Pattern-Preserving *k*-Anonymization of Sequences and its Application to Mobility Data Mining

Ruggero G. Pensa[1], Anna Monreale[2], Fabio Pinelli[1], and Dino Pedreschi[2]

[1] ISTI - CNR, Area della Ricerca di Pisa,
Via Giuseppe Moruzzi, 1 - 56124 Pisa, Italy
[2] Computer Science Dep., University of Pisa,
Largo Pontecorvo, 3 - 56127 Pisa, Italy
{firstname.lastname}@isti.cnr.it

Abstract. Sequential pattern mining is a major research field in knowledge discovery and data mining. Thanks to the increasing availability of transaction data, it is now possible to provide new and improved services based on users' and customers' behavior. However, this puts the citizen's privacy at risk. Thus, it is important to develop new privacy-preserving data mining techniques that do not alter the analysis results significantly. In this paper we propose a new approach for anonymizing sequential data by hiding infrequent, and thus potentially sensible, subsequences. Our approach guarantees that the disclosed data are *k*-anonymous and preserve the quality of extracted patterns. An application to a real-world moving object database is presented, which shows the effectiveness of our approach also in complex contexts.

1 Introduction

In the last decade, many KDD techniques have been developed that provide new means for improving personalized services through the discovery of patterns and models which represent typical or unexpected customer's and user's behavior. The exponential growth of available personal data, as well as the refinement of data mining techniques, lead to new and intriguing possibilities. On the other hand, the collection and the disclosure of personal, often sensible, information increase the risk of citizen's privacy violation. For this reason, many recent research works have focused on privacy-preserving data mining [5, 24, 16, 18], proposing novel techniques that allow to extract knowledge while trying to protect the privacy of users and customers (or respondents) represented in the data[3]. This may involve techniques that return anonymized data mining results, or that provide anonymized datasets to the companies/research institution in charge of their analysis.

[3] In statistics, the problem has been extensively studied in the field of *statistical disclosure control*.

A major and rising field in data mining research concerns the analysis of sequence databases. User's actions as well as customer transactions are often stored together with their timestamps, making the temporal sequentiality of the events a powerful source of information. For instance, web logs provide the full activity of each website visitors during each browser session. Moreover, the spreading of mobile devices, such as mobile phone, GPS devices and RFIDs, has become a great source of spatio-temporal data. Companies and public institutions can now study the sequential behavior of their customers/citizens to improve their offers and services. A lot of advanced techniques have been investigated to extract patterns and models in databases of sequences [4, 27, 23], as well as in databases of moving objects (trajectories) [13]. For both legal and ethical reasons, the data owners (or custodians) should not compromise the privacy of their customers and users, and therefore should reveal as little as possible their personal sensible information. Hiding personal identifiers, such as personal IDs or quasi-identifiers (i.e., attributes that can be linked to external information to re-identify the individual to whom the information refers) may not be sufficient in the case of sequential data. If a small sequence of actions is easily referable to a few persons, an attacker may access to the whole action sequences involving these persons. For instance, if a malicious data user has access to the daylight city traffic data, and he knows that John Smith often goes from the commercial zone A to the general hospital B, and the sequence $A \Rightarrow B$ appears few times in the dataset, he can easily identify the entire sequence of locations crossed by John Smith during the day, and guess his daily behavior. Existing k-Anonymity techniques do not take into consideration the intrinsic sensibility of sequential data. Some other approaches have been proposed that requires that sensible sequences have to be pre-defined [2, 1]. Other approaches use collaborative data mining techniques [17], or propose to mine models instead of the data [15], but they do not ensure that sensible sequences can not be extracted.

In this paper, we propose a new technique that provides an anonymized dataset of sequences, while preserving sequential pattern mining results. We use a method which combines k-anonymity (the disclosed dataset is such that any sequence is undistinguishable with at least $k-1$ other sequences) and sequence hiding approaches. Our approach consists in a reformulation of the anonymization problem as the problem of hiding k-infrequent sequences, i.e., transforming the original sequence database in such way that the sequences with support less than k in the original dataset can not be mined any longer. In the hypothesis that an attacker knows part of the sequence belonging to a person, and that s/he also know that this person is present in the database, s/he has a probability of $1/k$ of reconstructing the entire sequence. Our approach is formally defined in the general setting of sequences of items, or events. To illustrate its effectiveness and practicality in a realistic and complex domain, we put at work our anonymization technique in the scenario of moving object data analysis, and applied it to a large-scale, real-life dataset of GPS trajectories of vehicles with on-board GPS receivers, tracked in the city of Milan, Italy. The results of our experiments, where we compare the set of sequential patterns obtained before and after the application of our anonymization technique, show that we can substantially preserve such frequent sequential patterns, while guaranteeing that the disclosed data are k-anonymous.

The rest of the paper is organized as follows. Section 2 briefly discusses the relevant related works on privacy-preserving data mining. Section 3 introduces and explains our Privacy-Preserving k-Anonymization (P2kA) framework. The algorithmic details are given in Section 4, together with explanations on a toy example consisting of a small set of sequences. The experimental results of our application to a moving object dataset are presented and discussed in Section 5. Finally, Section 6 concludes.

