Hiding Sequences

Osman Abul Maurizio Atzori Francesco Bonchi Fosca Giannotti

Pisa KDD Laboratory
ISTI - CNR, Area della Ricerca di Pisa
Via Giuseppe Moruzzi, 1 - 56124 Pisa, Italy
e-mail: {name.surname}@isti.cnr.it

Abstract

The process of discovering relevant patterns holding in a database, was first indicated as a threat to database security by O' Leary in [20]. Since then, many different approaches for knowledge hiding have emerged over the years, mainly in the context of association rules and frequent itemsets mining. Following many real-world data and applications demands, in this paper we shift the problem of knowledge hiding to contexts where both the data and the extracted knowledge have a sequential structure. We provide problem statement, some theoretical issues including NP-hardness of the problem, a polynomial sanitization algorithm and an experimental evaluation. Finally we discuss possible extensions that will allow to use this work as a basic building block for more complex kinds of patterns and applications.

1. Introduction

Privacy preserving data mining, i.e., the study of data mining side-effects on privacy, has rapidly become a hot and lively research area [8, 20, 3, 27], which has seen the proliferation of many completely different approaches having different objectives, application contexts and using different techniques. The chronologically first approach in privacy preserving data mining was aimed at avoiding the identification of the original database rows (by means of data *perturbation* or *obfuscation*) while at the same time allowing the reconstruction of the data distribution at an aggregate level, and thus the production of valid mining models [3, 1, 10, 11, 23].

Another approach, named *knowledge hiding*, aims at hiding some knowledge (i.e. rules/patterns) considered sensitive, that could be inferred from the data which is going to be published. This hiding is usually obtained by *sanitizing* the database in input in such a way that the sensitive knowledge can no longer be inferred, while the original database is changed as less as possible.

A novel approach has recently emerged in privacy preserving data mining, studying the privacy threats (and possibly appropriate solutions) arising when *publishing the data mining results* themselves instead of the data [16, 6, 5, 13].

Clearly, this is not meant to be an exhaustive overview of the existing approaches. The aim is to show that there is a large variety of different approaches, that however have a common aspect: the kind of data and patterns considered. Due to the inherent challenges of protecting privacy while discovering knowledge, most of the work so far has focussed on simple, flat relations and on classical data mining tasks, such as decision trees, clustering, association rules and frequent itemsets. But nowadays, the real-world applications call for more advanced analysis of more structured and more complex data.

Consider spatio-temporal geo-referenced data daily collected by telecommunication companies exploiting mobile phones and other location-aware devices. The increasing availability of *space-time trajectories* of these personal devices and their human companions is expected to enable novel classes of applications, where the discovery of consumable, concise, and applicable knowledge is the key step. These mobile trajectories contain detailed information about personal and vehicular mobile behaviour, and therefore offer interesting practical opportunities to find behavioral patterns, to be used for instance in traffic and sustainable mobility management, e.g., to study the accessibility to services. Clearly, in these applications privacy is a concern, since location data enables intrusive inferences, which may reveal habits, social customs, religious and sexual preferences of individuals, and can be used for unauthorized advertisement and user profiling.

Similarly, consider *web usage log data* that contain traces of sequences of actions taken by a user, or *biomedical patient data* that usually contain clinical measures at different moment in time.

In each of these applications, (*i*) both the data and the kind of patterns of interest have some structure (i.e., sequences in time), (*ii*) there exist substantial privacy threats, as well as (*iii*) evident potential usefulness of knowledge discovered from these data. Therefore, privacy preserving data mining, in order to move towards maturity, should close the gap between the theory developed so far and the real-world data and applications. As a preliminary step in this direction, in this paper we shift the problem of knowledge hiding from the usual *frequent itemsets*, to contexts where both the data and the extracted knowledge have a *sequential structure*. We do not believe that this simple work alone could close the gap towards real-world applications. We believe instead, that it could represent a basilar building block that must be further developed and instantiated to the various application domains: in Section 7, for instance, we discuss the case of mobility data analysis, where both the data and the patterns extracted are trajectories, i.e., a sequence of positions (spatial locations) that are temporally close to each other and pertain to the same individual.

2. Related Work

Many different approaches for knowledge hiding have emerged over the years, mainly in the context of association rules and frequent itemsets mining, but to the best of our knowledge, no work has yet addressed the problem of hiding sequential patterns.

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 [4]. 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 is thus proposed: greedy search is performed for each sensitive itemset through its ancestors, selecting at each level the parent with the maximum support and setting it as the new itemset to be hidden. At the end of the process, a singleton items is selected. The algorithm searches through the common list of transactions that support both the selected item and the initial sensitive itemset to identify the transaction that affects the minimum number of 2-itemsets, and removes the selected item from this transaction. Then it propagates the results of this modification to the graph of frequent itemsets.

In the work [9] the objective is to hide individual sensitive rules instead of all rules produced by some sensitive itemsets. The authors propose three strategies which aim at either hiding the frequent sets that participate in these rules, or reducing the rules' importance by setting their confidence below the minimum confidence threshold. The decrement of the confidence of a rule is achieved by either increasing the support of the rule's antecedent through transactions that partially support it, or decreasing the support of the rule's consequent in transactions supporting both the antecedent and the consequent. For all three approaches the authors make the assumption that only rules supported by disjoint itemsets must be hidden. An extension of the work in [9] is presented in [28].

In [24, 25] the notion of "*unknowns*" is introduced as a mean to prevent discovery of association rules. An efficient, scalable, one-scan, heuristic algorithm, called *Sliding Window Algorithm* (SWA) is introduced in [21]. An important contribution of this work is the extension of the notion of a *disclosure threshold*: instead of using a unique threshold for the entire sanitization process, a distinct threshold is given for each sensitive rule.

The work in [22] propose two distortion-based heuristic techniques for selectively hiding sensitive rules. The hiding process may introduce a number of side effects, either by generating rules which were previously unknown, or by eliminating existing non-sensitive rules. A technique for hiding "*maximal*" sensitive patterns using a correlation matrix was introduced in [17]. Instead of selecting individual transactions and sanitizing them, the authors propose a methodology for directly constructing a sanitization matrix M by observing the relationship that holds between sensitive patterns and non-sensitive ones. This matrix is then multiplied by the original database $\mathcal{D}$, yielding a new *sanitized* database $\mathcal{D}'$ which achieves to address the privacy concerns.

