Hiding Co-Occurring Frequent Itemsets

Osman Abul*
TOBB University of Economics and Technology
Ankara, Turkey
osmanabul@etu.edu.tr

ABSTRACT

Knowledge hiding, hiding rules/patterns that are inferable from published data and attributed sensitive, is extensively studied in the literature in the context of frequent itemsets and association rules mining from transactional data. The research in this thread is focused mainly on developing sophisticated methods that achieve less distortion in data quality. With this work, we extend frequent itemset hiding to co-occurring frequent itemset hiding problem. Co-occurring frequent itemsets are those itemsets that co-exist in the output of frequent itemset mining. What is different from the classical frequent hiding is the new sensitivity definition: an itemset set is sensitive if its itemsets appear altogether within the frequent itemset mining results. In other words, co-occurrence is defined with reference to the mining results but not to the raw input dataset, and thus it is a kind of meta-knowledge. Our notion of co-occurrence is also very different from association rules as itemsets in an association rule need to be frequently present in the same set of transactions, but the co-occurrence need not necessarily require the joint occurrence in the same set of transactions.

In this paper, we briefly review the frequent itemset/association hiding problems and define the co-occurrence hiding along with the real world motivations. We explore its fundamental properties and show that frequent itemset hiding is a special case of the co-occurring frequent itemsets hiding. As a solution, we propose a two-stage sanitization framework, essentially a reduction, where an instance of the frequent itemset hiding is constructed in the first stage and the instance is solved in the second stage. Since the task is shown to be NP-Hard and the reduction is one-to-many, we propose heuristics only for the first stage as the second stage is a well-established field. Finally, an experimental evaluation is carried out on a couple of datasets, and the results are presented.

*O. Abul is fully supported by TUBITAK under the grant number 108E016

1. INTRODUCTION

Since the introduction of privacy preserving data mining in early nineties [18], it has received considerable attraction from the data mining research community [7, 10, 12, 19, 24]. In its nutshell, privacy preserving data mining is the study of data mining side-effects on privacy, secrecy and sensitivity. The related literature has seen the proliferation of various task formulations, many completely different solution approaches with different objectives, techniques and application domains. The tasks range from privacy-aware dataset regeneration [26] to privacy-aware data mining results publication [9] to privacy-aware database sharing [8]. Privacy-aware database sharing, in the context of data mining, is the study of how to securely publish databases for the analysis of third parties. The main concern is to not to disclose any sensitive knowledge that can be extracted from the dataset by means of complex analysis like data mining. *Knowledge hiding* aims at hiding some knowledge considered sensitive from shared databases.

Knowledge hiding refers to the activity of concealing some sensitive knowledge that hold in a database that is going to be published. Following the existence identification of some sensitive knowledge in the database to be released, the data publisher faces with the secrecy/utility tradeoff and has three basic options: (i) do not share the database, (ii) share it as is, or (iii) share it after suppressing sensitive knowledge. Clearly, the first option is completely safe but no external parties can utilize the dataset to extract nonsensitive knowledge of their interest, while the second one is unsafe as the data receiver may easily surface the sensitive knowledge by employing data mining but it allows full utilization of the dataset by third parties. The third option is to allow data receivers to build valid mining models from the released version of the database, while ensuring recovery of the sensitive knowledge impossible. Clearly, knowledge hiding problem is only relevant with the last option of the database publishing policy.

Knowledge hiding is usually obtained by *sanitizing* the database in such a way that the sensitive knowledge can no longer be inferred, while the original database is changed as less as possible. Many different approaches for knowledge hiding have emerged over the years, mainly in the context of association rules and frequent itemsets mining. The form of sensitive knowledge in frequent itemset (associations rules, resp.) mining is a list of user specified frequent itemsets (association rules, resp.). The respective hiding processes transform the database such that the sensitive frequent itemsets and association rules are impossible to recover. The implicit assumption in the scenario is that the form of sensitive knowledge exactly matches the form of knowledge that appear in the mining result. However, this is not true in all real-world problems as sensitive

knowledge can be defined in meta-knowledge level for instance. The observation is the main motivation of this study. As a first attempt, we define *co-occurring frequent itemsets* as the form of sensitive knowledge and frequent itemsets as the form of mining result.

