the relational context and focused on protection against identity disclosure. In [25, 26] the authors introduce k-anonymity and use generalization and suppression as their two basic tools for anonymizing a dataset. *Incognito* [15] and *Mondrian* [16] are two well known algorithms that guarantee k-anonymity for a relation table by transforming the original data using global (full-domain) and local recoding, respectively. [21] demonstrates that the information loss, when providing k-anonymity, can be reduced by using *natural* domain generalization hierarchies (as opposed to user-defined ones).

To address the problem of attribute disclosure, where a person can be associated with a sensitive value, the concept of ℓ-diversity [20] was introduced. *Anatomy* [30] provides ℓ-diversity and lies closer to our work, in the sense that it does not generalize or suppress the data, but instead it disassociates them by publishing them separately. Still, the anonymization approach is restricted to relational data and it does not protect against identity disclosure. *Slicing*, a more flexible version of *Anatomy* appears in [18]. *Slicing* guarantees l-diversity as *Anatomy*, but instead of completely separating sensitive attributes from quasi-identifies, it might publish some quasi-identifiers without disassociating them from sensitive values, if the diversity guarantee is not violated. Moreover, *Slicing*, disassociates quasi-identifiers to increase protection from membership disclosure. By disassociating quasi identifiers, an adversary is faced with several options for reconstructing each record, thus she cannot be certain that a specific record existed in the original data. The data transformation is similar to the approach of our work, but there are significant differences: a) there is no protection against identity disclosure and b) the disassociation between quasi-identifiers does not provide any privacy guarantee, and it takes place only if the impact on information loss is limited. Protection against membership disclosure is facilitated, but not guaranteed; it is roughly estimated using the number of attribute combinations, and not guaranteed by considering the possible initial datasets as in our work. The issues of empty and duplicate records are not addressed. Our work differs from *Slicing* mainly because it uses the disassociation of quasi-identifiers to provide a guarantee against identity, and because it addresses *sparse* multidimensional values.[2].

A similar idea, the vertical fragmentation of relational tables, is employed in a different context to guarantee user anonymity in [7]. The proposed technique distributes a relational table to different servers. In each server, only a subset of the relation's attributes are available unencrypted. The subsets that are available without encryption are chosen so that sensitive associations between attributes, captured by *confidentiality constraints*, are broken. Fragmentation is similar to the basic idea in our work and in [30, 18], but the anonymization model is very different since it focuses on known confidentiality constraints; attacks based on background knowledge are not considered.

More recently, a stronger privacy preservation paradigm, differential privacy, has been proposed [10]. Differential privacy is independent of adversary's background knowledge and it roughly requires that the existence of every single record in the data does not have a significant impact in any query. Finally, the work of [8], although focusing at the protection of associations in sparse bipartite graphs, is related to our work because of the way they define their semantics. The anonymization technique of [8] replaces each node of the graph with a safe group of labels, allowing in this way the anonymized graph to be matched to multiple possible initial graphs.

Privacy on set-valued data. The works that lie closer to this paper are those for privacy on set-valued data. Most works that provide protection against identity disclosure rely on generalization. An efficient algorithm for classical k-anonymity in a set-value context appears in [13]. [27, 28] introduce the k^m anonymity guarantee, which is used and extended in this paper. The authors provide algorithms for anonymizing the data that, unlike our approach, are based on generalization, employing both local and global recoding. In [4] an algorithm for providing k^m-anonymity using only

[2]In [18] there is an application of *Slicing* to the Netflix data [23], which are sparse. This is achieved by padding all null values with the average of the corresponding attribute values. This technique works only for specific types of data processing and cannot address of sparse data in general.

suppression is proposed. The authors have a similar motivation to our work and focus on web search query logs, which they anonymize by removing terms that violate k^m-anonymity. The proposed method preserves original terms, but due to the large tail of the term support distribution in such logs, it removes 90% of the terms even for low k and m values. In a different setting, [22] studied multirelational k-anonymity, which can be translated to a problem similar to the one studied here, but the anonymization procedure still relies on generalization. [31] provide protection both against identity and attribute disclosure by relying on suppression.

Protection against attribute disclosure is provided both by generalization and disassociation transformations. The work of [11] extends [30] to provide ℓ-diversity for transactional datasets with a large number of items per transaction, but it does not depart from the anonymization framework of [30]; it still has a separate set of quasi-identifiers and sensitive values. The basic idea of [11] is to create equivalence classes where the quasi-identifiers are published separately from the sensitive values and their supports. [5] provides a more elaborate ℓ-diversity guarantee for sparse multidimensional data, termed ρ-uncertainty, where sensitive items can act as quasi-identifiers too. Still, unlike our approach, generalization and suppression are employed to anonymize the data.

There have been few works that investigate the publication of set-valued data under differential privacy guaranties. [14] focuses on the anonymization of web search logs, using the AOL data [3]. The proposed method that guarantees differential privacy but it only publishes query terms and not records. Moreover, the anonymization procedure completely hides all terms that are infrequent, which are the majority of terms in AOL data. In [6] a method for publishing itemsets instead of isolated terms from a set-valued collection of data is proposed. The *DiffPart* algorithm follows a top down approach, which starts from the unification of the whole domain and refines it by partitioning it to subdomains, if the item combinations can be published without breaching differential privacy.

Our work lies closer to [11, 30, 18] in the sense that it does not suppress or generalize the data but instead it severs the links between values attributed to the same entity. Unlike [11, 30, 18] we focus on identity protection, and not simply on separating sensitive values from quasi-identifiers. The work of [18] has the most similar data transformation, but it solves a different problem and does not address the peculiarities of sparse multidimensional data. Our privacy guarantee comes from [27], but we follow a completely different path with respect to the data transformation and the type of targeted data utility.

9. CONCLUSIONS

In this paper, we proposed a novel anonymization method for sparse multidimensional data. Our method guarantees k^m-anonymity, for the published dataset using a novel data transformation called *disassociation*. Instead of eliminating identifying information by not publishing many original terms, either by suppressing or generalizing them, we partition the records so that the existence of certain terms in a record is obscured. This transformation introduces a different type of information loss from existing methods, making it a valuable alternative when the original terms are important.

10. REFERENCES

[1] C. Aggarwal. On k-anonymity and the curse of dimensionality. In *VLDB*, pp. 901-909, 2005.

[2] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi. Anonymity preserving pattern discovery. *VLDB Journal*, 17(4):703-727, 2008.

[3] M. Barbaro and T. Zeller. A face is exposed for AOL searcher no. 4417749. New York Times, 2006.

[4] T. Burghardt, K. Böhm, A. Guttmann, and C. Clifton. Anonymous search histories featuring personalized advertisement - balancing privacy with economic interests. *TDP*, 4(1):31-50, 2011.

[5] J. Cao, P. Karras, C. Raissi, and K.-L. Tan. ρ-uncertainty: inference-proof transaction anonymization. *PVLDB*, 3(1-2):1033-1044, 2010.

[6] R. Chen, M. Noman, B. C. Fung, B. C. Desai, and L. Xiong. Publishing set-valued data via differential privacy. *PVLDB*, 4(11):1087-1098, 2011.

[7] V. Ciriani, S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Combining fragmentation and encryption to protect privacy in data storage. *TISSEC*, 13(3):1-33, 2010.

[8] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang. Anonymizing bipartite graph data using safe groupings. *PVLDB*, 1(1):833-844, 2008.

[9] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, pp. 864-875, 2004.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. *TCC*, pp. 265-284, 2006.

[11] G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In *ICDE*, pp. 715-724, 2008.

[12] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *VLDB*, pp. 420-431, 1995.

[13] Y. He and J. F. Naughton. Anonymization of set-valued data via top-down, local generalization. *PVLDB*, 2(1):934-945, 2009.

[14] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing search queries and clicks privately. In *WWW*, pp. 171-180, 2009.

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

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

[17] J. Li, R. C.-W. Wong, A. W.-C. Fu, and J. Pei. Anonymization by local recoding in data with attribute hierarchical taxonomies. *TKDE*, 20(9):1181-1194, 2008.

[18] T. Li, N. Li, J. Zhang, and I. Molloy. Slicing: a new approach to privacy preserving data publishing. *TKDE*, 24(3):561-574, 2012.

[19] G. Loukides, A. Gkoulalas-Divanis, and B. Malin. Anonymization of electronic medical records for validating genome-wide association studies. *PNAS*, 17:7898-7903, 2010.

[20] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: privacy beyond k-anonymity. In *ICDE*, pp. 24, 2006.

[21] M. Nergiz and C. Clifton. Thoughts on k-anonymization. *DKE*, 63(3):622-645, 2007.

[22] M. Nergiz, C. Clifton, and A. Nergiz. Multirelational k-anonymity. In *ICDE*, pp. 1417-1421, 2007.

[23] Netflix Prize FAQ. http://www.netflixprize.com/faq, 2009.

[24] H. Pang, X. Ding, and X. Xiao. Embellishing text search queries to protect user privacy. *PVLDB*, 3(1-2):598-607, 2010.

[25] P. Samarati. Protecting respondents' identities in microdata release. *TKDE*, 13(6):1010-1027, 2001.

[26] L. Sweeney. k-anonymity: a model for protecting privacy. *IJUFKS*, 10(5):557-570, 2002.

[27] M. Terrovitis, N. Mamoulis, and P. Kalnis. Privacy-preserving anonymization of set-valued data. *PVLDB*, 1(1):115-125, 2008.

[28] M. Terrovitis, N. Mamoulis, and P. Kalnis. Local and global recoding methods for anonymizing set-valued data. *VLDB Journal*, 20(1):83-106, 2010.

[29] K. Wang, C. Xu, and B. Liu. Clustering transactions using large items. In *CIKM*, pp. 483-490, 1999.

[30] X. Xiao and Y. Tao. Anatomy: simple and effective privacy preservation. In *VLDB*, pp. 139-150, 2006.

[31] Y. Xu, K. Wang, A. W.-C. Fu, and P. S. Yu. Anonymizing transaction databases for publication. In *KDD*, pp. 767-775, 2008.

[32] 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*, pp. 72-83, 2009.

[33] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *KDD*, pp. 401-406, 2001.

Baquara: A Holistic Ontological Framework for Movement Analysis using Linked Data

Renato Fileto

PPGCC/INE - CTC, Federal University of Santa Catarina, Florianópolis, SC, Brazil
r.fileto@ufsc.br

Marcelo Krüger

CTTMar, University of Itajaí Valley (UNIVALI), São José, SC, Brazil
marcelo_kruger@univali.br

Nikos Pelekis[1], Yannis Theodoridis[2]

[1]Department of Statistics & Insurance Science, [2]Department of Informatics,
University of Piraeus, Greece
{npelekis,ytheod}@unipi.gr

Chiara Renso

KDDLab - IST/CNR, Pisa, Italy
chiara.renso@isti.cnr.it

Abstract. Movement understanding frequently requires further information and knowledge than what can be obtained from bare spatio-temporal traces. Despite recent progress in trajectory data management, there is still a gap between the spatio-temporal aspects and the semantics involved. This gap hinders trajectory analysis benefiting from growing collections of linked data, with well-defined and widely agreed semantics, already available on the Web. This article introduces Baquara, an ontology with rich constructs, associated with a system architecture and an approach to narrow this gap. The Baquara ontology functions as a conceptual framework for semantic enrichment of movement data with annotations based on linked data. The proposed architecture and approach reveal new possibilities for trajectory analysis, using database management systems and triple stores extended with spatial data and operators. The viability of the proposal and the expressiveness of the Baquara ontology and enabled queries are investigated in a case study using real sets of trajectories and linked data.

Keywords: Moving objects trajectories, semantic enrichment, linked data, movement analysis, ontology.

adfa, p. 1, 2011.

1 Introduction

Nowadays, large amounts of data about trajectories of moving objects can be gathered by using a variety of devices (e.g., smart phones equipped with GPS or just connected to a GSM network, vehicles equipped with RFID) and information systems (e.g., social Web sites that can detect changing locations of their users). A discrete raw trajectory is a time ordered sequence of spatio-temporal points (x_1, y_1, t_1) ... (x_n, y_n, t_n) $(n > 1)$, where each x_i, y_i is a pair of spatial coordinates and each t_i is an instant, representing sampled positions visited by the moving object. A sample point can be associated with keywords that carry information about the movement in that point. Lots of information can be extracted from such data, with a myriad of applications [4,8]. For instance, information extraction methods [8] can find *episodes* in raw trajectory data, i.e., maximal trajectory segments complying with a predicate [7]. However, these methods usually consider only the spatio-temporal component of trajectories, and do not address the specific content of episodes, whereas episodes can be thought of as segments of trajectories that carry specific semantics (e.g., a stop to take a picture of a monument or to take part in a sports event).

Effective analysis of movement must consider the semantics of the trajectories and the reality in which they occur. Some conceptual models have been proposed for trajectories databases [12,2]. They have introduced relevant ideas for semantic trajectories analysis, such as the concepts of stops and moves [12], later generalized to episodes [8], and dimensions for trajectories analysis [2] (e.g., goal, behavior, transportation means). Ontologies have also been proposed to support reasoning on knowledge bases describing trajectories [13,11]. However, these works do not address the automatic enrichment of trajectories with semantically precise information about specific places (e.g., restaurants, hotels, touristic spots), events (e.g., sport events, cultural events), and other relevant entities of the open dynamic world in which trajectories occur. In this article, a *semantic trajectory* is a sequence of episodes linked to specific concepts and/or instances via ontological relationships that can describe their precise semantics. Such semantic enrichment requires lots of continuously updated information, with well-defined and widely agreed semantics.

This article introduces Baquara, an ontology with an associated architecture and an approach to enable semantic enrichment and analysis of trajectories with vast and growing collections of linked data available in the Web. The proposed ontology has a rich repertoire of constructs to semantically describe trajectories and their relevant episodes with linked data. Baquara plays the role of a conceptual bridge between movement analysis and the semantic Web, by allowing movement data and associated knowledge to be connected and queried together. The proposed approach enables queries that refer to specific entities and classes taken from linked open data sources.

Our approach takes as input raw movement data associated with conventional data that may not have precise semantics (e.g., tag "Rio" may refer to a city, a state, or even a nightclub, among other possibilities). The linked data that help to describe and analyze the trajectories are selected according to the spatio-temporal scope of the movement to be analyzed, and the application domain (traffic analysis, tourism, emergency planning, etc.). Several methods can be used to find connections of

movement data with linked data, including lexical and spatio-temporal matching (e.g., tag "Rio de Janeiro" associated with an episode occurring inside that city). After relevant episodes have been extracted and semantically enriched with linked data, powerful queries can be executed in the resulting knowledge base. Such queries could be from very specific, e.g., "Select the trajectories with at least one episode related to a touristic place called *Corcovado* in *Rio de Janeiro* city, even though the episode happens up to 10 kilometers away from *Corcovado*", or abstract enough to only refer to concepts, e.g., "Select the trajectories that have a stop related to any *sport event*".

The contributions of this article can be summarized as follows:

- An ontological framework for movement data, called Baquara, is proposed, which enables semantic enrichment and analysis of trajectories of moving objects with vast and growing collections of linked data available in the Web.
- Queries on a semantically-annotated movement data collection can be stated in our approach by using Baquara constructs in SPARQL[1] extended with spatial operators [1,6], among other language options.
- The viability of the proposed approach and the expressiveness of the enabled queries are investigated in a case study in which tagged movement data are described, according to the Baquara ontology, using linked data from DBPedia and LinkedGeoData.

The rest of this article is organized as follows. Section 2 discusses related work and contributions. Section 3 describes the proposed Baquara ontology. Section 4 presents a system architecture and a general approach for semantic enrichment of movement data with linked data. Section 5 presents a case study that illustrates the use of the proposed approach. Finally, Section 6 summarizes our contributions and future work.

2 Related Work

A pioneering work on conceptual modeling of spatio-temporal objects is MADS (Modeling Application Data with Spatio-temporal features) [9]. MADS extends the basic ER model with spatio-temporal constructs with its key premise being that spatial and temporal concepts are orthogonal. It uses the object-relationship paradigm, including the features of the ODMG (Object Database Management Group) data model, and provides spatial and temporal attributes, data types and relationships, offering a wide range of conceptual constructs to model the spatio-temporal world. A more recent contribution with focus on conceptual modeling of spatio-temporal objects changing their geographical positions but not their shapes over time comes from Spaccapietra et al. [12]. This model represents semantic trajectories as stops and moves, i.e., trajectory segments in which the object is stationary or moving, respectively. It has been the first attempt to embed semantics in the movement representation, but it lacks generality since other relevant semantic aspects are not explicitly taken into account. An extension of the "Stop-Move" model towards overcoming

[1] http://www.w3.org/TR/2013/REC-sparql11-query-20130321

these limitations comes from the CONSTAnT conceptual model [2], which defines several semantic dimensions for movement analysis (e.g., goal, behavior).

Although the conceptual modeling of trajectories have seen a "convergence" to the "Stop-Move" model [8], ontologies for movement data did not find so far an agreed approach. Due to lack of space we cannot mention here all the proposals for spatio-temporal ontologies and we focus only on the ones most related to our approach. The trajectory ontology proposed by Yan et al. [13] includes three modules: the Geometric Trajectory Ontology describes the spatio-temporal features of a trajectory; the Geographic Ontology describes the geographic objects; and the Domain Application Ontology describes the thematic objects of the application. These ontologies are integrated into a unique ontology that supports conjunctive queries in a traffic application. The proposal of [11] exploits a movement ontology for querying and mining trajectory data enriched with geographic and application information. Here the ontology has been used to infer application-dependent behavior from raw and mined trajectory data. Although Baquara can be seen as an extension of these approaches, it advances one important step further since it introduces rich ontological constructs, algorithms, and the use of linked data in a semi-automatic process to semantically enrich trajectories. These extensions to the widely adopted "Stop-Move" model provide a great improvement in terms of expressive power, as shown in the case study section.

3 The Baquara Ontology

The core of our approach to support semantic enrichment of movement data for trajectory analysis is the so-called Baquara[2] ontology. **Fig. 1** shows the high level concepts (classes) of this ontology, and the major semantic relationships between them. Each labeled rectangle represents a concept. Nesting denotes subsumption (IS-A relationship), i.e., a nested concept (e.g., `Episode`) is a subclass of its enclosing concept (e.g., `SemanticTrace`). The plus sign on the top left corner of a rectangle indicates that the respective concept is further specialized in Baquara. A dashed line between concepts denotes a semantic relationship, such as composition (PART_OF) or a specific relationship (e.g., between an `Event` and a `Place` where it occurs). Lines linking two concepts in opposite directions denote inverse relationships.

The Baquara ontology has been designed to serve as a conceptual framework for describing semantic trajectories in several application domains, ranging from urban transportation to animal ecology. The current version of Baquara has more than 100 classes and more than 200 properties. It includes all the major constructs needed for the description and analysis of trajectories. However, it can also be adapted to specific domains, if necessary, by adding class specializations and object properties.

[2] The word Baquara, from the Tupi-Guarani languages, means knowledgeable, informed.

3.1 Places, Events, and Moving Objects

Baquara ontology uses the W3C's Time[3] ontology as the `Time` conceptualization, and a `Geometry` conceptualization that is compatible with that of the OGC's Geospatial Features[4]. Alternatives for these conceptualizations can also be considered, as standards evolve in the Web of data. These ontologies are used to define instances of `Place` and `Event`. These concepts, defined in the following, can be used to describe the spatio-temporal scope of the movement data, as well as the places and events of interest, for any particular application domain.

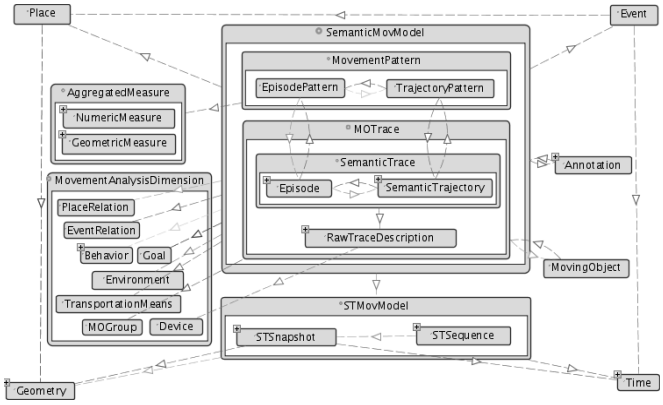


Fig. 1. An overview of the Baquara ontology

A `Place` (Definition 1) is a spatial feature relevant for movement analysis. The place's geometry, as defined by OGC's Geospatial Features, can be simple (point, line, or region) or complex (set of points, lines, or regions). Specializations of `Place` relevant for the tourism domain may, for example, include `City`, `Airport`, `Hotel`, `Restaurant`, and `LandMark`.

Definition 1. A **`Place`** is a tuple having a `Geometry`, and at least a `name`.

An `Event` (Definition 2) represents a circumstance that lasts for a time and that is relevant for movement analysis in an applicaton domain. An event's time can be simple (instant or period) or complex (set of instants or periods), as defined by the W3C's Time ontology. Subclasses of `Event` relevant for tourism may, for example, include `Season` and `SocialEvent`. Such categories can be specialized, e.g., `SocialEvent` can be specialized to `Party`, `Meeting`, etc. Conversely, `Carnival` can be regarded as a subclass of `Party`. Instances of `Event` (and its subclasses in any abstraction level) can be related to specific instances of `Place`, as indicated by the dashed line linking these classes. Thus, an instance of city whose name is "Rio de Janeiro", can be semantically related to events occurring there, named, for example "Pan American Games of Rio 2007" and "First Strike of Rio's Public Buses in 2013".

[3] http://www.w3.org/TR/owl-time
[4] http://www.opengeospatial.org/standards/sfa

On the other hand, events like "Christmas holidays" may not be associated with any particular place, because they occur in many places.

Definition 2. An **`Event`** is a tuple having a `Time`, at least a `name`, and zero or more relations with places.

A `MovingObject` (Definition 3) is anything that moves and that can be distinguished from other moving objects through its `MOid` (moving object identifier). In Baquara, a `MovingObject` can be a `Person`, a `Vehicle`, an `Animal`, or even a `Storm` (that can be represented as a moving region).

Definition 3. A **`MovingObject`** is a tuple having an `MOid`, and collections of `RawTraceDescription` and `SemanticTrajectory` instances.

The key constituents of the above definition, `RawTraceDescription` and, more important, `SemanticTrajectory`, are presented in detail in the following.

3.2 Movement Models and Description Statements

The entities used to represent movement in Baquara are subclasses of `STMovModel` and `SemanticMovModel`. A `STMovModel` (Spatio-temporal Model of Movement) just describes spatio-temporal representations of movement. It can refer to raw traces, processed traces (e.g., resulting of data cleansing and map matching), or even semantic trajectories and trajectory patterns having spatio-temporal counterparts. Such data are efficiently managed by moving object data management systems, such as Hermes [10]. The abstract class `STMovModel` has two concrete subclasses, `STSnapshot` and `STSequence`. A `STSnapshot` refer to a particular spatio-temporal situation that is represented by a `Geometry` and a `Time`. A `STSequence` is a sequence of instances of `STSnapshot` ordered according to their non-overlapping values of time. It is worth to note that the Baquara ontology does not restrict the geometries of moving objects. Baquara allows moving objects to be points, lines, or regions (e.g., storms). However, most case studies and moving object data management systems focus on point geometries.

A `SemanticMovModel` can be a `MOTrace`, or a `MovementPattern`. A `MOTrace` (Moving Object Trace) is an abstract class that describes movement done by a specific `MovingObject`. It can be a `RawTraceDescription` of raw data collected by a `Device`, or a `SemanticTrace` built by post-processing raw data.

A `RawTraceDescription` (Definition 4) is used to semantically describe raw data about movement. For example, to indicate the sensor device used to collect the trace, and other relevant information, such as the spatio-temporal precision, and the sampling rate. The spatio-temporal representation of an episode may differ from that of raw data, due to necessary transformations, including data cleansing, interpolation between samples, and map matching. Therefore, each class, `Episode` and `RawTraceDescription`, has its own `STMovModel`.

Definition 4. A **`RawTraceDescription`** is a tuple with one `STMovModel`, and an indication of the device used to collect that spatio-temporal data.

A SemanticTrace is an abstract class whose concrete subclasses are SemanticTrajectory (Definition 5) and Episode (Definition 6). An Episode is any noticeable happening in a trajectory segment, such as a stop or a move, that can be detected by a trajectory segmentation process; for instance, a stop may be detected according to whether the segment exceeds or not some threshold values on movement predicates (area covered, speed, etc.). A SemanticTrajectory is a time ordered sequence of episodes.

Definition 5. A **SemanticTrajectory** is a tuple having an id and a time ordered sequence of Episode instances.

Definition 6. An **Episode** is a tuple with the RawTraceDescription of the raw data used to build it, and, optionally, one STMovModel.

Differently from a MOTrace, a MovementPattern describes a conceptual movement that is not associated with a specific MovingObject. It can happen or not in some movement database (e.g., a movement starting at home and ending at work). A MovementPattern is an abstract class that generalizes TrajectoryPattern (Definition 7) and EpisodePattern (Definition 8).

Definition 7. A **TrajectoryPattern** is a tuple with a time ordered sequence of EpisodePattern instances, and arbitrary numbers of complying semantic trajectories, and aggregated measures about these trajectories.

Definition 8. An **EpisodePattern** is a tuple with arbitrary numbers of complying episodes, and aggregated measures about these episodes**.**

A MovementPattern (either TrajectoryPattern or EpisodePattern) refers to its compliant SemanticTraces (SemanticTrajectory or Episode collections, respectively), i.e., the ones with compatible traits (e.g., trajectories whose first episode is a stop at a Market). A MovementPattern can also have aggregate measures of its compliant SemanticTrace collections. These measures can be numeric (e.g., the total distance traveled and the time spent by all compliant traces) or geometric (e.g., an aggregate geometry representing the compliant traces).

Finally, description statements (Definition 9) allow the semantic annotation of the previously described entities of the SemanticMovModel with linked data.

Definition 9. A **description statement** for an instance r of a class R is a triple DS(r,P,V) where P refers to a property defined for instances of R, and V is a value that can be a typed literal (string or number), an instance of a class, or a class itself.

Any instance r of a class R subsumed by SemanticMovModel can have an arbitrary number of description statements DS(r,P,V). The general property hasAnnotation can be used for general descriptions. For example, an Episode can be related to an annotation having the value “bus”. However, such an annotation does not have specific semantics; it does not specify, for instance, if the mentioned “bus” plays the role of transport means or it is just something that captured the interest of the moving object in the annotated episode.

Other properties and classes are predefined in Baquara for making description statements with specific semantics. For example, an Episode can alternatively be described with the predefined property usesTransportationMeans pointing to e.g., the class Bus subsumed by class TransportationMeans. The values of specific properties can be taken from SemanticAnalysisDimensions, i.e, hierarchies of semantically related concepts or instances. Examples of such dimensions also include Goal and Behavior, among others, defined in [2]. Baquara defines specific properties for those and other analysis dimensions, such as MOGroup (Moving Objects Group), EventRelation, and PlaceRelation. The former allows for movement analysis according to general traits of moving objects (e.g., their classes, such as Vehicle and Person), without explicitly identifying them. The latter describe the kinds of relations that a MovementModel (e.g., an Episode) can have with an Event or a Place, respectively. For instance, an Episode may happen in a particular Place during an Event. Alternatively, an Episode can happen when the moving object just observes a Place (maybe from distance) or prepares for taking part in an Event. Thus, distinct properties can link a MovementModel to a Place (e.g., the city "Rio de Janeiro") or an Event (e.g., the sports event "Pan American Games of Rio 2007").

4 Semantically Enriching Trajectories with Linked Data

The ontology described in Section 4 is a conceptual framework to support semantic enrichment and analysis of movement data. This section presents the data enrichment process based on that ontology. This process allows arbitrary techniques for information extraction from movement data (e.g., to find trajectory episodes). It exploits ontologies and linked data to delineate movement analysis dimensions with well-defined semantics, enriching the movement analysis possibilities.

4.1 Problem Description

Consider that movement data (MoD) is provided as a relation of the form:

$$\text{MoD}\ (\underline{\text{id}},\ \text{MOid(fk)},\ \text{T},\ \text{S},\ \text{A}_1, \cdots, \text{A}_n) \quad (1)$$

where id is the tuple identifier (primary key), MOid is the moving object identifier (foreign key), T is a validity time (instant or period) when the moving object had the position and shape represented by S, S is a geometry (point, line, region), and $\text{A}_1, \ldots, \text{A}_n$ ($n \geq 0$) are descriptive attributes (e.g., TransportationMeans, Tag, Goal). Note that such a relation can hold relevant episodes instead of raw data.

A MoD relation can carry lots of information. However, this spatio-temporal and descriptive information (when the latter is available) without links to knowledge about the geographic space, relevant events, possible goals, and transportation means, among other issues, may be not enough to explain movement and support intelligent analysis movement. Thus, a semantically rich model of movement, such as Baquara defined in Section 3, must be built from the MoD. The challenge is to build such a

model for large MoD relations, taking into account potentially huge amounts of information and knowledge about the relevant data analysis dimensions, which may vary according to the application domain. This problem can be divided in three tasks:

Task 1. *extract episodes* from raw MoD;
Task 2. *find connections (e.g., lexical and spatial) between* episodes and information about the environment in which the episodes occur; and
Task 3. *devise proper semantic relations* between episodes and environment information to build description statements that can support analyses.

When large amounts of data are involved, automated methods are necessary to solve each one of these tasks. A variety of methods are currently available to solve Task 1, mainly by processing spatio-temporal data about movement [8]. Methods to solve Tasks 2 and 3 are still open issues, particularly if the attributes in the MoD are too generic (e.g. `Tag`) to be directly mapped to specific analysis dimensions via specific properties. On the other hand, we argue that the vast and growing collections of linked data available in the semantic Web can supply the information needed to realize these tasks in many cases. Furthermore, there is a potential for Task 1 to benefit from descriptive attribute values with the well-defined semantics of linked data too.

4.2 Using Linked Data in the Semantic Enrichment Process

The semantic enrichment process proposed in this work takes MoD in the form of (1), along with ontologies and linked data from various sources, to semantically describe movement in accordance to the Baquara ontology described in Section 3. The Baquara ontology provides a conceptual model to represent trajectories, episodes, patterns, and other constructs referring to movement. Its concepts and properties allow the description of movement using linked data.

Fig. 2 illustrates a system architecture to support the proposed semantic enrichment process. This process starts by loading the Baquara ontology in the knowledge management system. Then, linked data with the same spatio-temporal scope as the MoD to be analyzed can be selected from several sources (e.g., by using their SPARQL endpoints or REST APIs). The collected knowledge, initially represented as RDF triples, is used in the semantic enrichment, warehousing, and mining of the MoD.

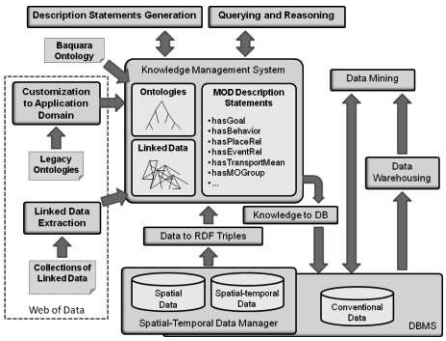


Fig. 2. The proposed architecture for semantic enrichment and analysis of MoD

The description statements generation takes MoD and linked data to derive description statements. This process is interlaced with the extraction of relevant episodes of the raw MoD contained in the spatio-temporal database. MoD can be converted into RDF triples, to process as knowledge when convenient. Conversely, the RDF triples of the resulting knowledge base with description statements based on linked data can be converted into a format that allows direct processing in a conventional or spatio-temporal DBMS, for efficiency purposes.

Algorithm 1 outlines a general method for automatically generating description statements. It takes as input `MovData`, a MoD relation (as described in Section 4.1), the Baquara ontology, and linked data with the same spatio-temporal scope as `MovData` (e.g., whose valid time and spatial extents are inside the minimum bounding box inclosing the `MovData`), for a particular application domain (e.g., tourism). It returns as output `DS`, a collection of description statements. The latter describes movement sample points or episodes, by connecting them to linked data via properties predefined or specialized in the Baquara ontology.

```
INPUT:  MovData(id,MOid,T,S,A1,…,An)  % Movement data
        LD   % Baquara ontology and collected linked data
OUTPUT: DS(id,property,value)   % Description statements
1.  DS = ∅
2.  FOR EACH r IN MovData DO
3.      Matches = FindMatches(r,LD);
4.      FOR EACH v IN Matches DO
5.          CandidateProperties = ChooseProperties(r,v);
6.          FOR EACH p IN CandidateProperties DO
7.              DS = DS + (r.id,p,v);
8. RETURN DS
```

Algorithm 1: General approach for generating description statements

Initially, the set of generated description statements (`DS`) is empty (line 1). Then, for each tuple `r` ∈ `MovData`, the automatic method `FindMatches(r,LD)` finds its matches with linked data in `LD` (lines 2 and 3). A variety of methods can be used for this purpose; such as spatial proximity and lexical similarity between conventional attribute values of the MoD and labels of linked data entities (e.g., the tag "Rio" and the labels of entities in linked data collections). Of course, problems can arise when looking for matches, such as ambiguities. Additional information, such as the classes of the moving objects and linked data resources can help solve these problems. Likewise, such information can help `ChooseProperties(r,v)` find properties to generate description statements for tuple `r` and linked data resource `v` (lines 4 to 7).

The current version of a prototype built to evaluate the proposed approach for MoD enrichment and analysis with linked data employs well-known spatio-temporal and text matching techniques [5,3,14] to find matches of MoD with linked data. It uses either the general `hasAnnotation` or the `isAt` (a place) property to generate description statements. The investigation of more sophisticated methods to generate specific description statements is theme for future work.

5 Case Study

We investigate the viability of our approach to semantically enrich and analyze MoD through a case study. In particular, we take raw trajectories and linked data about Brazil available at the Web, extract episodes, generate description statements, and evaluate some example queries in the resulting knowledge base, as described below.

5.1 Movement Data

The relation `RawFlickTrajsBrazil(id,MOid,Instant,Lat,Long,Tag)` is the MoD used in our case study. It was extracted from CoPhIR[7], a collection of data about images uploaded in Flickr[8]. The spatial coordinates `(Lat,Long)` refer to the positions indicated by the Flickr users when uploading their pictures. The `Instant` of each sample was collected by the devices used to take the pictures. Though the time of the devices may not be set correctly, it can be used to calculate the time intervals between consecutive sample points. We have verified by visual inspection that the coordinates may refer to the user's position when taking the picture or to a pictured object itself.

After extracting tuples with spatio-temporal points inside Brazil, we separated the trajectories as time ordered sequence points visited by each user during each day and, then, we eliminated trajectories with segments having speed higher than 500 km/h (i.e., following a simple trajectory reconstruction technique; evaluating more sophisticated approaches for trajectory reconstruction is beyond the scope of this article). The resulting raw data collection has 2143 trajectories owned by 564 distinct users, and consisting of 14504 sampled positions. These positions are associated with 12443 distinct tags. The total number of tuples in `RawFlickTrajsBrazil` is 117146, i.e., each spatio-temporal sample point is associated to 8.08 tags in average. We have extracted 971 stops from these trajectories. Each of these stops corresponds to a period of at least 30 minutes without moving more than 500 meters. These stops are associated to 6278 distinct tags, in a total of 45768 (stop,tag) pairs, i.e., around 47 different tag values associated to each stop, in average. After cleaning irrelevant tag values, we used Triplify[9] to transform the MoD from relations into RDF triples, in accordance to the conceptual model defined by the Baquara ontology.

5.2 Semantic Enrichment with Linked Data

We have accessed triple sets available on the Web pages, SPARQL endpoints, and REST endpoints to extract subsets of linked data from DBPedia[10] (a knowledge base built from Wikipedia[11]), LinkedGeoData[12] (a large collection of geographic entities'

[7] http://cophir.isti.cnr.it
[8] http://www.flickr.com
[9] http://triplify.org
[10] http://dbpedia.org
[11] http://www.wikipedia.org

descriptions taken from OpenStreetMap[13]), and GeoCodes[14] (another knowledge base about geographic entities). The extracted subsets are compatible with the spatio-temporal scope of the raw MoD considered for analyses, i.e., referring to Brazil and/or the time period between 2007 and 2008. The extracted linked data was stored in the triple repository Virtuoso[15], which supports spatial data extensions and GeoSPARQL[16] [1] queries with spatial operators.

Then we have investigated matches between labels of a selected collection of linked data and the tags associated with the sample points and episodes of our trajectory data collection. We have used just approximate string matching methods and geographic coordinates to help disambiguate in some cases. Most of the matches found are tag values from CoPhIR trajectories matching labels of entities classified as different kinds of places (`PopulatedPlace`, `NaturalPlace`, `Ammenity`, etc.) in the collected linked data. We have also found a smaller number of matches with specific kinds of events (`SportsEvent`, `SoccerTournament`, `FootballMatch`, etc.) and organizations (`SportsTeam`, `SoccerClub`, `SambaSchool`, etc.). This semantic enrichment process enabled us to generate some description statements to answer queries such as the ones presented in the following.

5.3 Analytical Queries

There follows some examples of GeoSPARQL queries that can be executed in the knowledge base resulting from the semantic enrichment of our collection of Flickr trajectory data. They use the `bq` namespace to refer to the Baquara ontology.

Query 1 (single episode query): Select trajectories with at least one episode that mentions and occurs up to 10 km from `Corcovado` (the mountain of Rio de Janeiro in whose top stands the statue of Christ the Redeemer).

```
SELECT ?trajectory WHERE {
 ?trajectory a bq:SemanticTrajectory;
  bq:hasEpisode ?episode.
 ?episode bq:hasAnnotation ?a. ?a bq:hasValue ?v.
 ?v a <http://dbpedia.org/ontology/Mountain>;
  rdfs:label "Corcovado"@pt; geo:geometry ?cGeo.
 ?rio a <http://dbpedia.org/ontology/City>;
  rdfs:label "Rio de Janeiro"@pt; geo:geometry ?rioGeo.
 FILTER(bif:st_intersects (?cGeo,?rioGeo,20) &&
        bif:st_intersects (?eGeo,?cGeo,10)) }
```

---

[12] http://linkedgeodata.org
[13] http://www.openstreetmap.org
[14] http://www.geocode.com
[15] http://virtuoso.openlinksw.com
[16] http://www.opengeospatial.org/standards/geosparql

Query 2 (multiple episodes query): Select trajectories with a stop in an `Amenity`, a stop mentioning a `SportsEvent`, and a stop lexically related to `"Beach"`.

```
SELECT ?trajectory WHERE {
 ?trajectory a bq:SemanticTrajectory;
   bq:hasEpisode ?s1, ?s2, ?s3.
 ?s1 a bq:Stop; bq:occursIn ?p1.
 ?s2 a bq:Stop; bq:hasAnnotation ?a2.
 ?s3 a bq:Stop; bq:hasAnnotation ?a3.
 ?a2 bq:hasValue ?v2. ?a3 bq:hasValue ?v3.
 ?p1 a <http://linkedgeodata.org/ontology/Amenity>.
 ?v2 a <http://dbpedia.org/resource/SportsEvent>.
 FILTER(regex(?v3,"Beach")) }
```

Commenting on the above queries, the raw trajectories extracted from Flickr have sample points annotated with the tag `"Corcovado"` that are far away from that mountain, probably because it can be seen and pictured from many positions in Rio. The FILTER clause in Query 1 ensures that the considered mountain labeled with `"Corcovado"@pt` is the one inside the city called `Rio de Janeiro` and that the stop mentioning that mountain occurs up to 10 km from it. Further, in Query 2, the bidding of the description statements of the stops `s1` and `s2` with values from specific classes from LinkedGeoData and DBpedia, respectively, are more precise than just matching strings, as it is done in the filter condition with `v3`.

6 Conclusions and Future Work

Vast collections of linked data about real world entities and events have been fed and continuously updated on the Web. However, their potential to leverage movement understanding has not been exploited yet. This article gives the following contributions towards using linked data to help movement analyses: (i) an ontology, called Baquara, for semantic trajectories enrichment with linked data; (ii) an architecture to narrow the gap between trajectory mining and the semantic Web; (iii) an automated method to derive semantic annotations from movement data with free annotations; (iv) examples of analytical queries enabled by this proposal through a case study with real data available at the Web. Though the queries presented in this article run on triple stores, the latter can be used just as means to handle knowledge. After semantic enrichment, the resulting knowledge can be converted into conventional and spatio-temporal databases, for more efficient analysis and mining.

In our future work we plan to: (i) evaluate the proposed approach with spatio-temporal and linked data from distinct domains; (ii) develop efficient methods to derive precise description statements from different data collections; and (iii) investigate the use of linked data for trajectories warehousing and trajectories mining.

Acknowledgments. This work has been supported by the European Union's IRSES-SEEK (grant 295179) and FP7-DATASIM (grant 270833) projects, and Brazil's

CNPq (grant 478634/2011-0) project. Nikos Pelekis' research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thales.

References

1. Battle, M.; Kolas, D. Enabling the Geospacial Semantic Web with Parliament and GeoSPARQL. Semantic Web Journal. 3(4) pp. 355–370 (2012)
2. Bogorny, V., Renso, C., Aquino, A., Siqueira, F. L., and Alvares, L. O. CONSTAnT - A Conceptual Data Model for Semantic Trajectories of Moving Objects. Transactions in GIS, 8(2) (2013)
3. Cohen, W.W., Ravikumar, P. D., Fienberg, S. E. A Comparison of String Distance Metrics for Name-Matching Tasks. In IIWeb: 73-78 (2003)
4. Giannotti, F., Nanni, M., Pedreschi, D., Pinelli, F., Renso, C., Rinzivillo, S., and Trasarti, R. Unveiling the complexity of human movement by querying and mining massive trajectory data. The VLDB Journal, 20(5):695-719 (2011)
5. Navarro, G. A guided tour to approximate string matching. *ACM* Computing Surveys 33(1): 31-88 (2001)
6. Kyzirakos, K., Karpathiotakis, M. Koubarakis, M. Strabon: A Semantic Geospatial DBMS. In 11th Int. Semantic Web Conf. (ISWC 2012), Boston, USA, pp. 11–15 (2012)
7. Mountain, D., Raper, J.F. Modelling human spatio-temporal behaviour: a challenge for location-based services, In 6th Int. Conf. on GeoComputation, Brisbane, Australia, 24–26, (2001)
8. Parent, C., Spaccapietra, S., Renso, C., Andrienko, G., Andrienko, N., Bogorny, V., Damiani, M. L., Gkoulalas-divanis, A., Macedo, J., Pelekis, N., Theodoridis, Y., and Yan, Z. Semantic trajectories modeling and analysis. ACM Computing Surveys, 45 (2013)
9. Parent, C., Spaccapietra, S., Zimányi, E. Conceptual Modeling for Traditional and Spatio-Temporal Applications: The MADS Approach, Springer (2006)
10. Pelekis, N., Frentzos, E., Giatrakos, N., Theodoridis, Y. HERMES: aggregative LBS via a trajectory DB engine. In SIGMOD Conf., pp.: 1255-1258 (2008)
11. Renso, C., Baglioni, M., de Macedo, J.A.F., Trasarti, R., Wachowicz, M. How you move reveals who you are: understanding human behavior by analyzing trajectory data. Knowledge and Information System Journal (KAIS), pp. 1-32, June (2012)
12. Spaccapietra, S., Parent, C., Damiani, M. L., de Macedo, J. A., Porto, F., and Vangenot, C. A conceptual view on trajectories. Data and Knowledge Engineering, 65(1):126–146 (2008)
13. Yan, Z., Macedo, J., Parent, C., Spaccapietra, S. Trajectory Ontologies and Queries. Transactions in GIS, 12(Suppl. 1): 75–91 (2008)
14. Yao, B., Li, F., Hadjieleftheriou, M., Hou, K. Approximate string search in spatial databases. In Int. IEEE Conf. on Data Engineering (ICDE), 545-556 (2010)

GRAMI: Frequent Subgraph and Pattern Mining in a Single Large Graph

Mohammed Elseidy
Ecole Polytechnique Fédérale de Lausanne
mohammed.elseidy@epfl.ch

Ehab Abdelhamid
King Abdullah University of Science and Technology
ehab.abdelhamid@kaust.edu.sa

Spiros Skiadopoulos *
University of Peloponnese
spiros@uop.gr

Panos Kalnis
King Abdullah University of Science and Technology
panos.kalnis@kaust.edu.sa

ABSTRACT

Mining frequent subgraphs is an important operation on graphs; it is defined as finding all subgraphs that appear frequently in a database according to a given frequency threshold. Most existing work assumes a database of many small graphs, but modern applications, such as social networks, citation graphs, or protein-protein interactions in bioinformatics, are modeled as a single large graph. In this paper we present GRAMI, a novel framework for frequent subgraph mining in a single large graph. GRAMI undertakes a novel approach that only finds the *minimal* set of instances to satisfy the frequency threshold and avoids the costly enumeration of *all* instances required by previous approaches. We accompany our approach with a heuristic and optimizations that significantly improve performance. Additionally, we present an extension of GRAMI that mines frequent patterns. Compared to subgraphs, patterns offer a more powerful version of matching that captures transitive interactions between graph nodes (like friend of a friend) which are very common in modern applications. Finally, we present CGRAMI, a version supporting structural and semantic constraints, and AGRAMI, an approximate version producing results with no false positives. Our experiments on real data demonstrate that our framework is up to 2 orders of magnitude faster and discovers more interesting patterns than existing approaches.