2 Related works

A lot of recent research works have focused on techniques for privacy-preserving data mining [5] and for privacy-preserving data publishing. Important techniques include perturbation, condensation, and data hiding with conceptual reconstruction. The first step before data publishing is to remove the personally identifying information. In [24] (and much earlier in statistics by T. Dalenius [9]), it has been shown that removing personally identifying information is not enough to protect privacy. In this work, Samarati and Sweeney propose a classification of the attributes in *quasi-identifiers* (i.e., attributes that can be linked to external information to re-identify the individual to whom the information refers, a concept that was already present in [10]), and sensitive attributes. Moreover, they propose the k-anonymity to generalize the values of quasi-identifier attributes in each record so that it is indistinguishable with at least $k-1$ other records with respect to the quasi-identifier, Recently, privacy-preserving data mining has been studied in conjunction with spatio-temporal data and trajectory mining [12, 8]. In the work presented in [3], the authors study the problem of anonymity preserving data publishing in moving objects databases. They propose the notion of $(k, \delta) - anonymity$ for moving objects databases. In particular, this is a novel concept of k-anonymity based on co-localization that exploits the inherent uncertainty of the moving objects whereabouts. The k-anonymity notion is also used in [22], where authors address privacy issues regarding the identification of individuals in static trajectory datasets. They provide privacy protection by: (1) first enforcing k-anonymity, meaning every released information refers to at least k users/trajectories, (2) then reconstructing randomly a representation of the original dataset from the anonymization. Although it has been shown that the k-anonymity framework presents some flaws and limitations [20], and that finding an optimal k-anonymization is NP-hard [6], the k-anonymity model is still practically relevant and in recent years a large research effort has been devoted to develop algorithms for k-anonymity [16, 18].

Existing work about anonymity of spatio-temporal moving points has been mainly developed in the context of location based services (LBS) [21, 26, 14, 7]. Works in [21, 26] use perturbation and obfuscation techniques to de-identify a given request or a location. In [14], anonymity is enforced on sensitive locations other than user location points or trajectories. In [7], anonymization process enforces points referring to same set of users to be anonymized together. However this work considers the anonymization of a request rather than the whole trajectory anonymization. In order to preserve the privacy for moving object data in [1] the authors propose a hiding technique. In particular, they address the problem of hiding sensitive trajectory patterns from a database of moving objects. A similar technique is used in [2], where Abul et al. address first the problem of hiding patterns that are a simple sequence of symbols and then they extend the proposed framework to the case of sequential patterns according to the classical definition [5]. A first work attacking the problem of limiting disclosure of sensitive rules by reducing their significance, while leaving unaltered or minimally affecting the significance of others, non-sensitive rules is [6]. One of the most important contributions of this paper is the proof that finding an optimal sanitization of a dataset is NP-hard. A heuristic using greedy search is thus proposed. In the work [11] the objective is to hide individual sensitive rules instead of all rules produced by some sensitive itemsets. The work in [25] proposes two distortion-based heuristic techniques for selectively hiding sensitive rules. An interesting work is presented in [15], where Jacquemont et al. propose a costless solution to privacy preserving for problems that may be stated as flow control problems, that is the case of frequent path discovery in Web sites and frequent route discovery in towns. They propose to model this flow of data in the form of a weighted automaton, for which they provide a probabilistic solution to discover frequent patterns (potentially with gaps) under constraints, without any information about the original data.

Essentially, in our work we present a new anonymization technique for preserving privacy and at same time, preserving also frequent sequential patterns (FSP) obtained by mining the anonymized data. The basic *frequent sequential pattern* problem, originally introduced in [4], is defined over a database of sequences D, where each element of each sequence is a time-stamped set of items — i.e., an *itemset.* Time-stamps determine the order of elements in the sequence. Then, the FSP problem consists in finding all the sequences that are *frequent* in the database, i.e., appear as subsequence of a large percentage of sequences of the database. Since its first definition, many algorithms for sequential patterns have been proposed, from the earliest in [4], to the more recent PrefixSpan [23] and SPADE [27].

3 Problem Definition

Let $L = \{l_1, l_2, \ldots, l_n\}$ denote a set of items (e.g, spatial locations or regions). A sequence $S = s_1 s_2 \ldots s_m$ ($s_i \in L$) is an ordered list of items, and an item can occur multiple times in a sequence. A sequence $T = t_1 t_2 \ldots t_w$ is a subsequence of S ($T \preceq S$) if there exist integers $1 \leq i_1 < \ldots < i_w \leq m$ such that $\forall 1 \leq j \leq w$ $t_j = s_{i_j}$. A sequence database $\mathcal{D}$ is a set of sequences $\mathcal{D} = \{S_1, S_2, \ldots, S_N\}$. The support of a sequence T in a database $\mathcal{D}$ is the number of sequences in the database containing T, i.e.:

$$supp_{\mathcal{D}}(T) = |\{S \;\; s.t. \;\; S \in \mathcal{D} \wedge T \preceq S\}|$$

Given a support threshold σ, a sequence T is called a σ-frequent sequential pattern in a sequence database $\mathcal{D}$ if $supp_{\mathcal{D}}(T) \geq \sigma$. The collection of all σ-frequent (sequential) patterns in $\mathcal{D}$ is denoted by $\mathcal{S}(\mathcal{D}, \sigma)$. The set of all subsequences supported by $\mathcal{D}$ is denoted by $\mathcal{S}(\mathcal{D})$.

Our goal is to provide an anomymized version of $\mathcal{D}$ that preserves as much as possible the collection of frequent patterns. We use a method which combines k-anonymity and sequence hiding approaches. Put in other words, we reformulate the anonymization problem — in the case of sequential data — as the problem of hiding k-infrequent sequences, i.e., transforming the original sequence database in such way that the sequences with support less than k in the original dataset can not be mined any longer. The disclosed dataset is such that any sequence is undistinguishable with at least $k-1$ other sequences. This goal is achieved by hiding all the subsequences which are not supported by at least k sequences in the database. Let $\mathcal{D}'$ denote the disclosed dataset. Given a positive integer k, the disclosed dataset $\mathcal{D}'$ is such that

$$\sum_{T \in \mathcal{S}(\mathcal{D}')} \delta[supp_{\mathcal{D}}(T) < k] \cdot supp_{\mathcal{D}'}(T) = 0$$

where $\delta[condition]$ is the Dirichlet function (which is equal to 1 if *condition* is true, 0 otherwise). In this paper we consider that any infrequent subsequence of items can potentially lead to the identification of the user (respondent). Thus, we do not need to specify any sensible subsequence preliminarily, as in [2, 1]. Moreover, we want to preserve frequent pattern mining results, in order to let the analysts investigate over frequent and interesting/unexpected behavior. The optimal **pattern-preserving k-anonymization problem** can be formulated as follows:

Definition 1 (optimal P2kA problem). *Given a sequence database $\mathcal{D}$, and a positive integer k, find a database $\mathcal{D}'$ such that*

1. *$\mathcal{D}'$ is k-anonymous, i.e.:*

$$\sum_{T \in \mathcal{S}(\mathcal{D}')} \delta[supp_{\mathcal{D}}(T) < k] \cdot supp_{\mathcal{D}'}(T) = 0$$

2. *the collection of all k-frequent pattern in $\mathcal{D}$ is preserved, i.e.:*

$$\begin{array}{l} \mathcal{S}(\mathcal{D}', k) = \mathcal{S}(\mathcal{D}, k) \\ \forall T \in \mathcal{S}(\mathcal{D}', k) \;\; supp_{\mathcal{D}'}(T) = supp_{\mathcal{D}'}(T) \end{array}$$

In this paper we present an algorithm which assures that (i) $\mathcal{D}'$ is k-anonymous and (ii) $\mathcal{S}(\mathcal{D}', k)$ and $\mathcal{S}(\mathcal{D}, k)$ are "similar". In particular the second condition of Definition 1 becomes:

$$\begin{array}{l} \mathcal{S}(\mathcal{D}', k) \subseteq \mathcal{S}(\mathcal{D}, k) \\ \forall T \in \mathcal{S}(\mathcal{D}', k) \;\; supp_{\mathcal{D}'}(T) \simeq supp_{\mathcal{D}'}(T) \end{array}$$

```
Algorithm 1: BF-P2kA($\mathcal{D}$, $k$)
  Input: A sequence database $\mathcal{D}$, a minimum support threshold $k$
  Output: A $k$-anonymous sequence database $\mathcal{D}'$
  $\mathcal{PT} = PrefixTreeConstruction(\mathcal{D})$;
  $\mathcal{PT}' = PTAnonymization(\mathcal{PT}, k)$
  $\mathcal{D}' = SequenceGeneration(\mathcal{PT}')$;
  return $\mathcal{D}'$
```

```
Algorithm 2: PTAnonymization($\mathcal{PT}$, $k$)
  Input: A prefix tree $\mathcal{PT}$, a minimum support threshold $k$
  Output: A $k$-anonymous prefix tree $\mathcal{PT}'$
  $\mathcal{L}_{cut} = \emptyset$;
  foreach $n$ in $Root(\mathcal{PT}).children$ do
      $\mathcal{L}_{cut} = \mathcal{L}_{cut} \cup TreePruning(n, \mathcal{PT}, k)$;
  end
  $\mathcal{PT}' = TreeReconstruction(\mathcal{PT}, \mathcal{L}_{cut})$;
  return $\mathcal{PT}'$
```

In the experimental section (see Section 5) we will express this similarity in terms of two measures which quantify how much the pattern support changes, and how many frequent pattern we miss. As a preliminary step towards an ”optimal” algorithm, we will show that our algorithm provides good results in term of pattern similarity (see Section 5), and guarantees that the disclosed dataset is k-anonymous.

4 The BF-P2kA algorithm

In this section we present our *BF-P2kA* (Brute Force Pattern-Preserving k-Anonymization) algorithm (Algorithm 1), which allows to anonymize a dataset of sequences $\mathcal{D}$. Our approach consists of three steps. During the first step, the sequences in the input dataset $\mathcal{D}$ are used to build a prefix tree $\mathcal{PT}$. The second step, given a minimum support threshold k, anonymizes the prefix tree. This means that sequences whose support is less than k are pruned from the prefix tree. Then part of these infrequent sequences is re-appended in the prefix tree. The third and last step post-process the anonymized prefix tree, as obtained in the previous step, to generate the anonymized dataset of sequences $\mathcal{D}'$.

Step I: Prefix Tree Construction The first step of the *BF-P2kA* algorithm (Algorithm 1) is the construction of a prefix tree $\mathcal{PT}$, given a list of sequences $\mathcal{D}$. The created prefix tree is a more compact structure than a list of sequences. It is defined as a triplet $\mathcal{PT} = (\mathcal{N}, \mathcal{E}, Root(\mathcal{PT}))$, where $\mathcal{N}$ is a finite set of labeled nodes, $\mathcal{E}$ is a set of edges and $Root(\mathcal{PT}) \in \mathcal{N}$ is a fictitious node and represents the root of the tree. Each node of the tree (except the root) has exactly one parent and it can be reached through a path, which is a sequence of

Algorithm 3: PrefixTreeConstruction($\mathcal{D}$)

```
Input: A sequence database $\mathcal{D}$
Output: A prefix tree $\mathcal{PT}$
foreach $T$ in $\mathcal{D}$ do
    $LP = LongestPrefixSearch(Root(\mathcal{PT}), T)$;
    Append $T$ to $LP$;
    foreach $v$ in $LP$ do
        $v.support = v.support + supp_{\mathcal{D}}(T)$;
    end
    foreach $v$ in $T \setminus LP$ do
        $v.support = supp_{\mathcal{D}}(T)$;
    end
end
return $\mathcal{PT}$
```

edges starting with the root node. An example of path for the node d (denoted $\mathcal{P}(d, \mathcal{PT})$) is the following:

$$\mathcal{P}(d, \mathcal{PT}) = (Root(\mathcal{PT}), a), (a, b), (b, c), (c, d).$$

Each node $v \in \mathcal{N}$, except $Root(\mathcal{PT})$, has entries in the form $\langle id, item, support, children\rangle$ where:

- *id* is the identifier of the node v
- *item* represents an item of a sequence
- *support* is the support of the sequence represented by the path from $Root(\mathcal{PT})$ to v
- *children* is the list of child nodes of v.