Co-occurring frequent itemsets is a group of itemsets that appear altogether in the mining results. In other words, the group is sensitive if every itemset in it appear in the mining results, and non-sensitive if any of the itemset is missed. This kind of sensitivity definition can be found in many contexts. As an example, consider the shopping domain where there are two deals (itemsets): D1=$\{popcorn, peanut\}$ and D2=$\{bread, butter\}$. Here the sensitive knowledge is that both deals have received customer kindness. So, to hide this meta-knowledge suppressing either of D1 or D2 suffices, but hiding both is not essential. As another example, consider the school domain where schools publish their annual performances. For most schools, it is acceptable that its students are mostly failing in social science or science courses, but failure in both tracks is unacceptable. To keep the school's reputation, hiding (any) one of the failure suffices. As it is clear, the sensitivity definition in these examples are different than that of found in frequent itemsets/associations. The notion is that sensitivity is not specified in isolation but with (mutual) reference to some other knowledge, i.e. a piece of knowledge is nonsensitive in its own but becomes sensitive if co-exists with some others. Co-occurring frequent itemset hiding is the subject of the paper.

The paper is structured as follows. The frequent itemset mining, association rules mining and respective hiding problems are reviewed in Section 2. The section 3 introduces the co-occurring frequent itemset hiding problem, and Section 4 gives an algorithmic framework to solve it. The framework is a two-stage approach where the first stage involves generating an instance of the frequent itemset hiding problem, through a reduction, and the second stage involves solving the instance using any algorithm developed for frequent itemset hiding. In Section 4, we propose four algorithms to be used in the first stage. The proposal is experimentally evaluated on two datasets in Section 5. We briefly cover the related work in Section 6. Finally, Section 7 concludes and gives our future work.

2. FREQUENT ITEMSET HIDING

Frequent itemsets/associations mining problem is first introduced in early 90s by Agrawal et al. [4,5] and later studied extensively by data mining community and has found diverse application areas. In this section, we briefly review the frequent itemsets/associations mining and the respective sensitive knowledge hiding problems.

DEFINITION 1 (FREQUENT ITEMSET MINING).
Let $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ be a set of distinct symbols called items, and itemset X be any non-empty subset of $\mathcal{I}$. X is called k-itemset if $|X| = k$. The transaction database, $\mathcal{D}$, is an unordered collection of transactions T; each of which is an itemset over $\mathcal{I}$, i.e. $T \in 2^{\mathcal{I}} - \emptyset$. The support set of an itemset X in $\mathcal{D}$, denoted $S_{\mathcal{D}}(X)$, is the set of transactions in $\mathcal{D}$ that accept X as a subset, i.e. $S_{\mathcal{D}}(X) = \{T : X \subseteq T \text{ and } T \in \mathcal{D}\}$. The cardinality of $S_{\mathcal{D}}(X)$ is called the support, denoted $sup_{\mathcal{D}}(X)$, of X in $\mathcal{D}$, i.e. $sup_{\mathcal{D}}(X) = |S_{\mathcal{D}}(X)|$. For a user-defined minimum support threshold σ (a non-negative integer), frequent itemset mining problem is to find all itemsets X having support of at least σ in $\mathcal{D}$. The resulting itemsets are called frequent (a.k.a. large) itemsets and formally denoted as $F_{(\mathcal{D},\sigma)} = \{X : X \subseteq \mathcal{I}, X \neq \emptyset, sup_{\mathcal{D}}(X) \geq \sigma\}$. A sample database containing nine transactions and all frequent itemsets mined (at $\sigma = 3$) are shown in Table 1.

Table 1: An example database (taken from [17])

	$\mathcal{D}$	$F_{(\mathcal{D},\sigma=3)}$
Tid	Items	Frequent itemset:support
1	a b c d e	a b d :3, a c d :4, b c d :3, c d e :3
2	a c d	a b:4, a c:5, a d:6, b c:4, b d:5, c d:6, c e:3, d e:3
3	a b d f g	a :7, b :6, c :7, d :8, e :3
4	b c d e	
5	a b d	
6	b c d f h	
7	a b c g	
8	a c d e	
9	a c d h	

The notion with itemsets is to study the co-occurrence of items (not to be mixed with the co-occurring frequent itemsets) in transaction databases, e.g. market-basket databases. Given a frequent itemset, itemset occurring in sizeable portion of the transactions, one can reason that there is a strong association between the items in the itemset. Therefore, such an itemset is a pattern (a piece of knowledge) that can be used for numerous purposes. For instance, in the market-basket case a frequent itemset is a pattern showing the consumer behaviors that can be used for marketing, e.g. offering new deals. Clearly, such a pattern has a commercial value and may be attributed sensitive.