1. INTRODUCTION

Graphs model complex relationships among objects in a variety of applications such as chemical, bioinformatics, computer vision, social networks, text retrieval and web analysis. Mining frequent subgraphs is a central and well studied problem in graphs, and plays a critical role in many data mining tasks that include graph classification [9], modeling of user profiles [11], graph clustering [15], database design [10] and index selection [31]. The goal of frequent subgraph mining is to find subgraphs whose appearances exceed a user defined threshold. This is useful in several real life applications. Consider for example protein-protein interaction (PPI) networks [5]. These networks are graphs where nodes represent proteins (and are labeled with their functionality) and edges represent interactions between these proteins. Such graphs are constantly updated to include new proteins and their interactions. A critical task for biologists is to predict the functionality (and add the corresponding label) of a new protein without experimental testing. The above task may be accurately preformed by mining frequent subgraphs with similar interactions to the new protein [5].

Consider the collaboration graph G of Fig. 1 and a user interested to mine important collaborations among authors. Typically, in such graphs, frequent subgraphs are most likely to show collaborations among authors having the same field of work (i.e., collaborations among DB researchers). In order to reveal more interesting subgraphs, the user would progressively reduce the frequency threshold until subgraphs showing interdisciplinary collaborations are discovered (i.e., among AI, DB and IR researchers). Lowering the frequency threshold increases the number of qualified intermediate results and intensifies the already expensive computations of the mining process. For example, a state-of-the-art method for frequent subgraph mining crashes after a day consuming 192GB for an input graph of 100K nodes and 1M edges. Therefore, the development of efficient frequent subgraph mining algorithms that support large graphs and low frequency thresholds is very crucial.

Existing literature considers two settings: transactional and single graph. The *transactional* case assumes a database of many, relatively small graphs, where each graph represents a transaction [18, 29]. A subgraph is frequent if it exists in at least τ transactions, where τ is a user-defined threshold. In this paper, the focus is on the *single-graph* setting that considers one large graph [17, 19, 20]. For this setting, a subgraph is frequent if it has at least τ appearances in the graph. Such a context is required in many modern applications, including social and PPI networks. The *single-graph* setting is a generalization of the transactional one, since a set of small graphs can be considered as connected components within a single large graph. Detecting frequent subgraphs in a single graph is more complicated because multiple instances of identical subgraphs may overlap. Moreover, it is more computationally demanding because complexity is exponential in the graph size.

The most straightforward method to evaluate frequency of a subgraph S in a graph G is to look for *isomorphisms* of S in G [12, 16, 19, 20]. Isomorphisms are exact matches of S in G that pair nodes, edges and labels. For example, in the collaboration graph G of Fig. 1, subgraph S_1 has three isomorphisms.

A typical method to mine frequent subgraphs in a single graph, is a *grow-and-store* method that proceeds with the following steps:

1. Find all nodes that appear at least τ times and store all of their appearances.
2. Extend the stored appearances to construct larger potential frequent subgraphs, evaluate their frequency, and store all the appearances of the new frequent subgraphs.

* Supported by EU/Greece Research Funding Program: Thales

Proceedings of the VLDB Endowment, Vol. 7, No. 7

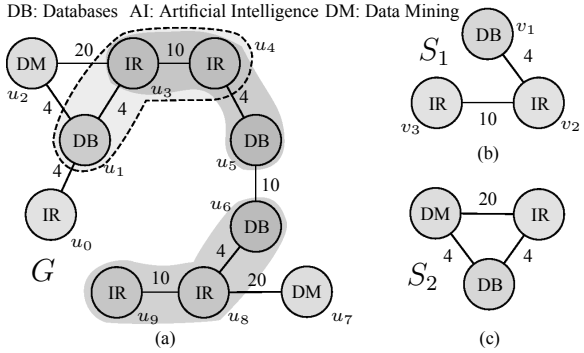


Figure 1: (a) A collaboration graph G; nodes correspond to authors (labeled with their field of work) and edges represent co-authorship (labeled with number of co-authored papers). (b) and (c) Subgraphs S_1 and S_2.

3. Repeat Step 2 until no more frequent subgraphs can be found.

Existing approaches such as SIGRAM [20] use variations of this grow-and-store method. These approaches take advantage of the stored appearances to evaluate the frequency of a subgraph. The main bottleneck of such algorithms is the creation and storage of all appearances of each subgraph. The number of such appearances depends on the size and the properties of the graph and the subgraph; it can be prohibitively large to compute and store, rendering grow-and-store solutions infeasible in practice.

In this work, we propose GRAMI (GRA*ph* MI*ning*); a novel framework that addresses the frequent subgraph mining problem. GRAMI undertakes a novel approach differentiating it from grow-and-store methods. First, it stores only the templates of frequent subgraphs, but not their appearances on the graph. This eliminates the limitations of the grow-and-store methods and allows GRAMI to mine large graphs and support low frequency thresholds. Also, it employs a novel method to evaluate the frequency of a subgraph. More specifically, GRAMI models the frequency evaluation as a constraint satisfaction problem (*CSP*). At each iteration, GRAMI solves the CSP until it finds the *minimal* set of appearances that are enough to evaluate subgraph frequency, and it ignores all remaining appearances. The process is repeated by extending the subgraphs until no more frequent subgraphs can be found.

Solving the CSP can still take exponential time in the worst case. In order to support large graphs in real-life applications, GRAMI employs a heuristic search and a series of optimizations that significantly improve performance. More specifically, GRAMI introduces novel optimizations that *(a)* prune large portions of the search space, *(b)* prioritize fast and postpone slow searches and *(c)* take advantage of special graph types and structures. By avoiding the exhaustive enumeration of appearances and using the proposed optimizations, GRAMI supports larger graphs and smaller frequency thresholds than existing approaches. For example, to compute the frequent patterns of the 100K nodes/1M edges graph that the state-of-the-art grow-and-store method crashed after a day, GRAMI needs only 16 minutes.

Additionally, we propose three extensions to the original GRAMI framework. The first one considers graphs such as social or research networks, that may contain incomplete information and transitive relationships. In such cases *indirect* relationships (like a friend of a friend) reveal neighborhood connectivity and proximity information. To explore these relationships, *patterns* were introduced [4, 17, 34]. Patterns establish a more powerful definition of matching, than subgraphs, that captures indirect connections by replacing edges with paths. To mine frequent patterns, we have appropriately extended GRAMI. For instance in Fig. 1, GRAMI may also consider $u_5 \overset{\cdots}{-} u_8 \overset{10}{-} u_9$ to be a match of S_1 since u_5 (labeled DB) is indirectly connected to u_8 (labeled IR). The second extension, CGRAMI, allows the user to define a set of *constraints*, both structural (e.g., the subgraph is allowed to have up to α edges) and semantic (e.g., a particular label cannot occur more than α times in the subgraph). The constraints are used to prune undesirable matches and limit the search space. The final extension, AGRAMI, is an *approximate* version, which approximates subgraph frequencies. The approximation method may miss some frequent subgraphs (i.e., has false negatives), but the returned results are *not* approximate (i.e., does not have false positives).

Noteworthily, GRAMI and its extensions support directed and undirected graphs and may be applied to both single and multiple labels (or weights) per node and edge.

In summary, our main contributions are:

- We propose GRAMI, a novel framework to mine frequent subgraphs in a large single graph. GRAMI is based on a novel idea that refrains from computing and storing large intermediate results (appearances of subgraphs). A key part of the underlying idea is to evaluate the frequency of subgraphs using CSP.
- We offer a heuristic search with novel optimizations that significantly improve GRAMI's performance by pruning the search space, postponing searches, and exploring special graph types.
- We develop a variation of GRAMI that is able to mine frequent patterns, a more powerful version of matching that is required in several modern applications.
- We present CGRAMI, a version that supports structural and semantic constraints, and AGRAMI, an approximate version which produces results with no false positives.
- We experimentally evaluate the performance of GRAMI and demonstrate that it is up to 2 orders of magnitude faster than existing methods in large real-life graphs.

The rest of the paper is organized as follows. Section 2 formalizes the problem. Section 3 presents GRAMI and its optimizations. Section 4 discusses the extensions of GRAMI. Section 5 presents the experimental evaluation. Section 6 surveys related work, and Section 7 concludes.

2. PRELIMINARIES

A *graph* $G = (V, E, L)$ consists of a set of nodes V, a set of edges E and a labeling function L that assigns labels to nodes and edges. A graph $S = (V_S, E_S, L_S)$ is a *subgraph* of a graph $G = (V, E, L)$ iff $V_S \subseteq V$, $E_S \subseteq E$ and $L_S(v) = L(v)$ for all $v \in V_S \cup E_S$. Fig. 1a illustrates an example of a collaboration graph. Node labels represent author's field of work (e.g., Databases) and edge labels represent the number of co-authored papers. To simplify presentation, all examples illustrate undirected graphs with a single label for each node. However, the proposed methods also support directed graphs and multiple labels per node/edge.

Definition 1 *Let $S = (V_S, E_S, L_S)$ be a subgraph of a graph $G = (V, E, L)$. A* subgraph isomorphism *of S to G is an injective function $f : V_S \rightarrow V$ satisfying* (a) *$L_S(v) = L(f(v))$ for all nodes $v \in V_S$, and* (b) *$(f(u), f(v)) \in E$ and $L_S(u, v) = L(f(u), f(v))$ for all edges $(u, v) \in E_S$.*

Intuitively, a subgraph isomorphism is a mapping from V_S to V such that each edge in E is mapped to a single edge in E_S and vice versa. This mapping preserves the labels on the nodes and edges. For example in Fig. 1, subgraph S_1 ($v_1 \overset{4}{-} v_2 \overset{10}{-} v_3$) has three isomorphisms with respect to graph G, namely $u_1 \overset{4}{-} u_3 \overset{10}{-} u_4$, $u_5 \overset{4}{-} u_4 \overset{10}{-} u_3$ and $u_6 \overset{4}{-} u_8 \overset{10}{-} u_9$.

The most intuitive way to measure the support of a subgraph in a graph is to count its isomorphisms. Unfortunately, such a metric is not *anti-monotone* since there are cases where a subgraph appears less times than its extension. For instance, in Fig. 1a the single node subgraph DB appears 3 times while its extension DB $\frac{4}{}$ IR appears 4 times. Having an anti-monotone support metric is of crucial importance since it allows the development of methods that effectively prune the search space; without an anti-monotone metric exhaustive search is unavoidable [12, 20]. The literature defines several anti-monotone support metrics such as *minimum image based* (*MNI*) [2], *harmful overlap* (*HO*) [12], and *maximum independent sets* (*MIS*) [20]. These metrics differ in the degree of overlap they allow between subgraph isomorphisms, and the complexity of their computation. In this paper, we adopt the *MNI* [2] metric mainly because it: *(a)* is the only metric that can be efficiently computed; the computation of *MIS* and *HO* are NP-complete [12, 20] and *(b)* provides a superset of the results of the alternative metrics; if we are interested in the *MIS* or *HO* metric we may pay their expensive computational cost and exclude the unqualified subgraphs [12]. Formally, the *MNI* metric is defined as follows [2].

Definition 2 *Let $f_1, \ldots, f_m$ be the set of isomorphisms of a subgraph $S(V_S, E_S, L_S)$ in a graph G. Also let $F(v) = \{f_1(v), \ldots, f_m(v)\}$ be the set that contains the (distinct) nodes in G whose functions $f_1, \ldots, f_m$ map a node $v \in V_S$. The* minimum image based support *(MNI) of S in G, denoted by $s_G(S)$, is defined as $s_G(S) = \min\{\, t \mid t = |F(v)| \text{ for all } v \in V_S\}$.*

For instance, for the subgraph S_1 of Fig. 1b and the graph G of Fig. 1a, we have $F(v_1) = \{u_1, u_5, u_6\}$, $F(v_2) = \{u_3, u_4, u_8\}$ and $F(v_3) = \{u_3, u_4, u_9\}$, thus $s_G(S_1) = 3$. To compare, the respective *MIS* metric is 2 since isomorphisms $u_1 \frac{4}{} u_3 \frac{10}{} u_4$ and $u_5 \frac{4}{} u_4 \frac{10}{} u_3$ overlap and the *MIS* metric regards them as one.

The frequent subgraph mining problem is defined as:

Problem 1 *Given a graph G and a minimum support threshold τ, the* frequent subgraph isomorphism mining problem *is defined as finding all subgraphs S in G such that $s_G(S) \geq \tau$.*

Problem 1 does not consider finding the actual number of appearances (i.e., frequency) provided that it is greater than τ. This is very useful in several applications [6, 20], but there are others that demand the exact number of appearances (like graph indexing [31]). Also note, that Problem 1 is computationally expensive since it relies on the NP-hard subgraph isomorphism problem [13].

Definition 1 enforces matching on both node and edge labels. For instance in Fig. 1, subgraph S_2 has only one isomorphism (formed by nodes u_1, u_2 and u_3). Recent research argues that this matching is rather restrictive, and relaxes it by allowing indirect relationships and differences between the edges of the graph and the subgraph [4, 17, 34]. Such frameworks may also consider subgraph $u_6 \frac{4}{} u_8 \frac{20}{} u_7$ to be a match of S_2 since DM and DB are indirectly connected. We refer to this match as a pattern. For mining frequent patterns, we adopt the pattern matching definition as outlined in [34]. Specifically, we employ a distance metric to measure the distance between two nodes. To this end, we may use any *metric* function, i.e., a function that satisfies the triangle inequality [34]. Typically, the distance function is computed based on the edge labels (or weights) but it may also be defined on other graph properties (e.g., the number of hops between two nodes).

For graph G of Fig. 1, we may use a distance function $\Delta_h(u, v)$ defined as the number of hops in the shortest path that connects u and v. For instance, $\Delta_h(u_0, u_3) = 2$. Alternatively, we may use $\Delta_p(u, v)$ defined as the minimum sum of the inverse of edge weights among the paths that connect u and v. For an example,

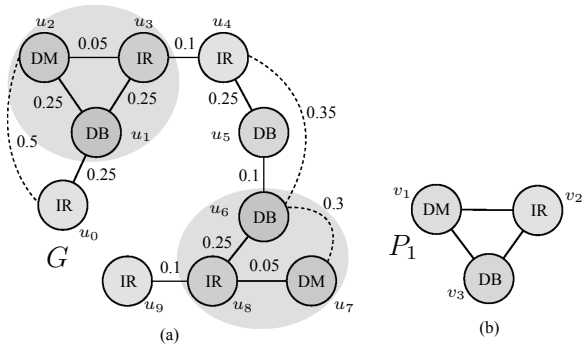


Figure 2: (a) The distance Δ_p for the graph G of Fig. 1. (b) A pattern P_1.

$\Delta_p(u_6, u_7) = 1/4 + 1/20 = 0.3$. Intuitively, a shorter distance denotes a stronger collaboration. Fig. 2 illustrates the values of Δ_p for the graph G of Fig. 1. Solid lines correspond to the original edges of the graph, while dotted lines illustrate some additional transitions (for figure clarity, we do not show all transitions).

Definition 3 *A graph $P = (V_P, E_P, L_P)$ is a* pattern *of a graph $G(V, E, L)$ iff $V_P \subseteq V$, $L_P(v) = L(v)$ for all $v \in V_P$ and $L_P(e) = \emptyset$ for all $e \in E_P$.*

In other words, a pattern is analogous to a subgraph but without considering edge labels. For instance, a pattern P_1 of the graph G is presented in Fig. 2b.

Definition 4 *Let $P = (V_P, E_P, L_P)$ be a pattern of a graph $G = (V, E, L)$, Δ be a distance metric function, and δ be a user-defined distance threshold. A* pattern embedding *of P to G is an injective function $\phi : V_P \rightarrow V$ satisfying* (a) *$L_P(v) = L(\phi(v))$ for all nodes $v \in V_P$ and* (b) *$\Delta(\phi(u), \phi(v)) \leq \delta$ for all edges $(u, v) \in E_P$.*

The minimum image based support for a pattern, denoted by $\sigma_G(P)$, can be computed as in Definition 2 by replacing the isomorphisms $f_1, \ldots, f_m$ with the pattern embeddings $\phi_1, \ldots, \phi_\mu$. For example consider Fig. 2; setting a threshold $\delta = 0.3$, we have $\sigma_G(P_1) = 2$. The corresponding embeddings are illustrated by the gray areas. Note that there are other possible matches to P_1 but only the indicated two satisfy the constraint $\Delta(\phi(u), \phi(v)) \leq \delta$.

Problem 2 *Given a graph G, a distance function Δ, a distance threshold δ, and a minimum support threshold τ, the* frequent pattern embedding mining problem *is defined as finding all patterns P of G such that $\sigma_G(P) \geq \tau$.*

3. THE GRAMI APPROACH

GRAMI proposed a novel technique that addresses the frequent subgraph mining problem without exhaustively enumerating all isomorphisms in the graph. To this end, GRAMI models the underlying problem as a *constraint satisfaction problem* (Section 3.1). Following, Section 3.2 applies the model to solve the frequent subgraph problem. Section 3.3 proposes several optimizations to enhance performance. The frequent pattern mining problem together with other interesting extensions are discussed in Section 4.

3.1 The CSP Model

A *constraint satisfaction problem (CSP)* is represented as a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where *(a)* $\mathcal{X}$ is an ordered set of variables, *(b)* $\mathcal{D}$ is a set of domains corresponding to variables $\mathcal{X}$, and *(c)* $\mathcal{C}$ is a set of constraints between the variables in $\mathcal{X}$. A *solution* for the CSP is an assignment to the variables in $\mathcal{X}$, such that all constraints in $\mathcal{C}$ are satisfied. The subgraph isomorphism problem (Definition 1) can be mapped to a CSP as follows.

Definition 5 *Let $S(V_S, E_S, L_S)$ be a subgraph of a graph $G(V, E, L)$. The* subgraph S to graph G CSP*, is a CSP $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where:*

1. *$\mathcal{X}$ contains a variable x_v for every node $v \in V_S$.*
2. *$\mathcal{D}$ is the set of domains for each variable $x_v \in \mathcal{X}$. Each domain is a subset of V.*
3. *Set $\mathcal{C}$ contains the following constraints:*
   a) $x_v \neq x_{v'}$, for all distinct variables $x_v, x_{v'} \in \mathcal{X}$.
   b) $L(x_v) = L_S(v)$, for every variable $x_v \in \mathcal{X}$.
   c) $L(x_v, x_{v'}) = L_S(v, v')$, for all $x_v, x_{v'} \in \mathcal{X}$ such that $(v, v') \in E_S$.

To simplify notation, whenever it is clear from the context, we use v to refer to a node of the subgraph and to the corresponding variable x_v of the CSP as we do in the following example.

Example 1 *Consider Fig. 1. The subgraph S_1 to graph G CSP is defined as:*

$$\left(\begin{array}{c} (v_1, v_2, v_3), \{\{u_0, \ldots, u_9\}, \ \ldots, \ \{u_0, \ldots, u_9\}\}, \\ \{v_1 \neq v_2 \neq v_3, \ L(v_1) = \mathrm{DB}, \ L(v_2) = L(v_3) = \mathrm{IR}, \\ L(v_1, v_2) = 4, \ L(v_2, v_3) = 10\} \end{array} \right)$$

The following proposition relates the subgraph to a graph CSP with the subgraph isomorphism f (Definition 1).

Proposition 1 *A solution of the subgraph S to graph G CSP corresponds to a subgraph isomorphism of S to G.*

Intuitively, a solution assigns a different node of G to each node of S, such that the labels of the corresponding nodes and edges match. For instance, a solution to the CSP of Example 1 is the assignment $(v_1, v_2, v_3) = (u_1, u_3, u_4)$.

Definition 6 *An assignment of a node u to a variable v is* valid *if and only if there exists a solution that assigns u to v. Note that each valid assignment corresponds to an isomorphism.*

In Example 1, $v_2 = u_3$ is a valid assignment; $v_2 = u_0$ is invalid.

Proposition 2 *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be the subgraph S to graph G CSP. The MNI support of S in G satisfies τ, i.e., $s_G(S) \geq \tau$, iff every variable in $\mathcal{X}$ has at least τ distinct valid assignments (i.e., isomorphisms of S in G).*

Proposition 2 is a key part of this work since it provides a method to determine if a subgraph S is frequent in G. To this end, we may consider the S to G CSP and check the number of valid assignments of every variable. If for every variable there exists τ or more valid assignments, then $s_G(S) \geq \tau$ and S is considered frequent. Continuing Example 1, we have $s_G(S_1) \geq 3$ since all domains contain at least 3 valid assignments (more specifically, the domains of variables v_1, v_2 and v_3 are $\{u_1, u_5, u_6\}$, $\{u_3, u_4, u_8\}$ and $\{u_4, u_3, u_9\}$ respectively).

3.2 Frequent Subgraph Mining

We now apply the CSP model presented in Section 3.1 to solve the frequent subgraph mining problem (Problem 1). We start by presenting Algorithms FREQUENTSUBGRAPHMINING and SUBGRAPHEXTENSION that are used in many related methods to generate candidate subgraphs [29, 20] and are illustrated for completeness. Then, we consider methods to measure the number of appearances (frequency) of these subgraphs. Algorithm ISFREQUENTCSP shows how we may address frequency evaluation without computing and storing all intermediate results. Algorithm ISFREQUENTHEURISTIC offers a heuristic approach and Algorithm ISFREQUENT supplements it with optimizations that highly improve performance. The frequent pattern embedding mining problem (Problem 2) is discussed in Section 4.

Algorithm: FREQUENTSUBGRAPHMINING

Input: A graph G and the frequency threshold τ
Output: All subgraphs S of G such that $s_G(S) \geq \tau$

```
1 result ← ∅
2 Let fEdges be the set of all frequent edges of G
3 foreach e ∈ fEdges do
4     result ← result ∪ SUBGRAPHEXTENSION(e, G, τ, fEdges)
5     Remove e from G and fEdges
6 return result
```

Algorithm: SUBGRAPHEXTENSION

Input: A subgraph S of a graph data G, the frequency threshold τ and the set of frequent edges $fEdges$ of G
Output: All frequent subgraphs of G that extend S

```
1 result ← S, candidateSet ← ∅
2 foreach edge e in fEdges and node u of S do
3     if e can be used to extend u then
4         Let ext be the extension of S with e
5         if ext is not already generated then
              candidateSet ← candidateSet ∪ ext
6 foreach c ∈ candidateSet do
7     if s_G(c) ≥ τ then
8         result ← result ∪ SUBGRAPHEXTENSION(c, G, τ, fEdges)
9 return result
```

FREQUENTSUBGRAPHMINING starts by identifying set $fEdges$ that contains all frequent edges (i.e., with support greater or equal to τ) in the graph. Based on the anti-monotone property, only these edges may participate in frequent subgraphs. For each frequent edge, SUBGRAPHEXTENSION is executed. This algorithm takes as input a subgraph S and tries to extend it with the frequent edges of $fEdges$ (Lines 2-5). All applicable extensions that have not been previously considered are stored in $candidateSet$. To exclude already generated extensions (Line 5) we adopt the *DFScode* canonical form as in GSPAN [29]. Then, SUBGRAPHEXTENSION (Lines 6-8) eliminates the members of $candidateSet$ that do not satisfy the support threshold τ since according to the anti-monotone property, their extensions are also infrequent. Finally, SUBGRAPHEXTENSION is recursively executed (Line 8) to further extend the frequent subgraphs.

According to Proposition 2, a subgraph S is frequent in G (i.e., $s_G(S) \geq \tau$) if there exist at least τ nodes in each domain $D_1, \ldots, D_n$ that are valid variable assignments (i.e., are part of a solution) for the corresponding variables $v_1, \ldots, v_n$. To evaluate frequency, we may use ISFREQUENTCSP that returns $true$ iff S is a frequent subgraph of G. Initially, ISFREQUENTCSP enforces *node and arc consistency* [22]. Node consistency excludes unqualified nodes from the domains (like nodes with different labels or with lower degree) and arc consistency ensures the consistency between the assignments of two variables. Specifically, for every constraint $C(v, v')$, arc consistency ensures that for every node in the domain of v there exists a node in the domain of v' satisfying $C(v, v')$. If, after node and arc consistency enforcement, the size of a domain is smaller than τ the algorithm returns $false$ (Line 3). Following, ISFREQUENTCSP considers every solution $\mathcal{S}ol$ and marks the nodes assigned to variables to the corresponding domains (Line 5). If all domains have at least τ marked nodes then (according to

Algorithm: ISFREQUENTCSP

Input: Graphs S and G and the frequency threshold τ
Output: $true$ if S is a frequent subgraph of G, $false$ otherwise

```
1 Consider the subgraph S to graph G CSP
2 Apply node and arc consistency
3 if the size of any domain is less than τ then return false
4 foreach solution Sol of the S to graph G CSP do
5     Mark all nodes of Sol in the corresponding domains
6     if all domains have at least τ marked nodes then return true
7 return false // Domain is exhausted
```

Algorithm: IsFrequentHeuristic

Input: Graphs S and G and the frequency threshold τ
Output: $true$ if S is a frequent subgraph of G, $false$ otherwise

1 Consider the subgraph S to graph G CSP
2 Apply node and arc consistency
3 **foreach** *variable* v *with domain* D **do**
4 $count \leftarrow 0$
5 Apply arc consistency
6 **if** *the size of any domain is less than* τ **then return** $false$
7 **foreach** *element* u *of* D **do**
8 **if** u *is already marked* **then** $count$++
9 **else if** *a solution* $\mathcal{S}ol$ *that assigns* u *to* v *exists* **then**
10 Mark all values of $\mathcal{S}ol$ in the corresponding domains
11 $count$++
12 **else** Remove u from the domain D
13 **if** $count = \tau$ **then** Move to the next v variable (Line 3)
14 **return** $false$ `// Domain is exhausted and` $count < \tau$
15 **return** $true$

Proposition 2) S is frequent in G. Otherwise, IsFrequentCsp continues with the following solution.

Complexity. Let N and n be the number of nodes of graph G and subgraph S respectively. The complexity of FrequentSubgraphMining is determined by the complexity of SubgraphExtension and IsFrequentCsp. The former computes all subgraphs of G, which takes $\mathcal{O}(2^{N^2})$ time. The latter evaluates frequency which is reduced to the computation of subgraph isomorphisms (a well-known *NP*-hard problem) and takes $\mathcal{O}(N^n)$ time. Overall, the complexity of the mining process is $\mathcal{O}(2^{N^2} \cdot N^n)$ time which is exponential in the problem size. Thus, it is of crucial importance to devise appropriate heuristics and optimizations that improve execution performance. Several works study the subgraph generation process and propose techniques that significantly improve performance [29, 20]. These techniques are implemented in Algorithm SubgraphExtension. In the following section, we consider the optimization of Algorithm IsFrequentCsp that computes subgraph isomorphisms.

3.3 Optimizing Frequency Evaluation

Algorithm IsFrequentCsp naively iterates over the solutions of the subgraph S to graph G CSP trying to find τ valid assignments for every variable. To guide this search process, we propose the heuristic illustrated in Algorithm IsFrequentHeuristic. Intuitively, the algorithm considers each variable at a time and searches for τ valid assignments. If these are found, it moves to the next variable and repeats the process. In more details, IsFrequentHeuristic starts by enforcing node and arc consistency. Then, the algorithm considers every variable and counts the valid assignments in its domain (stored in variable $count$). If, during the process, any variable domain remains with less than τ candidates, then the subgraph cannot be frequent, so the algorithm returns $false$ (Line 6 and 14). To count the valid assignments, IsFrequentHeuristic iterates over all nodes u in the domain D of a variable x and searches for a solution that assigns u to x. If the search is successful then $count$ is incremented by 1, and the process continues to the next node in D until the number of valid assignments ($count$) becomes τ, in which case the algorithm proceeds to the next domain (Line 13). On the other hand, if search is unsuccessful then u is removed from D and the algorithm continues with the next node in D. Updating D may trigger new inconsistencies in other domains, thus, arc consistency (Line 5) is checked again. IsFrequentHeuristic also implements the following optimization. Assume that for a domain D a solution was found for some node $u \in D$. Then, $count$ is incremented by 1 and all nodes (including u) that belong to this solution are *marked* in the respective

Algorithm: IsFrequent

Input: Graphs S and G and the frequency threshold τ
Output: $true$ if S is a frequent subgraph of G, $false$ otherwise

1 Consider the subgraph S to graph G CSP and apply node and arc consistency
`// Push-down pruning`
2 **foreach** *edge* e *of* S **do**
3 Let $S^{/e}$ be the graph after removing e from S
4 Remove the values of the domains in S that correspond to invalid assignments of $S^{/e}$
`// Unique labels`
5 **if** S *and* G *satisfy the unique labels optim. conditions* **then**
6 **if** *the size of any domain is less than* τ **then return** $false$
7 **else return** $true$
`// Automorphisms`
8 Compute the automorphisms of S
9 **foreach** *variable* x *and its domain* D **do**
10 $count \leftarrow 0$, $timedoutSearch \leftarrow \emptyset$
11 **if** *there is an automorphism with a computed domain* D' **then**
12 $D \leftarrow D'$ and move to the next x variable (Line 9)
13 Apply arc consistency
14 **if** *the size of a domain is less than* τ **then return** $false$
`// Lazy search`
15 **foreach** *element* u *of* D **do**
16 **if** u *is already marked* **then** $count$++
17 **else**
18 Search for a solution that assigns u to x for a given time threshold
19 **if** *search timeouts* **then** Save the search state in a structure $timedoutSearch$
20 **if** *a solution* $\mathcal{S}ol$ *is found* **then**
21 Mark all values of $\mathcal{S}ol$ to the corresponding domains
22 $count$++
23 **else** Remove u from the domain D and add u to the invalid assignments of D in S
24 **if** $count = \tau$ **then** Move to the next variable (Line 9)
`// Resume timed-out search if needed`
25 **if** $|timedoutSearch| + count \geq \tau$ **then**
`// Decompose`
26 Decompose graph S into a set of graphs $\mathbf{Set}$ that contain the newly added edge
27 **foreach** $s \in \mathbf{Set}$ **do** Remove invalid assignments of s from the respective domains of S
28 **foreach** $t \in timedoutSearch$ **do**
29 Resume search from the saved state t
30 **if** *a solution* $\mathcal{S}ol$ *is found* **then**
31 Mark all values of $\mathcal{S}ol$ to the corresponding domains
32 $count$++
33 **else** Remove u from the domain D and add u to the invalid assignments of D in S
34 **if** $count = \tau$ **then** Move to the next variable (Line 9)
35 **return** $false$ `// Domain is exhausted and` $count < \tau$
36 **return** $true$

domains (Line 10). Hence, if these nodes are considered in a later iteration of the algorithm, they are recognized as already belonging to a solution (Line 8). This precludes any further search.

In the following, we introduce Algorithm IsFrequent that enhances IsFrequentHeuristic through several optimizations that significantly improve execution performance. IsFrequent uses three novel optimizations, namely, *Push-down pruning*, *Lazy search* and *Unique labels*. Finally, IsFrequent specializes, for frequent mining, *Decomposition pruning* and *Automorphisms*, that are known to speed-up search [8] and frequent subgraph mining [1] respectively. In the sequel, we present the optimization techniques according to their execution order in the IsFrequent algorithm.

Push-down pruning. The subgraph generation tree is constructed by extending a parent subgraph with one edge at a time. Since the parent is a substructure of its children, those assignments that were pruned from the domains of the parent, cannot be valid as-

signments for any of its children. For example, Fig. 3a illustrates a part of a subgraph generation tree consisting of subgraph S_1 which is extended to S_2, S_3 and then to S_4 (via S_2). Assume that when considering subgraph S_1, ISFREQUENT excludes elements a_3, b_1, and a_3 from the domain of variables v_1, v_2, and v_3 respectively (depicted by light gray ovals in Fig. 3b). This information can be pushed down such that a_3, b_1, a_3 are also pruned from all descendants of S_1. This happens recursively; for instance, the assignments pruned because of S_2 are depicted by dark gray dotted ovals.

The same substructure may also appear in subgraphs that do not have an ancestor/descendant relationship. In the example of Fig. 3, S_4 is not a descendant of S_3; however, both contain substructure $A-B-A-C$. Since S_3 and S_4 are in different branches, pushing down the pruned assignments is not applicable. Instead, we use a hash table to store the pruned assignments of previously checked subgraphs. The hash key is the *DFScode* canonical representation of S_3 [29]. When S_4 is generated, the hash table is searched for matching substructures. If one is found, the corresponding invalid assignments are pruned from the domains of S_4. ISFREQUENT applies this optimization (Lines 2-4) using the invalid assignments populated while searching for valid nodes (Lines 23 and 33).

Saving the invalid assignments of subgraphs results in a significant performance gain for the following two reasons.

- Subgraphs (like S_4) take advantage of the respective pruning of smaller subgraphs (like S_1 and S_2) to prune invalid assignments. Thus, the domains of the subgraph variables are reduced avoiding the expensive search procedure (Lines 18 and 29). In many cases, a subgraph may be eliminated without search. For instance, in Fig. 3, assuming that $\tau = 3$, S_4 can be eliminated, because there are only two valid assignments of variable v_1 remaining in its domain.
- This domain reduction also speeds up the search process since it highly depends on the domain size. For instance, in Fig. 3, assuming that $\tau = 2$, when considering variable v_1, the search space has a size of $2{\cdot}2{\cdot}3{\cdot}4 = 48$ combinations (bottom of Fig. 3b), while without using this optimization the respective search space size is $5{\cdot}3{\cdot}5{\cdot}6 = 450$ combinations.

To perform push-down pruning, Line 3 constructs $\mathcal{O}(n^2)$ subgraphs $S^{/e}$ by removing an edge from S, (n is the number of nodes in S) and uses a hash lookup to remove the invalid assignment (Line 4). Thus, the overall complexity is $\mathcal{O}(n^2)$ time.

Unique labels. In the case of data graphs with a single label per node and subgraphs having a tree-like structure and unique node labels, the following optimization can be applied:

Proposition 3 *Let G be a graph with a single label per node, $S(V_S, E_S, L_S)$ be a subgraph of G, S's underlying undirected graph is a tree, and all of its node labels are unique, i.e., $L_S(v) \neq L_S(v')$ for all v and v' in V_S such that $v \neq v'$. To calculate $s_G(S)$ directly, it suffices to consider the S to G CSP and refine the domains of variables by enforcing node and arc consistency.*

PROOF: Since each graph node has a single label and the query has unique labels, no node can appear in more than one domain. For any S, we will use induction to prove that each value N in each domain of S (after applying the node and arc consistency constraints) is part of a valid solution. Let Q be a copy of S where all of S's directed edges are replaced with undirected ones. Q is connected, undirected, and acyclic, therefore it is a tree. Let Q be rooted at the node corresponding to N's domain.

- For Q with $height = 1$, N is guaranteed to be part of a valid solution (by definition of the node and arc consistency constraints and by considering the fact that the same node cannot appear in other domains).

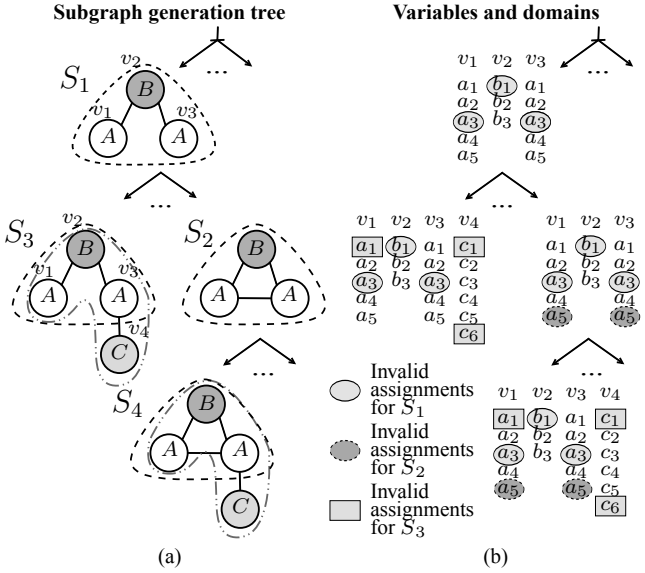


Figure 3: (a) Construction of the subgraph tree. (b) Variables and domains of the corresponding subtrees. Marked nodes represent the pruned assignments which are pushed down the tree.

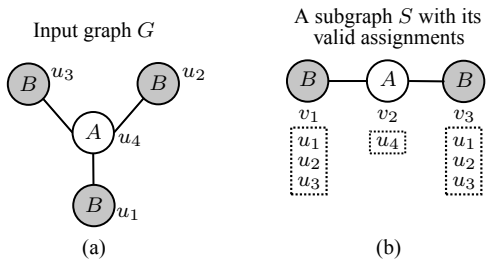


Figure 4: Automorphisms. (a) Input graph G. (b) Subgraph S and its valid assignments.

- For Q with $height = R$, let T be a subgraph of S and its underlying undirected graph is a subtree of Q sharing the same root but with $height = R - 1$. Let L be the set of T's leaf nodes and assume that T has a solution. Q is composed of T and the set of trees Z with height 1 (or 0) each rooted at a distinct node from L. Since each element in Z has a solution in G, and each solution joins with T's solution only by its corresponding root in Z, hence, a valid solution for S exists.

Note that the final step cannot be applied when the underlying undirected graph Q contains a cycle. For example if S is an undirected triangle of 3 nodes labeled (A, B, C) and the data graph G is undirected and contains 6 nodes forming a cycle: (A, B, C, A, B, C). When considering the S to G CSP after enforcing node and arc consistency the count $s_G(S)$ is 2, but, the correct result is 0. □

Example 2 *Consider the subgraph* DB−IR *and the graph G of Fig. 1. Let v_1 (resp. v_2) be the variable that corresponds to nodes labeled with* DB *(resp.* IR*). The initial domains are $D_{v_1} = D_{v_2} = \{u_0, \ldots, u_9\}$. After applying node and arc consistency we have $D_{v_1} = \{u_1, u_5, u_6\}$ and $D_{v_2} = \{u_0, u_3, u_4, u_8\}$ which encodes the actual isomorphisms of the subgraph to graph G.*

If the conditions hold (Line 5), GRAMI uses the current domain sizes to directly decide whether S is frequent or not (Lines 6-7). The overall process can be performed in $\mathcal{O}(n)$ time.

Automorphisms. Automorphism is an isomorphism of a graph to itself. Automorphisms appear because of symmetries. Following

[1], such symmetries in the subgraph can be used to prune equivalent branches and reduce the search space. For example, consider subgraph S of graph G presented in Fig. 4; S has automorphisms. To determine if S is frequent in G, while iterating over the domain of v_1, IsFrequent finds the assignment $(v_1, v_2, v_3) = (u_1, u_4, u_2)$ to be a solution (i.e., an isomorphism of S to G). Due to the symmetry of the subgraph S, assignment $(v_1, v_2, v_3) = (u_2, u_4, u_1)$ is also a solution. The benefits of this observation are twofold. First, we may identify the valid assignments of a variable more efficiently. More importantly, when we compute all valid assignments of a variable (like v_1) we also compute the valid assignments for its symmetric counterpart (i.e., v_3).

IsFrequent detects automorphisms in Line 8. This requires $\mathcal{O}(n^n)$ time where n is the number of nodes in subgraph S. In practice, despite the exponential worst-case bound, the cost of automorphisms is very low since the size of subgraph S is negligible compared to the size of the graph G.

Lazy search. Intuitively, to prove that a partial assignment does not contribute to any valid solution, the search algorithm has to exhaust all available options; a rather time consuming process. Thus, if a search for a solution that pertains to a specific partial assignment takes a long time, then this is probably because the partial assignment cannot contribute to a complete valid assignment. To address such cases, initially IsFrequent searches for a solution only for a limited time threshold (Line 18). The intuition of the optimization is that other assignments may produce much faster results that will help indicate if the subgraph is frequent ($s_G(S) \geq \tau$). In such a case, the result of the timed out search would be irrelevant, hence, there is no reason to waste time in further search. Nevertheless, this cannot guarantee that a timed out partial assignment will not eventually be essential for proving the frequency of the subgraph. Thus, if search is timed out, the algorithm stores the search state in the *timedoutSearch* set of nodes with incomplete check. These searches will only be resumed when the non-timed out cases are not sufficient to show that a subgraph is frequent. More specifically, timed-out searches are considered if after the time limited search, $count < \tau$ and $count$ plus the size of *timedoutSearch* (i.e., the number of timed out searches) surpasses the threshold τ (Line 25). Only then, the algorithm resumes each timed out search $t \in timedoutSearch$ from its saved state but without a time-out option until enough assignments are found to prove frequency (Line 34). Note that, if necessary, IsFrequent eventually searches the entire search space for each variable to provide the exact solution.

The complexity of Lazy search (Lines 15-24) can be done in $\mathcal{O}(N)$ time (note that the search of Line 18 takes constant time since it is performed for a specific time frame).

Decomposition pruning. The final optimization is performed in Lines 26 and 27. At this point, the algorithm is about to resume the timed out searches. To reduce the problem size, the algorithm decomposes the input subgraph S into a set of distinct subgraphs $\mathbf{Set}$. Recall that algorithm SubgraphExtension extends subgraphs by adding an edge e from the set of frequent edges *fEdges*. Set $\mathbf{Set}$ is constructed by removing one edge at a time from S and adding to $\mathbf{Set}$ the connected component that includes edge e. Any other decomposition has already been considered by the *Push-down pruning* optimization. Finding and removing invalid assignments from the domains of the elements of $\mathbf{Set}$ is a much easier task because they are smaller than the original subgraph S.

For example, consider Fig. 5. Subgraph S extends S' with edge $C-K$ and, thus, it is decomposed into $\mathbf{Set}$ that contains subgraphs S_1 to S_3. Let us assume that the variable corresponding to the new node labeled with K is v_k and the initial domain of v_k contains

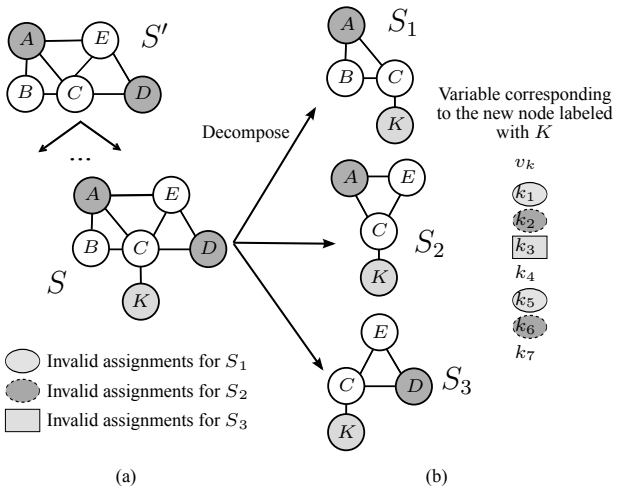


Figure 5: (a) Subgraph S is generated by extending S' with edge $C-K$. (b) S is decomposed into overlapping subgraphs S_1 to S_3 containing the newly extended edge $C-K$.

values k_1 to k_7. Further, assume that using subgraphs S_1, S_2 and S_3 we can exclude values $\{k_1, k_5\}$, $\{k_2, k_6\}$ and $\{k_3\}$ respectively. The decomposition optimization removes all these values from the domain of v_k, therefore, it only contains the values k_4 and k_7.

Decomposition pruning can be done in $\mathcal{O}(n^2)$. Resuming timed-out searches (Lines 28-34) requires solving a CSP on $n-1$ variables with domain of size N and can be done in $\mathcal{O}(N^{n-1})$ time.

Complexity analysis of IsFrequent. Let N and n be the number of nodes in G and S respectively. Push-down pruning, unique labels and automorphisms can be done in $\mathcal{O}(n^2)$, $\mathcal{O}(n)$ and $\mathcal{O}(n^n)$ respectively. Subgraph size is negligible in comparison to the data graph size, and thus these procedures are not expensive. IsFrequent applies arc consistency, lazy search and resumes timed-out search that can be done in $\mathcal{O}(Nn)$, $\mathcal{O}(N)$ and $\mathcal{O}(N^{n-1})$ respectively. Thus, the complexity of IsFrequent is determined by the resumed timed-out searches. More specifically, if p is the possibility expressing that a node in a domain of a variable is valid, then to find the required τ valid assignments we need to consider τ/p nodes and solve τ/p CSPs of size $n-1$ for each one of the n variables. In total, the complexity bound is $\mathcal{O}(n \cdot \tau/p \cdot N^{n-1})$.