The use of *border* in frequent itemset hiding was first introduced in [26], as a mean to track the impact of altering transactions. To do so, they compute both the *positive* and the *negative* borders of the itemsets lattice. During the hiding process, instead of considering every non-sensitive frequent itemset, the proposed methodology focuses on preserving the quality of the border, which directly reflects the quality of the sanitized database that is produced. A novel methodology for frequent itemset hiding based on borders' quality preservation, is also analyzed in [19]. In this paper, the authors propose an integer programming optimization algorithm for discovering the minimum number of transactions that need to be sanitized to hide the sensitive itemsets.

3. Hiding Sequential Patterns

In this section we first introduce some notation and then we formally provide *The Sequence Hiding Problem* statement. It is then proven that the problem of finding an optimal sanitization is NP-Hard.

3.1. Problem Definition

In this paper we focus on the discovery of patterns that are a simple sequence of symbols (or events, or positions). We will discuss in Section 7 how to extend the proposed framework to the case of *sequential patterns* according to the classical definition [2], i.e., sequences of sets. Let $\mathcal{D}$ be a database of sequences, where each $T \in \mathcal{D}$ is a finite sequence of symbols from an alphabet Σ: $T = \langle t_1, \ldots, t_{T_n} \rangle$ where $t_i \in \Sigma, \forall i \in \{1, \ldots, T_n\}$. We denote the set of all sequences as Σ^*. A sequence $U \in \Sigma^*$ is a subsequence of a sequence $V \in \Sigma^*$, denoted $U \sqsubseteq V$, if U can be obtained by deleting some symbols from V. More formally, $U = \langle u_1, \ldots, u_m \rangle$ is subsequence of $V = \langle v_1, \ldots, v_n \rangle$ if there are m indices $i_1 < \ldots < i_m$ such that $u_1 = v_{i_1}, \ldots, u_m = v_{i_m}$.

The support of a sequence $S \in \Sigma^*$ is the number of sequences in $\mathcal{D}$ that are supersequences of S: $sup_{\mathcal{D}}(S) = |\{T \in \mathcal{D} \,|\, S \sqsubseteq T\}|$.

The classical problem of mining frequent patterns requires, given a database $\mathcal{D}$ and a minimum support threshold σ, to compute all patterns that have a support larger than σ: $\mathcal{F}(\mathcal{D}, \sigma) = \{S \in \Sigma^* \,|\, sup_{\mathcal{D}}(S) \geq \sigma\}$.

The sequence hiding problem instead is defined as follows.

Problem 1 (The Sequence Hiding Problem) *Let $\mathcal{S}_h = \{S_1, \ldots, S_n\}$ with $S_i \in \Sigma^*, \forall i \in \{1, \ldots, n\}$, be the set of sensitive sequences that must be hidden from $\mathcal{D}$. Given a disclosure threshold ψ, the Sequence Hiding Problem requires to transform $\mathcal{D}$ in a database $\mathcal{D}'$ such that:*

1. *$\forall S_i \in \mathcal{S}_h, sup_{\mathcal{D}'}(S_i) \leq \psi$;*
2. *$\sum_{S \in \Sigma^* \setminus \mathcal{S}_h} |sup_{\mathcal{D}}(S) - sup_{\mathcal{D}'}(S)|$ is minimized.*

The problem requires to sanitize the input database $\mathcal{D}$ in such a way that a set of sensitive patterns $\mathcal{S}_h$ is hidden, while the most of the information in $\mathcal{D}$ is maintained. The resulting database $\mathcal{D}'$, is the released one.

The first requirement asks all sensitive patterns to be *hidden* in $\mathcal{D}'$, i.e., they must have a support not more than the given disclosure threshold.

The second requirement asks to minimize the sanitization effects on all non-sensitive patterns (regardless of their frequency). Note that this is equivalent to keep $\mathcal{D}'$ as similar as possible to $\mathcal{D}$. This is a very general definition which does not say *how* the sanitization is actually performed.

In the following, for sake of presentation, we assume:

1. to sanitize all occurrences of sensitive patterns, i.e., $\psi = 0$;
2. to sanitize sequences in the input database by means of an abstract operator, called *marking* that replaces a selected symbol in a position with a special symbol Δ which is not in Σ.

In the following section we show that the problem of sanitizing a sequence $T \in \mathcal{D}$, introducing the smallest number of Δ is a NP-Hard problem. Before that, we need to introduce the concept of matching set.

Definition 1 (Matching Set) *Given two sequences $S \in \mathcal{S}_h$ and $T \in \mathcal{D}$, we define the matching set of S in T, denoted $\mathcal{M}_S^T$, as the set of all sets with size $|S|$ of indices for which $S \sqsubseteq T$. For instance, let $S = \langle a, b, c \rangle$ and $T = \langle a, a, b, c, c, b, a, e \rangle$, in this case we got $\mathcal{M}_S^T = \{(1,3,4), (1,3,5), (2,3,4), (2,3,5)\}$. Moreover, given a sequence $T \in \mathcal{D}$ we define $\mathcal{M}_{\mathcal{S}_h}^T = \bigcup_{S \in \mathcal{S}_h} \mathcal{M}_S^T$.*

The notion of matching set is important to identify the point in the input database where the sanitization process should act. Clearly, if for a given sequence $T \in \mathcal{D}$ there is no match, i.e., $\mathcal{M}_{\mathcal{S}_h}^T = \emptyset$, then T does not support any sensitive sequence and thus it can be disclosed as it is. Otherwise it should be transformed such that all the matches in $\mathcal{M}_{\mathcal{S}_h}^T$ are removed. Given a sequence $T \in \mathcal{D}$ such that $\mathcal{M}_S^T \neq \emptyset$, we need to introduce a certain number of Δ symbols in T in such a way that it is sanitized.

3.2. Optimal Sanitization is NP-Hard

We now prove that the problem of identifying the smallest possible set of positions to be sanitized by replacing their symbol with Δ is a NP-Hard problem.