The *PrefixTreeConstruction* algorithm (see Algorithm 3) for each sequence of items T searches in $\mathcal{PT}$ the path which corresponds to the longest prefix of the sequence T. Next, it appends, to the last node of the longest prefix found, a branch which represents the remaining elements of T, updating the involved node attributes accordingly. In particular, it updates the support of each node belonging to the common prefix by adding the support of the sequence T in $\mathcal{D}$, and sets the support of the remaining nodes to $supp_{\mathcal{D}}(T)$.

Step II: Prefix Tree Anonymization The main phase of our approach is the second one. This phase is described by the *Tree Anonymization* Algorithm (Algorithm 2). Before describing this algorithm we introduce some notions which are needed to better explain our method.

Definition 2 (minimum prefix). *Let $S = s_1 s_2 \ldots s_n$ and $T = t_1 t_2 \ldots t_k$ be two sequences such that T is a subsequences of S and s_p is the first item of S such that $T \preceq s_1 s_2 \ldots s_p$. The sequence $S' = s_1 \ldots s_p$ is the* minimum prefix *of S containing the sub-sequence T.*

```
Algorithm 4: TreePruning($n$, $\mathcal{PT}$, $k$)
Input: A node $n$, a prefix tree $\mathcal{PT}$, a minimum support threshold $k$
Output: A list of infrequent sequences $\mathcal{L}_{cut}$
$\mathcal{L}_{cut} = \emptyset$;
if $n.support < k$ then
    $\mathcal{L}_{cut} =$ the set of all sequences in $PathTree(\mathcal{PT}, n)$;
    foreach $j \in \mathcal{P}(n, \mathcal{PT})$ do
        $j.support = j.support - n.support$;
    end
    $\mathcal{PT} = \mathcal{PT} \setminus$ the subtree induced by $n$;
else
    foreach $j \in n.children$ do
        $\mathcal{L}_{cut} \cup TreePruning(j, \mathcal{PT}, k)$;
    end
end
return $\mathcal{L}_{cut}$
```

Example 1. Let us consider the sequences

$$S = ABCDECDF$$
$$T = ACD$$

The sequence $S' = ABCD$ is the minimum prefix of S containing the subsequence T.

Definition 3 (path tree). *Let $\mathcal{PT}$ be a prefix tree, let n be a node in the prefix tree $\mathcal{PT}$. The* path tree *of n in $\mathcal{PT}$ (denoted by $PathTree(\mathcal{PT}, n)$) is the subtree induced by the set of nodes belonging to $\mathcal{P}(n, \mathcal{PT})$ plus the subtree induced by n.*

We recall now the well-known notions of Levenshtein distance [19] and Longest Common Subsequence, which are used in our algorithm.

Definition 4 (Levenshtein distance). *Let S and T be two sequences. The **Levenshtein (edit) distance** between S and T is given by the minimum number of operations needed to transform a sequences into the other, where an operation is an insertion, deletion, or substitution of a single element.*

Definition 5. *Let $\mathcal{T}$ be a set of sequences. The **Longest Common Subsequence** (LCS) is the longest subsequence common to all sequences in $\mathcal{T}$.*

The first step of the Algorithm 2 is the pruning of the prefix tree with respect to the minimum support threshold given in input. This operation is executed thanks to the *TreePruning* function (see Algorithm 4). Indeed, this function modifies the tree by pruning all the infrequent subtrees and updating the support of the path to the last frequent node. In particular, it visits the tree and, when the support of a given node n is less than the minimum support threshold k, it computes all the sequences represented by the paths which contain

```
Algorithm 5: TreeReconstruction($\mathcal{PT}$, $\mathcal{L}_{cut}$)
  Input: A prefix tree $\mathcal{PT}$, a list of infrequent sequences $\mathcal{L}_{cut}$
  Output: An anonymized reconstructed prefix tree $\mathcal{PT}'$
  foreach distinct $S \in \mathcal{L}_{cut}$ do
      $cand = ClosestLCS(S, \mathcal{PT})$;
      $L$ = the set of nodes in $\mathcal{PT}$ belonging to the first minimum prefix
      containing $cand$;
      if $L$ is not empty then
          foreach $node \in L$ do
              $node.support = node.support + supp_{\mathcal{L}_{cut}}(S)$;
          end
      end
  end
  return $\mathcal{PT}$
```

the node n and which start from the root and reach the leaves of the sub-tree with root n. Note that for construction each node of this sub-tree has support less than k. All the computed sequences and their supports are inserted in to the list $\mathcal{L}_{cut}$. Next, the subtree with root n is cut from the tree. Therefore, the procedure *TreePruning* returns a pruned prefix tree and the list $\mathcal{L}_{cut}$. After the pruning step, the algorithm redistributes the infrequent sequences in $\mathcal{L}_{cut}$ into the pruned tree, using the *TreeReconstruction* function (see Algorithm 5). In particular, for each infrequent sequence S in $\mathcal{L}_{cut}$, it computes the LCS between S and every sequence represented by the tree. Suppose that T is the sequence such that the computed LCS is subsequence of T. Thus, the *TreeReconstruction* function selects the path of the tree that represents the minimum prefix of T containing the LCS, and increases the support of the related nodes by adding the support of S in $\mathcal{L}_{cut}$. If there are more LCSs having the same length, the function *ClosestLCS* function returns the LCS and the sequence in $\mathcal{PT}$ such that the Levenshtein distance between them is minimum. This choice allows to increase the support of a limited set of nodes not belonging to the LCS, thus reducing the noise.