4. GraMi EXTENSIONS

Generalization to pattern mining. Section 3 models the subgraph isomorphism problem (Definition 1) as a subgraph to graph CSP (Definition 5). Similarly, a pattern embedding ϕ (Definition 4) can be mapped to a CSP by replacing Condition 3c of Definition 5 as follows.

3c) $\Delta(x_v, x_{v'}) \leq \delta$, *for every* $x_v, x_{v'} \in \mathcal{X}$ *such that* $(v, v') \in E_P$ *(where* Δ *is the distance metric and* δ *is the distance threshold).*

Whenever it is clear from the context, we use v to refer to a node of the pattern and x_v to refer to the corresponding variable of the CSP as we do in the following example.

Example 3 *Consider Fig. 2. For* $\delta = 0.3$*, the pattern* P_1 *of graph G CSP is defined as:*

$$\left(\begin{array}{c} (v_1, v_2, v_3),\ \{\{u_0, \ldots, u_9\},\ \ldots,\ \{u_0, \ldots, u_9\}\}, \\ \{\ v_1 \neq v_2 \neq v_3,\ L(v_1) = \text{DM},\ L(v_2) = \text{IR},\ L(v_3) = \text{DB}, \\ \Delta(v_1, v_2) \leq 0.3,\ \Delta(v_2, v_3) \leq 0.3,\ \Delta(v_1, v_3) \leq 0.3\ \} \end{array} \right)$$

The notations for a solution (Proposition 1) and valid (or invalid) assignments (Definition 6) are easily extended to support pattern to

Table 1: Definitions of the anti-monotonic structural constraints for pattern P, implemented in CGRAMI

$\|V_P\| \leq \alpha$	Number of nodes should not exceed α
$\|E_P\| \leq \alpha$	Number of edges should not exceed α
$\max(degree(V_P)) \leq \alpha$	The maximum node degree is α

Table 2: Definitions of the anti-monotonic semantic constraints for pattern P, implemented in CGRAMI

$(\forall v \in V_P)(L(v) \in \mathcal{L})$	P contains only labels from $\mathcal{L}$
$(\forall v \in V_P)(L(v) \notin \mathcal{L})$	P does not contain any label from $\mathcal{L}$
$(\forall v, v' \in E_P)(L(v, v') \in \mathcal{E})$	P contains only edges from $\mathcal{E}$
$(\forall v, v' \in E_P)(L(v, v') \notin \mathcal{E})$	P does not contain any edges from $\mathcal{E}$
$(\neg subgraph(P', P))$	Pattern P must not contain a specific subgraph P'
$(\forall v \in V_P)(\text{count}(L(v)) \leq \alpha)$	A node label cannot appear more than α times in P

graph CSPs. For instance, assignment $(v_1, v_2, v_3) = (u_7, u_8, u_6)$ is a solution of the CSP of Example 3 and a pattern embedding of P_1 to G. Moreover, $v_2 = u_3$ is a valid assignment while $v_2 = u_0$ is invalid (and thus, cannot be extended to a solution).

Proposition 4 *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be the pattern P to graph G CSP. The MNI support of P in G satisfies τ, i.e., $\sigma_G(S) \geq \tau$, iff every variable in $\mathcal{X}$ has at least τ distinct valid assignments (i.e., embeddings of P in G).*

Continuing Example 3, we have $\sigma_G(P_1) \geq 2$ since all domains contain at least 2 valid assignments (the domains of variables v_1, v_2 and v_3 are $\{u_2, u_7\}$, $\{u_3, u_8\}$ and $\{u_1, u_6\}$ respectively).

To address the frequent pattern mining problem (Problem 2), we can also employ Algorithms ISFREQUENTHEURISTIC and ISFREQUENT, with the following additional preprocessing step. For each frequent node, we precompute the set of nodes that are reachable within distance δ. We run a distance-bound Dijkstra algorithm from each frequent node to find the shortest path to the reachable nodes, where the path distance is defined by the distance function Δ; the algorithm terminates when the distance of the shortest path exceeds δ. All optimizations of Section 3.3 apply directly in this setting as well. To avoid confusion, we use GRAMI for the subgraph mining problem and GRAMI(δ) for the pattern mining problem.

User-defined constraints. Typically, frequent patterns show interactions between nodes bearing the same label. For instance, in citation graphs, most collaborations are among authors working in the same field. In many applications, interactions among nodes of different types (like interdisciplinary collaborations) are more interesting and important [33]. To allow the user to focus on the interesting patterns, we developed CGRAMI, a version of GRAMI that supports two types of user-defined constraints: *(a) Structural*, such as "the number of vertices in pattern P should be at most α' and *(b) Semantic*, such as "P must not contain specific labels".

Although not a requirement, it is desirable that the user-defined constraints are anti-monotonic. In such cases, the constraints can be pushed down in the subgraph extension search tree to early prune large parts of the search space, thus accelerating the process. Tables 1 and 2 present a set of useful structural and semantic anti-monotonic constraints that are supported by CGRAMI.

Approximate mining. Frequent subgraph mining is a computationally intensive task since it is dominated by the NP-hard subgraph isomorphism problem. Thus, its performance is prohibitively expensive when applied to large graphs. Motivated by this, we introduce AGRAMI, an approximate version of our framework, which is able to scale to larger graphs. To maintain the quality of results, AGRAMI does not return any infrequent pattern (i.e., does not have false positives), although it may miss some frequent ones (i.e., may have false negatives). To achieve this, we modified the way ISFREQUENT handles time-outs (Line 18) as follows: we set the time-out to occur after $f(\alpha)$ iterations of the search. If a solution is found before this time-out, the *count* is updated as normal. On the other hand, if a time-out occurs it is assumed that the search was unsuccessful. If enough time-outs occur during the search of a specific domain such that its *count* remains less than τ, the pattern is considered to be infrequent. Parameter $f(\alpha) = \alpha^n \prod_1^n |D_i| + \beta$, where β is a constant, D_i are the domains of the variables, n is the number of variables and $0 < \alpha \leq 1$ is a user-defined approximation parameter. $\prod_1^n |D_i|$ grows exponentially; thus it has to be bounded by an exponential weight α^n. Increasing α decreases the approximation error at the expense of longer execution time. When $\alpha = 1$, AGRAMI becomes equivalent to GRAMI.

Table 3: Datasets and their characteristics

Dataset	Nodes	Distinct node labels	Edges	Density
Twitter	11,316,811	100	85,331,846	Dense
Patents	3,942,797	453	16,522,438	Medium
Aviation	101,185	6,173	133,087	Sparse
MiCo	100,000	29	1,080,298	Dense
CiteSeer	3,312	6	4,732	Medium

5. EXPERIMENTAL EVALUATION

In this section, we experimentally evaluate GRAMI and its extensions. For comparison, we have implemented GROWSTORE that follows a pattern *grow-and-store* approach [20, 29]. GROWSTORE uses the original code of GSPAN [29] and takes advantage of all its optimizations. The only difference is that GROWSTORE, similarly to GRAMI, use the efficient *MNI* metric. Both GROWSTORE and GRAMI are completely memory based. All experiments are conducted using Java JRE v1.6.0 on a Linux (Ubuntu 12) machine with 8 cores running at 2.67GHz with 192GB RAM and 1TB disk. Our experimental machine used an exotic memory size to accommodate the memory requirements of GROWSTORE; GRAMI may also run on ordinary machines with 4GB RAM for all datasets but Twitter.

Datasets. We experiment on several different workload settings by employing the following real graph datasets; their main characteristics are summarized in Table 3.

Twitter (`socialcomputing.asu.edu/datasets/Twitter`*).* This graph models the social news of Twitter and consists of ∼11M nodes and ∼85M edges. Each node represents a Twitter user and each edge represents an interaction between two users. The original graph does not have labels, so we randomly added labels to the nodes. The number of distinct labels was set to 100 and the randomization follows a Gaussian distribution.

Patents. This dataset models U.S. patents' citations and consists of a directed graph with ∼4M nodes and ∼16M edges. Each node represents a patent and each edge represents a citation. The graph is maintained by the National Bureau of Economic Research [32]. As a preprocessing step, we remove all unlabeled nodes.

MiCo. This dataset models the Microsoft co-authorship information and consists of an undirected graph with 100K nodes and ∼1M edges. Nodes represent authors and are labeled with the author's field of interest. Edges represent collaboration between two authors and are labeled with the number of co-authored papers. To populate MiCo we crawled the computer science collaboration graph from `academic.research.microsoft.com`.

CiteSeer (`cs.umd.edu/projects/linqs/projects/lbc`*).* CiteSeer represents a directed graph consisting of ∼3K publications (nodes) and ∼4K citations between them (edges). Each node has a single label representing a Computer Science area. Each edge has a label (0 to 100) that measures the similarity between the corresponding

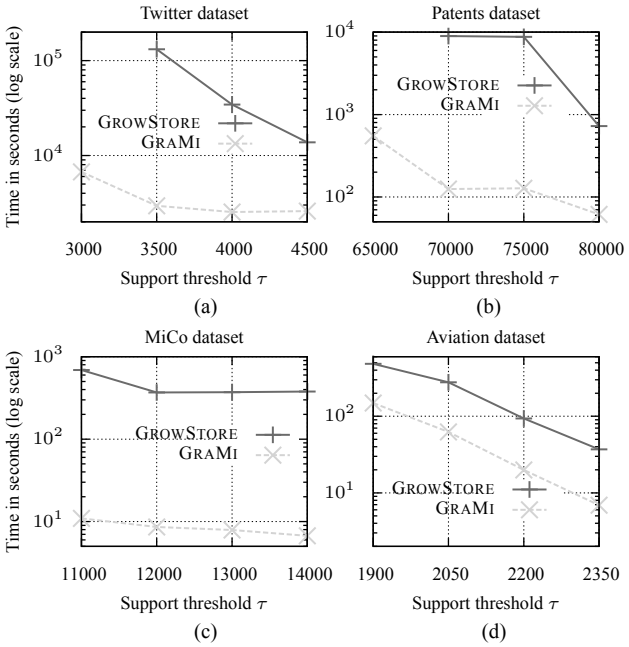


Figure 6: Performance of GRAMI and GROWSTORE

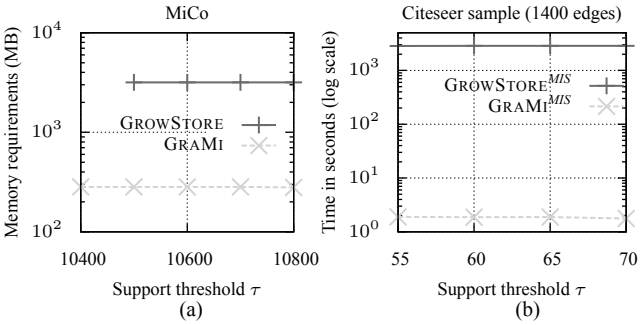


Figure 7: (a) Memory requirements for GRAMI and GROWSTORE and (b) Using *MIS* metric

pair of publications, a smaller label denotes a stronger similarity.

Aviation (`ailab.wsu.edu/subdue`). This dataset contains a list of records extracted from the aviation safety database and was used in [7, 20] for evaluation. Each record corresponds to an event and has several attributes (like event type, location, flight condition). This information is represented by a graph having two types of nodes and edges. The first type of nodes represents the events (and are labeled with the ids of the event) while the second represents attribute values (and are labeled with the actual value). The first type of edges links events and is labeled with their relationship (e.g., near to) while the second type links events with attribute values and is labeled with the attribute name. Aviation consists of 100K nodes and 133K edges. Note that Aviation is a fundamentally different dataset when compared with the previous ones. The Aviation graph has on average one edge per node, thus, it is very sparse. Also it has a very large number of distinct node labels.

Metrics. The support threshold τ is the key evaluation metric as it determines when a subgraph or a pattern is frequent. Decreasing τ results in an exponential increase in the number of possible candidates and thus exponential decrease in the performance of the mining algorithms. For a given time budget, an efficient algorithm should be able to solve mining problems for low τ values. When τ is given, efficiency is determined by the execution time.

To evaluate a result set, we consider the number and the maximum size of subgraphs/patterns in the set. Obviously, these values should be as large as possible.

Computing frequent subgraphs. Initially, we consider Problem 1 that mines frequent subgraph isomorphisms. Fig. 6 shows the performance of GROWSTORE and GRAMI on Twitter, Patents, MiCo and Aviation datasets. The number of results (intermediate and actual) grows exponentially when the support threshold τ decreases. Thus, the running time of all algorithms also grows exponentially. Unlike GROWSTORE, GRAMI does not need to enumerate all intermediate results, thus, it is more efficient. Our results indicate that GRAMI outperforms GROWSTORE by at least two orders of magnitude for Patents and MiCo datasets and by at least an order of magnitude for Twitter and Aviation datasets. For the larger datasets (Twitter and Patents) and for the lower τ (3K and 65K respectively), GROWSTORE was not able to produce results even when it was alloted 2 orders of magnitude more time than GRAMI.

Memory requirements. Fig.7a illustrates the memory requirements for GROWSTORE and GRAMI for the MiCo dataset. Since GROWSTORE needs to store all intermediate results, it consumes about an order of magnitude more memory. For τ=10, 400 the size of the intermediate results exceed the available memory (192GB), and hence GROWSTORE crashes. For this frequency, there is an increase in the number of the frequent subgraphs and thus an exponential increase in the number of intermediate candidates that need to be stored and checked for frequency. This trend also appears for the other datasets. GRAMI on the other hand is not affected by the increase in the output size. Most of the memory GRAMI uses, is required for the storage of the input graph G. The most costly data structure of ISFREQUENT is the hash table used by push-down pruning, but, still it does not exceed 2% for the overall required memory. Also the space needed to store timed-out searches (set $timedoutSearch$) was never above 1% of the total memory. For all our experiments, GRAMI could be also executed in machines with the typical memory size of 4GB except for the Twitter dataset.

Using *MIS* metric. In this experiment, we compare GROWSTORE^{MIS} the original version of GROWSTORE that uses the *MIS* metric with GRAMI^{MIS}, the modified version of GRAMI that also supports *MIS*. For the Aviation dataset, GRAMI^{MIS} takes slightly more time than GRAMI while GROWSTORE^{MIS} could not produce results even if it was alloted *three* orders of magnitude more time than GRAMI^{MIS}. Interestingly, GROWSTORE^{MIS} cannot produce results in reasonable time even for the much smaller Citeseer dataset. To achieve a comparison, we have constructed a new dataset by randomly sampling 1400 edges from the Citeseer dataset. The results are illustrated in Fig. 7b. Clearly, GRAMI^{MIS} outperforms GROWSTORE^{MIS} by up to 3 orders of magnitude.

Computing frequent patterns. We now consider Problem 2 that mines frequent pattern embeddings. We evaluate the performance of GROWSTORE and GRAMI(δ) for several values of the distance threshold δ. We use the CiteSeer dataset and distance function $\Delta_h(u, v)$ defined as the number of hops in the shortest path that connects u and v. For GRAMI(δ), we test on two different distance thresholds namely 1 and 4. Intuitively, for $\delta = 1$ (respectively $\delta = 4$) two pattern nodes that are connected with an edge may be matched with two graph nodes that are one hop (respectively four hops) away. GROWSTORE can only find matches that are only one hop away. Thus, only GROWSTORE and GRAMI(1) are directly comparable since they both compute the same results. As shown in Fig. 8a, GRAMI(1) is an order of magnitude faster than GROWSTORE (note the logarithmic scale). As expected GRAMI(4) com-

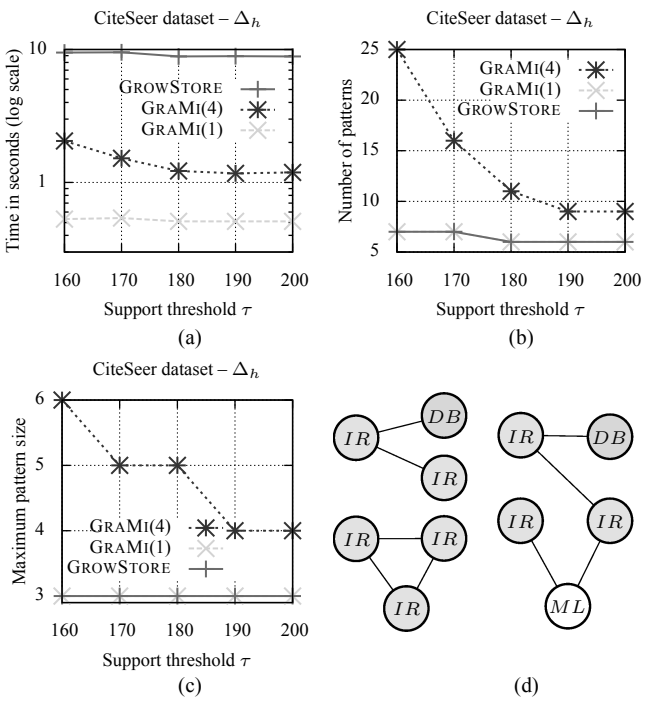


Figure 8: Performance evaluation for mining frequent patterns in CiteSeer dataset comparing between GROWSTORE and GRAMI(δ) where δ is the distance threshold

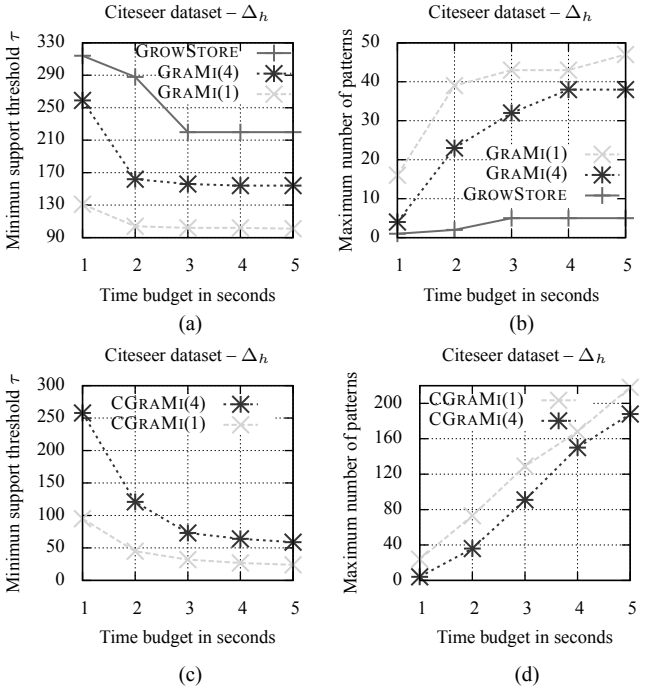


Figure 9: Comparing (a,c) the minimum support threshold and (b,d) the maximum number of frequent patterns that can be achieved within an allotted time budget. For (a,b) we used GRAMI(δ) and for (c,d) we use CGRAMI(δ) constrained to reject patterns with more than 4 nodes with the same label

putes more and larger patterns than GROWSTORE and GRAMI(1) (Figs. 8b and 8c). An example of a frequent pattern discovered by GRAMI is illustrated on the right of Fig. 8d and contains 5 nodes involving 3 different Computer Science areas. To compare, GROWSTORE computes the 3 nodes patterns at the left of Fig. 8d that involve 1 and 2 areas. To compute these results, GRAMI(4) takes more time than GRAMI(1) but is still faster than GROWSTORE.

To further illustrate the benefits of GRAMI(δ) we have conducted another set of experiments (Fig. 9). The aim of the experiments is to illustrate the properties of the patterns that can be generated within a specific time budget. Figs. 9a,b, consider the Citeseer dataset with the distance function Δ_h and compare between GROWSTORE, GRAMI(1) and GRAMI(4). Specifically, Fig. 9a shows the minimum support threshold τ that can be achieved, when the above algorithms are allotted a time budget that ranges from 1 to 5 seconds (lower is better). For this budget range, Fig. 9b illustrates the number of result patterns (higher is better). In both cases, GRAMI(1) and GRAMI(4) accomplish lower thresholds and result in more patterns than GROWSTORE.

CGRAMI: User-defined constraints. CGRAMI supports the addition of constraints on the returned results (Section 4). Using these constraints, the focus can be on more interesting pattern types like the ones that show interactions between nodes of a different type. To evaluate CGRAMI, we use the experimental setting of Fig. 9a,b. The only difference is that we now use CGRAMI(δ) with a constraint that does not allow more than 4 nodes with the same label in a pattern. The corresponding results are illustrated in Fig. 9c,d and are directly comparable to Fig. 9a,b. In every case and within the same time budget allowed for both GRAMI and CGRAMI, CGRAMI results in a significantly lower minimum support threshold τ and significantly larger frequent patterns set. For instance, for the Citeseer dataset with a time budget of 3 seconds, CGRAMI(1) achieves a 3 times lower threshold and almost 3 times more patterns than GRAMI. Additionally, CGRAMI generates patterns having about 3 times more label interactions than GRAMI.

AGRAMI: Approximate mining. AGRAMI, which offers approximate subgraph and pattern mining (Section 4), can be tuned by the approximation parameter α, $0 < \alpha \leq 1$ (value 1 means no approximation). Fig. 10 illustrates the performance of GRAMI and AGRAMI for several values for the α parameter in the Patents and MiCo datasets. We evaluate two parameters, execution time and recall, i.e., the percentage of subgraphs returned by AGRAMI with respect to the actual complete set of frequent subgraphs. For the Patents dataset, the performance gain is significant, nearly an order of magnitude for both $\alpha = 2{\cdot}10^{-5}$ and $\alpha = 3{\cdot}10^{-5}$. For $\alpha = 3{\cdot}10^{-5}$ the recall is always 100% (i.e., AGRAMI provides all subgraphs) except for $\tau = 63.600$ that is 95%. For $\alpha = 2{\cdot}10^{-5}$ the recall is always over 90%. For the MiCo dataset, the performance gain is significant, nearly an order of magnitude when $\alpha = 4{\cdot}10^{-4}$ and

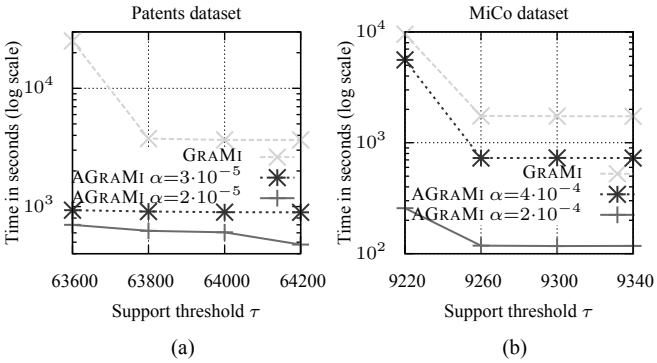


Figure 10: Performance evaluation of GRAMI and AGRAMI with different values for the approximation parameter.

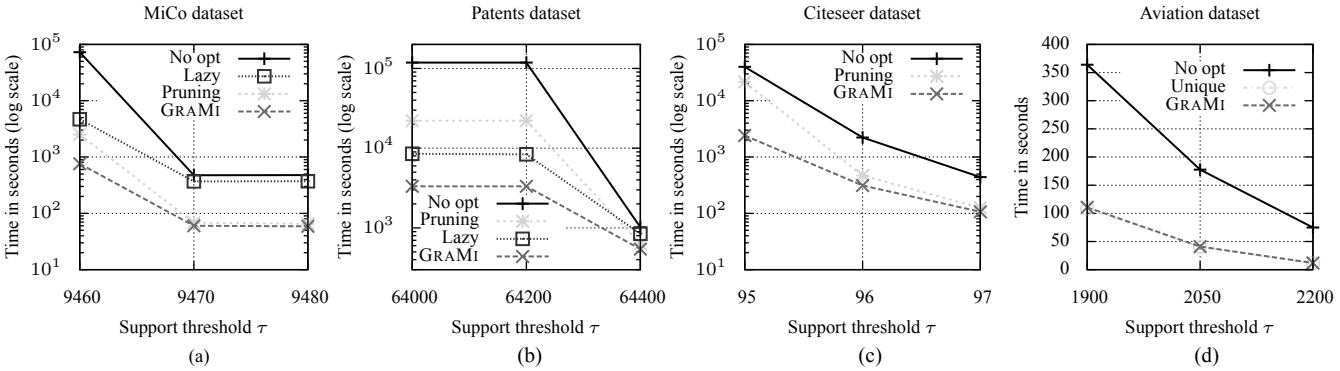


Figure 11: The effect of optimizations. No opt: **Algorithm** ISFREQUENTHEURISTIC **(Section 3.2).** Lazy: **Lazy search and decomposition optimizations enabled.** Pruning: **Only pruning push-down optimization enabled.** Unique: **Only unique labels optimization enabled.** GRAMI: **All optimization enabled (Algorithm** ISFREQUENT**).**

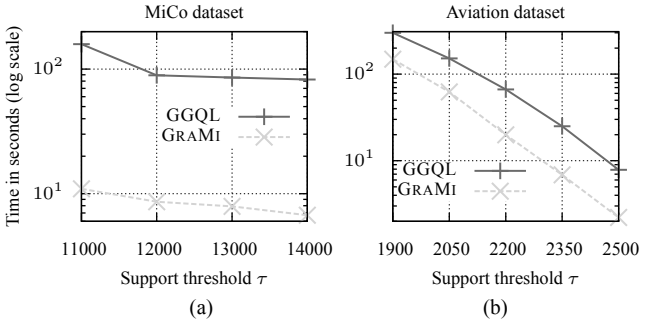


Figure 12: Performance comparison between GRAMI **and** GGQL**; a modified version of** GRAMI **that replaces** ISFREQUENT **with a counting function based on GraphQL**

nearly two orders of magnitude when $\alpha = 2{\cdot}10^{-4}$. Interestingly, the recall is always 100%.

Optimizations. This experiment demonstrates the effect of the optimizations discussed in Section 3.3 on mining the different datasets. A summary is illustrated in Fig. 11. For the MiCo dataset, the most effective optimization is *Push-down pruning* (denoted by Pruning in Fig. 11a) that achieves an improvement of up to 2 orders of magnitude. Following that, are the *Lazy search* and the *Decomposition pruning* optimizations, both are combined and denoted by Lazy in Fig. 11a. The two optimizations accomplish an improvement of up to an order of magnitude. Last comes the *Automorphism* and *Unique labels* optimizations that achieve only 4% improvement, since most of the frequent subgraphs in the MiCo dataset neither have automorphisms nor unique labels. For presentation clarity in Fig. 11a, we do not illustrate the results of the last two optimization methods. A similar trend also applies to Patents and Citeseer datasets (Figs. 11b and 11c).

For the Aviation dataset (Fig. 11d), a different optimization trend is noticed since this dataset is fundamentally different than MiCo Patents and Citeseer. In this case, the most effective optimization is Unique labels (denoted by Unique in Fig. 11d). As discussed earlier, the Aviation dataset is extremely sparse and has a very large number of distinct node labels, thus, the Unique label optimization is very effective. In contrast to the previous cases, all other optimizations do not offer any improvement and are not illustrated.

Comparison with subgraph isomorphism techniques. To address the frequent data mining problem, we may also employ subgraph isomorphism techniques [21]. For comparison, we have implemented GGQL; a modified version of GRAMI that replaces ISFREQUENT with a frequency evaluation function based on GRAPHQL [16]; one of the fastest state-of-the-art subgraph isomorphism techniques [21]. Clearly, as illustrated in Fig. 12, GRAMI outperforms GGQL by at least 3 times and up to more than an order of magnitude. This is easily justifiable since GRAMI uses several optimizations and visits only the necessary nodes in the input graph to solve the frequent subgraph mining problem.

6. RELATED WORK

This section discusses related work in many different directions.

Transactional mining. This setting is concerned with mining frequent subgraphs on a dataset of many, usually small, graphs. FSG [18] construct new candidate patterns by joining smaller frequent ones. The drawback of this approach is the costly join operation and the pruning of false positives. GSPAN [29] proposes a variation of the pattern *growth* approach. It uses an extension mechanism, where subgraphs grow directly from a single subgraph instead of joining two previous subgraphs. Other methods focus on particular subsets of frequent subgraphs. MARGIN [26] returns maximal subgraphs only, whereas CLOSEGRAPH [30] generates subgraphs that have strictly smaller support than any of their parts. LEAP [28] and GRAPHSIG [24], on the other hand, discover important subgraphs that are not necessarily frequent.

Although GRAMI focuses on the single large graph setting, it may be easily specialized to also support graph transactions.

Single graph mining. On the equally important single graph setting there exists less work. The major difference is the definition of an appropriate anti-monotone support metric (Section 2). SIGRAM [20] uses the *MIS* metric and proposes an algorithm that finds frequent connected subgraphs in a single, labeled, sparse and undirected graph. SIGRAM follows a *grow-and-store* approach, i.e., it needs to store intermediate results in order to evaluate frequencies. Overall, SIGRAM needs to enumerate all isomorphisms and relies on the expensive computation of *MIS* (which is *NP*-complete), thus the method is very expensive in practice.

Since the number of intermediate embeddings increases exponentially with the graph size, such approaches do not scale for large graphs. In contrast, GRAMI does not need to construct all the isomorphisms, hence, it can scale to much larger graphs. More importantly, GRAMI supports frequent subgraph and pattern mining (Problems 1 and 2 respectively). Thus, it allows for exact isomorphism matching and the more general distance-constrained pattern matching. Additionally, GRAMI supports constraint-based mining and works on directed, undirected, single and multi-labeled graphs.

Approximate mining. There is work on approximate techniques for solving the frequent subgraph mining problem as well. In GREW [19], the authors propose a heuristic approach that prunes large

parts of the search space, but discovers only a small subset of the answers. GAPPROX[3] employs an approximate version of the *MIS* metric. It mainly relies on enumerating all intermediate isomorphisms but allows approximate matches. SEUS [14] is another approximate method that constructs a compact summary of the input graph. This facilitates pruning many infrequent candidates, however, it is only useful when the input graph contains few and very frequent subgraphs. SUBDUE [7] is a branch-and-bound technique that mines subgraphs that can be used to compress the original graph. Finally, Khan *et al.* [17] propose proximity patterns, which relax the connectivity constraint of subgraphs and identify frequent patterns that cannot be found by other approaches.

In contrast to the existing work, AGRAMI, approximate version of GRAMI, may miss some frequent subgraphs, but the returned results do not have false positives.

Subgraph isomorphism. The frequent subgraph mining problem relies on the computation of subgraph isomorphisms. This problem is NP-complete and the first practical algorithm that addresses this problem follows a backtracking technique [27]. Since then, several performance enhancements were proposed, ranging from CSP based techniques [23], search order optimization [16], indexing [31] and parallelization [25].

Although the state-of-the-art subgraph isomorphism techniques lead to significant improvements, they are not as effective in the frequent subgraph mining problem for two reasons: First, subgraph isomorphism techniques are effective in finding all appearances of a subgraph, while for the frequent subgraph mining task, it is sufficient to find the minimum appearances that satisfy the support threshold; this difference affects the way graph nodes are traversed, minimizing the number of node visits during search. Additionally, modern techniques employ global pruning and indexing techniques. Forming such structures on large graphs results in a huge and often unacceptable overhead. GRAMI is based on a novel CSP method that overcomes the previous shortcomings and outperforms state-of-the-art subgraph isomorphism techniques by up to an order of magnitude. This is experimentally validated in Section 5.

Pattern matching. There is work on pattern matching over graphs as well. R-JOIN [4] supports reachability queries in a directed graph; If two nodes v and v' are reachable in the query then their corresponding mappings u and u' in the graph must also be reachable. DISTANCE-JOIN [34] extends the idea to undirected graphs and accommodates constraints on the distance in the path. GRAMI presents an extension to support frequent pattern mining, the extended version adopts the pattern definition from [34].

7. CONCLUSIONS

Many important applications, ranging from bioinformatics to social network study and from personalized advertisement (e.g., recommendation systems) to security (e.g., identification of terrorist groups), depend on graph mining. This paper introduces GRAMI; a versatile algorithm for discovering frequent patterns in a single large graph, a significantly more difficult problem compared to the usual case of mining a set of small graph transactions. The modeling of the frequency evaluation operation as a constraint satisfaction problem is the crux idea of GRAMI. We complement this idea with a set of optimizations that allows for the efficient performance of GRAMI. We also implement a version that supports structural and semantic constraints and an approximate version that scales to larger graphs. Our experimental results with real datasets demonstrate the effectiveness of GRAMI which is up to 2 orders of magnitude faster than existing approaches while discovering larger and more interesting frequent patterns.

8. REFERENCES

[1] B. Bringmann. *Mining Patterns in Structured Data*. PhD thesis, KU Leuven, 2009.

[2] B. Bringmann and S. Nijssen. What is frequent in a single graph? In *Proc. of PAKDD*, pages 858–863, 2008.

[3] C. Chen, X. Yan, F. Zhu, and J. Han. GAPPROX: Mining frequent approximate patterns from a massive network. In *Proc. of ICDM*, pages 445–450, 2007.

[4] J. Cheng, J. X. Yu, B. Ding, P. S. Yu, and H. Wang. Fast graph pattern matching. In *Proc. of ICDE*, pages 913–922, 2008.

[5] Y.-R. Cho and A. Zhang. Predicting protein function by frequent functional association pattern mining in protein interaction networks. *Trans. Info. Tech. Biomed.*, 14(1):30–36, Jan. 2010.

[6] W.-T. Chu and M.-H. Tsai. Visual pattern discovery for architecture image classification and product image search. In *Proc. of ICMR*, pages 27:1–27:8, 2012.

[7] D. J. Cook and L. B. Holder. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, 1(1):231–255, 1994.

[8] S. de Givry, T. Schiex, and G. Verfaillie. Exploiting tree decomposition and soft local consistency in weighted CSP. In *Proc. of AAAI*, pages 22–27, 2006.

[9] M. Deshpande, M. Kuramochi, and G. Karypis. Frequent sub-structure-based approaches for classifying chemical compounds. In *Proc. of ICDM*, pages 35–42, 2003.

[10] A. Deutsch, M. Fernandez, and D. Suciu. Storing semistructured data with stored. In *Proc. of SIGMOD*, pages 431–442, 1999.

[11] C. Domshlak, R. I. Brafman, and S. E. Shimony. Preference-based configuration of web page content. In *Proc. of IJCAI*, pages 1451–1456, 2001.

[12] M. Fiedler and C. Borgelt. Subgraph support in a single large graph. In *Proc. of ICDMW*, pages 399–404, 2007.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[14] S. Ghazizadeh and S. S. Chawathe. Seus: Structure extraction using summaries. In *Proc. of DS*, pages 71–85, 2002.

[15] V. Guralnik and G. Karypis. A scalable algorithm for clustering sequential data. In *Proc. of ICDM*, pages 179–186, 2001.

[16] H. He and A. K. Singh. Graphs-at-a-time: query language and access methods for graph databases. In *Proc. of SIGMOD*, pages 405–418, 2008.

[17] A. Khan, X. Yan, and K.-L. Wu. Towards proximity pattern mining in large graphs. In *Proc. of SIGMOD*, pages 867–878, 2010.

[18] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proc. of ICDM*, pages 313–320, 2001.

[19] M. Kuramochi and G. Karypis. GREW - A scalable frequent subgraph discovery algorithm. In *Proc. of ICDM*, pages 439–442, 2004.

[20] M. Kuramochi and G. Karypis. Finding frequent patterns in a large sparse graph. *Data Mining and Knowledge Discovery*, 11(3):243–271, 2005.

[21] J. Lee, W.-S. Han, R. Kasperovics, and J.-H. Lee. An in-depth comparison of subgraph isomorphism algorithms in graph databases. *PVLDB*, 6(2):133–144, Dec. 2012.

[22] A. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[23] J. J. McGregor. Relational consistency algorithms and their application in finding subgraph and graph isomorphisms. *Information Sciences*, 19:228–250, 1979.

[24] S. Ranu and A. K. Singh. GRAPHSIG: A scalable approach to mining significant subgraphs in large graph databases. In *Proc. of ICDE*, pages 844–855, 2009.

[25] Z. Sun, H. Wang, H. Wang, B. Shao, and J. Li. Efficient subgraph matching on billion node graphs. *PVLDB*, 5(9):788–799, May 2012.

[26] L. T. Thomas, S. R. Valluri, and K. Karlapalem. MARGIN: Maximal frequent subgraph mining. *TKDD*, 4(3):10:1–10:42, 2010.

[27] J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of ACM*, 23:31–42, 1976.

[28] X. Yan, H. Cheng, J. Han, and P. S. Yu. Mining significant graph patterns by leap search. In *Proc. of SIGMOD*, pages 433–444, 2008.

[29] X. Yan and J. Han. GSPAN: Graph-based substructure pattern mining. In *Proc. of ICDM*, pages 721–724, 2002.

[30] X. Yan and J. Han. CLOSEGRAPH: mining closed frequent graph patterns. In *Proc. of SIGKDD*, pages 286–295, 2003.

[31] X. Yan, P. S. Yu, and J. Han. Graph indexing: a frequent structure-based approach. In *Proc. of SIGMOD*, pages 335–346, 2004.

[32] R. Zafarani and H. Liu. Social computing data repository at ASU, 2009.

[33] F. Zhu, X. Yan, J. Han, and P. S. Yu. GPRUNE: A constraint pushing framework for graph pattern mining. In *Proc. of PAKDD*, pages 388–400, 2007.

[34] L. Zou, L. Chen, and M. T. Özsu. Distance-join: pattern match query in a large graph database. *PVLDB*, 2(1):886–897, 2009.

The *Baquara*^{2} Knowledge-based Framework for Semantic Enrichment and Analysis of Movement Data

Renato Fileto[a], Cleto May[a], Chiara Renso[b], Nikos Pelekis[c], Douglas Klein[a], Yannis Theodoridis[d]

[a] *PPGCC/INE/CTC, Federal University of Santa Catarina, Florianópolis-SC, Brazil*
[b] *HPC Lab, ISTI/CNR, Pisa, Italy*
[c] *Department of Statistics and Insurance Science, University of Piraeus, Greece*
[d] *Department of Informatics, University of Piraeus, Greece*

Abstract

The analysis of movements frequently requires more than just spatio-temporal data. Thus, despite recent progresses in trajectories handling, there is still a gap between movement data and formal semantics. This gap hinders movement analyses benefiting of available knowledge, with well-defined and widely agreed semantics. This article describes the *Baquara*^{2} framework to help narrow this gap by exploiting knowledge bases to semantically enrich and analyze movement data. It provides an ontological model for structuring and abstracting movement data in a multilevel hierarchy of progressively detailed movement segments that generalize concepts such as trajectories, stops, and moves. *Baquara*^{2} also includes a general customizable process to annotate movement data with concepts and objects described in ontologies and Linked Open Data (LOD) collections. The resulting semantic annotations enables queries for movement analyzes based on application and domain specific knowledge. The proposed framework has been used in experiments to semantically enrich movement data collected from social media with geo-referenced LOD. The obtained results enable powerful queries that illustrate *Baquara*^{2} capabilities.

Keywords: trajectories of moving objects, social media, ontologies, linked open data, semantic enrichment, movement data analysis.

1. Introduction

Nowadays, large amounts of movement data (trajectories of moving objects, time-ordered sequences of posts on social media, geo-referenced Web logs) can be gathered by using a variety of devices (e.g., smart phones equipped with GPS or just connected to a GSM network, vehicles equipped with RFID). These data can be useful for several kinds of spatio-temporal information analysis and a myriad of useful applications, in domains ranging from tourism and marketing to traffic and home land security [1, 2, 3, 4, 5, 6]. However, even spatio-temporal positions associated with text (e.g., freely annotated trajectories, geo-referenced

tweets) lack well-defined semantics to support precise information analysis. For instance, the tag `"Rio"` in a social media post may refer to a city, a state, or even a restaurant or nightclub, among other possibilities. Therefore, to realize potential applications it is necessary to develop appropriate methods to semantically enrich movement data. The recent progresses in movement data handling [7, 8, 9, 10, 11] include plenty of significant contributions for structuring and analyzing movement data, but are mainly based on just spatio-temporal data. Notwithstanding, it is recognized by the scientific community that semantic issues, including the exploitation of textual and contextual information that can come associated with spatio-temporal coordinates, must be addressed yet to better understand and exploit movement data [12, 13, 14, 15, 16, 17].

This article is a major extension of [38]. It describes the *Baquara*^{2} conceptual framework for movement data enrichment and analysis, which includes a customizable process to semantically enrich movement data, and an upper ontology that provides a conceptual model to accommodate the enriched data and support knowledge-based queries for movement analysis. The *Baquara*^{2} ontology has a rich set of constructs to semantically describe progressively detailed movement segments, which are organized in an arbitrarily deep hierarchy and generalize concepts like trajectories and episodes (e.g., stops and moves). The *Baquara*^{2} semantic enrichment process aims to semantically annotate movement segments with references to concepts (classes) and objects (instances of concepts) of movement analysis facets, such as space, time, and goals. These facets can be built from extracts of description logics compatible knowledge-bases, such as LOD collections and their associated ontologies [18, 19].

The proposed ontology and semantic enrichment process can be used with a variety data. The adaptation points of the *Baquara*^{2} conceptual framework are the domain knowledge used to semantically describe movements, and the tasks of the semantic enrichment process. The domain knowledge employed can be selected according to the spatio-temporal scope of the movements to be analyzed and the application domain. The general enrichment process can be customized by choosing specific methods to perform its tasks, which vary with the semantic enrichment objectives and facets (places, goals, transportation means, etc.). This work illustrates such a customization, in the form of an algorithm that connects movement data to visited Places of Interest (PoIs), according to their spatial proximity and the lexical similarity between LOD properties and text that comes associated with spatio-temporal coordinates (e.g., tweets, textually annotated trajectory positions or stops).

Baquara^2 enables queries referring to concepts and/or objects used to semantically enrich the movement data, such as the following ones:

Query 1 *Select the social media users with at least one stop to visit a mountain called Corcovado in the city of Rio, followed by one stop in a marketplace, where he/she does at least one finer stop in a restaurant.*

Query 2 *Determine the percentage of European tourists in Brazil that make at least a stop in a nature reserve, where he/she does at least one finer stop in a handcraft shop.*

These queries can be expressed in languages like SPARQL[1], and its extensions with spatial operators, such as GeoSPARQL [20] and ST-SPARQL [21], among other alternatives. The viability of the proposal have been investigated in case studies using movement data extracted from Flickr[2] and Twitter[3], and semantically enriched with labeled geo-referenced places of several subclasses taken from DBPedia[4] [22] and LinkedGeoData[5] [23].

The rest of this article is organized as follows. Section 2 defines general constructs for structuring and abstracting movement data. Section 3 describes the *Baquara*[2] upper ontology, that provides conceptual support to annotate and query movement data according to a variety of semantic description facets. Section 4 describes a basic semantic enrichment process, a customized algorithm to semantically annotate movement data having associated text with visited PoIs, and the *Baquara*[2] general architecture to semantically enrich and analyze movement data using ontologies and LOD. Section 5 reports experiments that apply our proposal to semantically enrich and analyze movement data collected from social media with geo-referenced LOD. Finally, Section 6 discusses related work, and Section 7 summarizes our contributions and future work.

2. General Structures and Abstractions for Movement Data

This section defines general concepts necessary to understand, structure and abstract movement data in several refinement levels. First, we call *movement data* any collection of spatio-temporal data representing sampled or inferred positions of moving objects. This informal definition encompasses, among many other things, moving objects' trajectories, and temporally ordered sequences of social media users' geo-referenced posts. A *raw trajectory* (or just trajectory, for simplicity) is a temporally ordered sequence of spatio-temporal positions occupied by a moving object. It is possible to get accurate trajectories, by using state-of-the-art sensors and fine sampling rates (e.g., every second, every 3 meters). However, it is hard to gather large volumes of annotated trajectories, because annotating them is a laborious task [14, 24, 25, 26]. Nevertheless, some social media posts (e.g., geo-referenced tweets) can come with spatio-temporal coordinates and usually have plenty of attached textual contents (e.g., text, hash tags). These contents can be regarded as textual annotations that may provide hints to explain movements. A *system user's spatio-temporal trail* (or simply *trail*) is a temporally ordered sequence of geo-refferenced registries of interactions of a user with a particular system (e.g., Twitter, FourSquare, Facebook, Flickr, Instagran), with spatio-temporal coordinates and associated contents (e.g., tweet(s) text). Differently from trajectories, social media users' trails are usually sparse, due to the asynchronous nature of the users' posts, and

[1] http://www.w3.org/TR/sparql11-query
[2] https://www.flickr.com
[3] https://twitter.com
[4] http://dbpedia.org
[5] http://linkedgeodata.org

usually are less precise than trajectories, due to limitations of their gathering processes or other restrictions. Trajectories and trails can also be combined (e.g., spatio-temporally fused) to exploit the best traits of each one of these categories of movement data [27]. Combinations and derivatives of different kinds of movement data can also be regarded as movement data, provided that they still represent (changes of) positions of moving objects.