Since the introduction [4, 5] of the frequent itemset mining problem, the research community has mainly concentrated on developing fast algorithms for support counting of exponentially growing itemsets space. To this end, one of the most useful property is the anti-monotonic **Apriori** property: *if an itemset is not frequent none of its superset can be frequent*, or equivalently *if an itemset is frequent then all of its subsets are frequent too*. As a result, support counting of large portions of itemsets can be avoided to gain efficiency. After the Apriori algorithm [4] which is exploiting Apriori property, several other algorithms have been developed using different approaches, techniques and advanced data structures [3, 15, 21].

DEFINITION 2 (ASSOCIATION RULES MINING).
*Let $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ be a set of items, and itemsets X and Y be any non-empty subsets of $\mathcal{I}$. An association rule is an implication of the form $X \Rightarrow Y$ where $X \bigcap Y = \emptyset$. X is called the left-hand side (*lhs*) or antecedent and Y is called right-hand side (*rhs*) or consequent. The rule $X \Rightarrow Y$ is said to hold in transaction database $\mathcal{D}$ with support s and confidence c if (i) $sup_{\mathcal{D}}(X \bigcup Y) \geq s$ and (ii) $\frac{sup_{\mathcal{D}}(X \bigcup Y)}{sup_{\mathcal{D}}(X)} \geq c$. Given the minimum support threshold σ and the minimum confidence threshold $minconf$, association rules mining problem is to find all significant association rules holding in the database with at least σ support and $minconf$ confidence.*

Association rule mining problem contains frequent itemset mining problem as its subproblem. In fact, association rules are typically generated as a postprocessing to frequent itemsets mining results.

Even though the similarities exist between the two problems, the knowledge forms are different. For this reason, these two problems offer two different respective hiding problems as defined next.

PROBLEM 1 (FREQUENT ITEMSET HIDING).
Let $\mathcal{X}^s = \{X_1, X_2, \ldots, X_n\}$ with $X_i \in 2^{\mathcal{I}} - \emptyset, \forall i \in \{1, \ldots, n\}$, be the set of sensitive itemsets that must be hidden from $\mathcal{D}$. Given a disclosure threshold ψ, the Frequent Hiding Problem requires to transform $\mathcal{D}$ in a database $\mathcal{D}'$ such that:

1. $\forall X_i \in \mathcal{X}^s : sup_{\mathcal{D}'}(X_i) < \psi$;
2. $\sum_{X \in (2^{\mathcal{I}} - \emptyset)} |sup_{\mathcal{D}}(X) - sup_{\mathcal{D}'}(X)|$ *is minimized.*

The first requirement asks for lowering the support of sensitive itemsets below ψ level, so the receiver of $\mathcal{D}'$ can not mine any of the sensitive itemsets at $\sigma = \psi$ threshold. The second requirement is the minimization objective which claims for solutions destroying supports of itemsets as less as possible. This is another way of saying the similarity between $\mathcal{D}$ and $\mathcal{D}'$ maximized, which is very important to obtain valid mining models through $\mathcal{D}'$.

Suppose $\mathcal{X}^s = \{\{a, b, d\}, \{c, d\}\}$ is the sensitive knowledge that must be hidden from the database in Table 1 at $\psi = 3$. A solution (among many others) can be obtained by suppressing items d, c, d and c from transactions 1, 2, 4, and 6, respectively to give $\mathcal{D}'$. Then, $F_{(\mathcal{D}', \sigma=3)} = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, e\}\}$. Note that neither of sensitive frequent itemsets can be mined, thus hidden, (at $\sigma = 3$) from $\mathcal{D}'$, while most of the nonsensitive frequent itemsets can still be mined.

PROBLEM 2 (ASSOCIATION RULES HIDING).
Let $\mathcal{A}^s = \{X_1 \Rightarrow Y_1, X_2 \Rightarrow Y_2, \ldots, X_n \Rightarrow Y_n\}$ be the set of sensitive association rules that must be hidden from $\mathcal{D}$. Given the pair of disclosure threshold parameters (ψ_1, ψ_2), Association Rules Hiding Problem requires to transform $\mathcal{D}$ in a database $\mathcal{D}'$ such that:

1. *the support of every rule in $\mathcal{D}$ is less than ψ_1, i.e. $\forall X_i \Rightarrow Y_i \in \mathcal{A}^s : sup_{\mathcal{D}'}(X_i \bigcup Y_i) < \psi_1$; or the confidence of every rule is less than ψ_2, i.e. $\forall X_i \Rightarrow Y_i \in \mathcal{A}^s : \frac{sup_{\mathcal{D}'}(X_i \bigcup Y_i)}{sup_{\mathcal{D}'}(X_i)} < \psi_2$;*
2. $\sum_{X \in (2^{\mathcal{I}} - \emptyset)} |sup_{\mathcal{D}}(X) - sup_{\mathcal{D}'}(X)|$ *is minimized.*