Step III: Generation of anonymized sequences *PTAnonymization* algorithm returns an anonymized prefix tree, i.e., a prefix tree where only k-frequent subsequences are represented. The third step our method allows to generate the anonymized dataset $\mathcal{D}'$. This phase is performed by the *SequenceGeneration* procedure, which visits the anonymized prefix tree and generates all the represented sequences. Of course, while a sequence is generated the *Sequences-Generation* procedure considers the support of this sequence.

We show now that (i) our approach guarantees that the disclosed dataset $\mathcal{D}'$ is k-anonymous (i.e., patterns whose support is less than k in the original dataset $\mathcal{D}$ are not represented in $\mathcal{D}'$) and (ii) the set of sequential patterns in $\mathcal{D}'$ is a subset of those in $\mathcal{D}$.

s_1	A B C D E F
s_2	A B C D E F
s_3	A B C D E F
s_4	A D E F
s_5	A D E F
s_6	A D E F
s_7	B K S
s_8	B K
s_9	B K
s_{10}	D E J F

(a) A dataset of sequences

s'_1	A B C D E F
s'_2	A B C D E F
s'_3	A B C D E F
s'_4	A D E F
s'_5	A D E F
s'_6	A D E F
s'_7	A D E F
s'_8	B K
s'_9	B K
s'_{10}	B K

(b) Anonymized dataset of sequences

Fig. 1. A toy example

Theorem 1. *Let $\mathcal{D}$ be a dataset of sequences. Given a minimum support threshold k, the dataset $\mathcal{D}'$ returned by Algorithm 1 satisfies the following conditions:*

1. $\mathcal{D}'$ is k-anonymous, i.e.:

$$\sum_{T \in \mathcal{S}(\mathcal{D}')} \delta[supp_{\mathcal{D}}(T) < k] \cdot supp_{\mathcal{D}'}(T) = 0$$

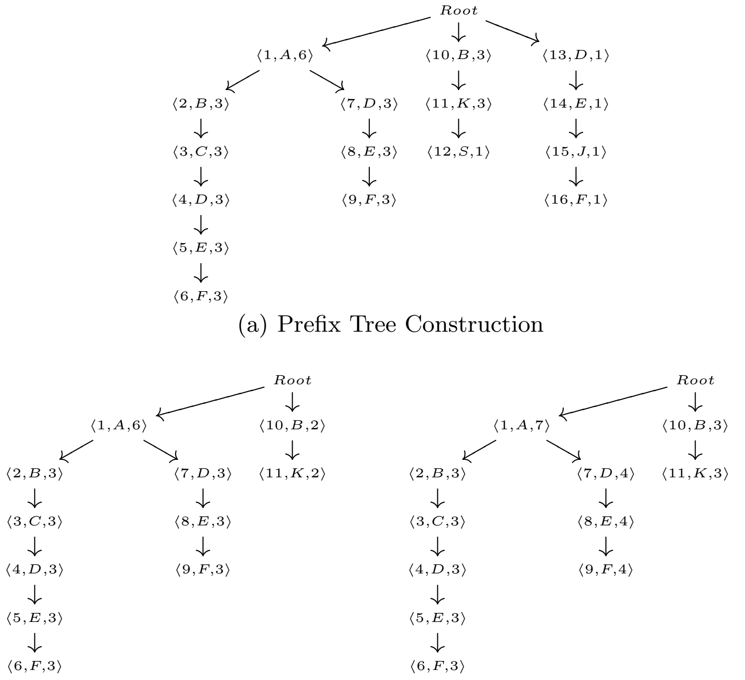
2. $\mathcal{S}(\mathcal{D}', k) \subseteq \mathcal{S}(\mathcal{D}, k)$

where $\mathcal{S}(\mathcal{D}, k)$ and $\mathcal{S}(\mathcal{D}', k)$ are the collections of k-frequent patterns respectively in $\mathcal{D}$ and $\mathcal{D}'$

Proof. (sketch)

1. By construction, the pruning step in Algorithm 2 prunes all the subtrees with support less than k, then the prefix tree $\mathcal{PT}$ only contains k-frequent sequences. Nevertheless, the reconstruction step (see Algorithm 5) does not change the tree structure of $\mathcal{PT}$, it only increases the support of existing sequences which are already k-frequent in $\mathcal{D}$. In conclusion, at the end of the second step of Algorithm 1, the sequential patterns which are represented in $\mathcal{PT}'$ are at least k-frequent in $\mathcal{D}$.
2. At the end of the pruning step in Algorithm 2, all infrequent branches in $\mathcal{PT}$ are cut off. However, this could also imply that some k-frequent sequential patterns are pruned out, if they are only supported by multiple infrequent paths in the prefix tree $\mathcal{PT}$. Then, the prefix tree $\mathcal{PT}$ contains a subset of the $\mathcal{S}(\mathcal{D}, k)$. Moreover, as already stated, during the reconstruction step the tree structure of $\mathcal{PT}$ is unchanged, i.e., patterns represented in $\mathcal{PT}'$ were still represented in $\mathcal{PT}$ after the pruning step. Finally, the set of sequential patterns supported by $\mathcal{D}'$ is a subset of those supported by $\mathcal{D}$.

Even if our approach does not assure that $\mathcal{S}(\mathcal{D}', k) = \mathcal{S}(\mathcal{D}, k)$, we will show in Section 5 that the difference between the two sets can be very small in practice.



(a) Prefix Tree Construction

(b) Pruned Prefix Tree

(c) Anonymized Prefix Tree

Fig. 2. Prefix tree processing

4.1 A running example

We present now an example which shows how our approach works. We consider the dataset of sequences in Figure 1(a) and a minimum support threshold equal to 2. During the first phase of our method the *PrefixTreeConstruction* algorithm builds the prefix tree depicted in Figure 2(a), which represents the sequences in a more compact way.