A moving object's positions sequence (MOPS) represents the known movement history of an object (e.g., a car monitored by GPS, a social media user) during a certain period of time as a temporally ordered sequence of spatio-temporal positions occupied by this object. A movement data segment or simply movement segment (MS) is an abstraction that refers to any continuous subsequence of a MOPS. These concepts generalize notions like social media user's trails (time-ordered sequences of posts), trajectories, and episodes (e.g., stops, moves). A MOPS can be successively segmented in several levels of detail. The MSs referring to successively smaller segments of a MOPS can be organized in a hierarchy with many refinement levels for information analyses purposes. Annotations can be associated with a MOPS, MS, or known spatio-temporal positions of a moving object to help describe its movements. Annotations of movement data and movement patterns are formally defined in Section 3.2.

The first step for movement analysis is to collect and time-order spatio-temporal positions of each moving object in a MOPS (Definition 1).

Definition 1. A **moving object's positions sequence (MOPS)** is a tuple:

$$mops = (idMO, PS, A)$$

where:

- $idMO$ is the unique identifier of a moving object;
- $PS = \langle p_1, \ldots, p_n \rangle$ is a time ordered sequence of spatio-temporal positions of the moving object identified by $idMO$;
- A is a set of annotations associated with the whole $mops$.

Each position p_i of PS $(i, n \in \mathbb{N}; 1 \leq i \leq n; n \geq 1)$ is a tuple of the form:

$$p_i = (i, geom, t, A_i)$$

where:

- i is the temporal order of the position p_i in PS;
- $geom$ is a geometry that represents the moving object identified by $idMO$ during the time t;
- $t = [begin, end]$, with $begin_i$ and end being instants of time and $begin \leq end$, is the time interval ($begin < end$) or instant ($begin = end$) when the position of the moving object identified by $idMO$ is represented in space by the geometry $geom$;
- A_i is a set of annotations associated with p_i.

An *idMO* identifies a moving object in a given data source. A moving object can sometimes be decomposed in a relevant moving entity (e.g., instance of person, animal, or vehicle) and a data gathering device (e.g., cell phone, GPS navigator) used to tack the moving entity positions. The entity id (possibly fake, for privacy reasons) of the same real world entity, can be different in different data sources. For example, though the same person can post in different social media systems (e.g., Twitter, Flickr, Facebook), her id can be different in each system. The same can happen to a device. In addition, a moving entity can hold several devices, and the same device may be held by different users at different times. Consequently, the movement data of the same real world entity and/or device taken from different sources is separated in distinct MOPS (each one with a different idMO). Identifying if movement positions coming from different data sources pertain to the same moving entity and/or device is beyond the scope of this work. This problem has been addressed as data fusion in [27].

A position p_i of a MOPS represents its location and shape in a given time. If the moving object shape can be neglected due to its small size compared to the space where it moves (e.g., a person or a car moving in a city), the geometry $p_i.geom$ can be a point. Otherwise, it can be a polygon or multi-polygon (e.g., representing a moving storm). The positions of a MOPS must be totally ordered by their respective times, and their times cannot overlap, i.e., the following constraint must apply to the positions sequence PS of any MOPS:

$$\forall p_i, p_{i+1} \in PS : p_i.t[end] < p_{i+1}.t[begin]$$

A MOPS can be segmented for information analysis purposes by using a variety of methods proposed in the literature to produce subsequences such as trajectories and episodes [14, 13, 11]. Although a structured trajectory can be defined as a time-ordered sequence of episodes [14], both trajectories and episodes refer to subsequences of movement positions satisfying particular predicates. For instance, the segmentation of a MOPS into trajectories can be determined according to constraints on time (e.g., a trajectory per day), space (e.g., segment according to some geographic boundaries or after reaching certain traveled distances), or both (e.g., segment every time there is a sampling gap or stop lasting longer than a given threshold) [28]. Episodes, such as *stops*/*moves*, on the other hand, can be determined by predicates like "speed below/above a certain threshold" [29] or "moving object inside a given region for a time period longer/shorter than a certain threshold" [30, 31]. In this work, we propose a generalized concept for constructs such as trajectories and episodes called movement segment (MS).

An MS (Definition 2) is an abstraction for a portion of the movement history of a moving object identified by *idMO*. Possible values for an MS' `type` include: `TRAIL` (time ordered sequence of social media user's posts), `TRAJ` (trajectory), `STOP` episode, and `MOVE` episode. An MS is associated to a subsequence of spatio-temporal positions $\langle p_i, \ldots, p_j \rangle$ of a positions sequence $mops.PS = \langle p_1, \ldots, p_n \rangle$. However, it can abstract these positions. The geometry $ms.geom$ of the MS ms is an approximation of the movement portion that ms refers to. For example, a

stop can be represented in the space by the centroid of the spatial coordinates of its constituent points, while a move can be represented by line segments. The time span $ms.ts$ temporally fits all the positions associated to ms.

Definition 2. A **movement segment (MS)** of a moving object's positions sequence $mops = (idMO, PS, A)$ is a tuple of the form:

$$ms = (idMO, idMS, type, geom, p_i, p_j, ts, father, level, prev, next, ord, A_{ms})$$

where:

$idMO$ is a moving object unique identifier;

$idMS$ is the unique identifier of ms;

$type$ is the type of ms;

$geom$ is the geometry used to abstractly represent ms in the space;

p_i, p_j are respectively the initial and final positions of the corresponding subsequence $\langle p_i, \ldots, p_j \rangle$ of the time ordered positions sequence $\langle p_1, \ldots, p_n \rangle$ of the moving object identified by $idMO$ $(i, j, n \in N; n \geq 1; 1 \leq i \leq j \leq n)$;

$ts = [b, e]$ $(b = p_i.t[begin], e = p_j.t[end])$ is time span of ms;

$father$ is the shortest movement segment of $idMO$ such that $ms.father \neq NULL \rightarrow ms.ts \subset ms.father.ts$;

$level$ is the distance of ms to its ancestry root, i.e., $ms.level = 0$ if $ms.father = NULL$, otherwise $ms.level = 1 + ms.father.level$;

$prev$ is the chronologically closest previous sibling of ms;

$next$ is the chronologically closest next sibling of ms;

ord is the distance of ms to its sibling that happened first in time plus 1, i.e., $ms.order = 1$ if $ms.prev = NULL$, otherwise $ms.order = 1 + ms.prev.order$;

A_{ms} is a possibly empty set of annotations associated to ms.

Sibling movement segments are those that have the same father, i.e., two movement segments ms and ms' are siblings if $ms.father = ms'.father$. The set of predecessors of a movement segment ms is given by the transitive closure $predecessors(ms) = prev(ms) \cup predecessors(ms.prev)$, with the stop condition $predecessors(NULL) = \emptyset$, and the set of successors of ms by the transitive closure $successors(ms) = next(ms) \cup successors(ms.next)$, with the stop condition $successors(NULL) = \emptyset$. The set of siblings of ms is $siblings(ms) = predecessors(ms) \cup successors(ms)$.

The *children* of a movement segment ms are the movement segments having ms as $father$, i.e., $children(ms) = \{ms' \mid ms'.father = ms.idMS\}$. The set of descendants of ms is given by the transitive closure $descendants(ms) = children(ms) \cup \{\cup_{ms' \in children(ms)} ms'\}$. The set of ancestors of ms is given by

the transitive closure $ancestors(ms) = ms.father \cup ancestors(ms.father)$ with $ancestors(NULL) = \emptyset$. The set of movement segments in the lineage of ms is $lineage(ms) = ancestors(ms) \cup descendants(ms)$.

Notice that, according to Definition 2, for any movement segment ms its time span $ms.ts$ must be contained in that of $ms.father$ if $ms.father \neq NULL$. In addition, sibling movement segments do not overlap in time, i.e.:

$$\forall ms', ms'' \in children(ms) : ms'.ts \cap ms''.ts = \emptyset$$

These restrictions ensure that the time span of an MS covers the time spans of all its descendants, and that sibling movement segments are always organized in a consistent total ordering in time, in the sense that no segment begins before its previous one finishes. Thus, movement segments can be arranged in a tree-like hierarchy to support information analysis at different levels of detail, determined by their time span, as stated by Definition 3.

Definition 3. A **movement segments hierarchy (MSH)** for a MOPS denoted by *mops* is a tree denoted by *msh* such that:

1. each node of the *msh* is a movement segment of *mops*;
2. the father of the unique root node of *msh* is $NULL$.

MSHs can support semantic analysis of movement data in different levels of detail. For example, at the root of the hierarchy a MOPS *mops* can be regarded as a sequence of semantic trajectories [14]. Each semantic trajectory can be refined in the next level by a sequence of episodes, each one referring to subsegments of a semantic trajectory that satisfy some kind of predicate (e.g., stops inside cities and moves between them). Then stops in each city can be further refined in finer stops (e.g., in places like an airport, a university, a shopping mall) and moves between such smaller stops that refine bigger stops in cities. Finally, stops in relatively big places of cities (e.g., a university or a marketplace), can be further segmented in lower level stops in smaller places (e.g., particular departments of the university or shops of the marketplace) and moves between such finest stops.

Figure 1 illustrates a hierarchy of movement segments. At the top level the corresponding MOPS is segmented and abstracted in sequences of trajectories inside big countries or world regions, such as Brazil, the US, and the EU. These trajectories are further segmented in stops and moves progressively detailed in the lower levels of the hierarchy. Stops are represented by circles, and moves by dashed lines between them, with arrows indicating the movement direction. Many of these movement segments have associated annotations, which begin with the sign @. In this example, the annotations indicate the places where trajectories or stops occur, and the transportation means of some moves. The portion of the second hierarchical level presented in Figure 1 details `Trajectory 3` by showing stops in some Brazilian cities and moves between them. The third hierarchical level details `stop 3.1` in `Rio`, with stops and moves in smaller places that are inside `Rio` , including `stop 3.1.1` at the `GIG airport`, followed

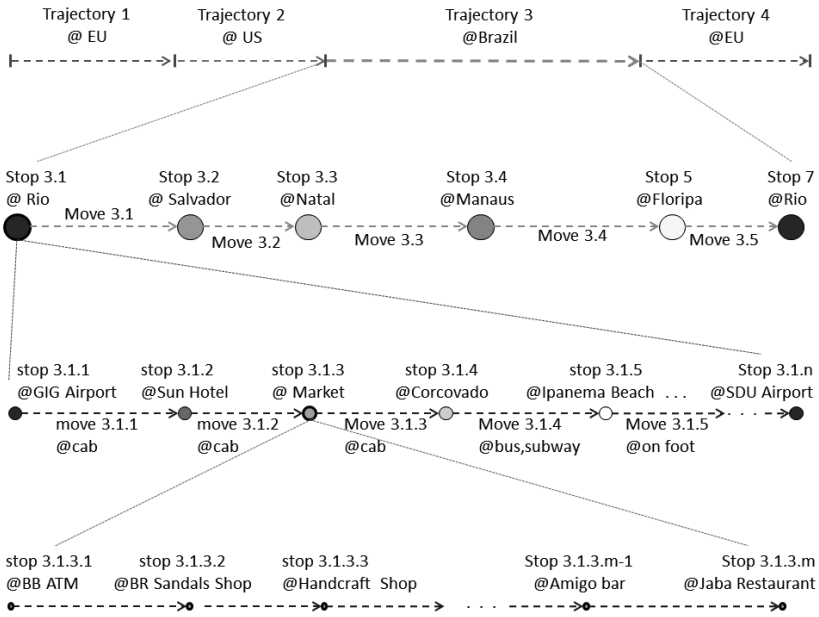


Figure 1: A movement segments hierarchy example

by stops at `Sun hotel`, `SC market`, `Corcovado`, `Ipanema Beach`, and so on. Finally, in the lowest level some details of the movement inside `SC Market` are presented, including `stop 3.1.3.1` at `BB ATM`, stops in some shops, and so on.

3. The *Baquara*^{2} ontology

The conceptual modeling core of our approach for semantic enrichment and analysis of movement data is the *Baquara*^{2} upper ontology. It has been designed to serve as a conceptual framework for describing movement segments in several application domains, ranging from urban transportation to animal ecology. Such adaptation can be done by specializing some of its pre-defined classes, and by creating new relationships among them.

Figure 2 shows the high level concepts (classes) of *Baquara*^{2} ontology, and the major semantic relationships between them. It has been produced by using the Protegé[6] ontology editor version 3.4 with the Jambalaya[7] graphic editor plugin. Each labeled rectangle represents a concept. Nesting denotes subsumption (`IS_A` relationship), i.e., each nested concept is a subclass of its enclosing concept. For instance `Episode IS_A MovementSegment`. The plus sign on the top left corner of a rectangle indicates that the respective concept can be

[6] http:/protege.stanford.edu
[7] http://protegewiki.stanford.edu/wiki/Jambalaya

further specialized, according to application domains and analyses needs. A dashed line between concepts denotes a semantic relationship, such as composition (`PART_OF`) or a specific relationship (e.g., between an `Event` and a `Place` where it occurs).

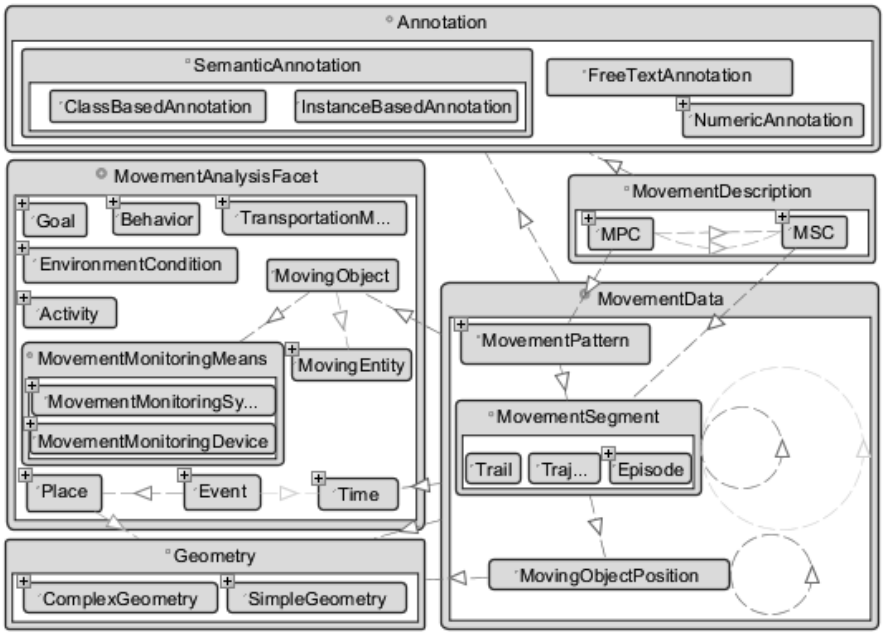


Figure 2: The backbone of the $Baquara^2$ ontology

The five main concepts of the $Baquara^2$ ontology are:

- `MovementData` that can be a `MovingObjectPosition`, a `MovementSegment`, or a `MovementPattern`;
- `Geometry` that can be a `SimpleGeometry` (e.g. a spatio-temporal point, a line segment) or a `ComplexGeometry` (e.g., a sequence of points, a sequence of line segments, or a sequence of line segments and points);
- `Annotation`, such as a `SemanticAnnotation`, a `FreeTextAnnotation`, or a `NumericAnnotation`, used to describe some `MovementData`;
- `MovementAnalysisFacet`, such as `Goal` or `TransportationMeans`, that semantically organize (with semantic relationships such as `IS_A`, `PART_OF`, or even domain specific relations) concepts and objects that can be referenced in semantic annotations;

- `MovementDescription` of movement patterns (`MPC - Movement Pattern Category`) or movement segments (`MSC - Movement Segment Category`) that use annotations to depict their common properties (e.g., moves of tourists by bus to visit historic places).

Geometries are well studied in spatial databases and geographic information systems [32]. Movement positions and movement segments have been defined in the previous section. Then, the following subsections describe in more detail the other constituents of the *Baquara*^{2} ontology, which are crucial for the semantic lifting with domain specific knowledge.

3.1. Movement Analysis Facets

Movement segments and other abstractions for describing movements like some movement patterns (Section 3.3) can be semantically annotated by linking them to concepts (classes) and/or objects (instances of classes) of semantic facets for movement description and analysis (Definition 4).

Definition 4. A **semantic facet** is a graph $G(V, E)$, where:

V is a set of resources, each one referring to a concept (class) or object (instance of a class);

E is a set of semantic relationships between resources of V.

Facets describe information and knowledge about relevant themes for movement analysis. Each facet has an intentional level (TBox) and an extensional level (ABox) based on description logics [33]. The first has at least one conceptualization hierarchy (classes organized according to their `IS_A` relationships), and the latter at least one objects hierarchy (instances that can be organized by some partial ordering relationship, such as `PART_OF`). Schemata and examples of facets and their derived dimensions for movement analysis in data warehouses can be found in [34]. Besides the *Baquara* pre-defined facets described in the following, that work also proposes facets based movement movement segments hierarchies, movement patterns hierarchies, and subsumption hierarchies of their respective categories.

The *Baquara*^{2} pre-defined facets cover those of the CONSTAnT model [16], namely: `Space` (e.g., `Place`), `Time`, `Goal` (e.g., `Eat`, `Watch a soccer game`), `Behavior` (e.g., `Flock` [35], `Chasing` [36], `Avoidance` [37]), `Transportation-Means`, `EnvironmentCondition` (e.g., `Windy, Sunny`), `Activity` (e.g., `Running`), and `MovingObject`. Baquara also includes the facet `Event` (e.g., `CulturalEvent`, `SportEvent`), and allows adding new specific facets for movement analysis in particular application domains.

A `MovingObject` is described in Baquara as an association between one `MovingEntity` (e.g., `Person`, `Animal`) and a `MovementMonitoringMeans`, that can be specialized in a `MovementMonitoringSystem` (e.g., `SocialMedia`), or a `MovementMonitoringDevice` (e.g., `CellPhone`), as discussed in Section 2.

Baquara[2] employs the OGCs Geospatial Features Model[8] and the W3Cs Time Ontology[9] as foundations to describe space (places and their relationships) and time (instants, periods of time, and their relationships), respectively. Their concepts and instances are used to describe the spatio-temporal scope of movement, as well as the places, times and events of interest for movement analysis. A place is a spatial feature relevant for movement data description and/or movement analysis. It can be anything with a geometry, and at least a name. A places geometry, represented in some coordinate system, can be simple (point, line, or region) or complex (set of points, lines, and/or regions). Specializations of place relevant for the tourism domain may, for example, include `Country`, `City`, `Airport`, `Hotel`, and `Cafe`. A time can be an instant or a period of time.

An event is any circumstance relevant for movement analysis in an given domain. It has at least one label, occurs in a time (instant or period), and may have relationship(s) with some place(s). For instance, `SportEvent` and `CulturalEvent` are subclasses of `Event` relevant for the tourism domain. A `CulturalEvent` can be specialized to `MusicFestival`, `DanceFestival`, etc. Conversely, `Carnival` can be regarded as a subclass of `TraditionalParty`. Instances of event (and its subclasses in any abstraction level) can be related to specific instances of place and time, as indicated by the dashed line linking these classes in the left bottom portion of Figure 2. For instance, the city `Rio` can be semantically related to events occurring there, such as `Pan American Games 2007` and `Word Cup Final 2014`. On the other hand, events like `Christmas holidays` may not be associated with any particular place, because they occur in many places. Remember that a facet represents descriptive information for movement analysis in the extensional level (e.g., `Maracanã`, `July 13 2014 5:00 PM - 8:00 PM`, `watch World Cup 2014 Final`), and the intentional level (e.g., `Stadium`, `Sunday evening`, `watch soccer game`).

Facets are crucial to semantically organize the resources (concepts or instances of concepts) used to describe movement segments in the semantic annotations defined in Section 3.2. They express semantic relationships between resources of knowledge bases (e.g., ontologies, LOD collection) used to semantically annotate movement data. Extracts of facets relevant for each particular application can be used as dimensions for movement analysis. Section 5.3 and [38] provide examples of SPARQL and geoSPARQL queries that exploit semantic annotations for movement analysis. Queries on a multidimensional model whose dimensions are extracts of facets be found in [34]. That work exploits facets based on movement movement segments hierarchies, movement patterns hierarchies, and subsumption hierarchies of their respective categories to support simpler and more efficient query expressions than the ones referring just to facets like the ones described in this paper. The improved queries efficiency is paid with pre-processing during the Extraction, Transformation, and Load (ETL) process of a movement data warehouse. Finally, a method to derive di-

[8] http://www.opengeospatial.org/standards/sfa
[9] http://www.w3.org/TR/owl-time

mensions tailored for particular sets of semantically annotated movement data by exploiting properties of currently available LOD collections used for annotating the respective movement data sets is presented in [19].

3.2. Movement Annotations

A set of annotations can be associated with movement segments to describe what is going on (e.g., place or event of interest, goal, environmental conditions), as stated by Definition 5.

Definition 5. An **annotation** is a triple of the form:

$$annot = (target, property, value)$$

where:

target is the annotated thing;

property is a descriptive property defined for instances of *target*;

value is a typed literal (e.g., text, number) or a reference to a resource (object or concept) described in a knowledge base.

The *target* is a reference to a movement object positions sequence (MOPS), a movement segment (MS), or a specific moving object position (defined in Section 2). The *target* can also be a movement pattern or a movement abstract description (defined in Section 3.3). The property indicates a relation of the *target* with the annotation value. For example, the property *occursAt* indicates that a movement segment (e.g., a stop) occurs at a particular place denoted by the value of this property.

An annotation can be free or semantic. The value of a free annotation is a literal, such as a string, free text, or a number. Thus, it may not have precise semantics. The value of a semantic annotation, on the other hand, must be a reference to resource, i.e., a concept (class) or an object (instance of a class) described in a description logics compatible knowledge base, to better describe its semantics [39]. In Baquara each semantic annotation is a reference to an object or a concept of a movement analysis facet described in Section 3.1.

3.3. Movement Patterns and Abstract Movement Descriptions

The *Baquara*[2] upper ontology also provides constructs to express movement patterns, and abstract movement descriptions (movement segment categories, and movement pattern categories). A **movement pattern (MP)** is as a collection of movement segments that satisfies some predicate, based on: spatio-temporal constraints of movement segments and/or semantic, ordering, and timing constraints on related segments. Examples of spatio-temporal constraints include moving clusters and meetings [36]). Semantic constraints are expressed by the type of the movement segments and their associated semantic annotations. Ordering constraints refer to the exact or relative order of movement segments among their siblings or lineage. Timing constraints refer to the duration of some movement segment(s) or the elapsed time between them.

A **Movement Segment Category (MSC)** is an abstract description for movement segments (MS). One MSC can be represented with by a tuple analogous to the one proposed to represent an MS in Definition 2, but without any geometry (*geom*) or time span (*ts*), minimum and maximum allowed duration instead of a single exact duration (*dur*), the possibility of relative instead of absolute sibling order, and the possibility of any relatives instead of just immediate relatives (*predecessors* instead of just *prev*, *successors* instead of just *next*, *ancestors* instead of $father$, and many *children*) always referring to other MSCs instead of MSs.

A **Movement Pattern Category (MPC)** is an abstract description movement patterns (MP). One MPC can be expressed by a reference to an MSC and its related MSCs, along with possible (partial) ordering and/or timming restrictions. For example, any collection of segments $S = ms_i$ such that: (i) ms is a `Stop` of an `EuropeanPerson` in `Market`; (i) ms_i takes part in a meeting pattern; (ii) ms_i is preceded by at least a sibling `Stop` in an `Airport` and another one in a `Hotel`; (iii) ms_i is followed by a a sibling `Stop` in a `TouristicPlace` called `Corcovado`; and (iv) ms_i is detailed in a number of shorter stops, being at least one of them in a `Bar` for at most 1 hour, which is immediately followed by another short `Stop` in a `Restaurant`. Notice that *stop*3.1.3 presented in Figure 1 satisfies all these semantic and ordering constraints, and maybe the spatio-temporal constraint (take part in a meeting pattern) and the time constraint of the short stop in a `Bar` for at most 1 hour as well.

Abstract movement descriptions (MSCs and MPCs) semantically describe movement by relaying on annotations, without referring to any concrete movement segment of any existent moving object. However, many concrete movement segments and movement patterns can semantically match such a description. More formal definitions for restricted MSCs and MPCs, with detailed data structures for representing them can be found in [34], along with examples of their use for movement analysis. Generalized formal definitions for these abstract movement descriptions and the investigation of semantic consistency rules among them, and the semantic matching concrete movement segments and movement patterns with their respective categories are out of the scope of this article, and left to future work.

4. Semantic Enrichment and Analysis of Movement Data

This section describes a customizable process to semantically enrich movement data by using a description logics compatible knowledge base (KB), which can be built with (portions of) domain ontologies and LOD collections. The semantic enrichment process takes movement data (either raw or structured) that can come associated with text, as described in Section 2, and connects it to knowledge base resources for producing semantic annotations of movement data compliant with the *Baquara*[2] upper ontology described in Section 3. The movement data structuring in movement segments hierarchies and the construction of movement analysis facets of KB resources (concepts or objects) and

their semantic relationships can take place before, during, or after this semantic enrichment process. These issues are beyond the scope of this article, and addressed in other works [34, 19].

4.1. A General Process for Semantic Enrichment

Figure 3 illustrates the inputs, outputs, stages, and major usual tasks of the proposed process to semantically enrich movement data with domain knowledge. The inputs are: (i) movement data (e.g., movement segments or individual positions of moving objects) annotated with text (e.g., social media posts textual contents, text associated with individual positions or episodes of a trajectory); and (ii) resources that can be taken from domain ontologies and LOD collections (e.g., DBpedia, LinkedGeoData), and organized in a KB with a variety of semantic facets as those described in Section 3.1 (place, time, goal, etc.). The outputs are semantically enriched movement data, i.e., movement data with semantic annotations that refer to resources of particular semantic facets used to describe and analyze movement data. Such a resource can be a concept (e.g., bar, restaurant) or an object (instance of such a concept).

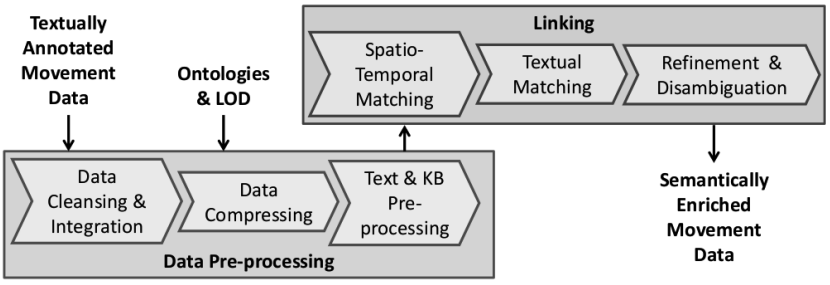


Figure 3: General process to semantically enrich movement data

The proposed semantic enrichment process can be realized in two stages: first the Data Pre-processing and then the Linking. Each stage includes tasks that are not rigid, but may be chosen and adapted for particular needs. These tasks are customizable in terms of the techniques used to accomplish them, parameters tunning, and even the existence of particular tasks and their relative order. Their customization can be done according to the characteristics of the data and knowledge provided as inputs, and the enrichment and analysis purposes. The execution of distinct tasks in each stage can be intertwined in practice.

The Data Pre-processing stage typically includes the tasks: Data Cleansing & Integration that filters out invalid data (e.g., outliers) and sometimes integrates data obtained from different sources (e.g., social media posts of different systems, social media posts with trajectories [27], LOD of different collections); Data & Compressing that can compact data for speeding up the following tasks, by using a variety of techniques such as those described in [13, 14, 11]; and Text & KB Pre-processing that prepares the textual data and KB resources for the linking

stage, by applying techniques such as textual contents filtering, classification, stemming, and named entities recognition [40, 41, 42, 43, 44].

Then, the Linking stage connects the movement data to the KB resources to generate semantic annotations. The specific tasks and techniques employed for solving this problem vary with the nature of the movement description facet. For example, Spatio-Temporal Matching is a crucial task to link movement data to facets like space and events, by taking into account the geographic extensions of places that are visited or where events take place, along with their operation times. In this case, Textual Matching may help to refine the matchings, before applying the final task of Refinement & Disambiguation. Entity linking [45, 44, 46] and other techniques can also be used in this stage.

Algorithm 1 is a customization of the proposed semantic enrichment process

Algorithm 1: Link movement segments to co-located resources

input : $S = \{s_0, ..., s_n\}$; // Pre-processed movement segments
$R = \{r_0, ..., r_m\}$; // Pre-processed resources set
$\tau_s \in \mathbb{R}^+$; // Spatial distance threshold in meters
$\tau_t \in \mathbb{R}^+$; // Textual similarity threshold

output: SA; // Semantic annotations of movement segments in S

```
1  begin
2      SA ← ∅;   // Semantic Annotations (SA) set initially empty
3      SJ ← (Π_{s←S.*, r←R.*, geoDist}(S ⋈_{(geoDist←dist(s.geom,r.geom))≤τ_s} R));
4      foreach s ∈ S do
5          k ← 0;          // Initialize best matching measures for s
6          minDist ← τ_s;
7          maxSim ← τ_t;
8          foreach (s, r, geoDist) ∈ SJ do
9              if geoDist ≤ minDist then
10                 textSim ← textualSimilarity(s.ppText, r.ppText);
11                 if textSim ≥ maxSim then
12                     if geoDist < minDist ∨ textSim > maxSim then
13                         k ← 0;    // Better matching resource r found
14                         minDist ← geoDist;
15                         maxSim ← textSim;
16                     k++;              // Increment number of matchings
17                     bestMatching[k] ← r;              // Add matching r
18         while k > 0 do
19             k−−;      // Create semantic annotations for segment s
20             SA ← SA ∪ (s, visits, bestMatching[k]);
21     return SA;
```

for linking movement segments that are individual positions or stops to PoIs. It exploits spatial proximity and textual similarity [47, 48, 49] to match movement segments with geo-referenced KB resources, and generate semantic annotations for the former linked to the latter. It employs the thresholds $\tau_s \in [0, \infty]$ and $\tau_t \in [0, 1]$ ($\tau_s, \tau_t \in R$) to filter the resources most likely to match each movement segment according to some geographic distance function, and some text similarity function, respectively. The spatial join of line 3) returns the pairs of movement segment (s) and resource (r) that are closer than τ_s along with the distance between them. Then, for each segment $s \in S$ the algorithm looks for the pairs (s,r) that are the closest in space, and among them those whose textual similarity is the highest (lines 4 to 17). Finally, the set $bestMatching$ of resources satisfying these conditions with respect to segment s is used to create the semantic annotations for s (lines 18 to 20).

In this version, Algorithm 1 generates semantic annotations of segments with the property *visits*. The investigation of methods to generate annotations for other properties (e.g., $comesFrom$ and $goesTo$ for moves, $hasGoal$ for stops and moves) is theme for future work. Nevertheless, notice that a number of optimizations and customizations can be easily done in this algorithm. For example, if tuples resulting of the spatial join of line 3 are grouped according to s, and the tuples for each s are put in ascending order of $geoDist$, then the second loop can be interrupted when $geoDist > minDist$. In addition, line 9 can be pushed down to be executed after the two line bellow it, to semantically annotate each movement segment s with the textually most similar resource(s) that are the closest to s. This algorithm can also be adapted to connected movement segments with PoI that are just mentioned or events that are visited or mentioned in the movement data associated with text. Recently proposed spatio-textual similarity joins [50, 51] can also be considered to speed-up the data processing. In addition, a variety of entity linking techniques [40, 45, 44, 46] can be employed to better refine and disambiguate generated links from movement segments to KB resources.

4.2. General Systems Architecture for Semantic Enrichment and Analysis

Figure 4 illustrates a general system architecture to realize the proposed approach for semantic enrichment and analyses of movement data. Firstly, the $Baquara^2$ ontology (or another one serving as a conceptual model for movement data enrichment and analysis) must be loaded in a KB handled by a Knowledge Management System. Secondly, domain specific knowledge (e.g., ontologies and LOD) with the same spatio-temporal scope as the movement data to be enriched and analyzed must be selected, and customized if necessary. The domain knowledge selection and customization can be done, for example, by using SPARQL endpoints or REST APIs of a variety of LOD collections available on the Web. Thirdly, the Semantic Enrichment Process described in Section 4.1 must be executed to generate the semantic annotations for the movement data using the collected domain specific knowledge.

The resulting semantically enriched movement data can be represented as a collection of RDF/RDFS triples, and maintained for Querying & Reasoning

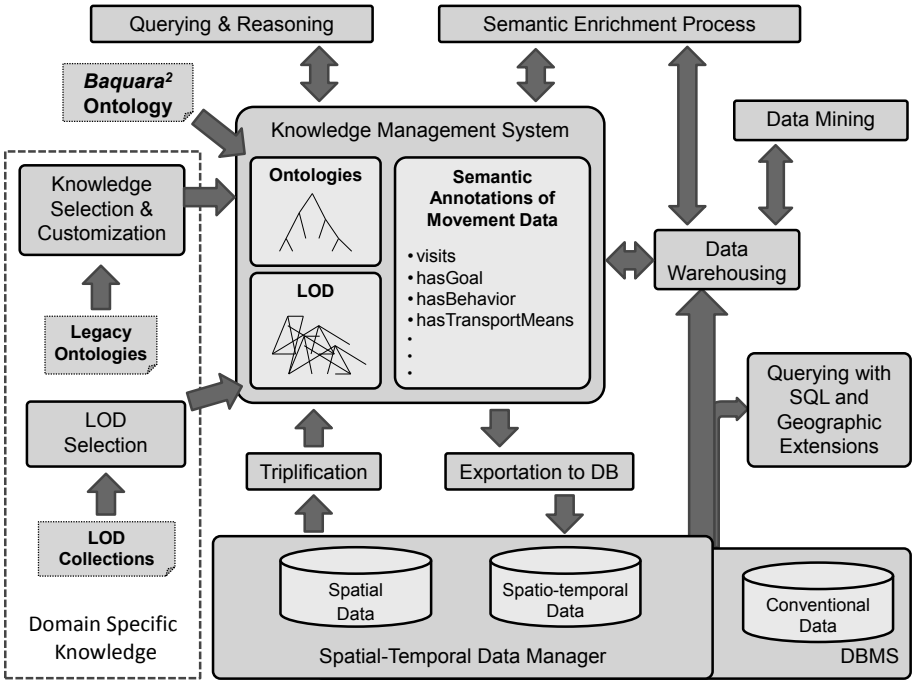


Figure 4: General architecture for semantic enrichment and analysis of movement data

in the KB. This KB can be queried by using languages such as SPARQL and geoSPARQL. It can also be analyzed or further enriched by using a variety of reasoners that can be connected to the KB. The general architecture of Figure 4 also includes a conventional (e.g., relational) Database Management System (DBMS) and a Spatio-Temporal Data Manager. Triplification can be used to convert spatio-temporal data (e.g., movement data) from these systems into RDF triples, for allowing their processing in the KB whenever necessary or convenient. Conversely, Exportation to DB can be used to convert RDF triples of the KB into conventional, spatial, and spatio-temporal data managed by systems that efficiently support Querying with SQL and Geographic Extensions, Data Warehousing, and Data Mining. This architecture allows data processing in the KB and/or conventional and spatio-temporal database, to render flexibility for the semantic enrichment process, as well as for querying, reasoning, data warehousing, and data mining with the semantically enriched movement data.

5. Experiments

The viability of the proposed approach to semantically enrich and analyze movement data has been investigated in two case studies, using data extracted from Flickr and Twitter, respectively, and LOD from DBpedia and LinkedGeoData. Specializations of the semantic enrichment process described in Sec-

tion 4.1 have been implemented on PostGIS[10], and applied to derive semantic annotations for stops and individual positions of moving objects. The resulting semantically annotated movement data supports queries in SQL with geographic extensions on PostGIS. They can also be converted into RDF triples, and stored in a KB compliant with the *Baquara*[2] ontology, to be queried with SPARQL and geoSPARQL, among other possibilities.

5.1. Flickr

Sample Flickr data for experiments were extracted from CoPhIR[11], by filtering tuples with spatio-temporal points inside Brazil, and eliminating moving objects positions sequences (MOPS) having any subsequence with speed higher than 500 km/h. The resulting raw data collection, that consists in 14,504 positions of 564 distinct users, was segmented in 2,143 user's trails (by just breaking each MOPS in a daily basis, as more sophisticated methods for trajectories reconstructing are beyond the scope of this paper). The positions are associated with 12,443 distinct tags. The total number of textual annotations is 117,146. Thus, each spatio-temporal sample point is associated to 8.08 tags in average. The user's daily trails were further segmented in 971 stops, each one corresponding to a period of at least 30 minutes without moving more than 500 meters. These stops are associated to 6,278 distinct tags, in a total of 45,768 (stop,tag) pairs, i.e., around 47 different tag values associated to each stop, in average. Figure 5 illustrates the distribution of the obtained trails across Brazil (left side), and the tags associated to a particular stop in the city of Rio. We have verified by visual inspection that the (sometimes inaccurate) coordinates usually refer to the position of the user when taking the picture, but sometimes to the pictured object itself.

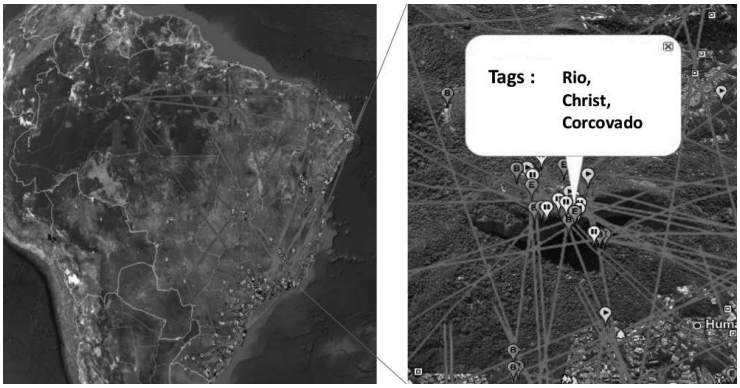


Figure 5: CoPhIR Flickr trails across Brazil (left), and close to Corcovado in Rio (right)

[10] http://postgis.net/
[11] http://cophir.isti.cnr.it

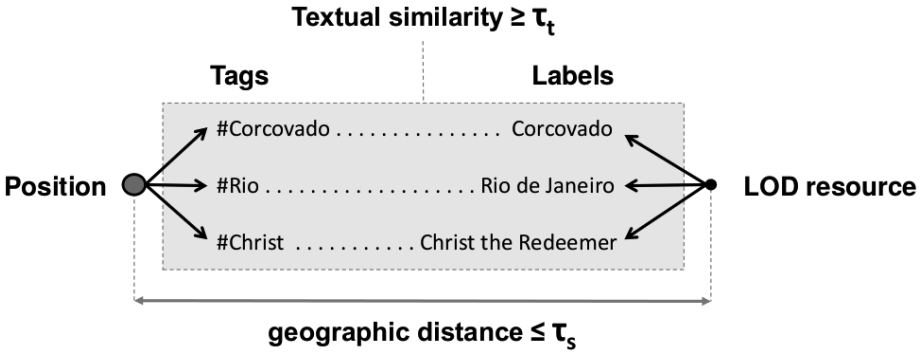


Figure 6: Linking a Flickr user tagged position to a geo-referenced labeled LOD resource

The Flickr users' positions have been semantically enriched with 97,242 resources of DBpedia and LinkedGeoData associated to geographic coordinates inside an MBR fitting the Brazilian territory, and having at least a label. The linking of these resources with Flickr positions was done with a variation of Algorithm 1 that employs Euclidean distance as geographic distance with a threshold $\tau_s = 1$ km to match the positions. For the pairs position-resource within 1 kilometer from each other, the set of tags associated to the position is then compared with the set of labels of the resource, by using soft-TFIDF [49] to compose Jaro-Winkler [48] similarities of pairs (tag,label) that are above the textual threshold τ_t in a unique similarity measure for the respective pair (position,resource). Figure 6 illustrates the linking criteria with the textual similarities considered between tags and labels of a pair (position,resource) referring to the place called `Corcovado` in the city of `Rio`. Finally, Figure 7 illustrates a situation in which the similarity between tags and labels is crucial to link the moving object's position to the correct resource. This position is in a densely populated area, and its spatial coordinates are not precise enough to decide what is the best matching among the many resources in the surroundings.

5.2. Twitter

Another variation of Algorithm 1 has been run on geo-referenced tweets whose geographic coordinates are inside an MBR fitting the Brazilian territory, which were collected during World Cup 2014 (6 June 2014 to 7 July 2014), via the Twitter API[12]. These 57,099,806 tweets were first filtered to take the ones originated from FourSquare, because they include mentions to named places where the FourSquare users have checked-in. Such names might be extracted by applying, for example, state-of-the-art named entity recognition (NER) methods [41, 42, 52] to textual contents of the tweets. However, these methods presented low precision and recall in our preliminary experiments. Further experiments

[12]https://dev.twitter.com/docs/api

Figure 7: Example of position-resource link enabled by Soft-TF-IDF textual similarity

with such methods on all collected tweets are left to future work. The 1,183,354 selected tweets have been compared with 97,242 LOD instances of LinkedGeoData whose coordinates are inside the same MBR fitting the Brazilian territory, in July 21 2014. The linking algorithm associates a tweet with a LinkedGeoData resource (e.g., an instance of shop) when the coordinates of the resource are up to τ_s meters away from the tweet, and the LinkedGeoData resource has at least a label value whose Jaro-Winkler similarity [49] with the name of the tweet location is higher than τ_t.

The first experiments with Twitter data just filtered the matchings satisfying the spatial and textual thresholds (τ_s and τ_t, respectively), i.e., without doing the refinements of lines 4 to 17 of Algorithm 1. Since the choice of algorithms and parameters setting are often crucial for performance, in the following we present and comment several results that help to understand how the change of algorithm details as well as spatial and textual parameters affects the number of resources that can potentially be linked, thus giving directions for properly choosing these values. Figure 8 shows the distribution of the percentage of the tweets associated with at least one LOD resource for distinct values of the thresholds τ_s and τ_t. The percentage of tweets with at least one resource in a radius of τ_s meters (column $\tau_t = 0$) jumps from 0.35% to $79,24\%$ by increasing τ_s from 1 to 1024 meters. Nevertheless, values of the textual threshold τ_t between 0.8 and 1 (range that properly filters similarities in our observations)

eliminate a considerable percentage of matchings. Figure 9 presents the average number of associated LOD resources per tweet for the same scales of values for τ_s and τ_t as those of Figure 8. Notice that $\tau_t \geq 0.8$ results in an average number of associations per tweet equal to 1 (in bold) or close to 1, meaning no or few ambiguities (the same movement segment associated to distinct LOD resources), respectively. However, for values of τ_s close to 1 km, almost half of the associations constitute ambiguities. On the other hand, filtering with low values for τ_s and high values for τ_t eliminates ambiguities, but may also eliminate a considerable number o valid matchings.