One can easily recognize that associations rules hiding problem can be easily reduced to frequent itemset hiding problem. This is simply because, solely decreasing the support solves both of the problems. However, association rules hiding has more flexibility, i.e. decreasing either support or confidence (note the disjunction in the first requirement of Problem 2) as opposed to support only decrease in frequent itemset hiding. A suite of algorithms for frequent itemsets/associations hiding is reviewed in Section 6.

3. CO-OCCURRENCE HIDING

In this section, we introduce the co-occurrence hiding problem in the context of frequent itemsets. Co-occurring frequent itemsets is a group of itemsets that appear altogether in the mining results. Given the dataset $\mathcal{D}$ and support threshold σ, the $F_{(\mathcal{D}, \sigma)}$ constitutes the extracted knowledge set. Over this knowledge set, some meta-knowledge forms, like the co-occurring frequent itemsets, can easily be defined. Clearly, some of these meta-knowledge can be attributed sensitive and thus need to be hidden before releasing the dataset.

DEFINITION 3 (CO-OCCURRING FREQUENT ITEMSETS).
Let C be any non-empty subset of $F_{(\mathcal{D}, \sigma)}$, i.e. $C \in 2^{F_{(\mathcal{D}, \sigma)}} - \emptyset$, then the set of itemsets in C are called co-occurring frequent itemsets. In other words, any non-empty subset of frequent itemsets is a co-occurring frequent itemsets. In the special case $C = \{X_1, X_2\}$ (i.e. $|C| = 2$), then X_1 and X_2 are called pairs of co-occurring frequent itemsets. The definition can be easily extended to triples, quadruples etc. Note that the definition still applies in the degenerate case of $C = \{X_1\}$, i.e. $|C| = 1$.

PROBLEM 3 (CO-OCCURRING FREQUENT ITEMSET HIDING).
Let $\mathcal{C}^s = \{C_1, C_2, \ldots, C_n\}$ be the set of sensitive co-occurring frequent itemset sets that must be hidden from $F_{(\mathcal{D}, \sigma)}$. Co-Occurrence Hiding Problem requires to transform $\mathcal{D}$ in a database $\mathcal{D}'$ such that:

1. $\forall C_i \in \mathcal{C}^s : C_i \not\subseteq F_{(\mathcal{D}', \sigma)}$;
2. $\sum_{X \in (2^{\mathcal{I}} - \emptyset)} |sup_{\mathcal{D}}(X) - sup_{\mathcal{D}'}(X)|$ *is minimized.*

Similar to frequent itemset hiding problem the first requirement in Problem 3 is a hard constraint but the second requirement is the optimization objective seeking best solution among all satisfying the first requirement. To satisfy the first requirement it is enough to exclude any one of the itemsets in every $\mathcal{C}^s$ from the frequent itemset mining results. The Proposition 1 formalizes this notion.

PROPOSITION 1.
Hiding any frequent itemset in each itemset set from $\mathcal{C}^s = \{C_1, C_2, \ldots, C_n\}$ satisfies the first requirement in Problem 3.
Proof: *Suppose $X_i \in C_j$ (for any i, j) is hidden from $F_{(\mathcal{D}, \sigma)}$, then $X_i \notin F_{(\mathcal{D}', \sigma)}$ by definition. $X_i \notin F_{(\mathcal{D}', \sigma)}$ and $X_i \in C_j$ together imply that $C_j \not\subseteq F_{(\mathcal{D}', \sigma)}$.* □

Suppose $\mathcal{C}^s = \{C_1\}$, where $C_1 = \{\{a, b, d\}, \{c, d\}\}$, is the sensitive knowledge that must be hidden from the database in Table 1 at $\sigma = 3$. A solution (among many others) can be obtained by suppressing item d from transaction 1 to give $\mathcal{D}'$. Then, $F_{(\mathcal{D}', \sigma=3)} = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}, \{c, e\}, \{a, c, d\}\}$. Note that one, $\{c, d\}$, of sensitive frequent itemsets can still be mined (at $\sigma = 3$) but the other, $\{a, b, d\}$, from $\mathcal{D}'$. So, the receiver of the $\mathcal{D}'$ can not discover the knowledge that itemsets $\{a, b, d\}$ and $\{c, d\}$ co-occur in $F_{(\mathcal{D}', \sigma=3)}$. Also note that, the quality of the solution is higher compared to the cost of hiding both frequent itemsets as required by Problem 1.