Problem 2 (Sequence Sanitization) Given: *A sequence T and a set of patterns $\mathcal{S}_h$ to be hidden.* **Objective:** *Find a set of position indices of T such that, replacing the symbols in the positions with Δ results in $\mathcal{M}_{\mathcal{S}_h}^T = \emptyset$.*

Theorem 1 *Optimal Sequence Sanitization Problem is NP-Hard.*

Proof: *We prove that our problem is NP-Hard by means of a reduction from the* Hitting Set Problem *(page 222 of the book by Garey and Johnson on NP-completeness [14]): given a set E and an a set of subsets C of it, find a smallest subset E' of E such that E' contains at least one element from every C.*

To make the reduction easy we assume (without loss of generality) that each $x \in C$ is a pair of distinct elements. The hitting set problem is NP-hard in this special form too. Without loss of generality let $E = \{1, 2, \ldots, n\}$.

The instance of the sequence sanitization problem is obtained as follows. Let $\Sigma = \{p_1, p_2, \ldots, p_n\}$*, where* $n = |E|$*;* $T = \langle p_1, p_2, \ldots, p_n \rangle$*; and* $\mathcal{S}_h = \{S_1, S_2, \ldots, S_{|C|}\}$ *where* S_i *is constructed solely from* i*'th element (let's denote it* C^i*) of* C *as follows:* $C^i = (j, k)$ *(this is by the assumption) where* $j, k \in E$ *and* $j < k$*, then* $S_i = \langle p_j, p_k \rangle$*.*

Now we show that the reduction above takes polynomial time and it is correct. Clearly the construction takes polynomial time, in fact it takes $O(n + |C|)$ *time, so linear in the* Hitting Set Problem *size. Now we show optimum solution for Optimal Sequence Sanitization Problem is exactly the optimum solution for* E' *for* Hitting Set Problem*. The notion of matching set serves in creating one to one correspondence and provides handling proof of both directions together.*

The proof is based on the fact that the solution to Hitting Set Problem E' *is equal to the indices of marking symbols introduced in optimal solution of Sequence Sanitization Problem as shown next. Note that size of matching set is* $|C|$ *and each* C^i *contributes exactly 1 to this matching set by the construction. Now consider moving an element* j *to* E'*, this in effect removes subsets of* C *containing* j *from further consideration. This corresponds to marking position* j *of* T *and this clearly removes all matchings having the position* j *either as a start or end position. When we continue the process this way, at the end the set* C *in effect becomes empty and this also means that the matching set* $\mathcal{M}_{\mathcal{S}_h}^T$ *is also empty.* □

Since the Optimal Sequence Sanitization Problem is NP-Hard we do not expect to find an optimal efficient solution. In other words, we need to resort to heuristics, as done in the next section.

4. A Polynomial Sanitization Algorithm

In this section we define a two-stage sanitization algorithm for hiding a set of sensitive sequences $\mathcal{S}_h$ from a database $\mathcal{D}$. During the first stage, the sequences in the input database are sanitized introducing the necessary Δ symbols. During the second stage, we either delete Δs or replace them with symbols from Σ. When the latter approach is chosen, we must take care of the possibility of re-generating fake patterns and also regenerating sensitive patterns.

In this paper, we do not consider neither deletion of Δs, nor replacement, we just focus on the marking operation. Let us point out that the second stage can be skipped totally in case we allow existence of marking symbols Δ in $\mathcal{D}'$: in this case, these symbols may be interpreted as missing values. Note that the marking operation here does not create new subsequences, thus there is no fake patterns introduced by this process.

Therefore, in the rest of this section we focus on the first stage, i.e, sanitizing by marking. Here we got two problems to address: on the local scale, given a sequence $T \in \mathcal{D}$, how to choose the positions to mark, and on the global scale, which sequences $T \in \mathcal{D}$ to sanitize. One heuristic is provided for both problems.

Intuitively, if there are small number of matches then the sanitization can be done with less distortion. So, the size of $\mathcal{M}_{\mathcal{S}_h}^T$ is a crucial issue. Unfortunately this number is exponential in the length of the sequence in the worst case.

Lemma 1 (Size of Matching Set) *The size of the matching set* $\mathcal{M}_S^T$ *is exponential in the length of sequence* T *in the worst case.*

Proof: *Consider the case where* S *and* T *are sequences of the same single symbol and nothing else. In this case we got that* $|\mathcal{M}_S^T| = \binom{|T|}{|S|}$*. The* middle binomial coefficient*, i.e., when* $k = n/2$ *is known for being the largest of the binomial coefficients* $\binom{n}{k}$*, which can be approximated by an application of Stirling's formula as:*

$$\binom{n}{\frac{n}{2}} \sim \sqrt{\frac{2}{\pi}} n^{-1/2} 2^n \quad \square$$

The key observation is that certain markings affect the number of matchings while others do not. In the following we use the usual array notation for sequences: i.e., $S[i]$ to denote the element of S at position i (with positions starting from 1).

Example 1 *Consider again the case* $S = \langle a, b, c \rangle$ *and* $T = \langle a, a, b, c, c, b, a, e \rangle$*. In this situation marking the symbol* e *(*$T[8]$*) does not affect the matching set while marking the symbol* b *in* $T[3]$ *position will cause* $\mathcal{M}_S^T = \emptyset$*. Note that the latter marking removes all the matching which is equivalent of hiding all sensitive pattern instances and thus provides sanitization. Also note that marking* $T[1]$ *reduces the number of matches without providing sanitization, while marking* $T[1]$ *and* $T[2]$ *together provides sanitization.*

Since there are many ways of providing sanitization by choosing different positions for marking, we need to choose the one with the minimum cost (the cost here is the number of non sensitive subsequences which are removed due to the marking). Our *local* heuristic is:

> *choose the marking position that is involved in most matches.*

This operation is iterated until $\mathcal{M}_{\mathcal{S}_h}^T = \emptyset$. We denote the number of matchings in which the ith position of T is involved as $\delta(T[i])$.