Spatial dist. threshold (τs : meters)	Textual similarity threshold (τt : Jaro-Winkler)											
	0	0.8	0.82	0.84	0.86	0.88	0.9	0.92	0.94	0.96	0.98	1
1	0.35%	0.18%	0.18%	0.18%	0.17%	0.14%	0.13%	0.13%	0.11%	0.09%	0.09%	0.09%
2	0.37%	0.19%	0.18%	0.18%	0.18%	0.15%	0.14%	0.13%	0.11%	0.10%	0.09%	0.09%
4	0.46%	0.21%	0.20%	0.20%	0.20%	0.16%	0.15%	0.15%	0.12%	0.11%	0.10%	0.10%
8	0.56%	0.24%	0.23%	0.23%	0.23%	0.19%	0.17%	0.16%	0.13%	0.12%	0.11%	0.11%
16	2.89%	1.17%	1.15%	1.11%	1.07%	0.98%	0.91%	0.84%	0.76%	0.71%	0.65%	0.62%
32	5.96%	2.14%	2.09%	2.04%	1.97%	1.82%	1.69%	1.56%	1.42%	1.33%	1.24%	1.18%
64	12.50%	3.78%	3.66%	3.56%	3.40%	3.18%	2.94%	2.73%	2.52%	2.37%	2.18%	2.10%
128	22.91%	5.14%	5.02%	4.88%	4.69%	4.39%	4.10%	3.71%	3.45%	3.19%	2.98%	2.88%
256	39.73%	6.62%	6.48%	6.27%	6.00%	5.56%	5.20%	4.69%	4.40%	4.11%	3.83%	3.72%
512	60.32%	8.84%	8.67%	8.32%	7.88%	7.34%	6.84%	6.17%	5.74%	5.41%	5.10%	4.97%
1024	79.24%	11.45%	11.20%	10.49%	9.78%	9.02%	8.42%	7.64%	7.15%	6.72%	6.33%	6.20%

Figure 8: Percentage of tweets associated with at least one LOD resource

Spatial dist. threshold (τs : meters)	Textual similarity threshold (τt : Jaro-Winkler)											
	0.00	0.80	0.82	0.84	0.86	0.88	0.90	0.92	0.94	0.96	0.98	1.00
1	1.01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
2	1.01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
4	1.08	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
8	1.08	1.00	1.00	1.00	1.00	1.00	1.00	1.01	**1.00**	**1.00**	**1.00**	**1.00**
16	1.18	1.01	1.01	1.01	1.01	1.01	1.01	1.00	1.00	1.00	1.00	1.00
32	1.37	1.03	1.03	1.04	1.03	1.04	1.04	1.04	1.02	1.01	1.01	1.01
64	1.91	1.05	1.05	1.04	1.04	1.04	1.04	1.04	1.02	1.01	1.02	1.02
128	2.86	1.12	1.11	1.11	1.10	1.10	1.10	1.11	1.09	1.07	1.06	1.06
256	4.60	1.19	1.19	1.18	1.17	1.16	1.16	1.17	1.16	1.11	1.11	1.11
512	8.79	1.30	1.30	1.27	1.24	1.22	1.20	1.21	1.20	1.17	1.16	1.17
1024	18.73	1.48	1.47	1.42	1.38	1.34	1.31	1.29	1.28	1.26	1.25	1.26

Figure 9: Average number of associated LOD resources per tweet

Figure 10 shows that the vast majority of the associated tweets are linked to just 1 LOD resource for $\tau_s = 16$ meters and $\tau_t = 0.9$ (left side), and that there are ambiguities in almost half of the associated tweets for $\tau_s = 1024$ meters and $\tau_t = 0.8$. It suggests that the textual similarity can play just a limited role on disambiguation. Notwithstanding, textual similarity can be crucial to make correct links, as in the scenario illustrated by Figure 11, in which the best matching resource is not the geographically closest to the tweet.

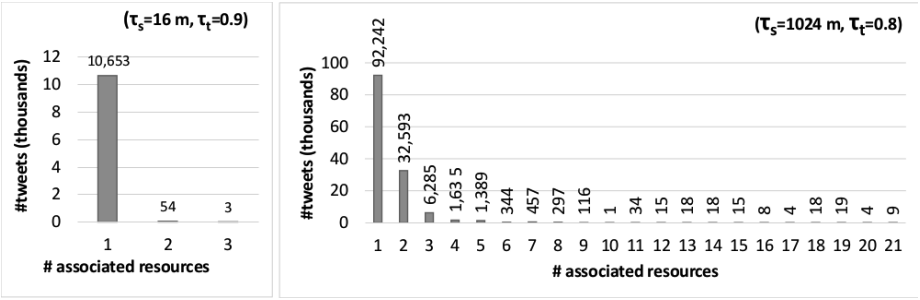


Figure 10: Distribution of the number of associated resources to tweets

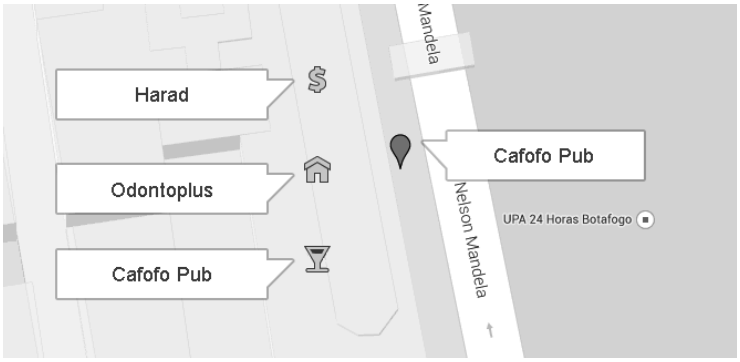
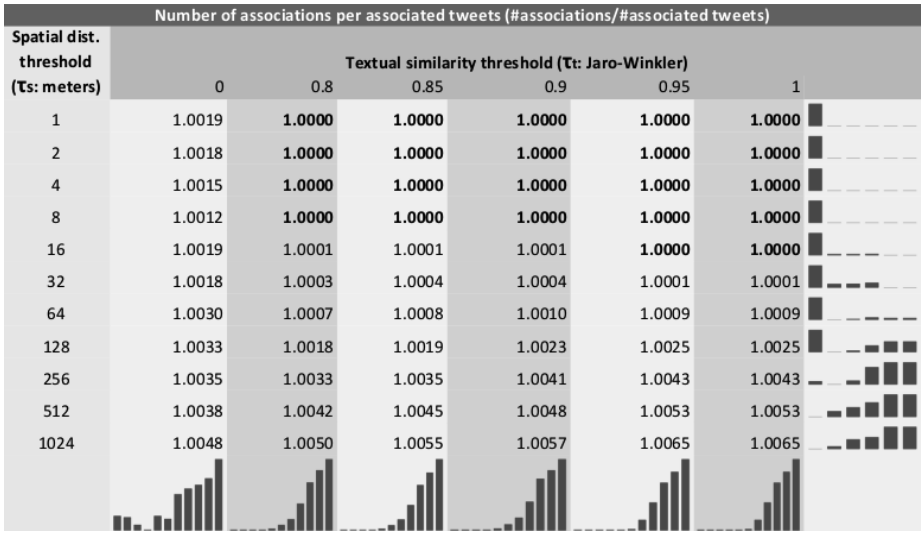


Figure 11: Example of a tweet position linked to most similarly labeled place

The experiments have been repeated with the same Twitter data and LOD as input, but performing the refinements of lines 4 to 17 of Algorithm 1 to link each movement segment ms only to the resource(s) that are the closest to ms in the geographic space, and whose location name is most similar to at least a label of that resource(s). Figure 12 presents the average number of associated resources per tweet for different values of τ_s and τ_t. In these results, the number of ambiguities do not vary monotonically with variations in τ_s and τ_t, because the refinement phase optimize the results for minimum τ_s and maximum τ_t, and the number of matching resources can vary for such optimized values. Notice that the average number of associated resources per tweet has decreased to values equal or closer to 1 than those in Figure 9, even for high values of τ_s. In fact,

the total number of ambiguities have been reduced to less than 0.5% for $\tau_s = 1$ km and $\tau_t = 0.8$, and is around 0.01% for $\tau_s = 16$ m and $\tau_t = 0.9$. In addition, some slight gains in the execution time have been observed in experiments that use the refinement portion of Algorithm 1, compared to just filtering by τ_s and τ_t. It happens because the textual similarity is only calculated in Algorithm 1 when a closer resource than the previously matched one(s) is found in line 9.



Number of associations per associated tweets (#associations/#associated tweets)

Spatial dist. threshold (τs: meters)	Textual similarity threshold (τt: Jaro-Winkler) 0	0.8	0.85	0.9	0.95	1
1	1.0019	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
2	1.0018	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
4	1.0015	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
8	1.0012	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
16	1.0019	1.0001	1.0001	1.0001	**1.0000**	**1.0000**
32	1.0018	1.0003	1.0004	1.0004	1.0001	1.0001
64	1.0030	1.0007	1.0008	1.0010	1.0009	1.0009
128	1.0033	1.0018	1.0019	1.0023	1.0025	1.0025
256	1.0035	1.0033	1.0035	1.0041	1.0043	1.0043
512	1.0038	1.0042	1.0045	1.0048	1.0053	1.0053
1024	1.0048	1.0050	1.0055	1.0057	1.0065	1.0065

Figure 12: Average number of associated LOD resources per tweet (refined)

Finally, Figure 13 presents the distribution of the top LOD resources linked to tweets collected during World Cup 2014, and whose geographic coordinates are inside the MBR fitting the Brazilian territory. The associations have been produced by using Algorithm 1, with $\tau_s = 16$ meters, and $\tau_t = 0.9$. The LOD instances appearing in Figure 13 (e.g., `dbp:Object100002684`) are visited PoIs (instances) directly associated to tweets by the algorithm. The LOD classes appearing in this figure (e.g., `geovcab:spatial#Feature`) have been indirectly associated to the tweets, by considering the property `rdf:type` of each associated LOD instance. Notice that the classes with the highest frequency are general ones such as `geovcab:spatial#Feature`, `lgd:meta/Node`, and `lgd:ontology/Amenity`. It happens because one LOD instance can be linked via `rdf:type` to many classes, and general classes are more prominent values of `rdf:type` in LOD collections such as DBPedia and LinkedGeoData. However, each LOD instance is usually associated to many classes (both general and specific ones), and most instances have at least one association with a specific class. Therefore, the high frequency of general classes does not hinder the ability to make interesting queries referring to more specific ones such as `lgd:ontology/Restaurant`, `lgd:ontology/Shop`, `dbpo:Stadium`, and `lgd:ontology/FastFood` (kind of restaurant). The direct matchings with LOD instances and indirect matchings with their types obtained in our experiments

proved helpful to bring forth interesting results for a variety of queries, such as the ones presented in the following subsection.

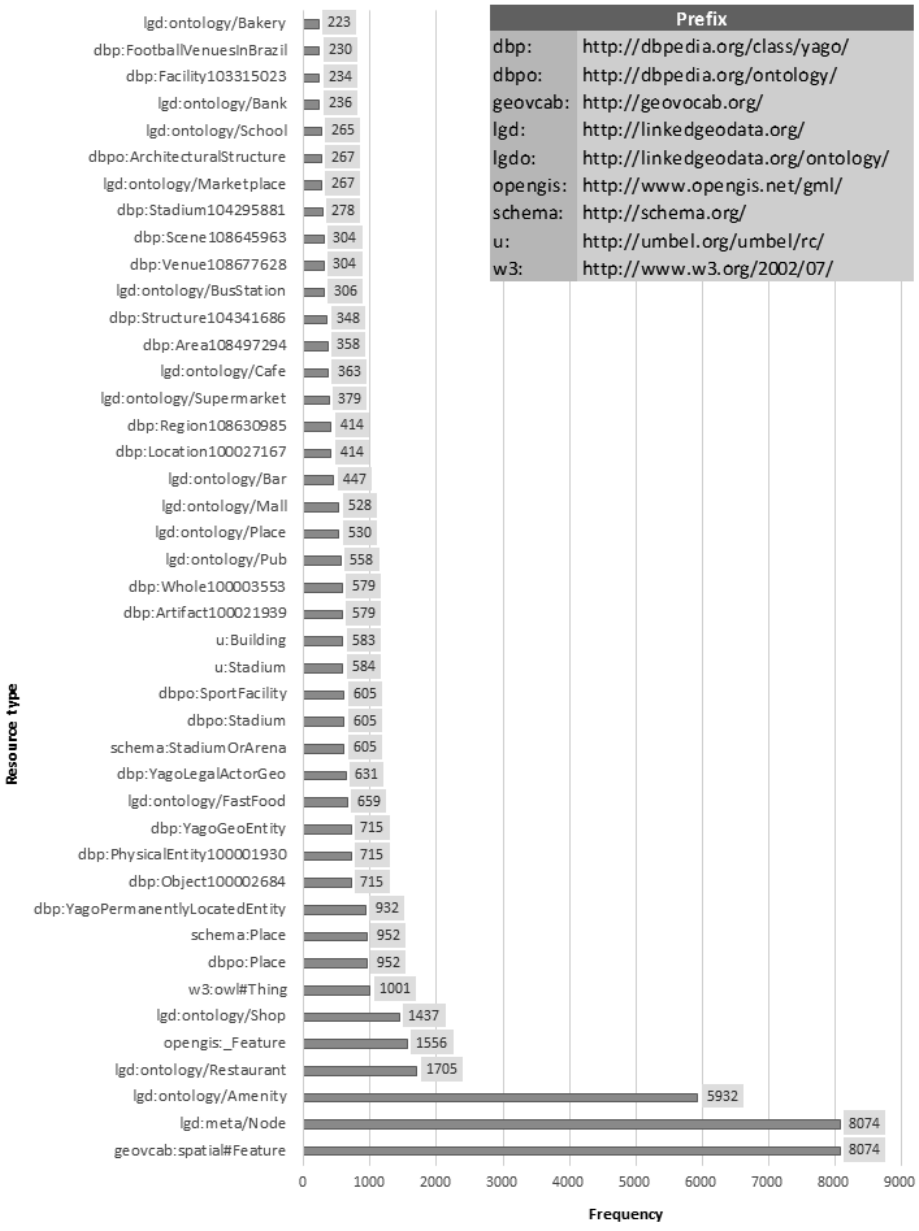


Figure 13: Top frequent LOD resources associated with tweets collected during the World Cup 2014 in Brazil

5.3. Example Queries

Baquara[2] ontology compliant knowledge bases (KBs) built with movement data extracted from sources such as Flickr and Twitter, and semantically enriched with LOD of collections such as DBPedia and LinkedGeoData can be stored in an RDF triples repository such as that of Virtuoso[13], which supports SPARQL and GeoSPARQL. Of course, besides Algorithm 1, further processing is necessary for producing hierarchies of semantically annotated movement segments as described in Section 2. A variety of methods and tools can help perform this task [53, 29, 30, 13, 14, 11, 31].

The following SPARQL queries complement the geoSPARQL ones presented in [38], by providing examples involving semantic constraints in multiple levels of movement segments hierarchies. Consider that the prefix `bq` refers to the URI of the *Baquara*[2] ontology[14], and that LOD from different sources have been pre-processed to consolidate the properties of resources linked by the `sameAs` property. The property `hasDuration` is available for movement segments in the *Baquara*[2] ontology to avoid having to calculate the movement segment duration in minutes in queries by using its time span.

Query 1 *Select the social media user's trails with at least one stop to visit a mountain called Corcovado in the city of Rio, followed by one stop in a marketplace, where he/she does at least one finer stop in a restaurant.*

```
SELECT ?t WHERE {
 ?t a bq:Trail.
 ?sc bq:father ?t;  a bq:Stop;  bq:ord ?sc_ord;
     bq:visits ?corcovado.
 ?corcovado a <http://schema.org/Mountain>;
            rdfs:label "Corcovado";
            <http://dbpedia.org/property/location> ?rio.
 ?rio a <http://dbpedia.org/ontology/City>;
      rdfs:label "Rio de Janeiro".
 ?sm bq:father ?t;  a bq:Stop;  bq:ord ?sm_ord;
     bq:visits ?mp.
 ?mp a <http://linkedgeodata.org/ontology/Marketplace>.
 ?sr  bq:father ?sm;  a bq:Stop;  bq:visits ?r.
 ?r a <http://linkedgeodata.org/ontology/Restaurant>.
 FILTER(?sc_ord < ?sm_ord)}
```

Query 2 *Determine the percentage of European's trails in Brazil that make at least a stop in a nature reserve, where he/she does at least one finer stop in a tourist shop.*

[13] http://virtuoso.openlinksw.com

[14] Prefix bq:<`http://www.seek-project.eu/Baquara02`>

```
SELECT COUNT(DISTINCT ?ts) / COUNT(DISTINCT ?t) WHERE {
 ?ts a bq:Trail;  bq:isOfMOclass bq:European;  bq:visits ?b.
 ?t a bq:Trail;  bq:isOfMOclass bq:European;  bq:visits ?b.
 ?b a <http://dbpedia.org/ontology/Country>;
       rdfs:label "Brazil".
 ?sr bq:father ?ts;  a bq:Stop; bq:visits ?r.
 ?r a <http://linkedgeodata.org/ontology/NatureReserve>.
 ?hs bq:father ?r;  a bq:Stop;  bq:visits ?s;
 ?s <http://linkedgeodata.org/ontology/TouristShop>.}
```

Query 3 *Select the* `Cities` *with the largest number of* `trails` *inside them, with at least one stop in a* `Marketplace`*, that is preceded by a* `stop` *in a* `Hotel`*, and followed by a* `stop` *in a* `Nightclub`*, that lasts at least 2 hours. The stop in the* `Marketplace` *must be detailed in at least one stop in a* `Pub`*.*

```
SELECT ?cityLabel, COUNT(DISTINCT ?t) AS ?nts WHERE {
 ?t a bq:Trail;  bq:visits ?city.
 ?city a <http://linkedgeodata.org/ontology/City>;
       rdfs:label ?cityLabel.
 ?sm bq:father ?t;  a bq:Stop;  bq:ord ?sm_ord;
     bq:visits ?mp.
 ?mp a <http://linkedgeodata.org/ontology/Marketplace>.
 ?sh bq:father ?t;  a bq:Stop;  bq:ord ?sh_ord;
      bq:visits ?h.
 ?h a <http://linkedgeodata.org/ontology/Hotel>.
 ?snc bq:father ?t;  a bq:Stop;  bq:ord ?snc_ord;
      bq:hasDuration ?snc_dur;   bq:visits ?nc.
 ?nc a <http://linkedgeodata.org/ontology/Nightclub>.
 ?sp bq:father ?sm;  a bq:Stop;   bq:visits ?p.
 ?p a <http://linkedgeodata.org/ontology/Pub>.
 FILTER((?snc_dur >= 120) && (?snc_ord > ?sm_ord) &&
        (?sh_ord < ?sm_ord))}
GROUP BY ?cityLabel  ORDER BY DESC(?nts)
```

6. Related Work

The present article extends and improves a previous work [38], that introduced the Baquara ontology, and firstly exploited links between movement data and LOD. First, it generalizes movement segments hierarchies to allow many levels of refinement. Secondly, it details on the *Baquara*[2] ontology and on the semantic linking process, providing also a customization of this process in the form of an algorithm to link movement segments with visited PoIs. Thirdly, it provides further experimental results with bigger amounts of social media data for assessing the effectiveness of the proposed linking algorithm.

A core contribution of this work is the conceptual model conveyed by the *Baquara*[2] ontology, which enables knowledge-based semantic description and analysis of movements in several abstraction levels. A pioneering work on conceptual modeling of spatio-temporal objects is MADS (Modeling Application Data with Spatio-temporal features) [54]. MADS extends the basic ER model with spatio-temporal constructs. The key MADS premise is that spatial and temporal concepts are orthogonal. MADS uses the object-relationship paradigm, including the features of the ODMG (Object Database Management Group) data model, and provides spatial and temporal data types, attributes, and relationships. It offers a wide range of conceptual constructs to model the spatio-temporal world. A more recent contribution with focus on conceptual modeling of trajectories (spatio-temporal objects changing their geographical positions but not their shapes) comes from Spaccapietra *et al.* [7]. This model represents semantic trajectories as stops and moves, i.e., trajectory segments in which the object is stationary or moving, respectively. It has been the first attempt to embed semantics in the movement representation, but it lacks generality since other relevant semantic aspects are not explicitly taken into account. An extension of the "Stop-Move" model towards overcoming these limitations comes from the CONSTAnT conceptual model [16], which defines several semantic dimensions for movement analysis (e.g., goal, behavior).

Although the conceptual modeling of trajectories have seen a "convergence" mainly to the "Stop-Move" model [7], ontologies for movement data did not find so far an agreed approach. Focusing only on the works most related to our approach, we recall, for example, Yan *et al.* [14]. The ontology proposed in that work includes three modules: the Geometric Trajectory Ontology describes the spatio-temporal features of a trajectory; the Geographic Ontology describes the geographic objects; and the Domain Application Ontology describes the thematic objects of the application. These ontology modules are integrated into a unique ontology that supports conjunctive queries in a traffic application.

The work presented in [55] introduces a design pattern for semantic trajectories to enable the publishing as Linked Data. They describe the geo-ontology design pattern in OWL expressing the basic features of a semantic trajectory like the spatio-temporal information as a sequence of fixes and the semantic information like Points of Interest visited and device information.

The proposal of [6] exploits a movement ontology for querying and mining trajectory data enriched with geographic and application information. Here the ontology has been used to infer application-dependent behavior from raw and mined trajectory data. A pioneering work on movement patterns is instead the one of Dodge *et al.* [36] where authors propose a taxonomy of movement patterns distinguishing generic patterns (e.g. moving clusters, co-location) that represents any form of movement behavior and can be extracted applying generic data mining algorithms from behavioral patterns (e.g. flock, leadership) where the movement has a clear semantics and can be considered higher level movement patterns. A recent survey on semantic trajectory modeling and analysis is reported in [13].

However, these works do not address the automatic enrichment of trajec-

tories with semantically precise information about specific places (e.g., restaurants, hotels, touristic spots), events (e.g., sport events, cultural events), and other relevant entities of the open dynamic world in which trajectories occur. In this article, movement segments are linked to specific concepts and/or instances via ontological relationships that can describe their precise semantics. Such semantic enrichment requires lots of continuously updated information, with well-defined and widely agreed semantics.

The conjugated use of textual and spatial information in social networks is another growing research theme. The recent work [56] proposes a method to exploit spatial proximity and users' common interests for querying location-based social networks. The problem is clearly NP-complete and the authors propose two efficient algorithms that explore the search space using two distinct criteria, that give good results in terms of performance compared to the state of the art.

Entity linking of social media data (e.g., tweets) is also a growing research theme, because in this context the task is particularly challenging: the text is noisy, short, and informal [42, 41]. Entity linking has been largely explored on the Web, mainly relying on the context around the entity [57, 58, 59]. However, these methods cannot be applied to tweets due to the insufficient context information. The interesting article [60] proposes a graph-based framework to collectively link all the named entity mentions in all the tweets posted by a user, with the assumption that each user has an underlying topic of interest distribution over various named entities.

In the field of semantic extraction from text, the challenge is to find patterns or associations in the text, in particular the co-occurrence of terms. An example of these kind of approach is [61], that proposes a generic framework for mining semantic associations in text. They base their idea on a co-occurrence graph and a set of primitives to mine four kinds of semantic associations, namely: topical anchors, semantic siblings, topical markers, and topic expansions.

7. Conclusions and Future Work

Vast amounts of linked open data (LOD) about real world entities and events have been fed and continuously updated on the Web. However, their potential to leverage movement understanding has not been fully exploited yet. This article proposes the *Baquara*^{2} knowledge-based framework as a bridge between movement analysis and knowledge bases, by allowing movement data and associated knowledge to be connected and queried together. It unleashes the use of growing collections of ontologies and LOD available in the Web to help semantically enrich and analyze a wide variety of movement data. The major contributions of this article are: (i) an ontology to support semantic enrichment and analysis of movement data, in several abstraction levels and according with several domain specific semantic facets; (ii) a customizable process to automatically create semantic annotations of movement data based on concepts and instances of existing knowledge bases (e.g., legacy ontologies, LOD collections) organized in semantic facets; (iii) a customization of the proposed process in the form of

an algorithm that connects movement data to PoIs based on spatial proximity and lexical similarity; and (iv) experimental results that show the viability of the proposal in case studies with real data available in the Web. Though this paper focus on knowledge management in engines such as triple stores, the latter constitute just an alternative means to handle movement data and knowledge. The semantically enriched movement data produced by using $Baquara^2$ can be stored and efficiently processed in alternative kinds of spatio-temporal database management systems.

In our future work we plan to: (i) further evaluate the performance and the effectiveness of the proposed framework in several application domains; (ii) collect data and judgments of volunteers to serve as ground true for analyzing the quality of the semantic annotations generated by the proposed methods, with measures such as precision and recall; (iii) develop efficient and effective methods to derive precise semantic annotations from different movement data, ontologies, and LOD collections; and (iv) investigate ways to use the semantic enriched movement data generated in accordance with the proposed framework in data warehousing and data mining.

Acknowledgements

This work has been supported by the European Union IRSES-SEEK (grant 295179) and FP7-DATASIM (grant 270833) projects. The Brazilian researchers have also been supported by CNPq (grant 478634/2011-0), CAPES, FAPESC, and FEESC. Nikos Pelekis research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thales. Thanks to Carina Friedrich Dorneles for the valuable hints in the area of textual similarity. Special thanks to Vania Bogorny for her support in difficult times, fruitful discussions on semantic enrichment of trajectories, and the opportunities she opened for our research group to collaborate with other ones around the world. Finally, we acknowledge the anonymous reviewers of the ER conference and of DKE for their precious comments that helped to improve this work.

References

[1] Z. Cheng, J. Caverlee, K. Lee, D. Z. Sui, Exploring millions of footprints in location sharing services, in: L. A. Adamic, R. A. Baeza-Yates, S. Counts (Eds.), ICWSM, The AAAI Press, 2011.

[2] Z. Yin, L. Cao, J. Han, J. Luo, T. S. Huang, Diversified trajectory pattern ranking in geo-tagged social media, in: SDM, SIAM / Omnipress, 2011, pp. 980–991.

[3] M. Azmandian, K. Singh, B. Gelsey, Y.-H. Chang, R. T. Maheswaran, Following human mobility using tweets, in: L. Cao, Y. Zeng, A. L. Symeonidis,

V. Gorodetsky, P. S. Yu, M. P. Singh (Eds.), ADMI, Vol. 7607 of LNCS, Springer, 2012, pp. 139–149.

[4] L. Gabrielli, S. Rinzivillo, F. Ronzano, D. Villatoro, From tweets to semantic trajectories: Mining anomalous urban mobility patterns, in: J. Nin, D. Villatoro (Eds.), CitiSens, Vol. 8313 of LNCS, Springer, 2013, pp. 26–35.

[5] S. Kumar, F. Morstatter, H. Liu, Twitter Data Analytics, Springer Briefs in Computer Science, Springer, 2014.

[6] C. Renso, M. Baglioni, J. A. F. de Macêdo, R. Trasarti, M. Wachowicz, How you move reveals who you are: understanding human behavior by analyzing trajectory data, Knowl. Inf. Syst. 37 (2) (2013) 331–362.

[7] S. Spaccapietra, C. Parent, M. L. Damiani, J. A. F. de Macêdo, F. Porto, C. Vangenot, A conceptual view on trajectories, Data Knowl. Eng. 65 (1) (2008) 126–146.

[8] M. Nanni, R. Trasarti, C. Renso, F. Giannotti, D. Pedreschi, Advanced knowledge discovery on movement data with the geopkdd system, in: EDBT, Vol. 426 of ACM International Conference Proceeding Series, ACM, 2010, pp. 693–696.

[9] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, C. Renso, S. Rinzivillo, R. Trasarti, Unveiling the complexity of human mobility by querying and mining massive trajectory data, VLDB J. 20 (5) (2011) 695–719.

[10] B. Furletti, L. Gabrielli, C. Renso, S. Rinzivillo, Analysis of GSM calls data for understanding user mobility behavior, in: BigData Conference, IEEE, 2013, pp. 550–555.

[11] N. Pelekis, Y. Theodoridis, Mobility Data Management and Exploration, Springer, 2014.

[12] S. Spaccapietra, C. Parent, Adding meaning to your steps (keynote paper), in: M. A. Jeusfeld, L. M. L. Delcambre, T. W. Ling (Eds.), ER, Vol. 6998 of LNCS, Springer, 2011, pp. 13–31.

[13] C. Parent, S. Spaccapietra, C. Renso, G. L. Andrienko, N. V. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. A. F. de Macêdo, N. Pelekis, Y. Theodoridis, Z. Yan, Semantic trajectories modeling and analysis, ACM Comput. Surv. 45 (4), article 42.

[14] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, K. Aberer, Semantic trajectories: Mobility data computation and annotation, ACM TIST 4 (3).

[15] A. S. Furtado, R. Fileto, C. Renso, Assessing the attractiveness of places with movement data, JIDM 4 (2) (2013) 124–133.

[16] V. Bogorny, C. Renso, A. R. de Aquino, F. de Lucca Siqueira, L. O. Alvares, CONSTAnT - A Conceptual Data Model for Semantic Trajectories of Moving Objects, T. GIS 18 (1) (2014) 66–88.

[17] N. Pelekis, Y. Theodoridis, D. Janssens, On the management and analysis of our lifesteps, SIGKDD Explorations 15 (1) (2013) 23–32.

[18] A. N. Ngomo, S. Auer, J. Lehmann, A. Zaveri, Introduction to linked data and its lifecycle on the web, in: Reasoning on the Web in the Big Data Era, Athens, Greece, 2014, pp. 1–99.

[19] J. A. P. Sacenti, R. Fileto, F. Salvini, A. Raffaetà, A. Roncato, Automatically tailoring semantics-enabled dimensions for movement data warehouses, in: DaWaK (to appear), 2015.

[20] R. Battle, D. Kolas, Enabling the geospatial Semantic Web with Parliament and GeoSPARQL, Semantic Web 3 (4) (2012) 355–370.

[21] K. Kyzirakos, M. Karpathiotakis, M. Koubarakis, Strabon: A semantic geospatial dbms, in: ISWC, Vol. 7649 of LNCS, Springer Berlin Heidelberg, 2012, pp. 295–311.

[22] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, C. Bizer, Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia, Semantic Web 6 (2) (2015) 167–195.

[23] C. Stadler, J. Lehmann, K. Höffner, S. Auer, Linkedgeodata: A core for a web of spatial open data, Semantic Web 3 (4) (2012) 333–354.

[24] S. Rinzivillo, F. de Lucca Siqueira, L. Gabrielli, C. Renso, V. Bogorny, Where Have You Been Today? Annotating Trajectories with DayTag, in: SSTD, Vol. 8098 of LNCS, Springer, 2013, pp. 467–471.

[25] A. Doulamis, N. Pelekis, Y. Theodoridis, Easytracker: An android application for capturing mobility behavior, 2012 16th Panhellenic Conference on Informatics 0 (2012) 357–362.

[26] G. Broll, H. Cao, P. Ebben, P. Holleis, K. Jacobs, J. Koolwaaij, M. Luther, B. Souville, Tripzoom: An app to improve your mobility behavior, in: Intl. Conf. on Mobile and Ubiquitous Multimedia, MUM, ACM, New York, NY, USA, 2012, pp. 57:1–57:4.

[27] R. G. B. Nabo, R. Fileto, C. Renso, M. Nanni, Annotating Trajectories by Fusing them with Social Media Users' Posts, in: Brazilian Symposium on Geoinformatics, GeoInfo, Campos do Jordão, SP, Brazil (to appear), 2014.

[28] G. Marketos, E. Frentzos, I. Ntoutsi, N. Pelekis, A. Raffaetà, Y. Theodoridis, Building Real World Trajectory Warehouses, in: MobiDE, ACM, 2008, pp. 8–15.

[29] J. A. M. R. Rocha, V. C. Times, G. Oliveira, L. O. Alvares, V. Bogorny, DB-SMoT: A direction-based spatio-temporal clustering method, in: IEEE Conf. of Intelligent Systems, IEEE, 2010, pp. 114–119.

[30] V. Bogorny, H. Avancini, B. C. de Paula, C. R. Kuplich, L. O. Alvares, Weka-STPM: a Software Architecture and Prototype for Semantic Trajectory Data Mining and Visualization, T. GIS 15 (2) (2011) 227–248.

[31] F. Moreno, A. Pineda, R. Fileto, V. Bogorny, SMoT+: Extending the SMoT Algorithm for Discovering Stops in Nested Sites, Computing and Informatics 33 (2) (2014) 327–342.

[32] P. Rigaux, M. Scholl, A. Voisard, Spatial databases - with applications to GIS, Elsevier, 2002.

[33] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.

[34] R. Fileto, A. Raffaetà, A. Roncato, J. A. P. Sacenti, C. May, D. Klein, A semantic model for movement data warehouses, in: 16th DOLAP, Shanghai, China, November 7, 2014.

[35] M. Wachowicz, R. Ong, C. Renso, M. Nanni, Finding moving flock patterns among pedestrians through collective coherence, International Journal of Geographical Information Science 25 (11) (2011) 1849–1864.

[36] S. Dodge, R. Weibel, A.-K. Lautenschütz, Towards a Taxonomy of Movement Patterns, Information Visualization 7 (3) (2008) 240–252.

[37] L. O. Alvares, A. M. Loy, C. Renso, V. Bogorny, An algorithm to identify avoidance behavior in moving object trajectories, Journal of the Brazilian Computer Society 17 (3) (2011) 193–203.

[38] R. Fileto, M. Krüger, N. Pelekis, Y. Theodoridis, C. Renso, Baquara: A Holistic Ontological Framework for Movement Analysis Using Linked Data, in: W. Ng, V. C. Storey, J. Trujillo (Eds.), ER, Vol. 8217 of LNCS, Springer, 2013, pp. 342–355.

[39] A. Kiryakov, B. Popov, I. Terziev, D. Manov, D. Ognyanoff, Semantic Annotation, Indexing, and Retrieval, Web Semantics: Science, Services and Agents on the World Wide Web 2 (1) (2004) 49–79.

[40] W. Zhang, J. Su, C. L. Tan, W. T. Wang, Entity linking leveraging: Automatically generated annotation, in: 23rd Intl. Conf. on Computational Linguistics, COLING, Association for Computational Linguistics, Stroudsburg, PA, USA, 2010, pp. 1290–1298.

[41] X. Liu, S. Zhang, F. Wei, M. Zhou, Recognizing named entities in tweets, in: 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Vol. 1 of HLT, Association for Computational Linguistics, Stroudsburg, PA, USA, 2011, pp. 359–367.

[42] A. Ritter, S. Clark, Mausam, O. Etzioni, Named entity recognition in tweets: An experimental study, in: Conf. on Empirical Methods in Natural Language Processing, EMNLP, Association for Computational Linguistics, Stroudsburg, PA, USA, 2011, pp. 1524–1534.

[43] J. Nothman, N. Ringland, W. Radford, T. Murphy, J. R. Curran, Learning multilingual named entity recognition from wikipedia, Artificial Intelligence 194 (0) (2013) 151 – 175.

[44] A. Gattani, D. S. Lamba, N. Garera, M. Tiwari, X. Chai, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, A. Doan, Entity extraction, linking, classification, and tagging for social media: A wikipedia-based approach, Proc. VLDB Endow. 6 (11) (2013) 1126–1137.

[45] X. Han, L. Sun, J. Zhao, Collective entity linking in web text: A graph-based method, in: Proc 34th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval, SIGIR, ACM, New York, NY, USA, 2011, pp. 765–774.

[46] Z. Guo, D. Barbosa, Entity linking with a unified semantic representation, in: 23rd Intl. Conference on World Wide Web, WWW Companion, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 2014, pp. 1305–1310.

[47] G. Navarro, A guided tour to approximate string matching, ACM Comput. Surv. 33 (1) (2001) 31–88.

[48] W. W. Cohen, P. D. Ravikumar, S. E. Fienberg, A comparison of string distance metrics for name-matching tasks, in: S. Kambhampati, C. A. Knoblock (Eds.), IIWeb, 2003, pp. 73–78.

[49] E. Moreau, F. Yvon, O. Cappé, Robust Similarity Measures for Named Entities Matching, in: 22nd Intl. Conf. on Computational Linguistics - Volume 1, COLING, Association for Computational Linguistics, Stroudsburg, PA, USA, 2008, pp. 593–600.

[50] P. Bouros, S. Ge, N. Mamoulis, Spatio-textual similarity joins, Proc. VLDB Endow. 6 (1) (2012) 1–12.

[51] S. Liu, G. Li, J. Feng, Star-join: Spatio-textual similarity join, in: 21st ACM Intl. Conf. on Information and Knowledge Management, CIKM '12, ACM, New York, NY, USA, 2012, pp. 2194–2198.

[52] L. Derczynski, D. Maynard, G. Rizzo, M. van Erp, G. Gorrell, R. Troncy, J. Petrak, K. Bontcheva, Analysis of named entity recognition and linking for tweets, Inf. Process. Manage. 51 (2) (2015) 32–49.

[53] A. T. Palma, V. Bogorny, B. Kuijpers, L. O. Alvares, A clustering-based approach for discovering interesting places in trajectories, in: R. L. Wainwright, H. Haddad (Eds.), SAC, ACM, 2008, pp. 863–868.

[54] C. Parent, S. Spaccapietra, E. Zimányi, Conceptual modeling for traditional and spatio-temporal applications - the MADS approach, Springer, 2006.

[55] Y. Hu, K. Janowicz, D. Carral, S. Scheider, W. Kuhn, G. Berg-Cross, P. Hitzler, M. Dean, D. Kolas, A geo-ontology design pattern for semantic trajectories, in: T. Tenbrink, J. Stell, A. Galton, Z. Wood (Eds.), Spatial Information Theory, Vol. 8116 of LNCS, Springer, 2013, pp. 438–456.

[56] Y. Li, D. Wu, J. Xu, B. Choi, W. Su, Spatial-aware interest group queries in location-based social networks, Data & Knowledge Engineering 92 (0) (2014) 20–38.

[57] S. Cucerzan, Large-scale named entity disambiguation based on wikipedia data, in: Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), June 28-30, Prague, Czech Republic, 2007, pp. 708–716.

[58] R. C. Bunescu, M. Pasca, Using encyclopedic knowledge for named entity disambiguation, in: EACL 2006, 11st Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, April 3-7, 2006, Trento, Italy, 2006.

[59] S. Kulkarni, A. Singh, G. Ramakrishnan, S. Chakrabarti, Collective annotation of wikipedia entities in web text, in: 15th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, KDD, ACM, New York, NY, USA, 2009, pp. 457–466.

[60] W. Shen, J. Wang, P. Luo, M. Wang, Linking named entities in tweets with knowledge base via user interest modeling, in: ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, KDD, ACM, New York, NY, USA, 2013, pp. 68–76.

[61] A. R. Rachakonda, S. Srinivasa, S. Kulkarni, M. Srinivasan, A generic framework and methodology for extracting semantics from co-occurrences, Data & Knowledge Engineering 92 (0) (2014) 39–9.

SECRETA: A System for Evaluating and Comparing RElational and Transaction Anonymization algorithms*

Giorgos Poulis
University of Peloponnese
poulis@uop.gr

Aris Gkoulalas-Divanis
IBM Research-Ireland
arisdiva@ie.ibm.com

Grigorios Loukides
Cardiff University
g.loukides@cs.cf.ac.uk

Spiros Skiadopoulos
University of Peloponnese
spiros@uop.gr

Christos Tryfonopoulos
University of Peloponnese
trifon@uop.gr

ABSTRACT

Publishing data about individuals, in a privacy-preserving way, has led to a large body of research. Meanwhile, algorithms for anonymizing datasets, with relational or transaction attributes, that preserve *data truthfulness*, have attracted significant interest from organizations. However, selecting the most appropriate algorithm is still far from trivial, and tools that assist data publishers in this task are needed. In response, we develop SECRETA, a system for analyzing the effectiveness and efficiency of anonymization algorithms. Our system allows data publishers to evaluate a specific algorithm, compare multiple algorithms, and combine algorithms for anonymizing datasets with both relational and transaction attributes. The analysis of the algorithm(s) is performed, in an interactive and progressive way, and results, including attribute statistics and various data utility indicators, are summarized and presented graphically.

1. INTRODUCTION

Publishing data about individuals is essential for applications, ranging from marketing to healthcare. Several marketing studies, for example, seek to find product combinations that appeal to customers with specific demographic profiles, while a large class of medical studies aims to discover associations between patient demographics and diseases. To enable these applications, data must be published in a way that preserves privacy and utility.

Towards this goal, numerous algorithms that prevent the disclosure of individuals' private and sensitive information, while maintaining *data truthfulness* (i.e., generate data that can be analyzed at a record level), have been proposed [4,6,7,10]. These algorithms work by transforming attribute values in a dataset (e.g., replacing them with more general values), and are applicable to either relational or transaction (set-valued) attributes. For example, an individual's year of birth is modeled as a relational attribute, while his/her purchased items are modeled as a transaction attribute. Furthermore, these algorithms can be combined, using a recent approach [9], to anonymize datasets with both relational and transaction attributes, referred to as RT-datasets.

While there is a growing interest for publishing protected and truthful data from governmental [8] and industrial organizations [1], selecting the most appropriate algorithm, for a given dataset and publishing scenario, remains a challenging and error-prone task. This is because both the effectiveness and efficiency of algorithms depend on: (a) *data characteristics* (e.g., the distribution of values in an attribute), (b) *various input parameters which affect the level of privacy and utility* (e.g., hierarchies that govern data transformation), and (c) *data utility requirements* (e.g., the need to accurately answer a certain query workload, or to adhere to constraints on the way values are transformed).

To assist data publishers in this task, we propose SECRETA, the first system for evaluating and comparing anonymization algorithms for relational, transaction, and RT datasets. Our system integrates 9 popular algorithms under a common, benchmark-oriented framework, and it allows data publishers to apply and analyze the performance of one or more of these algorithms. SECRETA operates in two modes, namely *Evaluation* and *Comparison*.

The Evaluation mode can be used to configure and evaluate the effectiveness of a given algorithm, with respect to data utility and privacy, as well as its efficiency. For capturing data utility, we employ several information loss measures [7, 12] and support data utility requirements. These requirements can be expressed using queries and/or *utility constraints* [7], which are specified by data publishers or generated automatically. Furthermore, SECRETA enables the use of 20 different combinations of algorithms to anonymize RT-datasets. The selection and management of these combinations is performed in an intuitive way that allows preserving different aspects of data utility.

The Comparison mode offers data publishers the ability to design and execute benchmarks for comparing multiple anonymization algorithms. These benchmarks facilitate an interactive and progressive comparison of sets of algorithms, with respect to their utility and efficiency. The results of the comparative analysis are summarized and presented graphically, allowing for fast and intuitive understanding of the effectiveness and efficiency of different algorithms.

To our knowledge, SECRETA is the only system that permits a comprehensive evaluation and comparison of recent anonymization techniques. The Cornell Anonymization Toolkit [11] demonstrates a single algorithm for relational data, also supported by SECRETA, while TIAMAT [3] does

*More details about the demo, together with additional screen shots, are available at: `http://secreta.uop.gr/`.

not support algorithms for transaction data, nor methods for anonymizing *RT*-datasets. Moreover, none of these systems employs utility requirements. We believe that the distinctive features of SECRETA can greatly assist data publishers in making informed decisions on publishing anonymized data.

2. OVERVIEW OF SECRETA

This section describes the components of our system, which we broadly divide into *frontend* and *backend* components. The frontend offers a Graphical User Interface (GUI), which enables users to: (a) issue anonymization requests, and (b) visualize and store experimental results. The backend consists of components for servicing anonymization requests and for conducting experimental evaluations. The architecture of SECRETA is presented in Figure 1.

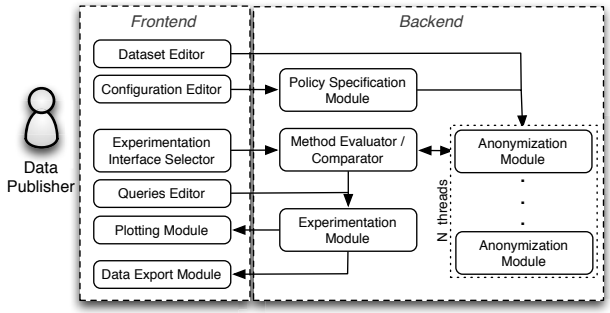


Figure 1: Architecture of SECRETA

2.1 Frontend of SECRETA

The frontend is implemented using the QT framework (`https://qt-project.org`). Using the provided GUI, users can: (a) select datasets for anonymization, (b) specify hierarchies and query workloads, (c) select and configure anonymization algorithms, (d) execute experiments and visualize the experimental results, and (e) export anonymized datasets and experimental results, in a variety of formats. In what follows, we detail the components of the frontend.

Dataset Editor: It enables users to select datasets for anonymization. The datasets can have relational and/or transaction attributes, and they need to be provided in a Comma-Separated Values (CSV) format. Once a dataset is loaded to the Dataset Editor, the user can modify it (edit attribute names and values, add/delete rows and attributes, etc.) and store the changes. The user can also generate data visualizations, such as histograms of attributes. Figure 2 shows a loaded dataset and some visualizations.

Configuration Editor: It allows users to select hierarchies and to specify utility and privacy policies. Hierarchies are used by all anonymization algorithms, except COAT [7] and PCTA [5], whereas utility and privacy policies are only used by these two algorithms to model such requirements. Hierarchies and policies can be uploaded from a file, or automatically derived from the data, using the algorithms in [7].

Queries Editor: This component allows specifying query workloads, which will be used to evaluate the utility of anonymized data in query answering. The system supports the same type of queries as [12], and uses Average Relative Error (ARE) [12] as a defacto utility indicator. The query workloads can be loaded from a file and edited by the user, or be inserted directly using the GUI (see Figure 2).