We would like to emphasize that co-occurring itemsets and association rules are fundamentally different concepts. To illustrate, consider a co-occurring itemset set $C_1 = \{X_1, X_2\}$ and an association rule $A_1 = \{X_1 \Rightarrow X_2\}$. A_1 requires both X_1 and X_2 appear

together in the same set of transactions and the union, $X_1 \bigcup X_2$, to be frequent; but C_1 has no such a requirement, i.e. X_1 and X_2 may appear independently in different transactions as long as both X_1 and X_2 are frequent.

In the next section, we propose a framework to solve Problem 3 which exploits Proposition 1.

4. A SANITIZATION FRAMEWORK

In this section, we propose a framework as a solution to Problem 3. The key notion is exploiting the utility of Proposition 1 in Proposition 2 to obtain an instance of Problem 1 from the instance of Problem 3.

PROPOSITION 2.
Given an instance ($\mathcal{C}^s = \{C_1, C_2, \ldots, C_n\}$ and σ) of Problem 3, an instance ($\mathcal{X}^s = \{X_1, X_2, \ldots, X_n\}$ and ψ) of Problem 1 can be obtained as follows, and hence the solution to the latter is also a solution to the former.
Proof: *The (one-to-many) reduction is as follows.*

- $X_i \leftarrow Y \; s.t. \; Y \in C_i, \forall i \in \{1, 2, \ldots, n\}$;
- $\psi \leftarrow \sigma$.

The correctness of the reduction follows from Proposition 1. □

The reduction is one-to-many mapping because any itemset in C_i can be assigned to X_i and all of them are correct albeit with different distortion costs. It is also clear that the reduction in Proposition 2 is the minimum requirement as hiding more than one itemset (from each C_i) also works but monotonically causes more distortion.

COROLLARY 1.
Problem 1 is a special case of Problem 3.
Proof: *From Proposition 2, it is clear that these two problems are equivalent in case in the instance ($\mathcal{C}^s = \{C_1, C_2, \ldots, C_n\}$ and σ) of Problem 3, every C_i is restricted to be singletons, i.e. $|C_i| = 1, \forall i \in \{1, 2, \ldots, n\}$.* □

COROLLARY 2.
Problem 3 is NP-Hard.
Proof: *NP-Hardness follow from Corollary 1 as generalization of any NP-Hard problem is NP-Hard too [14] and Problem 1 is proven to be NP-Hard in [8].* □

The Corollary 1 allows us to use any algorithm developed for Problem 3 to solve Problem 1, but the opposite is not immediate. However, in the Algorithm 1 we develop a two-stage framework which enables algorithms for Problem 1 to be employed to solve Problem 3. The framework exploits the proof of Proposition 2, the minimization of which is equivalent to the hitting set problem known to be NP-Hard [14]. The (generic) function $SelectSensitiveSet$ (in Line 1) obtains a hitting set from $\mathcal{C}^s$ by (potentially) employing $\mathcal{D}$ and σ in its heuristic. The hitting set, $\mathcal{X}^s$, is fed to (generic) $FISHider$ (in Line 3) which is any algorithm solving standard frequent itemset hiding problem. As a result, both stages in the framework are NP-Hard problems and thus each require heuristics or approximations.

Algorithm 1 Co-occurring Frequent Itemset Hiding Framework

Input: $\mathcal{D}, \mathcal{C}^s, \sigma$
Output: $\mathcal{D}'$
1: $\mathcal{X}^s \leftarrow SelectSensitiveSet(\mathcal{D}, \mathcal{C}^s, \sigma)$
2: $\psi \leftarrow \sigma$
3: $\mathcal{D}' \leftarrow FISHider(\mathcal{D}, \mathcal{X}^s, \psi)$
4: return $\mathcal{D}'$

There are several algorithms proposed to solve the standard frequent itemset hiding problem (see Section 6). Any of them can be substituted for the generic $FISHider$ function. This is the main reason that we focus on the first stage rather than the second one in this work. As the correspondence between the problem in the first stage and the hitting set problem is already established, we look for custom heuristics solving the problem in the first stage.