Example 2 *Consider again the case* $S = \langle a, b, c \rangle$ *and* $T = \langle a, a, b, c, c, b, a, e \rangle$*. We got that* $\delta(T[1]) = 2$*,* $\delta(T[2]) = 2$*,* $\delta(T[3]) = 4$*, and so on. So, we choose position* $T[3]$ *for marking (this causes* $T = \langle a, a, \Delta, c, c, b, a, e \rangle$*). Since this selection removes all matchings, no further iteration is needed.*

The considered heuristic requires the matching set $\mathcal{M}^T_{\mathcal{S}_h}$ to be computed and then, for each position of T, the number of occurrences in $\mathcal{M}^T_{\mathcal{S}_h}$ counted. It is shown (Lemma 1) that the size of $\mathcal{M}^T_{\mathcal{S}_h}$ may grow exponentially and the time required to produce this set is thus also exponential. So, the procedure of generating this set and then counting occurrences is not feasible. However, we note that the heuristic only requires the knowledge of number of matching each position is involved in, and not necessarily the $\mathcal{M}^T_{\mathcal{S}_h}$ itself. Fortunately, this can be computed in polynomial time as articulated next.

Lemma 2 (Computation of matching set size) *The computation of matching set size can be done in polynomial time.*

Proof: *Consider $S \in \mathcal{S}_h$ of length m and $T \in \mathcal{D}$ of length n. Let $P^{1..n}_{1..m}$ denote the size of $\mathcal{M}^T_S$, and, for instance $P^{1..n-1}_{1..m}$ denote the size of the matching set when the last element of T is removed. It is clear that if $S[m] \neq T[n]$, then $P^{1..n}_{1..m} = P^{1..n-1}_{1..m}$. On the other hand, when $S[m] \neq T[n]$, then $P^{1..n}_{1..m} = P^{1..n-1}_{1..m} + P^{1..n-1}_{1..m-1}$. The last equation holds since, there are two options either match last symbols or not and these two events are disjoint and exhaustive. In the boundary conditions, $P^j_0 = 1, \forall j \in \{0, 1, 2, \ldots, n\}$ and $P^0_i = 0, \forall i \in \{0, 1, 2, \ldots, m\}$. By employing dynamic programming, an algorithm with the complexity of $O(n * m)$ can be obtained.* □

The key issue now is how to compute $\delta(T[i])$ for each position i in T.

Theorem 2 *Computing $\delta(T[i])$ for each position i in T, can be done in polynomial time.*

Proof: *It is sufficient to show that this can be done in polynomial time for any given position i. All matchings in $\mathcal{M}^T_{\mathcal{S}_h}$ can be dichotomized based on whether including i or not. Consider $T' = T \setminus T[i]$, i.e., t from which we remove the ith element. The proof is based on two observations; 1) the set of matchings not involving i in the original matching set does not change 2) deletion of an element does not create any new subsequence.*

So, $\delta(T[i])$ can be computed as $\left|\mathcal{M}^T_{\mathcal{S}_h}\right| - \left|\mathcal{M}^{T'}_{\mathcal{S}_h}\right|$. □

In the case $\psi = 0$ (sensitive sequences must be completely removed from $\mathcal{D}$), we apply the same algorithm considered above for all sequences in $\mathcal{D}$. Otherwise, we have to select the set of sequences to be sanitized. We rely on the following *global* heuristic:

> *sort the sequences in $\mathcal{D}$ in ascending order of matching set size, and remove all matchings in top $|\mathcal{D}| - \psi$ input sequences.*

We consider size of matching set as a sorting criteria because if the matching set is small then we can sanitize the respective sequence with less distortion.

Algorithm 1 Sequence Hiding Algorithm

Input: $\mathcal{D}$, $\mathcal{S}_h$, ψ
Output: $\mathcal{D}'$

1: $\mathcal{D}' \leftarrow \emptyset$
2: **for all** $T \in \mathcal{D}$ **do**
3: Compute size of $\mathcal{M}^T_{\mathcal{S}_h}$
4: $\mathcal{D} \leftarrow$ Sort $\mathcal{D}$ in ascending order w.r.t. $\mathcal{M}^T_{\mathcal{S}_h}$
5: $\mathcal{D}_{sanitize} \leftarrow$ Top $|\mathcal{D}| - \psi$ sequences in $\mathcal{D}$
6: **for all** $T \in \mathcal{D}_{sanitize}$ **do**
7: $T' \leftarrow Sanitize(T, \mathcal{S}_h)$
8: $\mathcal{D}' \leftarrow \mathcal{D}' \bigcup T'$
9: $\mathcal{D}' \leftarrow \mathcal{D}' \bigcup (\mathcal{D} \setminus \mathcal{D}_{sanitize})$
10: $\mathcal{D}' \leftarrow ReplaceMarkingSymbol(\mathcal{D}', \mathcal{S}_h, \psi)$

Algorithm 1 summarizes the method that we introduced. It computes the matching set size for all $T \in \mathcal{D}$ using the strategy described in Lemma 2, and then it sorts the sequences in ascending order. The first $|\mathcal{D}| - \psi$ sequences in the order are selected to be sanitized.

The sanitization procedure $Sanitize(T, \mathcal{S}_h)$ eliminates all occurrences of $\mathcal{S}_h$ within a sequence T by introducing the Δ symbol in some chosen positions. The positions are selected according to our heuristic: i.e., first select the position with the largest $\delta(T[i])$. Therefore for each position i of T it computes $\delta(T[i])$ using the algorithm described in the proof of Theorem 2, then select the best position and sanitize it by changing the actual symbol with Δ. The process is iterated until $\mathcal{M}^T_{\mathcal{S}_h} = \emptyset$.

5. Handling Constraints

Sensitive patterns to be hidden can be further specified by means of constraints, such as the usual *minimum gap*, *maximum gap* and *maximum window* constraints. We denote minimum and maximum gap constraints by putting two integers on an arrow: $\rightarrow^{Mg}_{mg}$ such that $Mg \geq mg$, where Mg is maximum gap and mg is minimum gap. For instance, we could consider sensitive the pattern $a \rightarrow^0 b \rightarrow^6_2 c$, i.e., a directly followed by b, which in turn is followed by c after at least 2 and at most 6 other events. Let us consider again the input sequence $T = \langle a, a, b, c, c, b, a, e\rangle$: while it was supporting the pattern $\langle a, b, c\rangle$ (with a matching set of cardinality 4), it is not supporting the pattern $a \rightarrow^0 b \rightarrow^6_2 c$ due to the minimum gap constraint $b \rightarrow_2 c$. In fact, c appears after b only with a gap of 0 or 1.