Experimentation Interface Selector: This component selects the operation mode of SECRETA. Figure 3 shows an interface of the Evaluation mode, in which users can evaluate a given algorithm, while Figure 4 shows an interface of the Comparison mode, which allows users to compare multiple algorithms. Through these interfaces, users can select and configure the algorithm(s) to obtain the anonymized data, store the anonymized dataset(s), and generate visualizations that present the performance of the algorithm(s).

Plotting Module: This module is based on the QWT library (`http://qwt.sourceforge.net/`) and supports a series of data visualizations that help users analyze their data and understand the performance of anonymization algorithms, when they are applied with different configuration settings. Specifically, users can visualize information about: (a) *the original/anonymized dataset* (e.g., histograms of attributes, relative difference of the frequency between an original and a generalized value), and (b) *anonymization results*, for *single* and *varying* parameter execution. In single parameter execution, the results are derived with fixed, user-specified parameters and include frequencies of generalized values in relational or set-valued attributes, runtime, etc. In varying parameter execution, the user selects the start/end values and step of a parameter that varies, as well as fixed values for other parameters. The plotted results include data utility indicators and runtime vs. the varying parameter.

Data Export Module: This module allows exporting datasets, hierarchies, policies, and query workloads, in CSV format, and graphs, in PDF, JPG, BMP or PNG format.

2.2 Backend of SECRETA

The backend of our system is implemented in C++. For each mode of operation, SECRETA invokes one or more instances of the Anonymization Module with the specified algorithm and parameters. The anonymization results are collected by the Method Evaluator/Comparator component and forwarded to the Experimentation Module. From there, results are forwarded to the Plotting Module, for visualization, and/or to the Data Export Module, for data export.

Policy Specification Module: This module invokes algorithms that automatically generate hierarchies [10], as well as the strategies in [7], which generate privacy and utility policies. The hierarchies and/or policies are used by the Anonymization Module (to be described later).

Method Evaluator/Comparator: This component implements the functionality that is necessary for supporting the interfaces of the Evaluation and of the Comparison mode. Based on the selected interface, anonymization algorithm(s) and parameters, this component invokes one or more instances (threads) of the Anonymization Module. After all instances finish, the Method Evaluator/Comparator component collects the anonymization results and forwards them to the Experimentation Module.

Anonymization Module: This component is responsible for executing an anonymization algorithm with the specified configuration. SECRETA supports 9 algorithms; 4 of them are applicable to datasets with relational atrtributes (Incog-

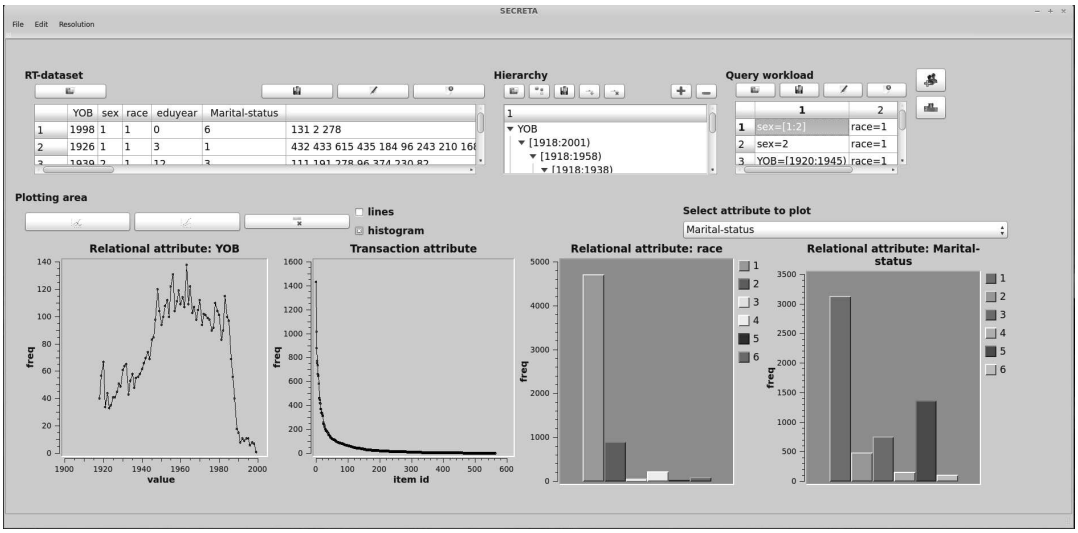


Figure 2: Main screen of SECRETA

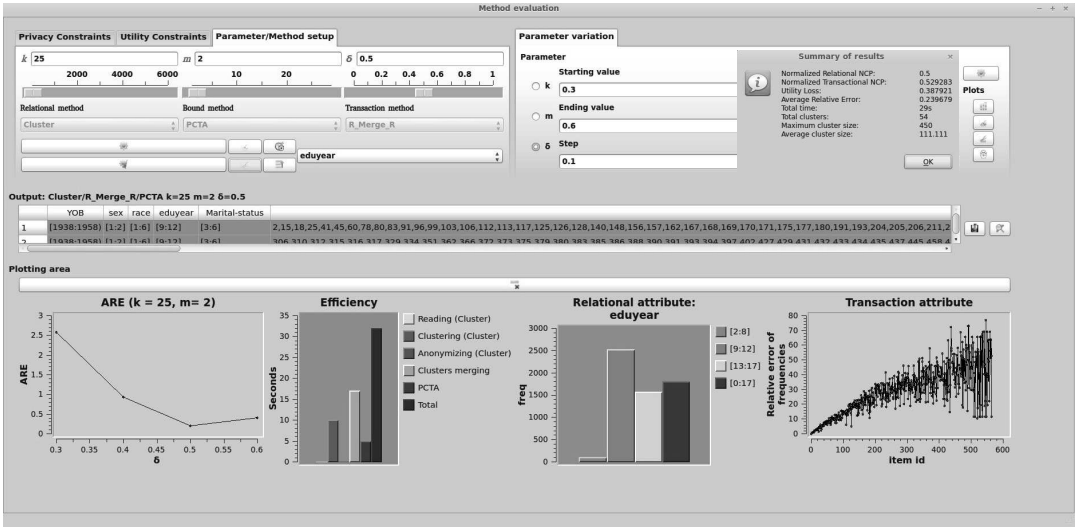


Figure 3: Evaluation mode: Method evaluation screen of SECRETA

nito [6], Cluster [9], Top-down [4], and Full subtree bottom-up), and 5 to datasets with transaction attributes (COAT [7], PCTA [5], Apriori, LRA and VPA [10]). Additionally, it supports 3 bounding methods ($\mathbf{R}\text{MERGE}_r$, $\mathbf{T}\text{MERGE}_r$, $\mathbf{RT}\text{MERGE}_r$) [9], which enable the anonymization of *RT*-datasets by combining two algorithms, each designed for a different attribute type (e.g., Incognito and COAT).

Experimentation Module: This module is responsible for producing visualizations of the anonymization results and of the performance of the anonymization algorithm(s), in the case of *single* and *varying* parameter execution. For visualizations involving the computation of ARE, input is used from the Queries Editor module. The produced visualizations are presented to the user, through the Plotting Module, and can be stored to disk, using the Data Export module.

3. DEMONSTRATION PLAN

During the demonstration, attendees will be able to use SECRETA to: (a) create, edit and analyze a dataset, and (b) execute two different scenarios that demonstrate the modes, functionality range, and potential of the system.

Using the Dataset Editor: The demonstration will start by allowing the user to load a ready-to-use *RT*-dataset. After that, the user will be able to edit the attribute names of the dataset, as well as the values in some records. These operations can be performed directly from the input area (top-left pane in Figure 2), and the user may overwrite the existing dataset with a modified one, or export it to a file. Subsequently, the user will analyze the dataset by plotting histograms of the frequency of values in any attribute (bottom pane in Figure 2).

Using the Configuration and Queries Editor: The user will load a predefined hierarchy from a file. This hierarchy is fully browsable and editable, through the hierarchy area (top-mid pane in Figure 2). Then, the user will

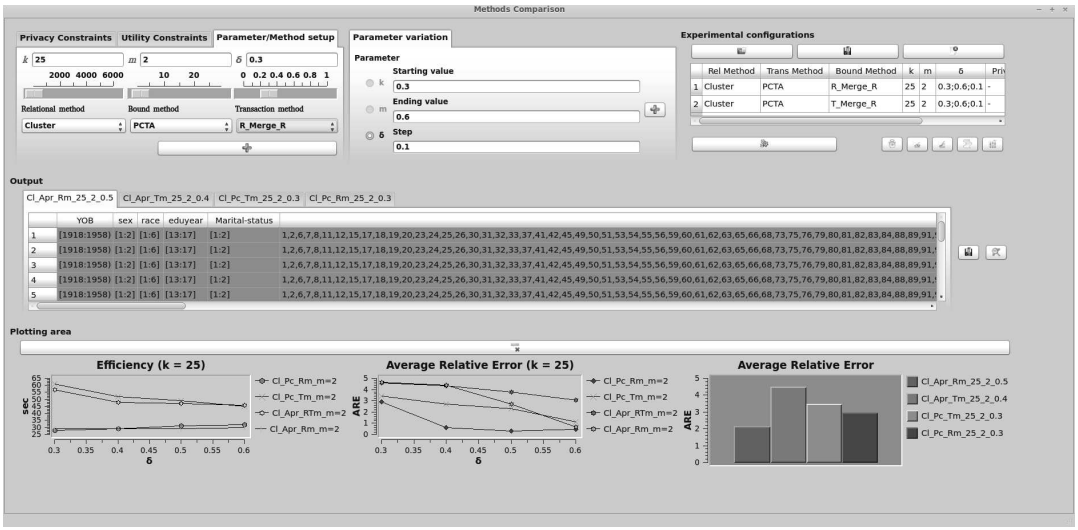


Figure 4: Comparison mode: Methods comparison screen of SECRETA

load a preconstructed query workload from a file, edit the query values using the query workload area (top-right pane in Figure 2), and follow either of the two following scenarios.

Evaluating a method for *RT*-datasets: In this scenario, the users will configure, apply, and evaluate a method, in a series of steps. First, they will use the "Method evaluation" interface (Figure 3) and set the values for parameters k, m, δ, by inputting them directly in the form, or by using the corresponding slider (top-left pane in Figure 3). Then, they may select two algorithms, one for anonymizing the relational attributes, and one for the transaction attribute, and a bounding method for combining the selected algorithms.

Next, the users will initiate the anonymization process. When this process ends, a message box with a summary of results will be presented and the anonymized dataset will be displayed in the output area (middle pane in Figure 3). Last, the users will select a number of data visualizations. These visualizations will be presented in the plotting area (bottom pane in Figure 3) and may illustrate any combination of the following: (a) ARE scores for various parameters (e.g., for varying δ and fixed k and m), (b) the time needed to execute the algorithm and its different phases, (c) the frequency of all generalized values, in a selected relational attribute, and (d) the relative error between the frequency of the transaction attribute values, in the original and the anonymized dataset.

Comparing methods for *RT*-datasets: In this scenario, the users will compare multiple anonymization methods. Using the "Methods comparison" interface (shown in Figure 4), they will: (a) select algorithms for anonymizing each type of attributes, as well as a bounding method, (b) set the values for parameters that will be fixed, as described above (top-left pane in Figure 4), and (c) choose a varying parameter (top-mid pane in Figure 4), along with its start/end value and step. The choices for (a) to (c) comprise a configuration, which will be added into the experimenter area (top-right pane in Figure 4). Similar configurations will be created by the users for at least another method. After the methods are applied, the users will select various graphs, which will be displayed in the plotting area (bottom pane in Figure 4).

4. CONCLUSION

In this paper, we presented SECRETA, a system that helps data publishers analyze the performance of anonymization algorithms and make informed decisions on publishing anonymized data. Our system allows evaluating and comparing a range of different algorithms, in an interactive and progressing way. In the future, we will extend our system, by incorporating additional algorithms, such as those in [2].

Acknowledgements

G. Poulis is supported by Heraclitus II, S. Skiadopoulos by EICOS/Thalis, and G. Loukides by a Research Fellowship from the Royal Academy of Engineering.

5. REFERENCES

[1] `http://www-03.ibm.com/software/products/us/en/infosphere-optim-data-privacy/`.

[2] J. Cao, P. Karras, C. Raïssi, and K. Tan. *rho*-uncertainty: Inference-proof transaction anonymization. *PVLDB*, 3(1):1033–1044, 2010.

[3] C. Dai, G. Ghinita, E. Bertino, J.-W. Byun, and N. Li. Tiamat: a tool for interactive analysis of microdata anonymization techniques. *PVLDB*, 2(2), 2009.

[4] B. Fung, K. Wang, and P. Yu. Top-down specialization for information and privacy preservation. In *ICDE*, 2005.

[5] A. Gkoulalas-Divanis and G. Loukides. Utility-guided clustering-based transaction data anonymization. *TDP*, 5(1):223–251, 2012.

[6] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Incognito: efficient full-domain k-anonymity. In *SIGMOD*, 2005.

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

[8] National Institutes of Health, 2013. Data repositories. `http://www.nlm.nih.gov/NIHbmic/nih_data_sharing_repositories.html`.

[9] G. Poulis, G. Loukides, A. Gkoulalas-Divanis, and S. Skiadopoulos. Anonymizing data with relational and transaction attributes. In *ECML/PKDD*, 2013.

[10] M. Terrovitis, N. Mamoulis, and P. Kalnis. Local and global recoding methods for anonymizing set-valued data. *VLDB J.*, 20(1):83–106, 2011.

[11] X. Xiao, G. Wang, and J. Gehrke. Interactive anonymization of sensitive data. In *SIGMOD*, 2009.

[12] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A.-C. Fu. Utility-based anonymization using local recoding. In *KDD*, pages 785–790, 2006.

Anonymizing data with relational and transaction attributes

Giorgos Poulis[1], Grigorios Loukides[2], Aris Gkoulalas-Divanis[3], and Spiros Skiadopoulos[1]

[1] University of Peloponnese {poulis, spiros}@uop.gr
[2] Cardiff University g.loukides@cs.cf.ac.uk
[3] IBM Research - Ireland arisdiva@ie.ibm.com

Abstract. Publishing datasets about individuals that contain both relational and transaction (i.e., set-valued) attributes is essential to support many applications, ranging from healthcare to marketing. However, preserving the privacy and utility of these datasets is challenging, as it requires *(i)* guarding against attackers, whose knowledge spans both attribute types, and *(ii)* minimizing the overall information loss. Existing anonymization techniques are not applicable to such datasets, and the problem cannot be tackled based on popular, multi-objective optimization strategies. This work proposes the first approach to address this problem. Based on this approach, we develop two frameworks to offer privacy, with bounded information loss in one attribute type and minimal information loss in the other. To realize each framework, we propose privacy algorithms that effectively preserve data utility, as verified by extensive experiments.

1 Introduction

Privacy-preserving data mining has emerged to address privacy concerns related to the collection, analysis, and sharing of data and aims at preventing the disclosure of individuals' private and sensitive information from the published data. Publishing datasets containing both relational and transaction attributes, *RT-datasets* for short, is essential in many real-world applications. Several marketing studies, for example, need to find product combinations that appeal to specific types of customers. Consider the RT-dataset in Fig. 1a, where each record corresponds to a customer. Age, Origin and Gender are relational attributes, whereas Purchased-products is a transaction attribute that contains a *set* of *items*, representing commercial transactions. Such studies may require finding all customers below 30 years old who purchased products E and F. Another application is in healthcare, where several medical studies require analyzing patient demographics and diagnosis information together. In such RT-datasets, patients features (e.g., demographics) are modeled as relational attributes and diagnosis as a transaction attribute. In all these applications, the privacy protection of data needs to performed without adding *fake* or removing *truthful* information [5,16]. This precludes the application of ϵ-*differential privacy* [3], which only allows releasing noisy answers to user queries or noisy summary statistics, as well as *suppression* [19], which deletes values prior to data release.

		Relational attributes			Transaction attribute
Id	Name	Age	Origin	Gender	Purchased-products
0	John	19	France	Male	E F B G
1	Steve	22	Greece	Male	E F D H
2	Mary	28	Germany	Female	B C E G
3	Zoe	39	Spain	Female	F D H
4	Ann	70	Algeria	Female	E G
5	Jim	55	Nigeria	Male	A F H

(a)

	Relational attributes			Transaction attribute
Id	Age	Origin	Gender	Purchased-products
0	[19:22]	Europe	Male	E F (A,B,C,D) G
1	[19:22]	Europe	Male	E F (A,B,C,D) H
2	[28:39]	Europe	Female	E (A B,C D) G
3	[28:39]	Europe	Female	F (A,B,C,D) H
4	[55:70]	Africa	All	E G
5	[55:70]	Africa	All	F (A,B,C,D) H

(b)

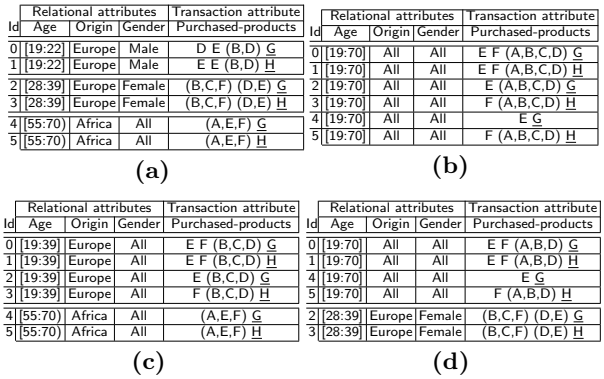
Fig. 1: (a) An RT-dataset with patient demographics and IDs of purchased products, and (b) a 2-anonymous dataset with respect to relational attributes and 2^2-anonymous with respect to the transaction attribute. Identifiers Id and Name are not published.

A plethora of methods can be used to preserve the privacy of datasets containing only relational or only transaction attributes [9,12,15,18]. However, there are currently no methods for anonymizing RT-datasets, and simply anonymizing each attribute type separately, using existing methods (e.g., [9,12,15,18]), is not enough. This is because information concerning *both* relational and transaction attributes may lead to *identity disclosure* (i.e., the association of an individual to their record) [15]. Consider, for example, the dataset in Fig. 1a which is anonymized by applying the methods of [18] and [8] to the relational and transaction attributes, as shown in Fig. 1b. An attacker, who knows that Jim is a 55-year-old Male from Nigeria who purchased F, can associate Jim with record 5 in Fig. 1b. Thwarting identity disclosure is essential to comply with legislation, e.g., HIPAA, and to help future data collection. At the same time, many applications require preventing *attribute disclosure* (i.e., the association of an individual with sensitive information). In medical data publishing, for example, this ensures that patients are not associated with sensitive diagnoses [17].

Furthermore, anonymized RT-datasets need to have minimal information loss in relational and in transaction attributes. However, these two requirements are conflicting, and the problem is difficult to address using multi-objective optimization strategies [1]. In fact, these strategies are either inapplicable or incur excessive information loss, as we show in Section 3.

Contributions. Our work makes the following specific contributions:

- We introduce the problem of anonymizing RT-datasets and propose the first approach to tackle it. Our privacy model prevents an attacker, who knows the set of an individual's values in the relational attributes and up to m items in the transaction attribute, from linking the individual to their record.
- We develop an approach for producing (k, k^m)-anonymous RT-datasets with bounded information loss in one attribute type and minimal information loss in the other. Following this approach, we propose two frameworks which employ *generalization* [15] and are based on a three-phase process: *(i)* creating k-anonymous clusters with respect to the relational attributes, *(ii)* merging these clusters in a way that helps anonymizing RT-datasets with low information loss, and *(iii)* enforcing (k, k^m)-anonymity to each merged cluster.
- We propose a family of algorithms to implement the second phase in each framework. These algorithms operate by building clusters, which can be made (k, k^m)-anonymous with minimal information loss, and preserve different aspects of data utility.



Id	Relational attributes: Age	Origin	Gender	Transaction attribute: Purchased-products
0	[19:22]	Europe	Male	D E (B,D) G
1	[19:22]	Europe	Male	E E (B,D) H
2	[28:39]	Europe	Female	(B,C,F) (D,E) G
3	[28:39]	Europe	Female	(B,C,F) (D,E) H
4	[55:70)	Africa	All	(A,E,F) G
5	[55:70)	Africa	All	(A,E,F) H

(a)

Id	Relational attributes: Age	Origin	Gender	Transaction attribute: Purchased-products
0	[19:70]	All	All	E F (A,B,C,D) G
1	[19:70]	All	All	E F (A,B,C,D) H
2	[19:70]	All	All	E (A,B,C,D) G
3	[19:70]	All	All	F (A,B,C,D) H
4	[19:70]	All	All	E G
5	[19:70]	All	All	F (A,B,C,D) H

(b)

Id	Relational attributes: Age	Origin	Gender	Transaction attribute: Purchased-products
0	[19:39]	Europe	All	E F (B,C,D) G
1	[19:39]	Europe	All	E F (B,C,D) H
2	[19:39]	Europe	All	E (B,C,D) G
3	[19:39]	Europe	All	F (B,C,D) H
4	[55:70)	Africa	All	(A,E,F) G
5	[55:70)	Africa	All	(A,E,F) H

(c)

Id	Relational attributes: Age	Origin	Gender	Transaction attribute: Purchased-products
0	[19:70]	All	All	E F (A,B,D) G
1	[19:70]	All	All	E F (A,B,D) H
4	[19:70]	All	All	E G
5	[19:70]	All	All	F (A,B,D) H
2	[28:39]	Europe	Female	(B,C,F) (D,E) G
3	[28:39]	Europe	Female	(B,C,F) (D,E) H

(d)

Fig. 2: The $(2, 2^2)$-anonymous datasets from applying (a) **R**FIRST, and (b) **T**FIRST to the dataset of Fig. 1a, and (c) $\mathbf{R}\text{MERGE}_R$, and (d) $\mathbf{R}\text{MERGE}_T$, to the clusters of Fig. 2a

– We investigate the effectiveness of our approach by conducting experiments on two real-world RT-datasets. Our results verify that the proposed approach is effective at preserving data utility.

PAPER ORGANIZATION. Section 2 defines concepts used in this work. Section 3 clarifies why popular multi-objective optimization strategies are unsuited for enforcing (k, k^m)-anonymity and formulates the target problems. Sections 4 and 5 present our approach and an instance of it. Sections 6 and 7 present experiments and discuss related work, and Section 8 concludes the paper.

2 *RT*-datasets and their anonymity

RT-DATASETS. An *RT-dataset* D consists of records containing relational attributes $R_1, \ldots, R_v$, which are single-valued, and a transaction attribute T, which is set-valued. For convenience, we consider that: *(i)* identifiers have been removed from D, and *(ii)* a single transaction attribute T is contained in D[1].

(k, k^m)-ANONYMITY. We propose (k, k^m)-anonymity to guard against *identity disclosure*. To prevent both identity and attribute disclosure, (k, k^m)-anonymity can be extended, as we explain in Section 5.

Before defining (k, k^m)-anonymity, we associate each record r in an RT-dataset D with a *group* of records $G(r)$, as shown below.

Definition 1. *For a record $r \in D$, its* group $G(r)$ *is a set of records that contains r and each record $q \in D$, such that $q[R_1, \ldots, R_v] = r[R_1, \ldots, R_v]$ and $q[T] \cap I = r[T] \cap I$, where I is any set of m or fewer items of $r[T]$*[2].

Group $G(r)$ contains r and all records that are indistinguishable from r to an attacker, who knows the values of r in relational attributes *and* up to m items

[1] Multiple transaction attributes $T_1, \ldots, T_u$ can be transformed to a single transaction attribute T, whose domain contains every item in the domain of $T_1, \ldots, T_u$, preceded by the domain name, i.e., $dom(T) = \{d.t \mid d = T_i \text{ and } t \in dom(T_i), i \in [1, u]\}$.

[2] Expression $r[A]$ is a shortcut for the projection $\pi_A(r)$.

in the transaction attribute. The size of $G(r)$, denoted with $|G(r)|$, represents the risk of associating an individual with a record r. Thus, to provide privacy, we may lower-bound $|G(r)|$. This idea is captured by (k, k^m)-anonymity.

Definition 2. *A group of records $G(r)$ is* (k, k^m)-anonymous*, if and only if $|G(r)| \geq k$, for each record r in $G(r)$. An RT-dataset D is (k, k^m)-anonymous, if and only if the group $G(r)$ of each record $r \in D$ is (k, k^m)-anonymous.*

For example, in Fig. 2a groups {0,1} (=G(0)=G(1)), {2,3} (=G(2)=G(3)) and {4,5} (=G(4)=G(5)) are $(2, 2^2)$-anonymous, rendering the whole dataset $(2, 2^2)$-anonymous. Note that in each group, all records have the same values in the relational attributes, as required by Definition 1, but do not necessarily have the same items in the transaction attribute Purchased-products (see Fig. 2b).

The notion of (k, k^m)-anonymity for RT-datasets extends and combines relational k-anonymity [15] and transactional k^m-anonymity [17].

Proposition 1. *Let $D[R_1, \ldots, R_v]$ and $D[T]$ be the relational and transaction part of an RT-dataset D, respectively. If D is (k, k^m)-anonymous, then $D[R_1, \ldots, R_v]$ is k-anonymous and $D[T]$ is k^m-anonymous.*

Proposition 1 shows that (k, k^m)-anonymity provides the same protection as k-anonymity [15], for relational attributes, and as k^m-anonymity [17], for transaction attributes. *Unfortunately, the inverse does not hold.* That is, an RT-dataset may be k and k^m but not (k, k^m)-anonymous. For instance, let D be the dataset of Fig. 1b. Note that D[Age, Origin, Gender] is 2-anonymous and D[Purchased-products] is 2^2-anonymous, but D is not $(2, 2^2)$-anonymous.

Generalization. We employ the generalization functions defined below.

Definition 3. *A* relational generalization function $\mathcal{R}$ *maps a value v in a relational attribute R to a generalized value $\tilde{v}$, which is a range of values, if R is numerical, or a collection of values, if R is categorical.*

Definition 4. *A* transaction generalization function $\mathcal{T}$ *maps an item u in the transaction attribute T to a generalized item $\tilde{u}$. The generalized item $\tilde{u}$ is a non-empty subset of items in T that contains u.*

The way relational values and transactional items are generalized is fundamentally different, as they have different semantics [19]. Specifically, a generalized value bears *atomic* semantics and is interpreted as a *single value* in a range or a collection of values, whereas a generalized item bears *set* semantics and is interpreted as *any non-empty subset* of the items mapped to it [12]. For instance, the generalized value [19:22] in Age, in the record 0 in Fig. 2a, means that the actual Age is in $[19, 22]$. Contrary, the generalized item (B, D) in Purchased-products means that B, or D, or both products were bought. Given a record r, the function $\mathcal{R}$ is applied to a single value $v \in R$, and all records in the k-anonymous group $G(r)$ must have the same generalized value in R. On the other hand, the function $\mathcal{T}$ is applied to one of the potentially many items in T, and the records in the k^m-anonymous $G(r)$ may not have the same generalized items.

DATA UTILITY MEASURES. In this work, we consider two general data utility measures; **Rum**, for relational attributes, and **Tum**, for the transaction attribute. These measures satisfy Properties 1, 2 and 3.

Property 1. Lower values in **Rum** and **Tum** imply better data utility.

Property 2. **Rum** is *monotonic* to subset relationships. More formally, given two groups G and G' having at least k records, and a relational generalization function $\mathcal{R}$, it holds that $\mathbf{Rum}(\mathcal{R}(G) \cup \mathcal{R}(G')) \leq \mathbf{Rum}(\mathcal{R}(G \cup G'))$.

Property 2 suggests that data utility is preserved better, when we generalize the relational values of small groups, and is consistent with prior work on relational data anonymization [2,6]. Intuitively, this is because the group $G \cup G'$ contains more distinct values in a relational attribute R than G or G', and thus more generalization is needed to make its values indistinguishable.

A broad class of measures, such as *NCP*, the measures expressed as Minkowski norms [6], *Discernability* [1], and the *Normalized average equivalence class size metric* [9], satisfy Property 2 [6], and can be used as **Rum**.

Property 3. **Tum** is *anti-monotonic* to subset relationships. More formally, given two groups G and G' having at least k records, and a transaction generalization function $\mathcal{T}$ that satisfies Definition 4 and *(i)* maps each item in the group it is applied to a generalized item that is not necessarily unique, and *(ii)* constructs the mapping with the minimum **Tum**, it holds that $\mathbf{Tum}(\mathcal{T}(G) \cup \mathcal{T}(G')) \geq \mathbf{Tum}(\mathcal{T}(G \cup G'))$.

Property 3 suggests that generalizing large groups can preserve transaction data utility better, and is consistent with earlier works [12,17]. Intuitively, this is because, all mappings between items and generalized items constructed by $\mathcal{T}$ when applied to G and G' separately (Case I) can also be constructed when $\mathcal{T}$ is applied to $G \cup G'$ (Case II), but there can be mappings that can only be considered in Case II. Thus, the mapping with the minimum **Tum** in Case I cannot have lower **Tum** than the corresponding mapping in Case II.

3 Challenges of enforcing (k, k^m)-anonymity

LACK OF OPTIMAL SOLUTION. Constructing a (k, k^m)-anonymous RT-dataset D with minimum information loss is far from trivial. Lemma 1 follows from Theorem 1 and shows that there is no (k, k^m)-anonymous version of D with minimum (i.e., optimal) **Rum** and **Tum**, for any D of realistic size.

Theorem 1. *Let $\mathcal{D}_R$ and $\mathcal{D}_T$ be the optimal (k, k^m)-anonymous version of an RT-dataset D with respect to* **Rum** *and* **Tum***, respectively. Then, no group in $\mathcal{D}_R$ contains more than $2k-1$ records, and $\mathcal{D}_T$ is comprised of a single group.*

Proof. (Sketch) The proof that no group in $\mathcal{D}_R$ contains more than $2k-1$ records is based on Property 2, and has been given in [6]. The proof that $\mathcal{D}_T$ is comprised of a single group is similar and, it is based on Property 3.

Lemma 1. *There is no optimal (k, k^m)-anonymous version $\mathcal{D}$ of an RT-dataset D with respect to both* **Rum** *and* **Tum**, *unless* $|D| \in [k, 2k-1]$.

Inadequacy of popular optimization strategies. Constructing useful (k, k^m)-anonymous RT-datasets requires minimizing information loss with respect to both **Rum** and **Tum**. Such multi-objective optimization problems are typically solved using the *lexicographical*, the *conventional weighted-formula*, or the *Pareto optimal* approach [4]. We will highlight why these approaches are not adequate for our problem.

Lexicographical. In this approach, the optimization objectives are ranked and optimized in order of priority. In our case, we can prioritize the lowering of information loss in *(i)* the relational attributes (i.e., minimal **Rum**), or *(ii)* the transaction attribute (i.e., minimal **Tum**).

Given an RT-dataset D and anonymization parameters k and m, an algorithm that implements strategy *(i)* is **R**first. This algorithm partitions D into a set of k-anonymous groups $\mathcal{C}$, with respect to the relational attributes (e.g., using [18]), and applies $\mathcal{T}$ to generalize items in each group of records in $\mathcal{C}$, separately (e.g., using [17]). Symmetrically, to implement strategy *(ii)*, we may use an algorithm **T**first, which first partitions D into a set of k^m-anonymous groups (e.g., using the LRA algorithm [17]), and then applies a relational generalization function (see Definition 3) to each relational attribute, in each group.

Both **R**first and **T**first enforce (k, k^m)-anonymity, but produce vastly different results. For instance, Figs. 2a and 2b show $(2, 2^2)$-anonymous versions of the dataset in Fig. 1a, produced by **R**first and **T**first, repectively. Observe that **R**first did not generalize the relational attributes as heavily as **T**first but applied more generalization to the transaction attribute. This is because, **R**first constructs small groups, and does not control the grouping of items. Contrary, the groups created by **T**first contain records, whose items are not heavily generalized, unlike their values in the relational attributes. In either case, the purpose of producing anonymized RT-datasets that allow meaningful analysis of relational and transaction attributes together, is defeated.

Conventional weighted-formula. In this approach, all objectives are combined into a single one, using a weighted formula. The combined objective is then optimized by a single-objective optimization algorithm. For example, a clustering-based algorithm [13] would aim to minimize the weighted sum of **Rum** and **Tum**. However, this approach works only for *commensurable* objectives [4]. This is not the case for **Rum** and **Tum**, which are fundamentally different and have different properties (see Section 2). Therefore, this approach is not suitable.

Pareto optimal. This approach finds a set of solutions that are *non-dominated* [4], from which the most appropriate solution is selected by the data publisher, according to their preferences. However, the very large number of non-dominated solutions that can be constructed by flexible generalization functions, such as those in Definitions 3 and 4, render this approach impractical.

Problem formulation. To construct a (k, k^m)-anonymous version of an RT-dataset, we *either* upper-bound the information loss in relational attributes and

Algorithm: **Rum**-BOUND

```
// Initial cluster formation
{C_1, ..., C_n} := ClusterFormation(D, k)
D := {C_1, ..., C_n}
if Rum(D) > δ then return false
// Cluster merging
D := RMerge(D, T, δ)
// (k, k^m)-anonymization
for each cluster C ∈ D do
    D := (D \ C) ∪ T(C)
return D
```

Algorithm: **Tum**-BOUND

```
// Initial cluster formation
{C_1, ..., C_n} := ClusterFormation(D, k)
D := {C_1, ··· , C_n}
if Tum(T(D)) ≤ δ then return D
// Cluster merging
D := TMerge(D, T, δ)
// (k, k^m)-anonymization
for each cluster C ∈ D do
    D := (D \ C) ∪ T(C)
if Tum(D) > δ then return false
return D
```

seek to minimize the information loss in the transaction attribute (Problem 1), *or* upper-bound the information loss in the transaction attribute and seek to minimize the information loss in relational attributes (Problem 2).

Problem 1. Given an RT-dataset D, data utility measures **Rum** and **Tum**, parameters k and m, and a threshold δ, construct a (k, k^m)-anonymous version $\mathcal{D}$ of D, such that $\mathbf{Rum}(\mathcal{D}) \leq \delta$ and $\mathbf{Tum}(\mathcal{D})$ is minimized.

Problem 2. Given an RT-dataset D, data utility measures **Rum** and **Tum**, parameters k and m, and a threshold δ, construct a (k, k^m)-anonymous version $\mathcal{D}$ of D, such that $\mathbf{Tum}(\mathcal{D}) \leq \delta$ and $\mathbf{Rum}(\mathcal{D})$ is minimized.

Threshold δ must be specified by data publishers, as in [6]. Thus, constructing $\mathcal{D}$ might be infeasible for an arbitrary δ. Solving Problem 1 or Problem 2 ensures that $\mathcal{D}$ preserves privacy and utility, but it is NP-hard (proof follows from [12]).

4 Anonymization approach

We propose an approach that overcomes the deficiencies of the aforementioned optimization approaches and works in three phases:

Initial cluster formation: k-anonymous clusters with respect to relational attributes, which incur low information loss, are formed.

Cluster merging: Clusters are merged until the conditions set by Problems 1 or 2 are met.

(k, k^m)-*anonymization*: Each cluster becomes (k, k^m)-anonymous, by generalizing the its items with low **Tum**.

Based on our approach, we developed two anonymization frameworks, **Rum**-BOUND and **Tum**-BOUND, which address Problems 1 and 2, respectively. **Rum**-BOUND seeks to produce a dataset with minimal **Tum** and acceptable **Rum**, and implements the phases of our approach, as follows.

Initial cluster formation (Steps 1–3): Algorithm **Rum**-BOUND clusters D, using a function CLUSTERFORMATION, which can be implemented by any generalization-based k-anonymity algorithm [9,18,2]. This function produces a set of k-

anonymous clusters $C_1, \ldots, C_n$, from which a dataset $\mathcal{D}$ containing $C_1, \ldots, C_n$, is created (Step 2). The dataset $\mathcal{D}$ must have a lower **Rum** than δ, since subsequent steps of the algorithm cannot decrease **Rum** (see Property 2). If the dataset $\mathcal{D}$ does not satisfy this condition, it cannot be a solution to Problem 1, and false is returned (Step 3).

Cluster merging (Step 4): This phase is the crux of our framework. It is performed by a function **R**MERGE, which merges the clusters of $\mathcal{D}$ to produce a version that can be (k, k^m)-anonymized with minimal **Tum** and without violating δ. To implement **R**MERGE we propose three algorithms, namely **R**MERGE$_R$, **R**MERGE$_T$ and **R**MERGE$_{RT}$, which aim at minimizing **Tum** using different heuristics.

(k, k^m)-anonymization (Steps 5–7): In this phase, $\mathcal{D}$ is made (k, k^m)-anonymous, by applying a transaction generalization function $\mathcal{T}$ to each of its clusters.

Tum-BOUND, on the other hand, focuses on Problem 2 and aims at creating a dataset with minimal **Rum** and acceptable **Tum**. This framework has the following major differences from **Rum**-BOUND.

- At Step 3, after the formation of $\mathcal{D}$, **Tum**-BOUND checks if $\mathcal{D}$ has lower **Tum** than the threshold δ. In such case, $\mathcal{D}$ is a solution to Problem 2.
- At Step 4, function **T**MERGE merges clusters until the **Tum** threshold is reached, or no more merging is possible. To implement **T**MERGE we propose three algorithms: **T**MERGE$_R$, **T**MERGE$_R$ and **T**MERGE$_{RT}$, which aim at minimizing **Rum** using different heuristics.
- At Step 7, **Tum**-BOUND checks if $\mathbf{Tum}(\mathcal{D}) > \delta$; in this case, we cannot satisfy Problem 2 conditions and, thus, return false.

CLUSTER-MERGING ALGORITHMS. We now present three algorithms that implement function **R**MERGE, which is responsible for the merging phase of **Rum**-BOUND (Step 4). Our algorithms are based on different merging heuristics. Specifically, **R**MERGE$_R$ merges clusters with similar relational values, **R**MERGE$_T$ with similar transaction items and **R**MERGE$_{RT}$ takes a middle line between these two algorithms. In all cases, relational generalization is performed by a set of functions $\mathcal{G} = \{\mathcal{L}_1, \ldots, \mathcal{L}_v\}$, one for each relational attribute (Definition 3) and transaction generalization is performed by function $\mathcal{T}$ (Definition 4).

RMERGE$_R$ selects the cluster C with the minimum $\mathbf{Rum}(C)$ as a seed (Step 2). Cluster C contains relational values that are not highly generalized and is expected to be merged with a low relational utility loss. The algorithm locates the cluster C' with the most similar relational values to C (Step 3) and constructs a temporary dataset $\mathcal{D}_{tmp}$ that reflects the merging of C and C' (Step 4). If $\mathcal{D}_{tmp}$ does not violate the **Rum** threshold, it is assigned to $\mathcal{D}$ (Step 5).

RMERGE$_T$ starts by selecting the same seed C as **R**MERGE$_R$ (Step 2) and seeks a cluster C' that contains similar transaction items to C and, when merged with C, results in a dataset with **Rum** no higher than δ. To this end, **R**MERGE$_T$ merges C with every other cluster C_i in $\mathcal{D} \backslash C$ and orders the clusters by increasing $\mathbf{Tum}(\mathcal{T}(C \cup C_i))$ (Step 3). This allows efficiently finding the best merging for minimizing **Tum** that does not violate $\mathbf{Rum}(\mathcal{D}) \leq \delta$. The algorithm considers

Algorithm: **R**MERGE$_R$

```
1  while D changes do
2      Select, as a seed, the cluster C ∈ D with minimum Rum(C)
3      Find the cluster C' ∈ D that minimizes Rum(G(C ∪ C')) .
4      D_tmp := ((D \ C) \ C') ∪ G(C ∪ C')
5      if Rum(D_tmp) ≤ δ then
           D := D_tmp
6  return D
```

Algorithm: **R**MERGE$_T$

```
1  while D changes do
2      Select, as a seed, the cluster C ∈ D with minimum Rum(C)
       // Find the appropriate cluster C' to be merged with C
3      Let {C_1, ..., C_t} be the set of clusters in D \ C ordered by increasing
       Tum(T(C ∪ C_i)), i ∈ [1, t)
4      for i := 1 to t do          // Test if C' = C_i
5          D_tmp := ((D \ C) \ C_i) ∪ G(C ∪ C_i)
6          if Rum(D_tmp) ≤ δ then   // C' is C_i
7              D := D_tmp
8              exit the for loop
9  return D
```

Algorithm: **R**MERGE$_{RT}$

```
1  while D changes do
2      Select, as a seed, the cluster C ∈ D with minimum Rum(C)
3      Let {C_1, ..., C_t} (resp. {Ĉ_1, ..., Ĉ_t}) be the set of clusters in D \ C ordered by
       increasing Rum(G(C ∪ C_i)) (resp. Tum(T(C ∪ Ĉ_i))), i ∈ [1, t)
       // Find the appropriate cluster C' to be merged with C
4      for i := 1 to t do
5          Find cluster C', that has the i-th minimum sum of indices u + v s.t.
           C_u ∈ {C_1, ..., C_t} and C_v ∈ {Ĉ_1, ..., Ĉ_t}
6          D_tmp := ((D \ C) \ C_i) ∪ G(C ∪ C_i)
7          if Rum(D_tmp) ≤ δ then
8              D := D_tmp
9              exit the for loop
10 return D
```

the clusters with increasing **Tum**$(\mathcal{T}(C \cup C_i))$ scores. The first cluster that gives a dataset with acceptable **Rum** is used for merging (Steps 4–5).

RMERGE$_{RT}$ combines the benefits of **R**MERGE$_R$ and **R**MERGE$_T$. It selects the same seed cluster C as **R**MERGE$_T$, and constructs two orderings, which sort the generalized merged clusters in ascending order of **Rum** and **Tum**, respectively (Step 3). Then, a cluster C' that is as close as possible to C, based on both orderings (i.e., it has the i-th minimum sum $(u + v)$, where u and v are the indices of C' in the $\{C_1, \ldots, C_t\}$ and orderings $\{\hat{C}_1, \ldots, \hat{C}_t\}$ repsectively), is found (Step 5). The next steps of **R**MERGE$_{RT}$ are the same as in **R**MERGE$_T$.

We now discuss **T**MERGE$_R$, **T**MERGE$_R$, and **T**MERGE$_{RT}$, used in **Tum**-BOUND. These algorithms perform cluster merging, until $\mathcal{D}$ satisfies the **Tum** threshold, or all possible mergings have been considered. The pseudocode of **R**MERGE$_R$ is the same as that of **T**MERGE$_R$, except that Step 5 in **R**MERGE$_R$ is replaced by the following steps. Note that D is returned if it satisfies the **Tum** threshold, because **Rum** cannot be improved by further cluster merging (Property 2).

```
5  if Tum(D_tmp) ≤ δ then
■      D := D_tmp
■      return D
```

The pseudocode of **T**MERGE$_R$ and **T**MERGE$_{RT}$ can be derived by replacing the same steps with Steps 5 and 7 in **T**MERGE$_R$ and **T**MERGE$_{RT}$, respectively.

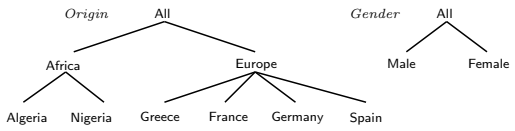


Fig. 3: Hierarchies for the dataset of Fig. 1a

The runtime cost of anonymization is $O(\mathcal{F} + |\mathcal{C}|^2 \cdot (\mathcal{K}_\mathcal{R} + \mathcal{K}_\mathcal{T}))$, where $\mathcal{F}$ is the cost for initial cluster formation, $|\mathcal{C}|$ the number of clusters in $\mathcal{D}$, and $\mathcal{K}_\mathcal{R}$ and $\mathcal{K}_\mathcal{T}$ the cost of generalizing the relational and transaction part of a cluster.

5 Instantiating and extending the frameworks

Our frameworks can be parameterized by generalization functions, data utility measures, and initial cluster formation algorithms. This section presents such instantiations and strategies to improve their efficiency, as well as extensions of our frameworks to prevent both identity and attribute disclosure.

GENERALIZATION FUNCTIONS. We employ the *local recoding* [18] and *set-based generalization* [8,12]. As an example, the dataset in Fig. 1b has been created by applying these functions to the dataset in Fig. 1a, using the hierarchies in Fig. 3.