During the anonymization step, the prefix tree is modified by the *TreePruning* procedure with respect to the minimum support threshold. In particular, this procedure searches the tree for all nodes with support less than 2:

```
<12, S, 1>  <13, D, 1>.
```

Next, it selects the paths that contain these nodes and which start from the root and reach each leaves belonging to the subtrees of these nodes. Then, it generates all the sequences represented by these paths and inserts them into the list $\mathcal{L}_{cut}$:

```
(B K S, 1)    (D E J F, 1).
```

Finally, the *TreePruning* procedure eliminates from the tree the subtrees induced by the infrequent nodes listed above and updates the support of the remaining nodes. The prefix tree obtained after the pruning step is shown in the Figure 2(b).

The infrequent sequences within $\mathcal{L}_{cut}$ are then redistributed in this way:

1. `(B K S, 1)` increases the support of the following nodes
   `<10, B, 2>   <11, K, 2>`
   and thus we obtain
   `<10, B, 3>   <11, K, 3>`
2. `(D E J F, 1)` increases the support of the following nodes of the tree
   `<1, A, 6>   <7, D, 3>   <8, E, 3>   <9, F, 3>`
   therefore we obtain
   `<1, A, 7>   <7, D, 4>   <8, E, 4>   <9, F, 4>`.

The prefix tree obtained after the anonymization step is shown in Figure 2(c). Finally, the *SequencesGeneration* procedure provides the anonymized sequence dataset shown in Figure 1(b).

5 Experiments & Results

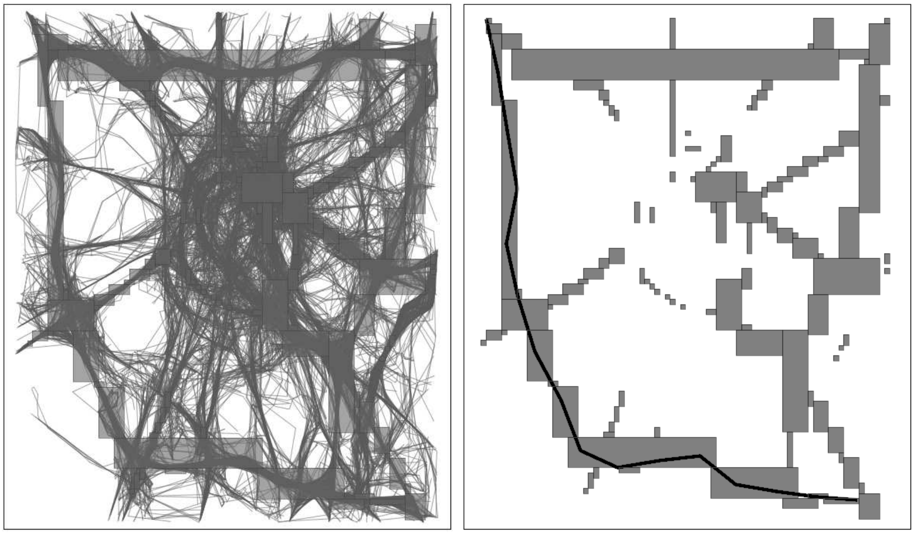
In this section, we present an application to a moving objects dataset. Object trajectories are first transformed into sequences of crossed locations, and then processed with our anonymization approach. In the following, we discuss the results over multiple instances of the original data, for different anonymization degrees.

5.1 Data Preparation

In this section, we explain the procedure used to obtain the input datasets. We got a set of GPS trajectories of cars from the european project *GeoPKDD*[4] that cover a week of traffic in Milan. Essentially, each trajectory is a sequence of pairs of coordinates x and y with relative timestamp. Obviously, performing our algorithm over sequences of points is practically useless because it is impossible to find a set of points that exactly matches enough times for being considered frequent with respect to any values of k. Thus, to overcome this problem, we use the definition of Regions of Interest given in [13], where the authors discretize the working space through a regular grid with cells of small size. Then the density of each cell is computed by considering each single trajectory and incrementing the density of all the cells that contain any of its points. Finally a set of RoI's is extracted by means of a simple heuristics using a density threshold.

As a result, a set of Roi's provides a coverage of dense cells through different sized, disjoint, rectangular regions with some form of local maximality. In particular, for each region they consider the average density of its cells, instead of its overall density (which is generally higher), and larger rectangles are preferred only if they add dense regions.

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



(a) (b)

Fig. 3. Trajectories and regions.

Once the set of RoI's has been extracted, we preprocess all the input trajectories translating each one from a sequence of points to a sequence of RoI's. The order of visit is maintained by means of timestamps. An example of this simple procedure of translation is shown in Fig. 3 — on the left we can see all the trajectories and a set of RoI's extracted; on the right we show a trajectory and we evidence which RoI's it crosses. This new dataset represents the input dataset for the anonymization algorithm.

The datasets used in our experiments are built using all the trajectories in the dataset described above with different density thresholds. These values have been chosen in order to obtain an adequate number of RoI's, since low density values correspond to few big regions, and higher values produce few small regions. In that way, we obtain different sets of RoI's meaning different sets of items in the input sequences. Table 1 summarizes the datasets used in our experiments. Notice that the number of trajectories is different among the datasets because we lose those trajectories that do not cross any region.