It is worth noting that these constraints are not specified on the patterns itself, but are on the *oc-*

currences of a pattern within an input sequence. In other terms, a constraint is a function $\mathcal{C} : \Sigma^* \times \Sigma^* \rightarrow \{true, false\}$ (or alternatively a function $\mathcal{C} : \mathcal{D} \times \mathcal{S}_h \rightarrow \{true, false\}$) . Note also that minimum and maximum gap constraints are *local* constraints, that is satisfaction of constraints in two different arrows in the same pattern is independent. On the other hand, the maximum window constraints, i.e., the gap between the position of the first and the last symbol in the pattern (denoted Ws), is a *global* constraint, which is satisfied or not by the occurrence of the whole pattern in an input sequence.

In the rest of this section we extend the framework introduced in the previous sections, to handle this kind of constraints. A fundamental step in our framework is the computation of the size of the matching set. We next describe how this computation changes when constraints are specified.

First we must introduce an auxiliary concept. Given $S \in \mathcal{S}_h$ and $T \in \mathcal{D}$ let P_k^j be the number of matches of the prefix of length k of S ending exactly in $T[j]$.

Example 3 *Consider again* $T = \langle a, a, b, c, c, b, a, e\rangle$ *and* $S = \langle a, b, c\rangle$. *In this case we got that, for instance,* $P_2^3 = 2$, *because the length 2 prefix of* S *(i.e.,* $\langle a, b\rangle$*), has 2 matches ending exactly in* $T[3]$.

Lemma 3 P_k^j *can be computed in polynomial time.*

Proof: *The algorithm is by iterating on* j *and* k. *The base cases are:* $P_0^0 = 1$, $P_0^{j>0} = 0$ *and* $P_{k>0}^0 = 0$.

Computing $\mathbf{P_k^{j+1}}$. *This is zero if* $S[k] \neq T[j+1]$*; but if* $S[k] = T[j+1]$, *then* $P_k^{j+1} = \sum_{l=1..j} P_{k-1}^l$. *The last result is correct since it gives the total number of different partial matches of* $S[1, \ldots, k-1]$ *to* $T[1, \ldots, j]$. *Note that, by construction, no match is counted twice.*

Computing $\mathbf{P_{k+1}^j}$. *This is the dual of the previous case. That is, it is the sum of all distinct matches of* $S[1, \ldots, k]$ *in* $T[1, \ldots, j-1]$. *By definition it is* $P_{k+1}^j = \sum_{l=1..j-1} P_k^l$ *if* $S[k+1] = T[j]$ *and zero otherwise.*

The cost of whole procedure provided above is $O(n^2 m)$. *That is we need to fill all entries of a* $n \times m$ *table and each such operation requires at most* n *summations.* □

Note that the computation in Lemma 3 produces more information than the computation in Lemma 2. This is because, it is always easy to get number of matches just by summing the contributions of last matches as these are disjoint and collectively exhaustive events. Therefore, we use the computation in Lemma 3 to push constraints in Algorithm 1.

Lemma 4 (Min and Max gap constraints) *The computation of number of matches ending exactly at an index of* T, *in case of min and max gap constraints, can be done using the same strategy employed in the proof of Lemma 3.*

Proof: *The key observation is that no constraint can increase the number of unconstrained matches ending in any position* j *of* T. *Let's define* Q_k^j *as the analog of* P_k^j *in the constrained case. It is clear that if* $P_k^j = 0$, *then* $Q_k^j = 0$ *from the above observation. Let's suppose* $P_k^j > 0$, *then we compute the corresponding entry in* Q_k^j *as follows. Note that, it is already the case that* $S[k] = T[j]$, *so it suffices to find and count all matches of* $S[k-1]$ *in* $T[1, \ldots, j-1]$. *This in turn reduces to* $S[k-1] = T[l], l \in \{1, 2, \ldots, j-1\}$. *That is the index* l *shows the location of match for* $S[k-1]$ *on* T. *If the respective min gap (*mg*) and max gap (*Mg*) constraints are specified, however, this corresponds to constrain the span of index* l. *Indeed, its new span is the set* $\{max((j-1) - Mg, 1), \ldots, max(1, (j-1) - mg)\}$. *So,* $Q_k^j = \sum_{l=(j-1)-Mg..(j-1)-mg} Q_{k-1}^l$. *Note that the computation of* Q *has the same complexity as* P *and its entries are all same to respective entries of* P *in case no min gap and max gap constraints specified.* □

The maximum window constraint can be forced to find the number of matches for each position j of T. To compute it, we will make use of the method described in Lemma 3. The next Lemma formalizes this.

Lemma 5 (Max window constraint) *The computation of number of matches ending exactly at an index of* T, *in case of Max window constraint, can be done using the same strategy employed in the proof of Lemma 3.*

Proof: *The main observation is that for any ending index* j *of matching, the index of the first match should be between* $l = j - Ws + 1$ *(if it is negative or zero it is truncated to 1) and* j. *If we construct the table* P *(as described in Lemma 3) for the* $T[l, \ldots, j]$ *and* S, *the entry* P_m^{j-l-1} *gives the number of matches ending at position* $T[j]$ *under the maximum window constraint.* □

In the case of a conjunction of min/max gap and maximum window constraints, we can just use table Q_k^j (specified in Lemma 4), instead of P_k^j (specified in Lemma 3), within the computation described in Lemma 5.