DATA UTILITY MEASURES. To measure data utility in relational and transaction attributes, we used *Normalized Certainty Penalty* (*NCP*) [18] and *Utility Loss* (*UL*) [12], respectively. The *NCP* for a generalized value $\tilde{v}$, a record r, and an RT-dataset D, is defined as: $NCP_R(\tilde{v}) = \begin{cases} 0, |\tilde{v}| = 1 \\ |\tilde{v}|/|R|, \text{otherwise} \end{cases}$, $NCP(r) = \sum_{i \in [1,v]} w_i \cdot NCP_{R_i}(r[R_i])$ and $NCP(D) = \frac{\sum_{r \in D} NCP(r)}{|D|}$ resp., where $|R|$ denotes the number of leaves in the hierarchy for a categorical attribute R (or domain size for a numerical attribute R), $|\tilde{v}|$ denotes the number of leaves of the subtree rooted at $\tilde{v}$ in the hierarchy for a categorical R (or the length of the range for a numerical R), and $w_i \in [0, 1]$ is a weight that measures the importance of an attribute. The *UL* for a generalized item $\tilde{u}$, a record r, and an RT-dataset D, is defined as: $UL(\tilde{u}) = (2^{|\tilde{u}|} - 1) \cdot w(\tilde{u})$, $UL(r) = \frac{\sum_{\forall \tilde{u} \in r} UL(\tilde{u})}{2^{\sigma(r)} - 1}$ and $UL(D) = \frac{\sum_{\forall r \in D} UL(r)}{|D|}$ resp., where $|\tilde{u}|$ is the number of items mapped to $\tilde{u}$, $w(\tilde{u}) \in [0, 1]$ a weight reflecting the importance of $\tilde{u}$ [12], and $\sigma(r)$ the sum of sizes of all generalized items in r.

INITIAL CLUSTER FORMATION WITH CLUSTER. The initial cluster formation phase should be implemented using algorithms that create many small clusters, with low **Rum**, because this increases the chance of constructing a (k, k^m)-anonymous dataset with good data utility. Thus, we employ CLUSTER, an algorithm that is instantiated with *NCP* and local recoding, and it is inspired by the algorithm in [2]. The time complexity of CLUSTER is $O(\frac{|D|^2}{k} \cdot \log(|D|))$.

EFFICIENCY OPTIMIZATION STRATEGIES. To improve the efficiency of cluster-merging algorithms, we compute $\mathbf{Rum}(\mathcal{D}_{tmp})$ incrementally, thereby avoiding to access all records in $\mathcal{D}_{tmp}$, after a cluster merging. This can be performed for all measures in Section 2, but we illustrate it for *NCP*. We use a list λ of tuples $<|C|, NCP(r_c))>$, for each cluster C in $\mathcal{D}_{tmp}$ and any record r_c in C, which is initialized based on $\mathcal{D}$. Observe that $NCP(\mathcal{D}_{tmp}) = \frac{\sum_{\forall C \in \mathcal{D}_{tmp}} (|C| \cdot NCP(r_c))}{|D|}$, and

Algorithm: CLUSTER

$\mathcal{C} := \emptyset$
// Create clusters of size k
while $|D| \geq k$ **do**
  Select, as a seed, a random record s from D
  Add s and each record $r \in D$ having one of the lowest $k-1$ values in $NCP(\mathcal{G}(\{s, r\}))$ to cluster C
  Add cluster C to $\mathcal{C}$ and remove its records from D
// Accommodate the remaining $|D|$ mod k records
for *each record* $r \in D$ **do**
  Add r to the cluster $C \in \mathcal{C}$ that minimizes $NCP(\mathcal{G}(C \cup r))$
Apply $\mathcal{G}$ to the relational values of each cluster in $\mathcal{C}$
// Extend clusters
for *each cluster* $C \in \mathcal{C}$ **do**
  Let S be the set of clusters in $\mathcal{C}$ with the same values in relational attributes as C.
  Extend C with the records of S and remove each cluster in S from $\mathcal{C}$.
return $\mathcal{C}$

it can be updated, after C and C' are merged, by adding: $\frac{(|C|+|C'|)\cdot NCP(r_{c\cup c'})}{|D|} - \frac{|C|\cdot NCP(r_c)-|C'|\cdot NCP(r_{c'})}{|D|}$. This requires accessing only the records in $C \cup C'$.

The efficiency of $\mathbf{R}$MERGE$_T$, $\mathbf{R}$MERGE$_{RT}$, $\mathbf{T}$MERGE$_R$, and $\mathbf{T}$MERGE$_{RT}$ can be further improved by avoiding computing $\mathbf{Tum}(\mathcal{T}(C \cup C_1)), \ldots, \mathbf{Tum}(\mathcal{T}(C \cup C_t))$. For this purpose, we merge clusters using *Bit-vector Transaction Distance* (*BTD*). The *BTD* for records r_1, r_2 is defined as $BTD(r_1, r_2) = \frac{ones(b_1 \veebar b_2)+1}{ones(b_1 \wedge b_2)+1}$. $ones(b_1 \vee b_2)$, where b_1 and b_2 are the bit-vector based representations of $r_1[T]$ and $r_2[T]$, $\veebar$, $\wedge$ and $\vee$ are the Boolean operators, for XOR, AND, and OR, and the function *ones* counts the number of 1 bits in a bit-vector. The *BTD* of a cluster C is defined as $BTD(C) = \max\{BTD(r_1, r_2)|$ for all $r_1, r_2 \in C\}$. *BTD* helps enforcing (k, k^m)-anonymity with minimal **Tum**, as it favors the grouping of records with a small number of items, many of which are common.

PREVENTING BOTH IDENTITY AND ATTRIBUTE DISCLOSURE. To prevent both types of disclosure, we propose the concept of (k, ℓ^m)-diversity, defined below.

Let $G(r)$ be a group of records and $G(r')$ be a group with the same records as $G(r)$ projected over $\{R_1, \ldots, R_v, T'\}$, where T' contains only the nonsensitive items in T. $G(r)$ is (k, ℓ^m)-*diverse*, if and only if $G(r')$ is (k, k^m)-anonymous, and an attacker, who knows up to m nonsensitive items about an individual, cannot associate any record in $G(r)$ to any combination of sensitive items, with a probability greater than $\frac{1}{\ell}$. An RT-dataset D is (k, ℓ^m)-diverse, if and only if the group $G(r)$ of each record $r \in D$ is (k, ℓ^m)-diverse.

(k, ℓ^m)-diversity forestalls identity disclosure, and, additionally, the inference of any combination of sensitive items, based on ℓ^m-diversity [17]. Extending our anonymization frameworks to enforce (k, ℓ^m)-diversity requires: *(i)* applying **Tum** to nonsensitive items, and *(ii)* replacing the transaction generalization function $\mathcal{T}$, which enforces k^m-anonymity to each cluster, with one that applies ℓ^m-diversity. The ℓ^m-diversity version of AA [17] was used as such a function.

6 Experimental evaluation

In this section, we evaluate our algorithms in terms of data utility and efficiency, and demonstrate the benefit of choices made in their design.

Dataset	$\|D\|$	Rel. att.	$\|dom(T)\|$	Max, Avg # items/record
INFORMS	36553	5	619	17, 4.27
YOUTUBE	131780	6	936	37, 6.51

Table 1: Description of the datasets

EXPERIMENTAL SETUP. We implemented all algorithms in C++ and applied them to INFORMS (https://sites.google.com/site/informsdataminingcontest) and YOUTUBE (http://netsg.cs.sfu.ca/youtubedata) datasets, whose characteristics are shown in Table 1[4]. The default parameters were k=25, m=2, and δ=0.65, and hierarchies were created as in [17]. Our algorithms are referred to in abbreviated form (e.g., $\mathbf{RM}_R$ for $\mathbf{RMERGE}_R$) and were not compared against prior works, since they cannot (k, k^m)-anonymize RT-datasets. The algorithms that enforce (k, ℓ^m)-diversity are named after those based on (k, k^m)-anonymity. All experiments ran on an Intel i5 at 2.7 GHz with 8 GB of RAM.

DATA UTILITY. We evaluated data utility on INFORMS and YOUTUBE using k=25 and k=100, respectively, and varied δ in $[X, 1)$, where X is the NCP of the dataset produced by CLUSTER, for **Rum**-BOUND, or the UL, for **Tum**-BOUND. Data utility is captured using ARE [9,12,16], which is invariant of the way our algorithms work and reflects the average number of records that are retrieved incorrectly, as part of query answers. We used workloads of 100 queries, involving relational, transaction, or both attribute types, which retrieve random values and/or sets of 2 items by default [9,12]. Low ARE scores imply that anonymized data can be used to accurately estimate the number of co-occurrences of relational values and items. This statistic is an important building block of several data mining models.

Figs. 4a to 4g demonstrate the conflicting objectives of minimizing information loss in relational and transaction attributes, and that **Rum**-BOUND can produce useful data. By comparing Fig. 4a with 4c, and Fig. 4d with 4g, it can be seen that a small δ forces all algorithms to incur low information loss in the relational attributes, whereas a large δ favors the transaction attribute. Also, NCP is at most δ, in all tested cases, and data remain useful for queries involving both attribute types (see Figs. 4b, 4e, and 4f). We performed the same experiments for the **Tum**-BOUND and present a subset of them in Fig. 4h. Note that, increasing δ (i.e., the bound for UL), favors relational data, and that the information loss in the transaction attribute is low. Similar observations can be made for the (k, l^m)-diversity algorithms (see Fig. 5).

Next, we compared $\mathbf{RM}_R$, $\mathbf{RM}_T$, and $\mathbf{RM}_{RT}$. As shown in Fig. 4, $\mathbf{RM}_R$ incurred the lowest information loss in the transaction attribute, and the highest in the relational attributes, and $\mathbf{RM}_T$ had opposite trends. $\mathbf{RM}_{RT}$ allows more accurate query answering than $\mathbf{RM}_R$, in relational attributes, and than $\mathbf{RM}_T$, in the transaction attribute, as it merges clusters, based on both attribute types. Similar results were obtained for YOUTUBE (see Figs. 4d-4g), from comparing $\mathbf{TM}_T$,

[4] INFORMS contains the relational attributes {*month of birth, year of birth, race, years of education, income*}, and the transaction attribute *diagnosis_codes*. YOUTUBE contains the relational attributes {*age, category, length, rate, #ratings, #comments*}, and the transaction attribute *related_videos*.

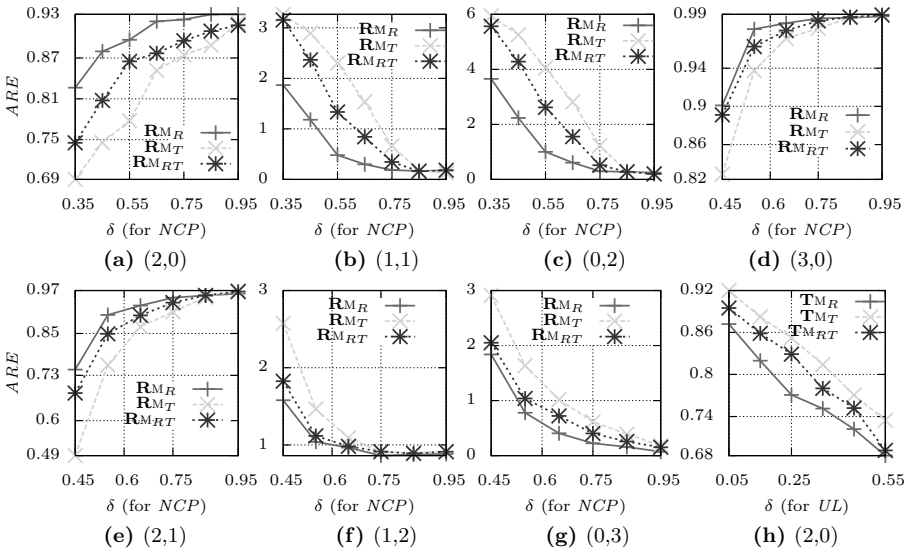


Fig. 4: ARE for queries involving (x, y) relational values and items. Figs. (a)-(c) are for INFORMS; (d)-(g) for YOUTUBE (**Rum**-BOUND). Fig. (h) is for INFORMS (**Tum**-BOUND)

$\mathbf{T}\text{M}_R$, and $\mathbf{T}\text{M}_{RT}$ (see e.g., Fig. 4h), and from comparing the (k, l^m)-diversity algorithms (see Figs. 5). Figs. 6a and 6b show the size of the largest cluster created by $\mathbf{R}\text{M}_R$, $\mathbf{R}\text{M}_T$, and $\mathbf{R}\text{M}_{RT}$, for varying δ. $\mathbf{R}\text{M}_R$ created the largest clusters, as it merges many clusters with similar relational values. These clusters have low *UL*, as shown in Figs. 6c and 6d. Furthermore, Figs. 6a and 6c, show that $\mathbf{R}\text{M}_{RT}$ created slightly larger clusters than $\mathbf{R}\text{M}_T$, which have lower *UL* scores. The results for $\mathbf{T}\text{M}_T$, $\mathbf{T}\text{M}_R$, and $\mathbf{T}\text{M}_{RT}$ and the (k, l^m)-diversity algorithms were similar.

EFFICIENCY. We studied the impact of dataset size using random subsets of INFORMS, whose records were contained in all larger sets. As can be seen in Fig. 7a, $\mathbf{R}\text{M}_T$ outperformed $\mathbf{R}\text{M}_R$ and $\mathbf{R}\text{M}_{RT}$, and it was more scalable, due to the use of the *BTD* measure. $\mathbf{R}\text{M}_{RT}$ was the slowest, because it computes two cluster orderings. $\mathbf{T}\text{M}_T$, $\mathbf{T}\text{M}_R$, and $\mathbf{T}\text{M}_R$ perform similarly to $\mathbf{R}\text{M}_R$, $\mathbf{R}\text{M}_T$, and $\mathbf{R}\text{M}_{RT}$ (their results were omitted). Fig. 7a shows the cost of CL. We also studied the impact of k using the largest dataset of the previous experiment. Fig. 7b shows that the runtime of $\mathbf{R}\text{M}_R$, $\mathbf{R}\text{M}_T$, and $\mathbf{R}\text{M}_{RT}$ improves with k, as fewer clusters are merged. $\mathbf{R}\text{M}_T$ was up to 2.2 times more efficient than $\mathbf{R}\text{M}_R$ and $\mathbf{R}\text{M}_{RT}$ was the least efficient. Fig. 7b shows that the runtime of CL improves with k. The cost of the (k, l^m)-diverse algorithms was similar (omitted).

BENEFITS OF ALGORITHMIC CHOICES. To show that *BTD* helps efficiency without degrading data utility, we developed the baseline algorithms $\mathbf{R}\text{M}_{T\,UL}$, $\mathbf{R}\text{M}_{T\,UL}$, $\mathbf{T}\text{M}_{T\,UL}$, and $\mathbf{T}\text{M}_{RT\,UL}$, which do not perform the optimization of Section 5. Due to their high runtime, a subset of INFORMS with $4K$ records was used. Observe in Figs. 7c and 7e that $\mathbf{R}\text{M}_T$ and $\mathbf{R}\text{M}_{RT}$ have the same *UL* scores with their

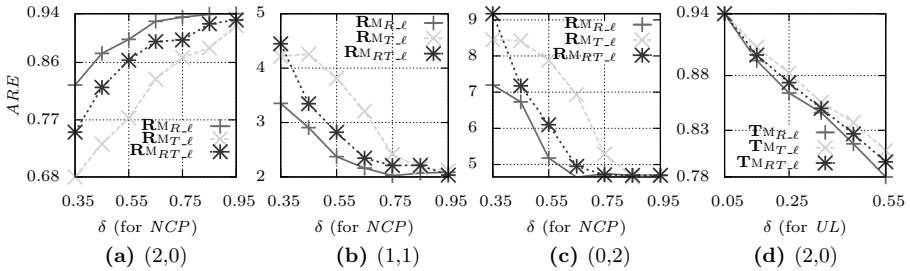


Fig. 5: ARE for queries involving (x, y) relational values and items. Figs. (a)-(c) are for INFORMS (**Rum**-BOUND); Fig. (d) is for INFORMS (**Tum**-BOUND)

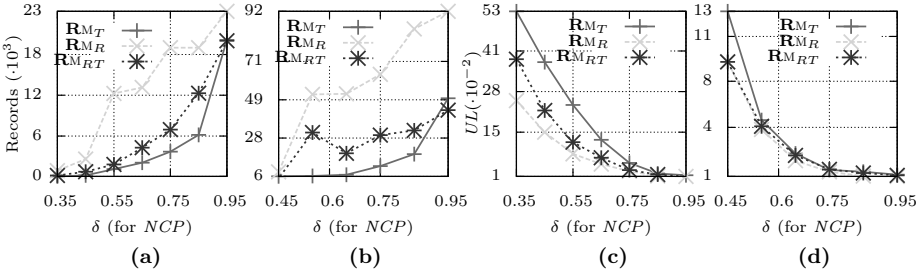


Fig. 6: Max. cluster size for (a) INFORMS, and (b) YOUTUBE, and *UL* for (c) INFORMS and (d) YOUTUBE (**Rum**-BOUND)

corresponding baseline algorithms, but are at least 10 times more efficient and scalable with respect to δ. Similar observations can be made from Figs. 7d and 7f, for $\mathbf{T}\text{M}_R$ and $\mathbf{T}\text{M}_{RT}$. Last, we show that *UL* decreases monotonically, as our algorithms merge clusters. Figs. 7g-7h show the results with $\delta = 1$ for the dataset used in the previous experiment. The fact that *UL* never increases shows that avoiding to compute $UL(\mathcal{T}(\mathcal{D}_{tmp}))$ after a cluster merge does not impact data utility but helps efficiency. The (k, l^m)-diversity algorithms performed similarly.

7 Related work

Preventing identity disclosure is crucial in many real-world applications [5,11] and can be achieved through k-anonymity [15]. This privacy principle can be enforced through various generalization-based algorithms (see [5] for a survey). Thwarting *attribute disclosure* may additionally be needed [14,19,17], and this can be achieved by applying other privacy models, such as l-diversity [14], together with k-anonymity.

Privacy-preserving transaction data publishing requires new privacy models and algorithms, due to the high dimensionality and sparsity of transaction data [19,7,17]. k^m-anonymity is a model for protecting transaction data against attackers, who know up to m items about an individual [17]. Under this condition, which is often satisfied in applications [17,16,11], an individual cannot be associated with fewer than k records in the dataset. k^m-anonymity can be enforced using several algorithms [17,12,8], which can be incorporated into our frameworks. However, k^m-anonymity does not guarantee protection against stronger

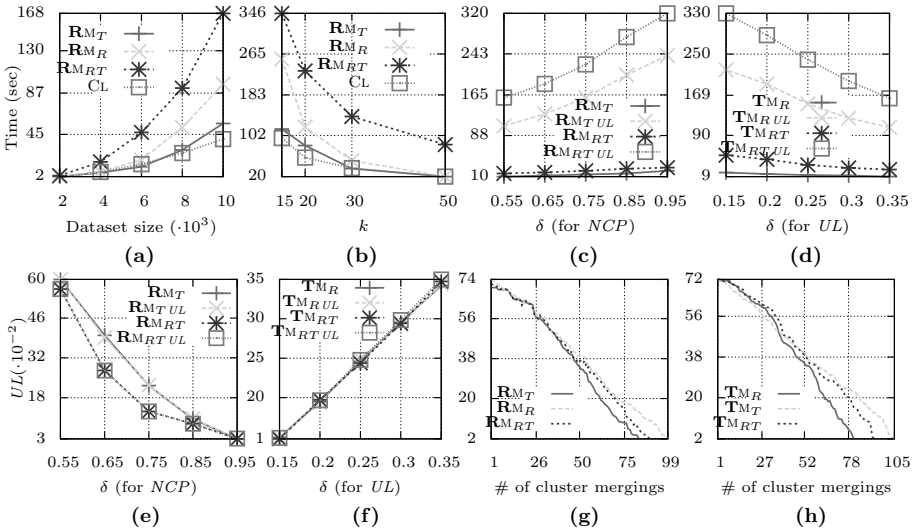


Fig. 7: Runtime (a) vs. $|D|$ and (b) vs. k. Impact of using BTD on runtime for (c) **Rum**-BOUND and (d) (**Tum**-BOUND), and on UL for (e) **Rum**-BOUND and (f) **Tum**-BOUND. UL vs. number of cluster mergings for (g) **Rum**-BOUND, and (h) **Tum**-BOUND

attackers, who know that an individual is associated with exactly certain items [17,16]. This is because, by excluding records that have exactly these items from consideration, the attackers may be able to increase the probability of associating an individual with their record to greater than $\frac{1}{k}$ (although not necessarily 1). A recent method [20] can guard against such attackers while preserving data utility based on a *nonreciprocal recoding* anonymization scheme. To thwart both identity and attribute disclosure in transaction data publishing, [17] proposes ℓ^m-diversity, which we also employ in our frameworks.

Our frameworks employ generalization, which incurs lower information loss than suppression [17] and helps preventing identity disclosure, contrary to bucketization [7]. Also, we seek to publish record-level and truthful data. Thus, we do not employ ϵ-differential privacy [3], nor disassociation [16]. However, the relationship between (k, k^m)-anonymization and relaxed differential privacy definitions is worth investigating to strengthen protection. For instance, Li et al. [10] proved that *safe* k-anonymization algorithms, which perform data grouping and recoding in a differentially private way, can satisfy a relaxed version of differential privacy when preceded by a random sampling step.

8 Conclusions

In this paper, we introduced the problem of anonymizing RT-datasets and proposed the first approach to protect such datasets, along with two frameworks for enforcing it. Three cluster-merging algorithms were developed, for each framework, which preserve different aspects of data utility. Last, we showed how our approach can be extended to prevent both identity and attribute disclosure.

Acknowledgements

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

References

1. R.J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *ICDE*, pages 217–228, 2005.
2. J-W. Byun, A. Kamra, E. Bertino, and N. Li. Efficient k-anonymization using clustering techniques. In *DASFAA*, pages 188–200, 2007.
3. C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.
4. A.A. Freitas. A critical review of multi-objective optimization in data mining: a position paper. *SIGKDD Explorations*, 6(2):77–86, 2004.
5. B.C.M. Fung, K. Wang, R. Chen, and P.S. Yu. Privacy-preserving data publishing: A survey on recent developments. *ACM Comput. Surv.*, 42, 2010.
6. G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. A framework for efficient data anonymization under privacy and accuracy constraints. *TODS*, 34(2), 2009.
7. G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In *ICDE*, pages 715–724, 2008.
8. A. Gkoulalas-Divanis and G. Loukides. Utility-guided clustering-based transaction data anonymization. *Trans. on Data Privacy*, 5(1):223–251, 2012.
9. K. LeFevre, D.J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, page 25, 2006.
10. N. Lii, W. Qardaji, and D. Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In *ASIACCS*, pages 32–33, 2012.
11. G. Loukides, A. Gkoulalas-Divanis, and B. Malin. Anonymization of electronic medical records for validating genome-wide association studies. *Proceedings of the National Academy of Sciences*, 17:7898–7903, 2010.
12. G. Loukides, A. Gkoulalas-Divanis, and B. Malin. COAT: Constraint-based anonymization of transactions. *Knowledge and Information Systems*, 28(2):251–282, 2011.
13. G. Loukides and J. Shao. Clustering-based k-anonymisation algorithms. In *DEXA*, pages 761–771, 2007.
14. A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, page 24, 2006.
15. P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *PODS*, page 188, 1998.
16. M. Terrovitis, J. Liagouris, N. Mamoulis, and S. Skiadopoulos. Privacy preservation by disassociation. *PVLDB*, 5(10):944–955, 2012.
17. M. Terrovitis, N. Mamoulis, and P. Kalnis. Local and global recoding methods for anonymizing set-valued data. *VLDB J.*, 20(1):83–106, 2011.
18. J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A.W-C. Fu. Utility-based anonymization using local recoding. In *KDD*, pages 785–790, 2006.
19. Y. Xu, K. Wang, A.W-C. Fu, and P.S. Yu. Anonymizing transaction databases for publication. In *KDD*, pages 767–775, 2008.
20. M. Xue, P. Karras, C. Raïssi, J. Vaidya, and K. Tan. Anonymizing set-valued data by nonreciprocal recoding. In *KDD*, pages 1050–1058, 2012.

Cloud-Based Data and Knowledge Management for Multi-Centre Biomedical Studies

Amalia Tsafara Christos Tryfonopoulos Spiros Skiadopoulos* Lefteris Zervakis*

Department of Informatics & Telecommunications
University of Peloponnese, GR 22100 Tripoli, Greece

{amtsafara,trifon,spiros,zervakis}@uop.gr

ABSTRACT

Among the basic research tools for (bio)medical science are epidemiological studies that typically involve a number of hospitals, clinics, and research centres scattered around the world, and are often referred to as multi-centre studies. Clearly, the effectiveness and importance of a multi-centre study increases with the number of participating centres and enrolled patients, but at the same time this natural distribution in the production of research data requires sophisticated data/knowledge management infrastructures to support the participating units. This kind of infrastructure is not only expensive to build and maintain, but also cannot be reused as it is often tailored to a specific study. In this work, we present a cloud-based system, that allows users without any computer science background to design, deploy, and administer platforms aimed for managing, sharing, and analysing clinical data from multi-centre studies. The proposed system provides a zero-administration, zero-cost online data/knowledge management tool that *(i)* enhances re-usability by introducing study templates, *(ii)* supports (bio)medical needs through specialised data types able to capture specialised knowledge like repeated therapies or treatments, and *(iii)* emphasises data filtering/export through an expressive yet simple graphical query engine.

1. INTRODUCTION

Large-scale epidemiological studies typically involve a number of different stakeholders, including hospitals, clinics, and research centres, physically distributed around the world, and are often referred to as *multi-centre studies*. These studies are invaluable as they collect large amounts of data from different regions, correlate them, and draw useful conclusions on important research questions. However, the physical distribution of the participants and the asynchronous nature of data acquisition pose a number of issues including the collection, organisation, and processing of data.

*Supported by EU and Greek funds through the EICOS project under the National Strategic Reference Framework Research Funding Program: Thales.

K-CAP'15, October 7–10, 2015, Palisades, NY, USA.
 ISBN 978-1-4503-3849-3/15/10
DOI: http://dx.doi.org/10.1145/2815833.2816949 ...$15.00.

To tackle data/knowledge fragmentation and address the issues arising from the coordination of geographically distributed participants, a number of platforms, that focus on the storage and management of (bio)medical data and knowledge, have been proposed [2, 4, 5, 6]. However, all these platforms are either designed for a *specific task or study* [2, 4, 6] (and are thus unusable in any other study), or require significant *computing infrastructure* and an *expert* in Information Technology (IT) for setup and tuning [5]. Typically, reconfiguring an existing platform for another study or setting one up from scratch results in *(i)* time-consuming meetings between scientists of different principles trying to understand each other's needs and *(ii)* resource-consuming IT infrastructure, that requires outsourcing to IT specialists and regular maintenance/upgrades to keep up with technological requirements. Due to these issues, a great number of multi-centre studies that lack the resources are still performed by resorting to *manual procedures*, such as collecting data on paper, exchanging data by post (either as hard copies, or as electronic copies stored in removable media), or emailing enormous (often outdated) spreadsheet files with (often sensitive) patient data among participants. Therefore, concerns like data freshness/integrity, participant coordination and degree of involvement, and timeliness of results are lost between versioning in exchanged spreadsheets, hard copies of patient data, and unanswered email requests.

In this work, we present a *cloud-based* service for *managing, sharing, and organising* clinical and patient data from *multi-centre studies*. The proposed system covers all the functional requirements posed by multi-centre studies, and enables researchers to easily organise and share data and knowledge generated by the research activity. We propose an innovative, integrated framework for creating platforms for multi-centre studies that enables users with *no prior IT knowledge* to *(i) design and launch*, in an easy and transparent way, platforms tailored to the specific needs of their studies, *(ii)* perform basic and advanced *user management* tasks (manage users, assign user privileges and permissions, perform access control on data), *(iii) record, organise and manage* clinical/patient data by resorting to a number of built-in and customisable data entry forms, and *(iv) search and filter* information by using a powerful yet simple point-and-click mechanism that poses restrictions on the stored data and extracts the requested information in a number of formats/outputs including raw data, pie/column charts, and ready-to-process spreadsheets. Due to the cloud infrastructure, computational resources are allocated on demand, providing *elasticity* and *fault-tolerance* in a way that is transparent to the end-user.

The contributions of this work are twofold:

Figure 1: Platform creation and editing.

- We propose a *cloud-based, zero-cost, zero-administration* tool that offers both fundamental and advanced user and data/knowledge management functionality for multi-centre studies. To the best of our knowledge, this is the first cloud-based system that focuses on multi-centre studies and allows users to deploy their own platforms within minutes, alleviating the need to rely on expensive custom-made solutions that require IT infrastructure and skills to maintain.
- We present the architectural considerations and solutions behind the proposed tool, and describe a number of *novel services* that allow users without any prior IT knowledge to create, administer, launch, and use personalised platforms.

Our system is currently under beta testing for multi-centre studies led by the Hellenic Society for Chemotherapy and the University Hospital Attikon, and has already been used by more than 12 public hospitals in Greece.

The rest of the paper is organised as follows. Section 2 describes the implemented services and functionality, outlines the system architecture and describes a demonstration plan to be presented at the conference, while Section 3 discusses related work.

2. SYSTEM OVERVIEW

Our system supports three different types of users that correspond to three data access privileges. More specifically, we have *(i) IT administrators* (ITAs) that are responsible for the update and maintenance of the system and have access (edit, delete, or filter) to all data in the database, *(ii) study administrators* (SAs) that are typically in charge of an ongoing study and may access all data related to the specific study, and *(iii) participants* in an ongoing study that may access only the data added by them.

2.1 Supported Functionality

Questions and platforms. *Questions* are the backbone of multi-centre studies. To support the set of questions of a particular multi-centre study, we introduce the concept of *platforms*. Technically, a platform is a custom-made set of questions, together with all necessary user administration and data/knowledge components of a study. Platforms are typically created and administered by the SA. More specifically, the SA is able to *(a)* design and launch a new platform by inputting a platform name and defining a number of study questions and *(b)* rearrange, or delete questions through drag-and-drop actions. For each question, the SA specifies three key elements: the *data type*, the *question text*, and the *question values*. This process is performed with the design tool of Figure 1. Our system supports all standard

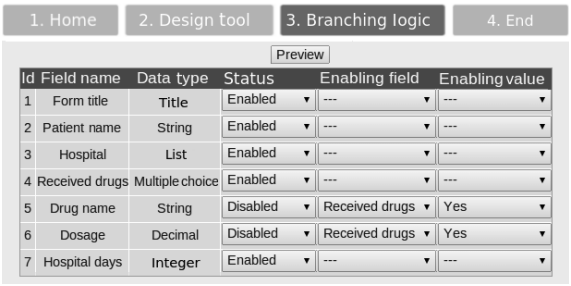


Figure 2: Platform branching logic.

data types such as *strings*, *integers*, *dates*, *decimal*, etc. Additionally, it supports *titles*, *multiple choices*, *lists* and *complex data types*.

Titles are used to introduce new questionnaire sections.

Multiple choices restrict possible answers to a fixed set.

Lists allow data input from dynamic, custom-made drop-down menus that the SA dynamically creates, stores, and edits during the platform design or maintenance. These lists may be shared across different studies of the same user and are used to ensure data consistency, enhance data integrity, and enforce validation of input. To define a new list, the SA specifies its unique name and defines its contents. Subsequently, when specifying a question, the SA sets the data type to list and selects one of the stored drop-down lists. For example in Figure 1, the third question specified is a user-defined list of hospitals participating in a study.

Complex datatypes support medical operations that involve groups of recurring questions – as in treatment plans or therapeutic protocols which usually consist of a set of medicines with periodically recorded data (e.g., name, start/end dates, outcome). The advantages of creating complex data types include better data modelling and knowledge capture, and thus richer query possibilities, and flexibility in the design of studies with complex/repeating medical processes.

Branching logic. SAs may also specify the *branching logic* of questions. This functionality allows SAs to specify the values in questions that are required to enable or disable following ones. For example in Figure 2, the SA specified that if the answer to the question Received drugs (Id = 4) is yes, the questions Drug name and Dosage (Id = 5 and Id = 6, respectively) should be enabled.

Templates. Typically, specific parts of a study may be used (as they are or with minor modifications) in other studies. For example, demographic data are a typical resuable part of many biomedical multi-centre studies since they hardly change among different studies. To support this reusability, the system offers the SA the possibility to create *platform templates* that may be used across different studies. Figure 3 shows a platform template meant for demographic data.

Editing an existing template involves two different scenarios: *(i) simple editing* that includes adding, deleting and modifying questions or their branching logic, and *(ii) structural editing* that relates to the modification of question order or data types. Such editing could affect the consistency of stored data/knowledge or cause compatibility problems between the stored data/knowledge and the new template and is thus limited to actions that do not cause such issues. For instance in Figure 1, the SA is not allowed to change Patient name to integer.

Figure 3: An example template for demographic data.

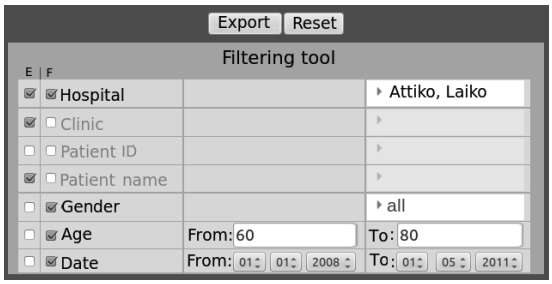


Figure 4: The filtering tool.

Filtering and chart tools. SAs are able to extract information from a study, while participants may only extract information added by them. Such information may be retrieved and presented as *(i)* a set of tuples (using the *filtering tool*), and *(ii)* a graph (using the *chart tool*).

The filtering tool (Figure 4) uses a powerful yet easy-to-use query issuing mechanism that allows users to *(i)* filter and retrieve and *(ii)* export to a third-party application stored records by applying constraints with simple point-and-click interactions. Data filter and export involves a *two-step process*. In the first step the user defines the query output by checking the questions that will be used for *data projection* (i.e., data to be exported). In the second step, he applies one or more *filtering conditions* on the data. The filtering conditions are introduced by presenting *all distinct values* stored for a specific question and allowing the user to define the ones that satisfy the filtering criteria. In this way, he may define *conjunctions* and *disjunctions* both on the questions and on the stored data. The query result may then be exported in a spreadsheet, or presented as a list of tuples. In Figure 4, the SA has selected to export the results for Hospitals, Clinics, and Patient names (notice the checked boxes in the first column) for all records that were input between 01/01/2008 and 01/05/2011 in Attiko or Laiko hospital and involve patients between 40 and 60 years of age (notice the specified constraints).

The filtering tool is able to capture the complex data types described earlier, allowing the user to *(i)* set more than one filters for every complex data type and *(ii)* include constructs like concurrent episodes of a diagnosis or treatment. Examples of supported queries include:

- Show patients older than 40 years of age with temperature greater than 38, who were diagnosed with microorganisms (Pseudomonas and Candida). Notice that the names of the microorganisms are values already stored in the database.
- Show patients between 40 and 60 years of age, who live in (England or France or Greece), have been diagnosed with microorganisms ((Pseudomonas and Candida) or (Pseudomonas and Salmonella) or (Citrobacter)), have undertaken therapy with (Ampicillin or Cefotaxime), and had progress as the eventual outcome of their condition. Notice the application of Boolean operators both on the questions, and on the data stored for each question.
- Show patients who suffer from Massive hemoptysis, had an initial episode of Bacteremia and accepted a dose between 2 and 4mg of antibiotic Amikacin, and subsequently had a second episode of Bacteremia. Notice that the query is applied to a complex data type (recurring episodes/treatments as described above).

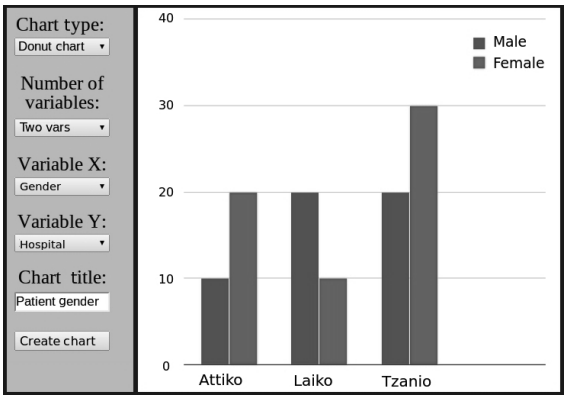


Figure 5: Using the chart tool with two variables.

The chart tool (Figure 5) may be used to extract and present information from stored data directly as a *pie, column, bar, or donut chart*. It supports the creation of single and multiple variable graphs depending on the constraints defined by the user. In this way the user may dynamically create graphs for queries with a single variable like "how many patients in the database are male/female", but also for queries with multiple variables like "how many patients are male/female in each hospital that participates in the study" (shown in Figure 5).

2.2 System Architecture

The main idea of the system is to allow users to design and build platforms, through a series of simple and adaptive processes. This can be done transparently through simple-to-follow wizards from users without any IT training, while the cloud-based architecture automatically adapts to the resources and infrastructure by relying on cloud elasticity.

The cloud functionality is provided by the open-source platform *ownCloud* (`http://owncloud.org/`), setup over a medium-sized computing infrastructure available at the university campus. Figure 6 presents a high-level view of the system architecture and the different types of modules implemented. The Cloud API is responsible for performing all necessary communication with the ownCloud platform and provides elasticity services, while the storage manager performs all necessary storage/retrieval operations to the data/knowledge base backend. Our backend implementation uses the LAMP framework as the backend infrastructure, while the rest of the modules have been developed using Javascript, PHP, and JQuery.

The Study Manager module is responsible for the creation, editing, and management of studies, and consists of a number of modules utilised to *(i)* manage the participants and the stored data associated with a study and *(ii)* filter/extract data requested by a SA. The Platform Manager

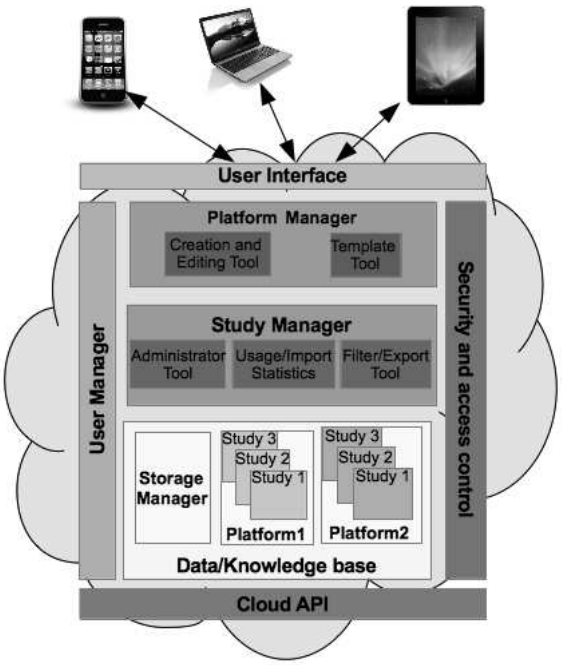


Figure 6: A high-level view of the architecture.

is used to create, edit, and manage platforms and templates utilised by different studies. The User Manager module is utilised by the ITA to create and manage the SAs, and also by the SAs to create and manage the participants (and their roles) in a specific study. The Security and Access Control Module enforces the security policies for the system and controls access privileges over the stored data. Security features include certificate and password-based authentication, single sign-on policy, and role-based user management. Finally, the User Interface module is responsible for identifying the hardware used to connect to the sytem (PC, tablet, smart phone) and adjust the viewing components accordingly.

Apart from the poster presentation we also plan to demonstrate the functionality of the system at the conference. The interested reader may find more information about the project at `www.uop.gr/~trifon/CloudStudy/`.

3. RELATED WORK

Over the years, many solutions aiming at the management and sharing of (bio)medical information have been proposed; in what follows we focus on approaches related to *electronic patient record (EPR)* systems specifically designed for multi-centre studies. Most of the proposed systems focus on a *specific study*, and put forward architectures and services tailored to the problem at hand. [10] presents an information system that may be used to manage multi-centre studies for cancer. Similarly, MSBase [3] introduces a web platform for collecting prospective data on patients with multiple sclerosis, while [2] presents a system for HIV/AIDS prevention and treatment. Finally, a number of EPR systems [4, 8, 9, 6], platforms (e.g., [5], HICDEP – `http://www.hicdep.org/`), and projects (e.g., CASCADE – `http://www.ctu.mrc.ac.uk/cascade/`, COBRED – `http://www.cobred.eu/`) have also been launched for supporting *(i)* different types of focused multi-centre studies [6] and *(ii)* clinical/biomedical data integration [7, 12, 11, 1].

The large number of existing specialised systems and the needs dictated by each different multi-centre study led researchers to the design of systems that are able to support *classes* of functionalities needed in multi-centre studies. The most prominent paradigms in this line of work are the REDCap project [5] and the Qure system [6]. REDCap provides a principled way of designing, constructing, and managing databases that are able to support multi-centre studies. However, to deploy a system for a specific study, the study coordinator needs to get in touch with the REDCap project, and inform the IT specialist on the specific needs and requirements of the study at hand. Subsequently, after an iterative and possibly long process of refinement and corrections in the database design, the REDCap IT expert will deploy the database and the users will be able to enter the data. Obviously, any subsequent changes in the specifications will result in the redesign of the database and the porting of the inserted data in the new database. On the other hand, the Qure system, while supporting online creation of study questionnaires, offers *(i)* limited data types (e.g., does not offer custom drop-down lists or complex data types), *(ii)* no data export functionality (e.g., pie/column charts, spreadsheets, SPSS compatible output), *(iii)* a query engine with limited expressiveness, and *(iv)* runs on dedicated hardware with no adaptation policy (e.g., elasticity of resources) and low quality-of-service guarantees.

4. REFERENCES

[1] Serge Abiteboul, Bogdan Alexe, Omar Benjelloun, Bogdan Cautis, Irini Fundulaki, Tova Milo, and Arnaud Sahuguet. An electronic patient record on steroids: Distributed, peer-to-peer, secure and privacy-conscious. In *VLDB*, 2004.

[2] A. Nucita et al. A global approach to the management of emr (electronic medical records) of patients with hiv/aids in sub-saharan africa: the experience of dream software. *BMC Medical Informatics and Decision Making*, 2009.

[3] H. Butzkueven et al. Msbase: an international, online registry and platform for collaborative outcomes research in multiple sclerosis. *Multiple Sclerosis*, 2006.

[4] H.S. Fraser et al. An information system and medical record to support hiv treatment in rural haiti. *British Medical Journal*, 2004.

[5] P.A. Harris, R. Taylor, R. Thielke, J. Payne, N. Gonzalez, and J.G. Conde. Research electronic data capture (redcap) âĂŞ a metadata-driven methodology and workflow process for providing translational research informatics. *Biomedical Informatics*, 2009.

[6] M. Jager, L. Kamm, D. Krushevskaja, H. Talvik, J. Veldemann, A. Vilgota, and J. Vilo. Flexible database platform for biomedical research with multiple user interfaces and a universal query engine. In *DB&IS*, 2008.

[7] Toralf Kirsten, Jörg Lange, and Erhard Rahm. An integrated platform for analyzing molecular-biological data within clinical studies. In *EDBT Workshops*, 2006.

[8] Arvind Kumar, Amey Purandare, Jay Chen, Arthur Meacham, and Lakshminarayanan Subramanian. Elmer: lightweight mobile health records. In *SIGMOD*, 2009.

[9] Wen-Syan Li, Jianfeng Yan, Ying Yan, and Jin Zhang. Xbase: cloud-enabled information appliance for healthcare. In *EDBT*, 2010.

[10] M. Martinez, J.M. Vazquez, M.G. Lopez, F.M. Arnal, B. Gonzalez-Conde, J. Pereira, and A. Pazos. Semantic integration of data in an information system for multicenter epidemiological studies on cancer. In *MIE2008*, 2008.

[11] Rakesh Nagarajan, Mushtaq Ahmed, and Aditya Phatak. Database challenges in the integration of biomedical data sets. In *VLDB*, 2004.

[12] James F. Terwilliger, Lois M. L. Delcambre, and Judith Logan. Context-sensitive clinical data integration. In *EDBT Workshops*, 2006.