There are many algorithms proposed for the hitting set problem and for its variants (weighted hitting set etc.), so any of them can be customized to substitute generic $SelectSensitiveSet$ function. In what follows, we present four greedy custom algorithms to replace $SelectSensitiveSet$. The reason that we focus on custom heuristics (rather than using standard hitting set algorithms) is to enable the use of some additional information like support and thresholds in the design to maintain data quality as much as possible.

4.1 Heuristics for SelectSensitiveSet

The first algorithm (Algorithm 2), called MOMSH, is based on maximum occurrence and minimum support heuristics. In each iteration of the first loop, an itemset occurring in maximum number of $\mathcal{C}^s$ is selected to be included in the hitting set. However, if there is no itemset shared among itemset sets in $\mathcal{C}^s$, then the loop breaks. At this time, it is guaranteed that there is no itemset shared by modified $\mathcal{C}^s$. So, the second loop independently considers each set in $\mathcal{C}^s$ and selects the itemset having the least support in the database. Clearly, the first loop exploits the idea that to minimize the hitting set cardinality, itemsets shared by most itemset sets should be included in the hitting set; this is the *maximum occurrence* heuristic. The second loop exploits the idea that selecting the itemset having the least support has more potential to attain less distortion; this is the *minimum support* heuristic.

The second algorithm (Algorithm 3), called MSH, exploits only the minimum support heuristic. It considers every co-occurring itemset set in $\mathcal{C}^s$ and includes the minimum support itemset in the hitting set. Then, it removes other itemset sets from $\mathcal{C}^s$ intersecting with the selected minimum support itemset, and the iteration continues until $\mathcal{C}^s$ becomes empty.

The third algorithm (Algorithm 4), called GMSH, exploits global minimum support heuristic. Global minimum support heuristic first computes the ground set, $\mathcal{S}$, of itemsets from $\mathcal{C}^s$ (Line 2). Then it picks the minimum support itemset from $\mathcal{S}$ and removes every itemset set from $\mathcal{C}^s$ containing this itemset. The itemset is included in the hitting set and the iteration continues until $\mathcal{C}^s$ becomes empty. The basic idea behind the heuristic is to give precedence (to be included in the hitting set) to those itemsets with the least support to keep the distortion small.

The fourth algorithm (Algorithm 5), called UA, is a very simple

Algorithm 2 MOMSH: Maximum occurrence and minimum support heuristics

Input: $\mathcal{D}, \mathcal{C}^s, \sigma$
Output: $\mathcal{X}^s$ // hitting set
$\mathcal{X}^s \leftarrow \emptyset$
$\mathcal{S} \leftarrow \bigcup_{C_i \in \mathcal{C}^s} C_i$
while $\mathcal{C}^s \neq \emptyset$ **do**
  $X \leftarrow \arg max_{X \in S} |\{X \in C | C \in \mathcal{C}^s\}|$
  $maxOcc \leftarrow max_{X \in S} |\{X \in C | C \in \mathcal{C}^s\}|$
  if $maxOcc >= 2$ **then**
    $\mathcal{C}^s \leftarrow \mathcal{C}^s \setminus \{C \in \mathcal{C}^s | X \in C\}$
    $\mathcal{X}^s \leftarrow \mathcal{X}^s \bigcup X$
  else
    break
for $C \in \mathcal{C}^s$ **do**
  $X \leftarrow \arg min_{X \in C} |sup_{\mathcal{D}}(X)|$
  $\mathcal{X}^s \leftarrow \mathcal{X}^s \bigcup X$
return $\mathcal{X}^s$

Algorithm 3 MSH: Minimum support heuristic

Input: $\mathcal{D}, \mathcal{C}^s, \sigma$
Output: $\mathcal{X}^s$ // hitting set
$\mathcal{X}^s \leftarrow \emptyset$
while $\mathcal{C}^s \neq \emptyset$ **do**
  $C \leftarrow first\ C \in \mathcal{C}^s$
  $X \leftarrow \arg min_{X \in C} |sup_{\mathcal{D}}(X)|$
  $\mathcal{C}^s \leftarrow \mathcal{C}^s \setminus \{C \in \mathcal{C}^s | X \in C\}$
  $\mathcal{X}^s \leftarrow \mathcal{X}^s \bigcup X$
return $\mathcal{X}^s$