6. Experimental Evaluation

In this section we report the experiments we conducted in order to assess the effectiveness of the proposed approach, in terms of distortion introduced by

the sanitization. We experimented on two datasets; a real truck movement data used in [12], containing 273 trajectories, and a synthetic car movement data, containing 300 trajectories, generated in our laboratory [15]. The former dataset is referred TRUCKS and the latter SYNTHETIC throughout. In both of these datasets, the movement sequences are discretized using 10 by 10 grid where locations are indexed with X_iY_j, where $i, j \in \{1, 2, \ldots, 10\}$. Thus the trajectories are discretized in sequences over an alphabet Σ of 100 symbols. This discretization results in 20.1 (resp. 6.8) locations (symbols), on average, per trajectory for TRUCKS (resp. SYNTHETIC) datasets.

Since there is no other algorithm for sequence sanitization in the literature, we measure the effectiveness of the proposed algorithm against random sanitization. Recall that the algorithm employs two orthogonal heuristics: (i) for selecting the positions to be sanitized within a given sequence; (ii) for selecting the subset of sequences to be sanitized. We can easily employ random settings along the two orthogonal dimensions to get different algorithms. This way it becomes possible to measure the individual and collective contribution of the two heuristics proposed. To this end, we name these four algorithms as HH (Heuristic-Heuristic), HR (Heuristic-Random), RH (Random-Heuristic), and RR (Random-Random). HR is interpreted as using the heuristic in a given sequence sanitization but selecting a random subset to be sanitized, and other 3 schemes are defined accordingly.

It is worth noting that in random settings, the choice of positions in a sequence, and the choice of sequences to be sanitized, are not completely blind: the random choice is actually performed only among reasonable choices, i.e., positions and sequences that *need* to be sanitized.

We tested three distortion measures as defined next:

- M1 (Data distortion): total number of marking symbols in $\mathcal{D}'$.
- M2 (Frequent Pattern Distortion):

$$\frac{|\mathcal{F}(\mathcal{D}, \sigma)| - |\mathcal{F}(\mathcal{D}', \sigma)|}{|\mathcal{F}(\mathcal{D}, \sigma)|}$$

- M3 (Frequent Pattern Support Distortion):

$$\frac{1}{|\mathcal{F}(\mathcal{D}', \sigma)|} \sum_{S \in \mathcal{F}(\mathcal{D}', \sigma)} \frac{sup_{\mathcal{D}}(S) - sup_{\mathcal{D}'}(S)}{sup_{\mathcal{D}}(S)}$$

Note that the measure M1 focuses on the distortion on the whole data, while measures M2 and M3 focus only on the distortion on the frequent patterns. Moreover, M1 is an absolute measure while M2 and M3 are relative measures ranging between 0 and 1.

In the experiments the sequences to be hidden are $\mathcal{S}_h = \{\langle X_6Y_3, X_7Y_2\rangle, \langle X_4Y_3, X_5Y_3\rangle\}$ for the TRUCKS dataset and $\mathcal{S}_h = \{\langle X_2Y_7, X_3Y_7\rangle, \langle X_5Y_7, X_5Y_6\rangle\}$ for the SYNTHETIC dataset. In the following we report the support of these sensitive patterns: this is an important information since it strongly influences the distortion introduced by the sanitization.

$\mathcal{D}$ = TRUCKS, $\lvert\mathcal{D}\rvert = 273$
$sup_{\mathcal{D}}(\langle X_6Y_3, X_7Y_2\rangle) = 36$
$sup_{\mathcal{D}}(\langle X_4Y_3, X_5Y_3\rangle) = 38$
$sup_{\mathcal{D}}(\langle X_6Y_3, X_7Y_2\rangle \vee \langle X_4Y_3, X_5Y_3\rangle) = 66$
$\mathcal{D}$ = SYNTHETIC, $\lvert\mathcal{D}\rvert = 300$
$sup_{\mathcal{D}}(\langle X_2Y_7, X_3Y_7\rangle) = 99$
$sup_{\mathcal{D}}(\langle X_5Y_7, X_5Y_6\rangle) = 172$
$sup_{\mathcal{D}}(\langle X_2Y_7, X_3Y_7\rangle \vee \langle X_5Y_7, X_5Y_6\rangle) = 200$

In Figure 1(a,. . . ,i) we report all experimental results where random cases are averaged over 10 runs. The threshold on the X-axis is always the disclosure threshold ψ. For the experiments on measures M2 and M3 we always use a minimum frequency threshold equivalent to the disclosure threshold: i.e., $\sigma = \psi$.

Figure 1(a) and (d) report the experimental results for the M1 measures. They show that HH performs consistently best and RR consistently worst for both datasets at all thresholds. Comparing HR and RH reveals that up to a certain threshold, sequence level sanitization heuristics is more important than heuristically selecting the subset to be sanitized. And vice versa after that disclosure threshold. This shows the importance of heuristics at both level too.

The effects of constraints at different levels of thresholds for HH algorithm and for M1 measure, are reported in Figure 1(g),(h) and (i). The plots show that constraints can help in reducing the unnecessary distortions. As it is clear from the figure, increasing level of constraints result in less distortion as well. One notable exception to this is with MaxWindow constraint at thresholds between 35 and 45 for no maxwindow and and maxwindow=10. At first this could seem meaningless, since we expect that the constraints always reduce distortion. Although this is almost always true (as indicated by the results), it is not guaranteed due to imperfectness of the heuristics.

Figure 1(b),(c) report the M2 and M3 distortion measures on the TRUCKS dataset, while Figure 1(e),(f) report the same measures on the SYNTHETIC dataset. The results confirm that HH algorithm achieves the best performance and RR achieves the worst for both measures on the TRUCKS dataset. This agrees with the results obtained for measure M1. However, achieving best performance for M1 do not imply the best performance for M2 and M3 as results show (Figure 1(e),(f)).