5.2 Results and discussion

Since our goal is to preserve local patterns (i.e., local subsequences) as much as possible, we compare the collections of pattern extracted before and after the anonymization process. To measure the similarity between two collection of patterns, we define two metrics:

Dataset	**Density threshold**	**N. of Regions**	**N.of Trajectories**	**Avg. Length**
1	0.01	113	82341	8.327
2	0.035	31	28663	9.152
3	0.037	21	24995	7.519
4	0.038	16	23744	6.239
5	0.039	10	10604	6.687
6	0.04	8	9213	6.863

Table 1. Input parameter

– SIM1 (Frequent Pattern Support Similarity): defined as

$$\frac{1}{|\mathcal{S}(\mathcal{D}',\sigma)|}\sum_{s\in\mathcal{S}(\mathcal{D}',\sigma)}\frac{\min\{\mathit{freq}(s,\mathcal{D}'),\mathit{freq}(s,\mathcal{D})\}}{\max\{\mathit{freq}(s,\mathcal{D}'),\mathit{freq}(s,\mathcal{D})\}}$$

– SIM2 (Frequent Pattern Collection Size Similarity): defined as

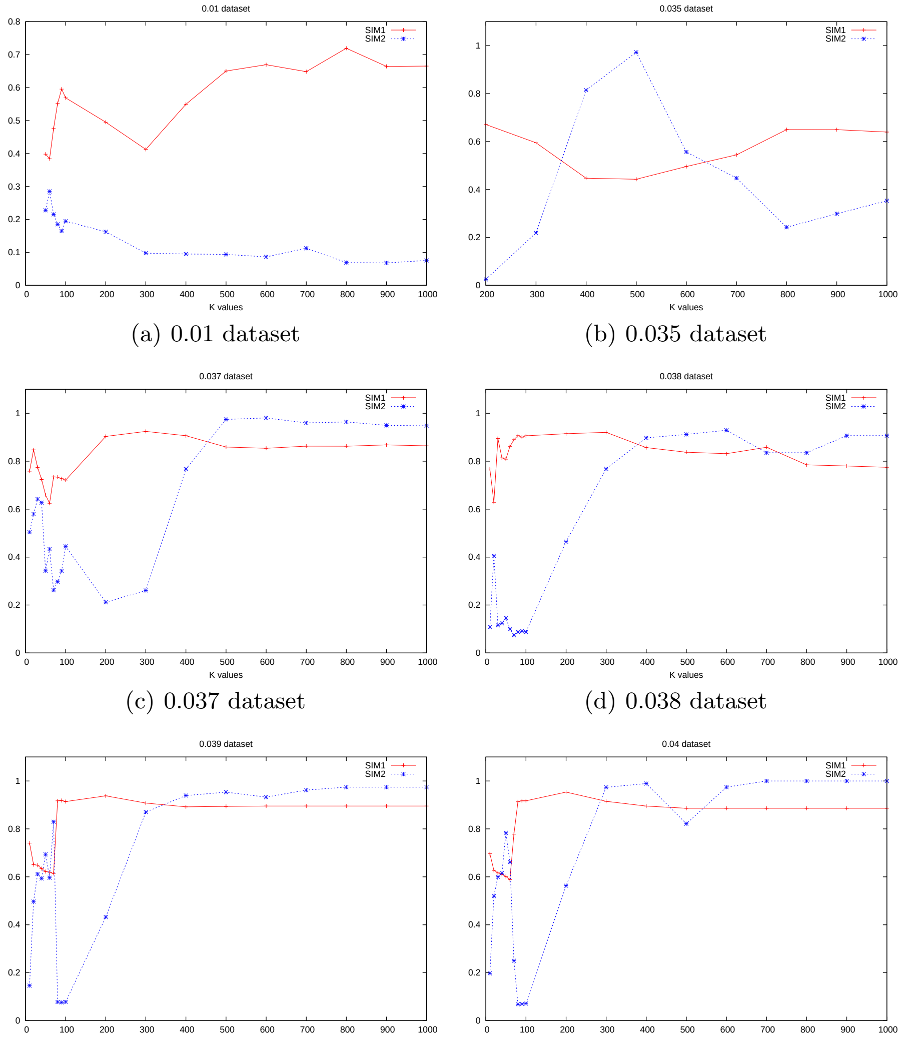
$$\frac{\min\{|\mathcal{S}(\mathcal{D}',\sigma)|,|\mathcal{S}(\mathcal{D},\sigma)|\}}{\max\{|\mathcal{S}(\mathcal{D}',\sigma)|,|\mathcal{S}(\mathcal{D},\sigma)|\}}$$

All these measures are defined between 0 and 1. When two collections of subsequences are identical, the two measures are all equal to 1.

Our experiments were conducted as follows: we first anonymized the six datasets using values of k between 10 and 1000. Then, for each value of k we compared the collection of frequent patterns extracted from the original dataset and the collection extracted from the k-anonymized dataset. In all these experiments, we used PrefixSpan [23] and the minimum support threshold was set to k.

In Figure 4 we report the results of all the experiments. We were unable to compare results for $k < 50$ and $k < 200$ for the two first datasets, since the related pattern collections are too huge and then untractable. As expected, some frequent patterns in $\mathcal{D}$ are missing in $\mathcal{D}'$. This is more evident in the first two datasets (see Figure 4(a) and 4(b)), while for higher density thresholds (Figure 4(c) to 4(f)) the value of SIM2 is closer to the maximum. This is in part due to the fact that when data are sparser, the anonymization algorithm tends to prune more sequences. Concerning the effective support similarity (SIM1), the results show that the higher the k threshold, the more similar the relative frequencies. Moreover, the SIM1 measure is quite high in general (the only exception is for the 0.035 dataset).

It is interesting to notice that, for some datasets, it is possible to identify an "optimum" minimum value of k. For instance, if we look at the similarity measures for the last dataset (see Figure 4(f)), $k = 300$ that preserves the number of frequent patterns, as well as their support. For the first dataset (see Figure 4(a)), two good choices are $k = 100$ and $k = 500$. This may possibly help the data publisher in deciding of a suitable value of k. A possible methodology would consist in finding the best tradeoff (w.r.t. the application) between the anonymization degree and the number of preserved patterns.