Distance-based k^m-anonymization of 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 publication of trajectory data opens up new directions in studying human behavior, but it is challenging to perform in a privacy-preserving way. This is mainly because, the identities of individuals, whose movement is recorded in the data, can be disclosed, even after removing identifying information. Existing works to anonymize trajectory data offer privacy, but at a high data utility cost. This is because, they either do not produce truthful data, which is important in many applications, or are limited in their privacy specification component. This paper proposes an approach that overcomes these shortcomings by adapting k^m-anonymity to trajectory data and by using distance-based generalization. We also develop an effective and efficient anonymization algorithm, which is based on the apriori principle. Our experiments verify that this algorithm preserves data utility well, and it is fast and scalable.**

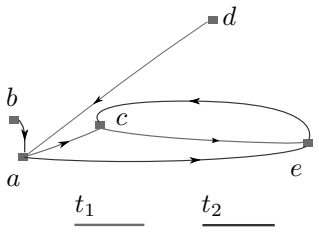
I. Introduction

The widespread use of GPS-enabled smartphones and location-based social networking applications, such as Foursquare (https://foursquare.com), opens up new opportunities in understanding human behavior 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 identifying information (ID) is not published [23]. The values that, in combination, may lead to identity disclosure are called *quasi-identifiers* (QI) [24], [22]. For example, let us assume that a location-based social network service, publishes the movement of users during a day in form of checkins in various locations. An example of this data is shown in Fig. 1a. If Mary's colleague, John, knows that sometime that day, Mary 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 visited d and then a, he can accurately link Mary with her 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 adversaries, but also the difference from well-studied set-valued data anonymity models, like k^m-anonymity [26] and privacy-constrained anonymization [17], [11]. In these models, value ordering is not significant; thus records are represented as unordered sets of items. For instance, if an adversary knew that someone visited location c and then e, they could link this individual only to record t_1 (Fig. 1b). On the other hand, if $\mathcal{T}$ was a set-valued dataset, records t_1, t_2 and t_4 would have items c and e, hiding this individual's identity among 3 records. Consequently, for any set of n items in a trajectory, there are $n!$ possible quasi-identifiers.

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)

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

This difference makes anonymizing trajectory datasets more challenging, as it drastically increases the number of potential quasi-identifiers. Existing methods operate either by (i) anonymizing each trajectory as a whole, thereby not assuming any specific background knowledge of attackers [1], [2], [18], [21], or (ii) by anonymizing parts of the user trajectories by considering attackers who can effectively link specific locations to individuals in order to re-identify them [25], [28]. The first category of approaches are based on *clustering and perturbation* [1], [2], [21], while the second category employs *generalization and suppression* of quasi-identifiers [19], [28], [20], [25].

The main drawback of clustering-based approaches is that they may lose information about the direction of movement of co-clustered trajectories, as well as cause excessive information loss, due to space translation. Moreover, applying perturbation to datasets, creates data that are not *truthful* and cannot be used in several applications [9]. Similarly, existing generalization-and-suppression based methods [19], [28], [20], [25] have the following limitations. First, they assume that quasi-identifiers are known to the data publisher prior to anonymization [28], [25] (e.g., defined by users) or that any combination of locations can act as a quasi-identifier [19]. Second, they require a location taxonomy to be specified by data publishers [20], based on locations' semantics. However, this taxonomy may not well reflect the distance between locations and therefore the anonymized data may incur a high amount of information loss. Last, some approaches assume that each location can be classified as either sensitive or non-sensitive [20]. In practice, however, this assumption may not hold, as location sensitivity depends on context (e.g. visiting a hospital may be sensitive for a patient, but not for a doctor).

Our proposed approach addresses the aforementioned short-

comings by adapting k^m-anonymity [26] to trajectory data and by applying generalization in a way that minimizes the distance between the original and the anonymized user trajectories. 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 publication. At the same time, our approach avoids the use for a location taxonomy and generalizes trajectories in a way that preserves data utility.

The rest of the paper is organized as follows. Section II discuss related work. Section III formulates the problem and Section IV presents our anonymization algorithm. Section V presents the experimental evaluation of our algorithm in terms of utility and efficiency. Finally, Section VI concludes the paper.

II. Related work

k-anonymity is a privacy model that prevents identity disclosure, by requiring at least k records of a dataset to have the same values over QI [24], [23], [16], [12], [13], [27]. Thus, a k-anonymous dataset upperbounds the probability of associating an individual to their record by $\frac{1}{k}$. To enforce k-anonymity, most works [22], [14], [15], [10] 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.

The k-anonymity principle has been recently considered in the context of publishing user trajectories, leading to several trajectory anonymization methods [4]. These methods operate either by (i) anonymizing each trajectory as a whole [1], [2], [18] or (ii) by anonymizing parts of the user trajectories by considering attackers who can link specific locations to individuals in order to perform identity disclosure [25], [28]. The approaches of the first category operate by grouping original trajectories into clusters of k members 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 anonymity algorithm proposed in [1] identifies trajectories that lie close to each other in time and employs space translation.

The second category of approaches considers attackers with background knowledge on ordered sequences of places of interest (POIs) visited by specific individuals. Terrovitis et al. [25] 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 released 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 adversary.

Yarovoy et al. [28] proposed a k-anonymity based approach for publishing user trajectories by considering time as a quasi-identifier and supporting privacy personalization. Unlike previous work that assumed that all users share a common quasi-identifier, [28] assumes that each user has a different set of POIs and times for which he or she requires 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. [19], 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 adversaries 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.

Contrary to related work on trajectory anonymization approaches operating through data generalization or suppression of quasi-identifiers, our method makes the realistic assumption that an adversary may have knowledge of up to m locations that a user has visited. To protect the trajectories, we employ a distance-based generalization approach that does not depend on a pre-specified location taxonomy. Moreover, in this work we refrain from classifying locations as sensitive or nonsensitive (QI), and prevent identity disclosure, based on any combination of up to m locations.

Recently, differential privacy [8] methods were proposed to anonymize sequential datasets [6], [7]. These methods focus on specific data analytic tasks, such as query answering or frequent pattern mining [3] and work by adding noise to the data. Thus, they harm data truthfulness, which is essential to preserve in many data analysis tasks [17].

III. Problem formulation

Let $\mathcal{L}$ be a set of locations (points of interest, touristic sites, shops, etc.).

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., $|t| = n$.

A trajectory represents the locations and the order these locations are visited by a moving object (individual, bus, taxi, etc.). In our setting a location may also model points in space (1D, 2D, 3D, etc.) and even incorporate a temporal dimension.

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 $f(1) < \cdots < f(\nu)$.

Thus, a subtrajectory is formed by removing some locations from the original trajectory, while maintaining the order of the remaining locations. For instance, the trajectory (a, e) is contained in $t_1 = (d, a, c, e)$ (Fig. 1).

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 Fig. 1a, we have $\sup((a,e),\mathcal{T}) = 3$. 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 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 $\frac{1}{k}$).

Example 1: Consider the dataset of trajectories $\mathcal{T}$ depicted in Fig. 1a. $\mathcal{T}$ is 2^1-anonymous and 1^3-anonymous. However, it is not 2^2-anonymous, as the subtrajectory (d,a) is contained only in the trajectory t_1 of $\mathcal{T}$.

To explain the way we generalize trajectories, we define the notion of generalized location as follows.

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

A *generalized location* is interpreted as any of it's locations. 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 location among $l_1,\ldots,l_v$.

To enforce k^m-anonymity, we either generalize a location l to a generalized location that contains l or leave l intact.

We are interested in generalization transformations that distort as little as possible the initial dataset $\mathcal{T}$. A common way to measure the distortion of a transformation is to measure the *distance* between the original and the transformed dataset [25], [21], [28]. In our case, the distance between the initial 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. In more detail, we have.

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:

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

where *EuclDist* 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 $\mathit{EuclDist}(a,a_1) = 1$ and $\mathit{EuclDist}(a,a_2) = 2$. 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 $\mathcal{D}_{traj}((a,b),(\{a,a_1,a_2\},b)) = (1+0)/2$.

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

The problem we consider can be expressed as follows.

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

In the rest of the paper, we present and evaluate a method to tackle Problem 1.

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 distance from l_1
8 Replace all occurrences of l_1 and l_2 in $\mathcal{T}'$ and s with $\{l_1,l_2\}$
9 **return** $\mathcal{T}'$

IV. ANONYMIZATION ALGORITHM

Given an input set of trajectories $\mathcal{T}$, we will present a method that transforms $\mathcal{T}$ into a k^m-anonymous set of trajectories $\mathcal{T}'$ corresponding to $\mathcal{T}$ by generalizing the locations of the trajectories that do not satisfy the k^m-anonymity metric.

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

In more detail, SEQANON proceeds as follows. First, SEQANON initializes $\mathcal{T}'$ (Step 1). Then, Steps 2 – 2 follow the apriori principle. Step 3 computes set $\mathcal{S}$ containing the subtrajectories s of $\mathcal{T}$ having size i (i.e., having i locations) and lower support than the anomymization parameter k (i.e., $\sup(s,\mathcal{T}') < k$). SEQANON considers the lower support subtrajectories of $\mathcal{S}$ first. This tactic improves the efficiency of

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)

Fig. 2: (a) Set $\mathcal{S}$ for subtrajectories of size $i = 2$ and the respective supports, (b) Transformed dataset $\mathcal{T}'$ after the processing of subtrajectory (d, a), and (c) The final 2^2-anonymous result $\mathcal{T}'$

the method and the quality of the results. Remedying the lower support subtrajectories commonly benefits higher support subtrajectories while at the same time, their generalization does not significantly affect the dataset. Continuing, for every such trajectory $s \in \mathcal{S}$, the algorithm finds the location l_1 of s with the minimum support (Step 6). Similarly to subtrajectories, we consider lower support locations first. Then, the algorithm searches the locations of $\mathcal{T}$ to detect the closest location l_2 to l_1 (Step 7). 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 anonymization parameter k.

The following is an example of SEQANON in operation.

Example 2: We will demonstrate the operation of SEQANON with input the dataset $\mathcal{T}$ of Fig. 1a and $k = m = 2$. The intermediate steps are illustrated in Fig. 2. The first iteration of the for loop (Steps 2 – 2) considers the subtrajectories of size $i = 1$. It is not hard to verify that all size 1 locations have support greater than $k = 2$, thus, the algorithm proceeds to $i = 2$. For this case, Step 3 computes the set of low support subtrajectories $\mathcal{S}$ (illustrated in Fig. 2a). SEQANON considers subtrajectory $s = (d, a)$, which is the first subtrajectory in $\mathcal{S}$. Then, Step 6 sets $l_1 = a$ (since a is the lowest support location of (d, a)) and Step 7 sets $l_2 = b$ (since location b is closer to a – see also the map of Fig. 1b). Finally, Step 8 replaces a and b with the generalized location $\{a, b\}$ in s and all the trajectories of $\mathcal{T}'$. After these replacement, we have $s = (d, \{a, b\})$ while $\mathcal{T}'$ is depicted in Fig. 2b. Since, for the changes values of s and $\mathcal{T}'$ we still have $\sup(s, \mathcal{T}') < k$, the while loop (Steps 5 – 8) is executed again. This time $l_1 = \{a, b\}$, $l_2 = c$ and after the replacements $\mathcal{T}'$ is depicted in Fig. 2c. The remaining steps of the algorithm SEQANON do not change $\mathcal{T}'$, thus, Fig. 2c illustrates the final output.

Complexity analysis. Algorithm SEQANON executes the **for** loop (Steps 2 – 8). For each iteration of this loop, set $\mathcal{S}$ is constructed and explored. The size of $\mathcal{S}$ is, in the worst case, $\mathcal{O}(|\mathcal{L}|^i)$, where $|\mathcal{L}|$ is the size of the location set used in $\mathcal{T}$ and i is the loop counter. The above bound is a very crude approximation. $\mathcal{O}(|\mathcal{L}|^i)$ is the size of $\mathcal{S}$ when all size i subtrajectories have support lower than k. This is hardly the case; the actual sutrajectories s with $\sup(s, \mathcal{T}') < k$ are a small fraction of $\mathcal{O}(|\mathcal{L}|^i)$. This number depends heavily on the dataset $\mathcal{T}$ and the value of the anonymization parameter k. To have a more precise bound we multiply with the factor $p(i, k, \mathcal{T})$ which measures the probability of a subtrajectory of size i having support less than k in dataset $\mathcal{T}$. Thus, the size of $\mathcal{S}$ is bounded by $\mathcal{O}(p(i, k, \mathcal{T}) \cdot |\mathcal{L}|^i)$. For each element s of $\mathcal{S}$ the **while** loop (Steps 5 – 8) is executed. This loop takes $\mathcal{O}(|s|) = \mathcal{O}(i)$ time in the worst case (i.e., when we are going to generalize all locations of the trajectory). Overall, each iteration of the **for** loop takes $\mathcal{O}(i \cdot p(i, k, \mathcal{T}) \cdot |\mathcal{L}|^i)$ time. Thus, in total, the complexity of the SEQANON algorithm is $\mathcal{O}\big(\sum_{i=1}^{m}(i \cdot p(i, k, \mathcal{T}) \cdot |\mathcal{L}|^i)\big) = \mathcal{O}\big(m \cdot p(k, \mathcal{T}) \cdot |\mathcal{L}|^m\big)$ where $p(k, \mathcal{T})$ averages $p(1, k, \mathcal{T}), \ldots, p(m, k, \mathcal{T})$.

V. EXPERIMENTAL EVALUATION

In this section, we evaluate our algorithm in terms of data utility and efficiency.

Experimental setup. We implemented our algorithm in C++ and tested it on an Intel Core i7 at 2.2 GHz with 6 GB of RAM. We generated synthetic trajectories of moving objects on Oldenburg city map using Brinkhoff's generator [5]. This setting has been used by many works [1], [21], [28], [25]. We then normalized trajectories so that all coordinates take values in a $10^3 \times 10^3$ map and we simulated trajectories corresponding to these routes as follows. The map was divided into 100 regions using a uniform grid. An 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 average length of the generated trajectories is 4.72. The default number of locations of $\mathcal{L}$ and trajectories of $\mathcal{T}$ are 100 and 18,143 respectively. Unless otherwise stated, m is set to 2.

Data utility. To measure data utility, we evaluated the number of published original locations and the number of generalized locations. For the generalized locations, we also measure their average size and distance. Initially, we vary the anonymization parameter k in [2, 100]. Our results are summarized in Fig. 3.

In Fig. 3a, we present the number of the original locations published (i.e., locations that were not generalized) as a function of k. As expected, increasing k led to fewer original locations published. In Fig. 3b, we illustrate 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 Fig. 3c. Finally, we present the average distance

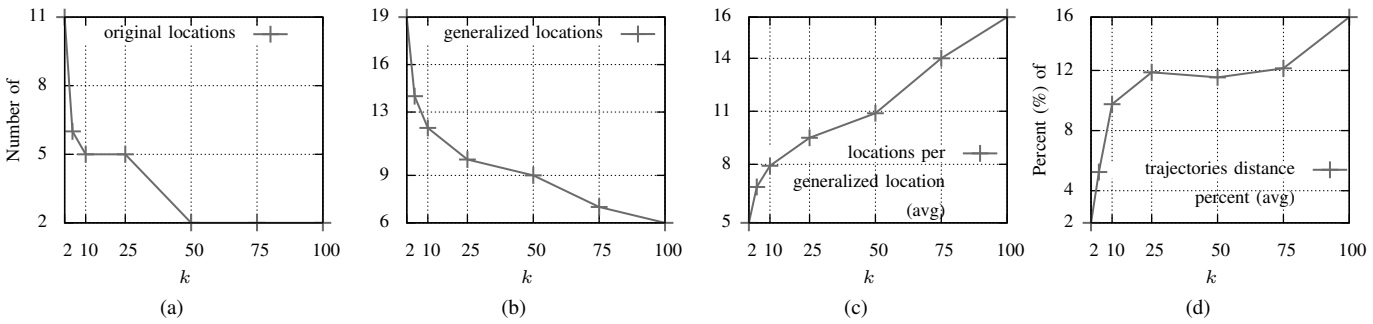


Fig. 3: number of (a) original (non generalized) locations published and (b) generalized locations published, (c) average number of generalized locations size, (d) average percent of distance in generalized locations

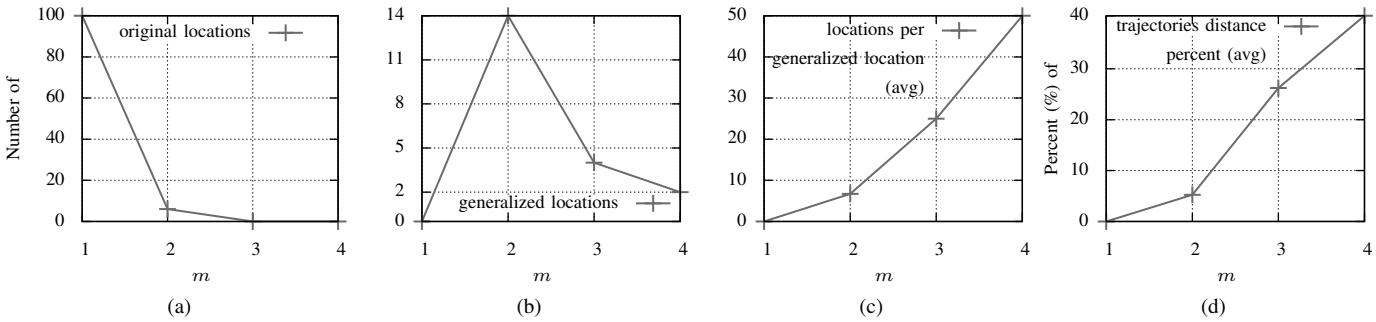


Fig. 4: number of (a) original (non generalized) locations published and (b) generalized locations published, (c) average number of generalized locations size, (d) average percent of distances in generalized locations

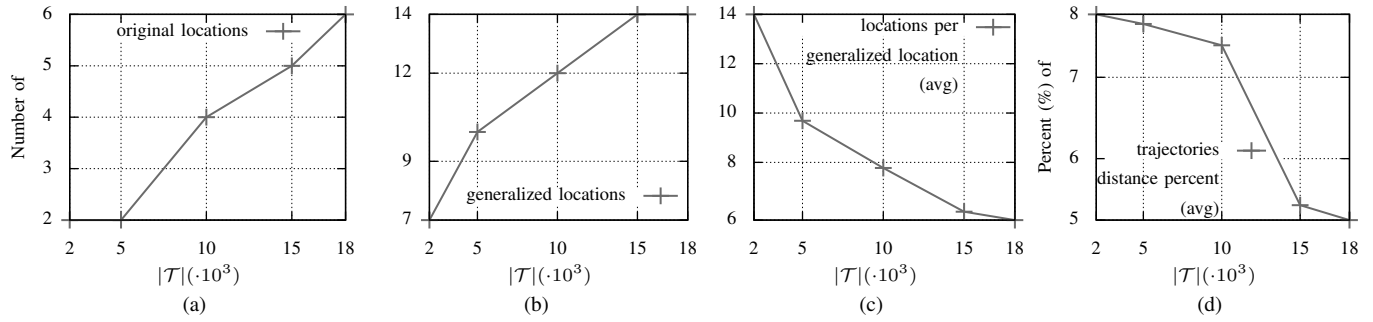


Fig. 5: number of (a) original (non generalized) locations published and (b) generalized locations published, (c) average number of generalized locations size, (d) average percent of distances in generalized locations

of all locations inside each generalized location from original location in $\mathcal{L}$. We normalize this distance as a percentage of the maximum possible distance (i.e., the distance between the furthermost points). This percentage quantifies the distance distortion in a generalized location. In Fig. 3d, 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 our algorithm focuses in minimizing the distance of locations in each generalized location, distortion is relatively low and increases slowly.

To show the impact of m on 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 Figs 4a-4d.

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 Fig. 5a, we illustrate the number of original locations published 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 Figs 5b-5d.

Efficiency. We studied the impact of the anonymization parameters k and m, and the dataset size on efficiency. To highlight the impact on efficiency of the apriori principle used by our algorithm, we created a version of Algorithm

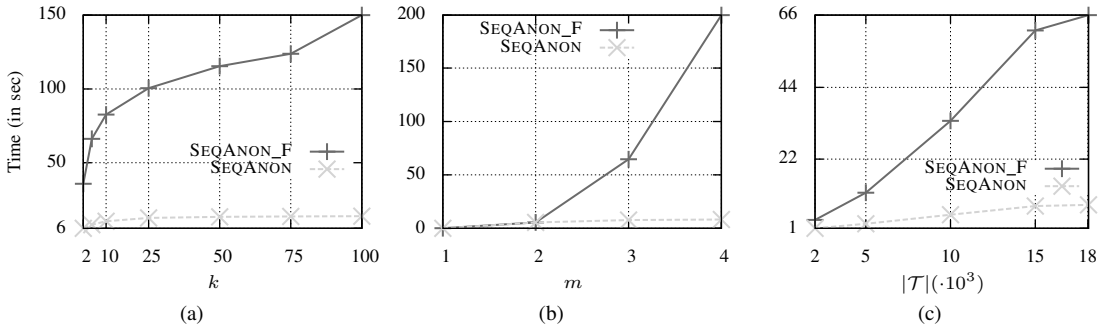


Fig. 6: Runtime of Algorithms SEQANON and SEQANON_F for (a) variable k, (b) variable m and (c) variable $|\mathcal{T}|$

SEQANON, denoted by 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, Algorithm SEQANON_F tries to eliminate directly subtrajectories of size m with low support. At first, we evaluated our algorithm using various k values, as in the experiments above. As we illustrate in Fig. 6a, the execution time increases with k. As expected, greater values for k lead to more subtrajectories with a lower support than k, resulting in a $\mathcal{S}$ of increased size (see also Step 3 of SEQANON). Similarly, m has the same affect on execution time. As m increases, our algorithm performs more iterations (Steps 2-2 of Algorithm SEQANON). In Fig. 6b, we show the impact of m on efficiency. Our algorithm significantly outperforms SEQANON_F, verifying that the apriori principle improves the efficiency rate for larger m values. Finally, in Fig. 6c, we illustrate the impact of dataset size in the execution time of SEQANON. As expected, larger datasets require longer processing time, since SEQANON needs to process more records.

VI. 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, as it employs distance-based generalization. We also developed an anonymization algorithm that performs well in terms of data utility preservation, and it is fast and scalable, due to the use of apriori principle.

ACKNOWLEDGEMENTS

G. Poulis is supported by the Research Funding Programm: Heraclitus II. S. Skiadopoulos is partially supported by EU/Greece the Research Funding Program: Thales. G. Loukides is partly 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. *IS*, 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. *SIGKDD Explorations*, 13(1):30–42, 2011.
[5] T. Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.
[6] 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.
[7] R. Chen, B. C. M. Fung, and B. C. Desai. Differentially private trajectory data publication. *CoRR*, abs/1112.2020, 2011.
[8] C. Dwork. Differential privacy. In *ICALP (2)*, pages 1–12, 2006.
[9] 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.
[10] B. C. M. Fung, K. Wang, and P. Yu. Top-down specialization for information and privacy preservation. In *ICDE*, pages 205–216, 2005.
[11] A. Gkoulalas-Divanis and G. Loukides. PCTA: privacy-constrained clustering-based transaction data anonymization. In *PAIS*, pages 1–10, 2011.
[12] A. Gkoulalas-Divanis and G. Loukides. Utility-guided clustering-based transaction data anonymization. *Transactions on Data Privacy*, 5(1):223–251, 2012.
[13] A. Gkoulalas-Divanis, Y. Saygin, and D. Pedreschi. Privacy in mobility data mining. *SIGKDD Explorations*, 13(1):4–5, 2011.
[14] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Incognito: efficient full-domain k-anonymity. In *SIGMOD*, pages 49–60, 2005.
[15] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, page 25, 2006.
[16] G. Loukides and A. Gkoulalas-Divanis. Utility-preserving transaction data anonymization with low information loss. *Expert Syst. Appl.*, 39(10):9764–9777, 2012.
[17] G. Loukides, A. Gkoulalas-Divanis, and B. Malin. COAT: Constraint-based anonymization of transactions. *Knowl. Inf. Systems*, 28(2):251–282, 2011.
[18] 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.
[19] A. Monreale, G. L. Andrienko, N. V. Andrienko, F. Giannotti, D. Pedreschi, S. Rinzivillo, and S. Wrobel. Movement data anonymity through generalization. *TDP*, 3(2):91–121, 2010.
[20] A. Monreale, R. Trasarti, D. Pedreschi, C. Renso, and V. Bogorny. C-safety: a framework for the anonymization of semantic trajectories. *TDP*, 4(2):73–101, 2011.
[21] M. E. Nergiz, M. Atzori, and Y. Saygin. Towards trajectory anonymization: a generalization-based approach. In *SPRINGL*, pages 52–61, 2008.
[22] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. Knowl. Data Eng.*, 13(6):1010–1027, 2001.
[23] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *PODS*, pages 188–, 1998.
[24] L. Sweeney. k-anonymity: A model for protecting privacy. *IJUFKS*, 10(5):557–570, 2002.
[25] M. Terrovitis and N. Mamoulis. Privacy preservation in the publication of trajectories. In *MDM*, pages 65–72, 2008.
[26] M. Terrovitis, N. Mamoulis, and P. Kalnis. Local and global recoding methods for anonymizing set-valued data. *VLDB J.*, 20(1):83–106, 2011.
[27] Y. Xu, K. Wang, A. W.-C. Fu, and P. S. Yu. Anonymizing transaction databases for publication. In *KDD*, pages 767–775, 2008.
[28] 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.

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

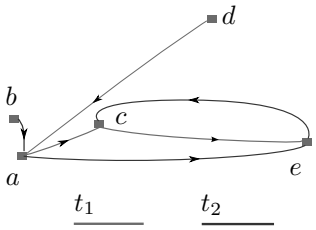
1 Introduction

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.

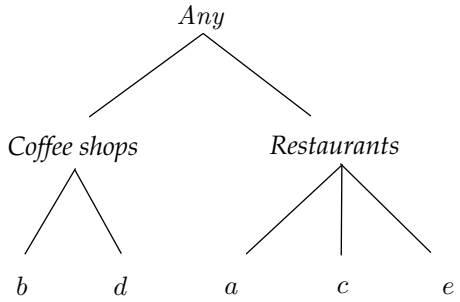


Figure 2: A location taxonomy

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

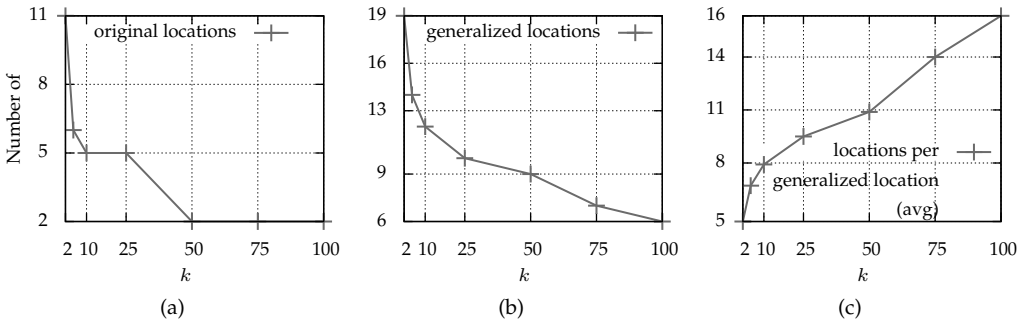


Figure 6: number of *(a)* original (non generalized) locations published, *(b)* generalized locations published and *(c)* average generalized location size

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.

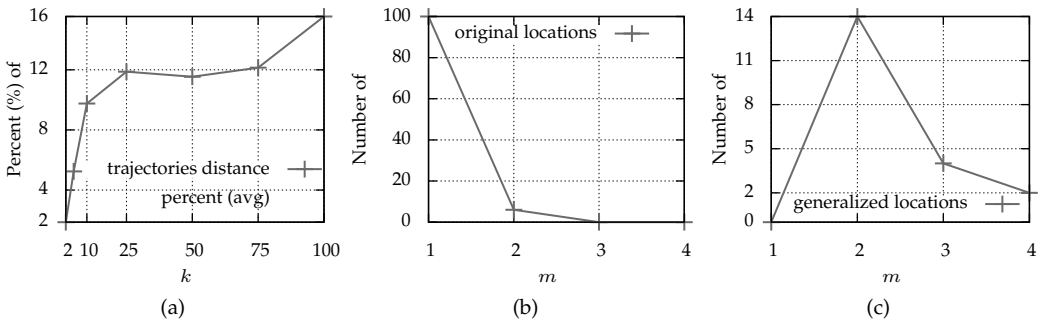


Figure 7: number of *(a)* average percent of distance in generalized locations, *(b)* original (non generalized) locations published and *(c)* generalized locations published

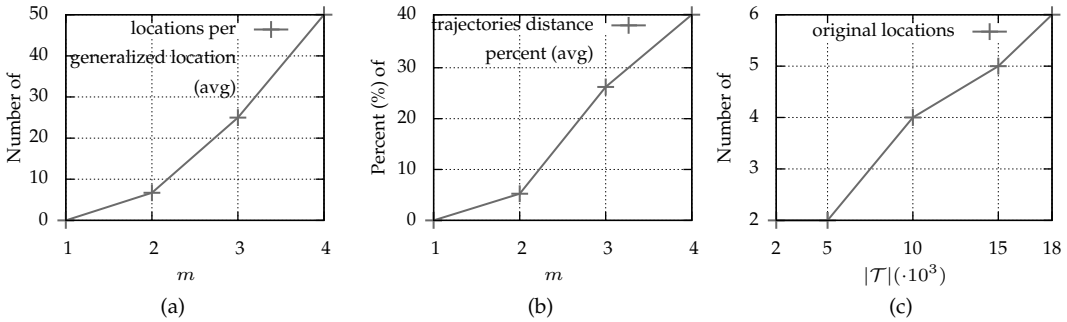


Figure 8: *(a)* average generalized location size, *(b)* average percent of distances in generalized locations and *(c)* number of original (non generalized) locations published

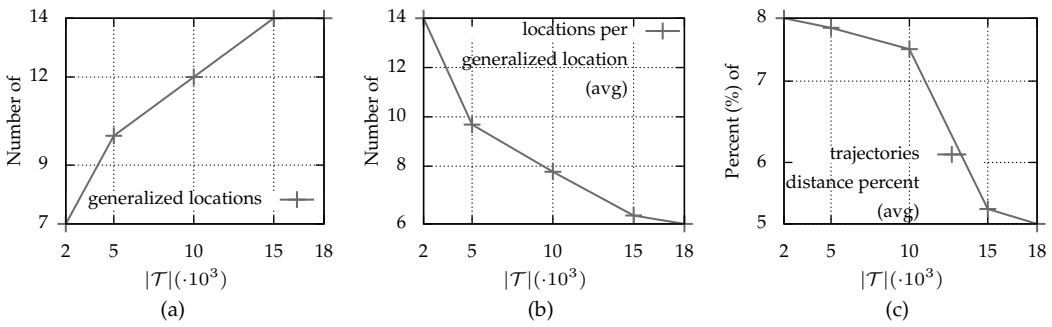


Figure 9: *(a)* number of generalized locations published, *(b)* average generalized location size and *(c)* average percent of distances in generalized locations

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.*

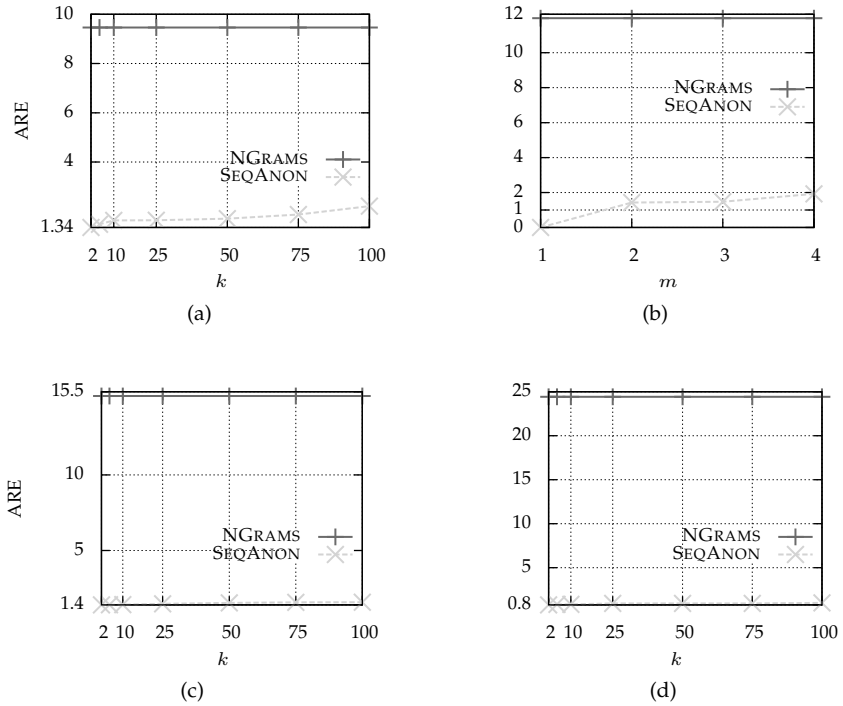


Figure 11: ARE for queries involving 100 randomly selected subtrajectories *(a)* with size in $[1, 2]$, in the Oldenburg dataset (varying k), *(b)* with size in $[1, 2]$, in the Oldenburg dataset (varying m), *(c)* with size 1, in the Gowalla dataset (varying k), and *(d)* with size 2 in the Gowalla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

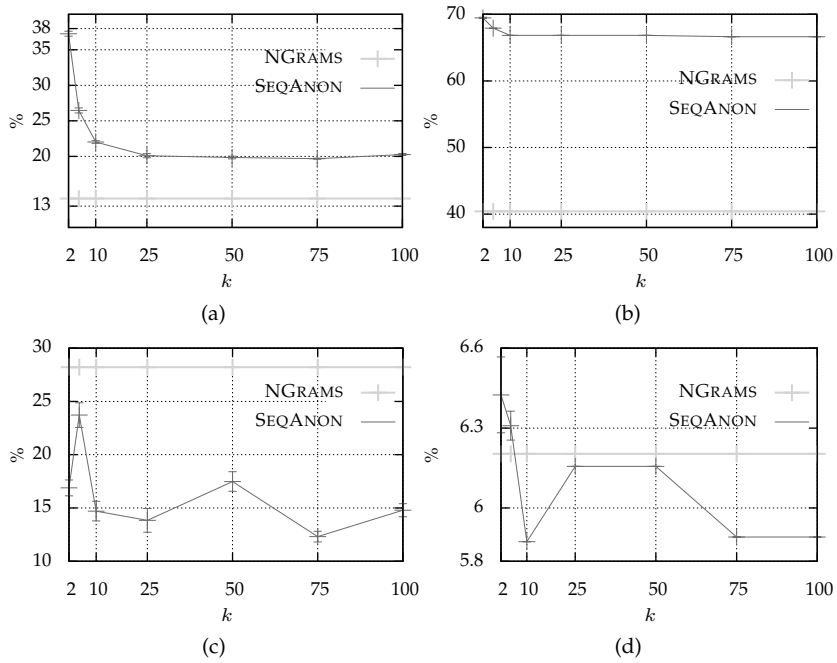


Figure 12: Percentage of frequent sequential patterns preserved in anonymous dataset (median) for varying k on *(a)* Oldenburg dataset, *(b)* Gowalla dataset, and percentage of frequent sequential patterns of the anonymous dataset that are not frequent on the original (median) for varying k on *(c)*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

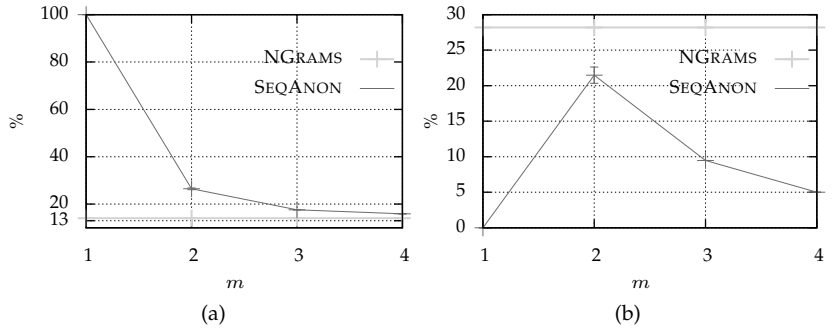


Figure 13: *(a)* Percentage of frequent sequential patterns preserved in anonymous dataset (median) and *(b)* percentage of frequent sequential patterns of the anonymous dataset that are not frequent on the original (median) on Oldenburg dataset for varying m.

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-

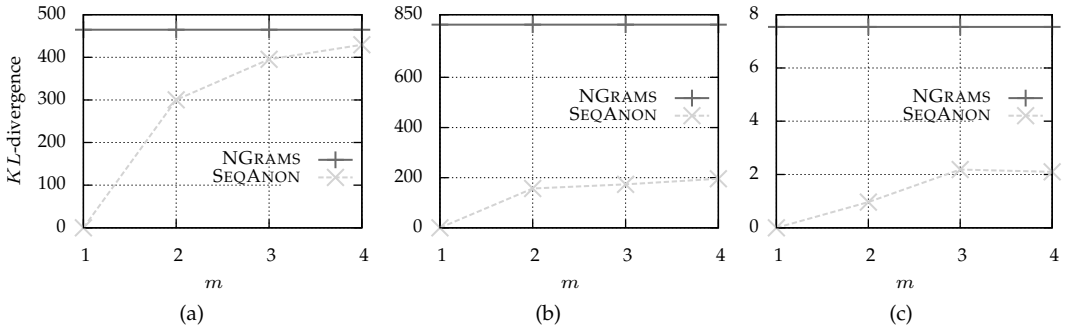


Figure 14: KL-divergence for the distribution of the support of *(a)* all locations in the Oldenburg dataset, *(b)* 100 randomly selected locations in the Gowalla dataset, and *(c)* 100 randomly selected subtrajectories of size 2 in the Gowalla dataset.

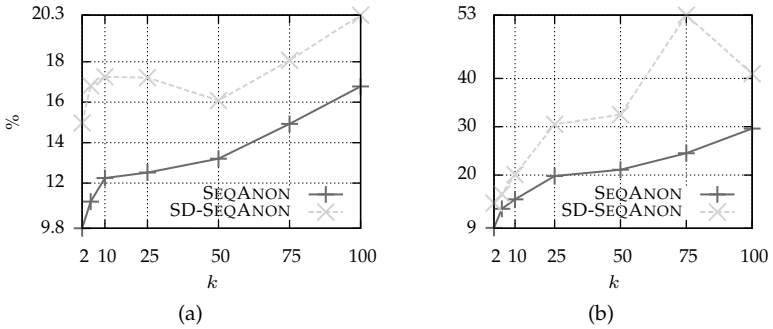


Figure 15: Average percent of distance in generalized locations for *(a)* Oldenburg and *(b)* Gowalla dataset.

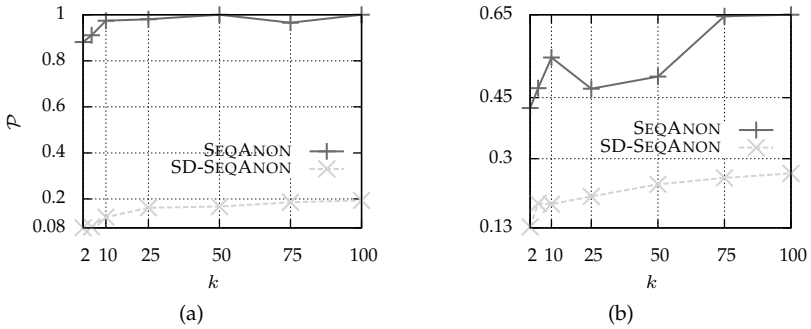


Figure 16: Semantic similarity penalty $\mathcal{P}(\mathcal{T}')$ for *(a)* Oldenburg and *(b)* Gowalla dataset.

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).

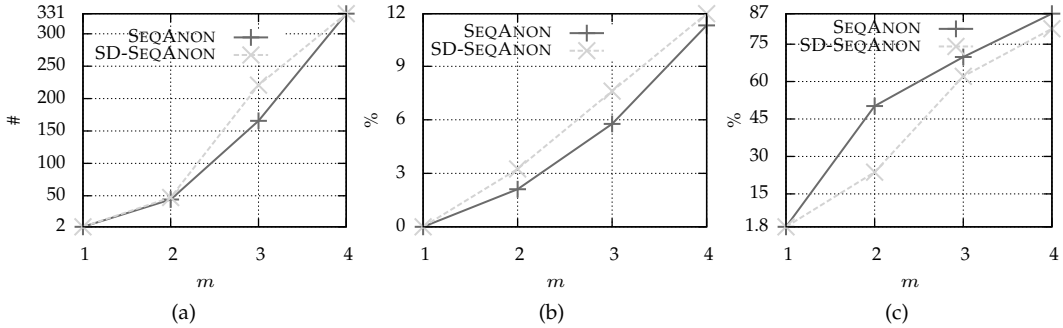


Figure 17: *(a)* average generalized location size, *(b)* average percent of distance in generalized locations and *(c)* average percent of similarity in generalized locations for $k = 50$ (Gowalla dataset).

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.

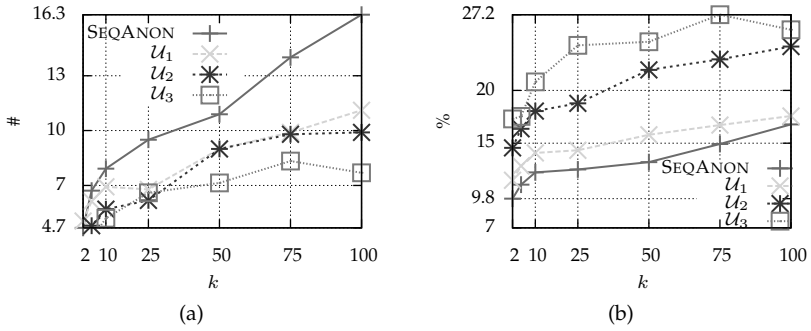


Figure 19: *(a)* average generalized location size and *(b)* average percent of distance in generalized locations, for the Oldenburg dataset.

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-

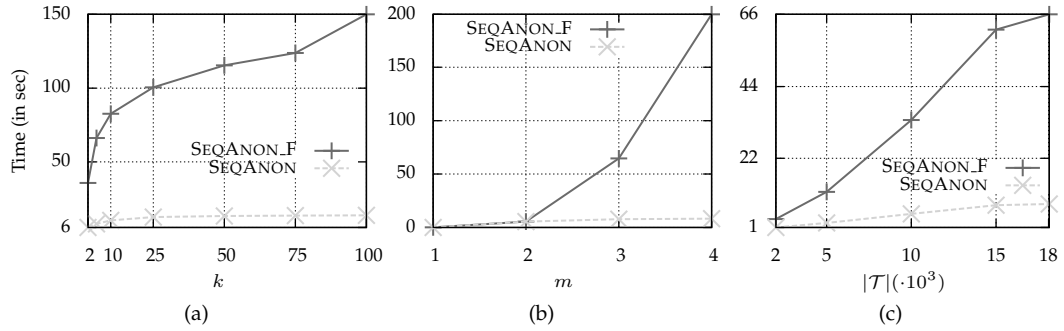


Figure 20: Runtime of SEQANON and SEQANON_F for the Oldenburg dataset and *(a)* varying k, *(b)* varying m, and *(c)* varying dataset size.

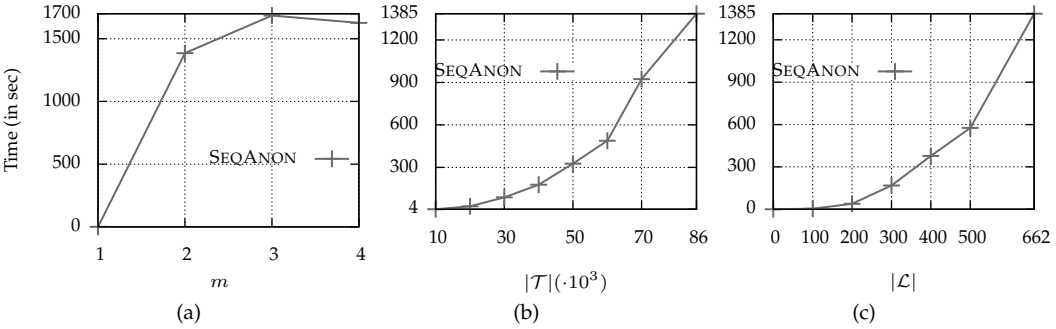


Figure 21: Runtime of SEQANON for the Gowalla dataset and *(a)* varying m, *(b)* varying dataset size, and *(c)* varying number of locations.

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.

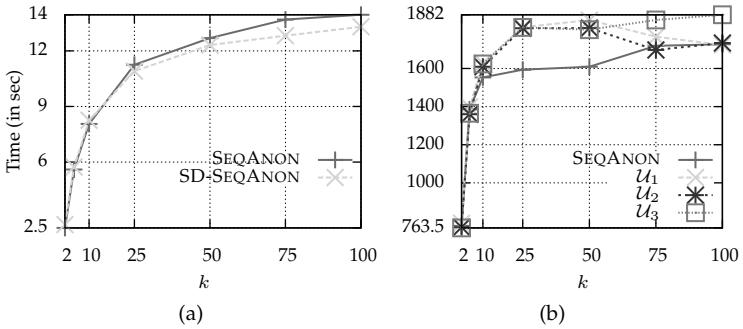


Figure 22: Runtime for varying k and for *(a)* SEQANON and SD-SEQANON (Oldenburg dataset), *(b)* SEQANON and U-SEQANON (Gowalla dataset).

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.