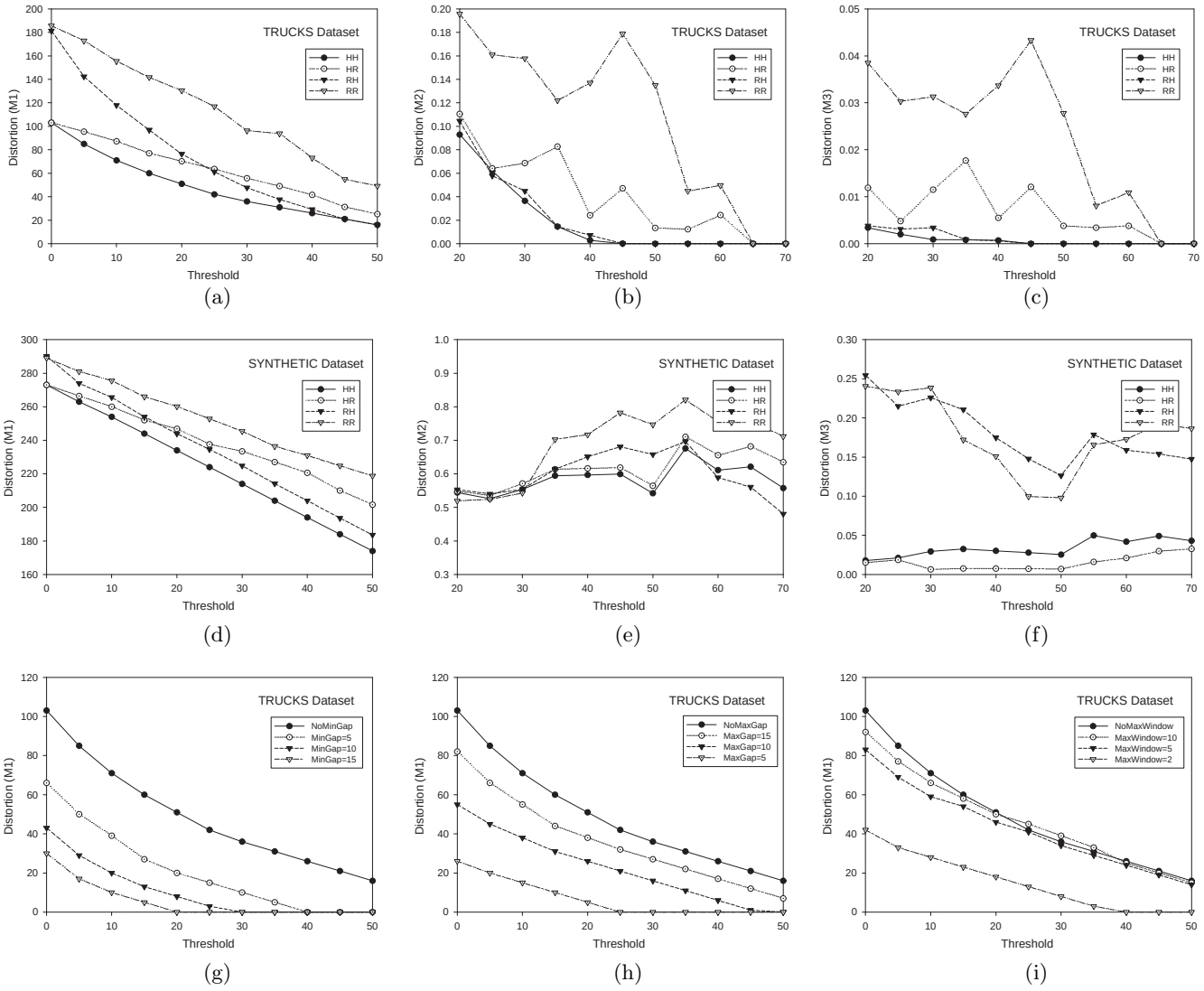

Figure 1. Distortion empirical evaluation.

Recall that the designed algorithm is aimed to reduce M1 not M2 and M3. But, on the average, it is also effective in reducing M2 and M3.

7. Extensions

In this section we describe two straightforward extensions of our method, namely (i) sequential pattern in the classical setting [2] (i.e., sequences of sets of items), and (ii) sequences of events annotated with real time tags. We then discuss a less straightforward roadmap, that we are pursuing in our ongoing work, towards spatio-temporal patterns hiding.

7.1. Itemset Sequences

In principle it is easy to extend both the constrained and unconstrained sequence hiding framework to itemset sequences. In this case, each element in position i of $T \in \mathcal{D}$ is not an item but a (non-empty) set of items, usually called itemsets, and similarly, each element in position j of $S \in \mathcal{S}_h$ is not an item but a (non-empty) set of items.

The main difference lies in how to find the matches of a sequence $S \in \mathcal{S}_h$ in a input sequence $T \in \mathcal{D}$. In this case, it is not an equality test of items but a set inclusion test, i.e. if $S[j] \subseteq T[i]$ then we got a match.

Even though the computation of matching set $\mathcal{M}^T_{\mathcal{S}_h}$ size under this extension is straightforward, the mark-

ing operation in the case of sequences of itemsets is more challenging. This is because there can be many possible alternatives for marking an itemset, in such a way that the matching itemsets of a sensitive sequence are no longer subset. Thus our heuristic needs to be modified for itemsets. One possible solution is first choosing the position in T to sanitize using the same heuristic proposed for simple sequences, and then, choosing a subset of items for marking in this itemset which reduces the matching set most. This is a two level hierarchical heuristic and can be implemented by keeping auxiliary data structures for each itemset in $T \in \mathcal{D}$.

7.2. Events with Real-time Tags

As a step towards spatio-temporal knowledge hiding we could consider sequences of events (either items or itemsets), where each event is labelled with a time tag. In this case we can simply rely on the min gap, max gap and max window constraints discussed in Section 5. In the new scheme these constraints are expressed in real times. The adaptation is straightforward, since our basic method needs only the indices computed over T for all three mentioned constraints. These indices can be easily located using the associated real time tags.

7.3. Spatio-Temporal Patterns

A trajectory is the continuous movement data of a mobile entity over a time course. Since trajectories may contain sensitive patterns, they may need to be sanitized before being published.

The simplest form of trajectory pattern is obtained by discretizing location information, and thus converting them to event sequences, similarly to what we have done in our experiments. These event sequences, with their associated time information can be treated as discussed in Section 7.2.

However, in general, more sophisticated form of spatio-temporal patterns can be mined from trajectory data [18, 7], for instance, without using predefined space or time discretization, but extracting the most interesting discretization as part of the knowledge discovery task. Moreover, a real-world model can be available as background knowledge: for instance, in the case of mobility data, the geographic map and the road network. Such background knowledge can be exploited to rediscover the hidden patterns, if the sanitization has not been performed properly, i.e., considering such background knowledge as a big constraint that the sanitized data must satisfy.