Algorithm 4 GMSH: Global minimum support heuristic

Input: $\mathcal{D}, \mathcal{C}^s, \sigma$
Output: $\mathcal{X}^s$ // hitting set
$\mathcal{X}^s \leftarrow \emptyset$
$\mathcal{S} \leftarrow \bigcup_{C_i \in \mathcal{C}^s} C_i$
while $\mathcal{C}^s \neq \emptyset$ **do**
  $X \leftarrow \arg min_{X \in S} sup_{\mathcal{D}}(X)$
  $\mathcal{C}^s \leftarrow \mathcal{C}^s \setminus \{C \in \mathcal{C}^s | X \in C\}$
  $\mathcal{X}^s \leftarrow \mathcal{X}^s \bigcup X$
  $S \leftarrow S \setminus X$
return $\mathcal{X}^s$

Algorithm 5 UA: Unite all

Input: $\mathcal{D}, \mathcal{C}^s, \sigma$
Output: $\mathcal{X}^s$ // hitting set
$\mathcal{X}^s \leftarrow \bigcup_{C_i \in \mathcal{C}^s} C_i$
return $\mathcal{X}^s$

union find algorithm, aimed at serving as a reference for the other heuristics.

5. EXPERIMENTAL EVALUATION

We experiment with two datasets; the running example given in Table 1, hereafter referred *Toy*, and Connect dataset from **FIMI** repository [1], hereafter referred *Connect*. The sensitive knowledge (somewhat arbitrarily selected among large support itemsets) to be hidden is fixed for each dataset and the disclosure threshold is varied in experiments. Dataset features and the selected sensitive knowledge are presented in Table 2.

Recalling that our framework consists of two generic sub procedures, $SelectSensitiveSet$ and $FISHider$, we fix $FISHider$ to cyclic hiding algorithm [8] and allow $SelectSensitiveSet$ to be one of four algorithms presented in Section 4. For each sensitive itemset, cyclic hiding algorithm goes through the dataset and finds a transaction supporting the itemset. Then, from the transaction it deletes (suppresses) an item which is the next item in the sensitive itemset. If the support of the sensitive itemset reduces below the disclosure threshold, the process restarts with the next sensitive itemset.

We tested with two distortion metrics, M1 and M2, as defined next. M1 measures the raw data distortion while M2 measures the distortion in knowledge space. Additionally, we also measure the size of the hitting set which is the output of $SelectSensitiveSet$ function in Algorithm 1.

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

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

Toy dataset results are presented in Figure 1 and Table 3. The results in Figure 1 (b) are obtained with frequent itemset mining done on the output of respective sanitization, i.e. $\sigma = \psi$. The results indicate that the proposed three heuristics perform better than *UniteAll*, indicating that they are promising. Note that the distortion gap is considerable even on this toy dataset. On the other hand, the utility of three heuristics are not immediate as they seem to perform very close. In fact, the relative merits are explored in bigger *Connect* dataset. Table 3 gives the hitting sets of the four algorithms. Since MSH and GMSH produce same hitting sets, they always give the same distortion in Figure 1.

Connect dataset results are presented in Figure 2 and Table 4. Since the connect dataset is very dense, frequent itemset mining with practical support thresholds (e.g. 5%) is not feasible as the output size grows to several gigabytes. Due to this, we raise minimum support thresholds to larger values (i.e. 60% and up). Even with these larger minimum supports, the output size grows to several megabytes. The results in Figure 2 (b) are obtained with frequent itemset mining done on the output of complete sanitization, i.e. support of the hitting set reduced to zero. Note that, this has the worst effect on the utility of data as more non-sensitive frequent itemsets are not frequent any more. The results indicate that the proposed three heuristics perform better than *UniteAll*, again indicating that they are promising. The results also indicate that

[1] http://fimi.cs.helsinki.fi/

Table 2: Dataset features and sensitive knowledge

$\mathcal{D}$	$\lvert\mathcal{D}\rvert$	$\mathcal{C}^s$:support
Toy	9	{ {a b d :3, b c :4}, {e :3 , b c :4}, {c d :6 , c e :3} }
Connect	67557	{ { 31 37 75 127 :60802, 13 16 37 91 :61017 } {121 106 19 55 75 127 109 91 :62137, 52 34 106 88 :61423}, {52 34 106 88 :61423, 34 72 88 19 55 75 109 :61941}, {13 16 37 91 :61017, 34 72 88 19 37 75 :62542, 52 34 106 88 :61423}, {121 106 19 55 75 127 109 91 :62137, 13 16 37 91 :61017}, {13 16 37 91 :61017, 16 85 72 106 88 19 127 109 :60812} }

MOMSH performs significantly better than MSH and GMSH. Note that MOMSH finds the minimum cardinality hitting set and thus attains lowest distortion. However, Figure 2 (b) shows that overall distortion is quite high (as high as 98%), suggesting that the utility of the released dataset in the complete sanitization is questionable even with the good sanitization solutions. In our experiments (results not reported), however we have seen that acceptable distortions are committed with relatively higher values of disclosure threshold. Indeed, it is not magic but the well-known utility/sensitivity tradeoff.