(a) 0.01 dataset

(b) 0.035 dataset

(c) 0.037 dataset

(d) 0.038 dataset

(e) 0.039 dataset

(f) 0.04 dataset

Fig. 4. Values of SIM1 and SIM2 for different location datasets

6 Conclusion and future work

In this paper, we introduced a new approach for anonymizing sequential datasets. Our approach provides k-anonymous data generalizing the sequence hiding approach. Through an experiment of application to a real-life mobility dataset, we showed that the proposed technique preserves sequential pattern mining results both in terms of number of extracted patterns and their support.

Further research will investigate over new approaches to preserve pattern mining results also in other hard contexts, such as sparse datasets or long sequences. One possible strategy might require the usage of a different and more compact data structure, instead of the prefix tree which is used here. Moreover, another investigation possibility could be oriented to a relaxed privacy constraint. Instead of guaranteeing the full satisfaction of k-anonymity, we could enable better pattern mining results despite of a less aggressive (and slightly more risky) pruning step. Concerning the application to mobility data, our approach does not consider yet the precious information carried by temporal annotations as well as the geographical proximity of locations/regions. A deep research effort will be undertaken to investigate on the possible extension of our approach towards a comprehensive privacy-preserving spatio-temporal framework.

Acknowledgements The authors wish to thank the anonymous reviewers for their precious comments, and Roberto Trasarti for his technical support. This work has been carried out within the European project GeoPKDD, which acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Sixth Framework Programme for Research of the European Commission, under FET-Open contract number 014915.

References

1. Osman Abul, Maurizio Atzori, Francesco Bonchi, and Fosca Giannotti. Hiding sensitive trajectory patterns. In *ICDM Workshops*, pages 693–698. IEEE Computer Society, 2007.
2. Osman Abul, Maurizio Atzori, Francesco Bonchi, and Fosca Giannotti. Hiding sequences. In *ICDE Workshops*, pages 147–156. IEEE Computer Society, 2007.
3. Osman Abul, Francesco Bonchi, and Mirco Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *ICDE*, pages 376–385. IEEE, 2008.
4. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of ICDE*, 1995.
5. R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD 2000)*, pages 439–450, 2000.
6. M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. S. Verykios. Disclosure limitation of sensitive rules. In *Proceedings of the 1999 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99)*, pages 45–52, 1999.
7. Claudio Bettini, Xiaoyang Sean Wang, and Sushil Jajodia. Protecting privacy against location-based personal identification. In *Proceedings VLDB Workshop SDM 2005*, volume 3674 of *LNCS*, pages 185–199. Springer, 2005.

8. Francesco Bonchi, Yücel Saygin, Vassilios S. Verykios, Maurizio Atzori, Aris Gkoulalas-Divanis, Selim. V. Kaya, and Erkay Savas. *Privacy in Spatiotemporal Data Mining*, pages 297–329. Springer Verlang, 2008.
9. Tore Dalenius. The invasion of privacy problem and statistics production — an overview. *Statistik Tidskrift*, 12:213–225, 1974.
10. Tore Dalenius. inding a needle in a haystack — or identifying anonymous census record. *Journal of Official Statistics*, 2(3):329–336, 1986.
11. Elena Dasseni, Vassilios S. Verykios, Ahmed K. Elmagarmid, and Elisa Bertino. Hiding association rules by using confidence and support. In Ira S. Moskowitz, editor, *Information Hiding*, volume 2137 of *LNCS*, pages 369–383. Springer, 2001.
12. Dino Pedreschi Fosca Giannotti, editor. *Mobility Data Mining and Privacy: Geographic Knowledge Discovery.* Springer Verlang, 2008.
13. Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In Pavel Berkhin, Rich Caruana, and Xindong Wu, editors, *KDD*, pages 330–339. ACM, 2007.
14. M. Gruteser and X. Liu. Protecting privacy in continuous location-tracking applications. *IEEE Security & Privacy Magazine*, 2(2):28–34, 2004.
15. Stéphanie Jacquemont, François Jacquenet, and Marc Sebban. Sequence mining without sequences: A new way for privacy preserving. In *ICTAI*, pages 347–354. IEEE Computer Society, 2006.
16. Roberto J. Bayardo Jr. and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *ICDE*, pages 217–228. IEEE Computer Society, 2005.
17. Seung-Woo Kim, Sanghyun Park, Jung-Im Won, and Sang-Wook Kim. Privacy preserving data mining of sequential patterns for network traffic data. *Inf. Sci.*, 178(3):694–713, 2008.
18. Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In Ling Liu, Andreas Reuter, Kyu-Young Whang, and Jianjun Zhang, editors, *ICDE*, page 25. IEEE Computer Society, 2006.
19. Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.
20. Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. *L*-diversity: Privacy beyond *k*-anonymity. *TKDD*, 1(1), 2007.
21. S. Menon, S. Sarkar, and S. Mukherjee. Maximizing accuracy of shared databases when concealing sensitive patterns. *Information Systems Research*, 16(3):256–270, 2005.
22. Mehmet Ercan Nergiz, Maurizio Atzori, and Yucel Saygin. Perturbation-driven anonymization of trajectories. Technical Report 2007-TR-017, ISTI-CNR, Pisa, Italy, 2007. 10 pages.
23. J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In *ICDE*, pages 215–225, 2001.
24. P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, SRI International, March 1998.
25. Yücel Saygin, Vassilios S. Verykios, and Chris Clifton. Using unknowns to prevent discovery of association rules. *SIGMOD Record*, 30(4):45–54, 2001.
26. V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *ACM SIGMOD Record*, 33(1):50–57, 2004.
27. M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, 2001.