Recapitulating, open issues that must be investigated in order to move towards spatio-temporal knowledge hiding, are:

- How to map the real-world background knowledge to a mathematical model.
- Private pattern language: this language should be expressive enough to define non-trivial spatio-temporal patterns. *Regular expressions* with time constraints can be a candidate.
- Basic operations for distortion: since the sanitization requires removal of pattern occurrences, we need some operations to do that. The simplest solution is finding all pattern occurrences and removing these trajectories as a whole. But there are more elegant operations like swapping locations, swapping partial trajectories, replacing locations, shifting locations in time, generalizations in time and space. Any sanitization operation chosen should respect the constraint given by the real-world background model.

8. Conclusion and Future Work

To the best of our knowledge, this is the first work addressing the problem of knowledge hiding in sequential pattern mining. We have proven that the optimal sequence hiding problem, sanitization is NP-Hard and, thus we have introduced a heuristic algorithm which aims less distortion while providing sanitization. The algorithm is flexible in the sense that it allows a disclosure threshold and three practical constrains (min gap, max gap and max window) to be specified. The carried out experiments demonstrate the effectiveness of the approach. We discussed its straightforward extensions to sequences of itemsets and sequences of events with real time tags. We also explained the road-map that we intend to follow towards spatio-temporal knowledge hiding.

Other issues worth of further investigations are:

- Efficiency: the basic theory developed is considered in terms of effectiveness with straightforward implementations. Efficient implementation is important especially for large datasets.
- Other alternatives heuristics such as: (i) auto-correlation of sequence (high auto-correlation means small number of distinct subsequences), (ii) length of sequence (long sequences potentially provide support to a larger number of subsequences).
- Multiple disclosure thresholds: in case the sensitivity level of patterns differs. Though there is a

very simple solution to this problem in the current algorithm (just take the minimum of all), it may be handled in a relatively novel way.

- Patterns as arbitrary regular expressions (REs): the work presented in this paper is for a subclass of REs. It is a particular interest to search for how arbitrary REs can be used in this framework.

Acknowledgment. This work is supported by the EU project *GeoPKDD (IST-6FP-014915)*. Osman Abul is fully supported by an ERCIM post-doc scholarship. Authors wish to thank Mirco Nanni for the useful discussion on dynamic programming.

References

[1] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the 20th ACM Symposium on Principles of Database Systems (PODS 2001)*, pages 247–255, 2001.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *Eleventh International Conference on Data Engineering (ICDE'95)*, pages 3–14, Taipei, Taiwan, 1995.

[3] 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.

[4] 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.

[5] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi. Anonymity preserving pattern discovery. *VLDB Journal*. Accepted for publication 2006.

[6] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi. Blocking anonymity threats raised by frequent itemset mining. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM 2005)*, pages 561–564, 2005.

[7] H. Cao, N. Mamoulis, and D. W. Cheung. Mining frequent spatio-temporal sequential patterns. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*.

[8] C. Clifton and D. Marks. Security and privacy implications of data mining. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data (SIGMOD'96)*, pages 15–19, Feb. 1996.

[9] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, and E. Bertino. Hiding association rules by using confidence and support. In *Proceedings of the 4th International Workshop on Information Hiding*, pages 369–383, 2001.

[10] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003)*, pages 505–510, 2003.

[11] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002)*, pages 343–364, 2002.

[12] E. Frentzos, K. Gratsias, N. Pelekis, and Y. Theodoridis. Nearest neighbor search on moving object trajectories. In *SSTD'05*, 2005.

[13] A. Friedman, A. Schuster, and R. Wolff. k-anonymous decision tree induction. In *Proc. of PKDD '06*, pages 151–162, Berlin, Germany, 2006. Springer-Verlag.

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

[15] F. Giannotti, A. Mazzoni, S. Puntoni, and C. Renso. Synthetic generation of cellular network positioning data. In *13th ACM International Symposium on Advances in Geografic Information Systems (ACMGIS'05)*, Bremen, Germany, 2005.

[16] M. Kantarcioglu, J. Jin, and C. Clifton. When do data mining results violate privacy? In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004)*, pages 599–604, 2004.

[17] G. Lee, C.-Y. Chang, and A. L. P. Chen. Hiding sensitive patterns in association rules mining. In *28th Annual International Computer Software and Applications Conference (COMPSAC 2004)*, pages 424–429, 2004.

[18] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D. W. Cheung. Mining, indexing, and querying historical spatiotemporal data. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004)*.

[19] S. Menon, S. Sarkar, and S. Mukherjee. Maximizing accuracy of shared databases when concealing sensitive patterns. *Information Systems Research*, 16(3):256–270, 2005.

[20] D. E. O'Leary. Knowledge discovery as a threat to database security. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 507–516. AAAI/MIT Press, 1991.

[21] S. R. M. Oliveira and O. R. Zaïane. Protecting sensitive knowledge by data sanitization. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM 2003)*, pages 211–218, 2003.

[22] E. D. Pontikakis, A. A. Tsitsonis, and V. S. Verykios. An experimental study of distortion-based techniques for association rule hiding. In *Proceedings of the 18th Conference on Database Security (DBSEC 2004)*, pages 325–339, 2004.

[23] S. Rizvi and J. R. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB 2002)*, 2002.

[24] Y. Saygin, V. S. Verykios, and C. Clifton. Using unknowns to prevent discovery of association rules. *ACM SIGMOD Record*, 30(4):45–54, 2001.

[25] Y. Saygin, V. S. Verykios, and A. K. Elmagarmid. Privacy preserving association rule mining. In *Proceedings of the 2002 International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems (RIDE 2002)*, 2002.

[26] X. Sun and P. S. Yu. A border-based approach for hiding sensitive frequent itemsets. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM 2005)*, pages 426–433, 2005.

[27] 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.

[28] V. S. Verykios, A. K. Emagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association rule hiding. *IEEE Transactions on Knowledge and Data Engineering*, 16(4):434–447, 2004.