6. RELATED WORK

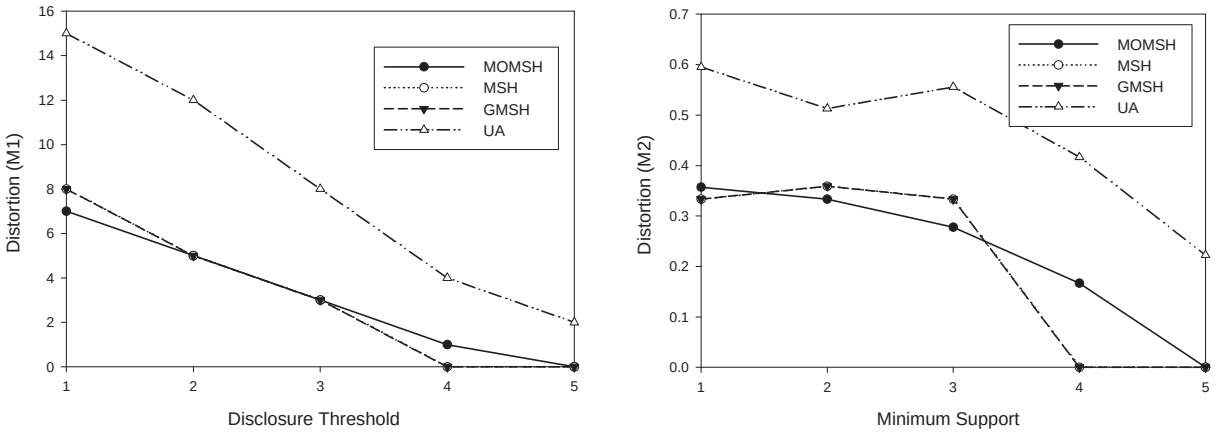
The work by Atallah et al. [8] proved that frequent itemset hiding problem (their problem is a simpler version of Problem 1) is NP-Hard. Their operation to solve the problem is to remove (suppress) items from transactions. They also proposed heuristics to solve the problem by selectively reducing the support of sensitive itemsets. The algorithm iterates over the set of sensitive frequent itemsets until none of them is frequent, and picks an itemset to hide at each step. Let's suppose that a k-itemset $X \in \mathcal{X}^s$ is selected to be hidden. A level-wise traversal, starting from X and ending at a singleton included in X, is carried out on the itemset lattice as described next. Over all $k-1$ subsets of X, the respective supports are reviewed and subset with the maximum support is chosen as the new point for the traversal. This process is iterated until $k=1$, at which time a singleton itemset is found. An occurrence of this singleton itemset is suppressed in one of the supporting transactions of X. The criteria for this decision is minimal effect on the number of 2-itemsets of the transaction, which is somewhat equal to distorting remaining large-itemsets that this transaction is providing support to. Since the support of X is reduced one at a time fashion, the whole operation is repeated until X's support reaches below the disclosure threshold. The heuristic hides all sensitive itemsets, i.e. no hiding failure, and the heuristic serves for minimally affecting the support of non-sensitive itemsets.

A border-based approach is presented in [23]. The idea is to preserve the shape of positive border during sanitization as much as possible. The algorithm first computes the minimal set of sensitive itemsets from the user-specified sensitive itemsets and hides each of them one-by-one fashion after ordering based on length (descending) and support (ascending). Next, the elements of positive border intersecting with the every single item in the sensitive itemset at hand is determined and a weight factor is computed for each candidate (a pair of a single item and a transaction supporting it). The minimum weight candidate is selected for suppression and the candidate selection is repeated until the sensitive itemset gets support less than the disclosure threshold. The weight measures the degree to which the candidate will distort the nonsensitive frequent itemsets. Another border-based approach is presented in [17].

A linear time (w.r.t. the $|\mathcal{D}|$) sanitization algorithm employing sliding window approach is presented in [20]. The algorithm is intended to hide the knowledge of association rules but it is actually hiding frequent itemsets. For each batch of window size K, the algorithm consists of five steps: (1) identifying sensitive and nonsensitive transactions in the batch, (2) selection of the victim item, (3) building the sensitive transaction list for every sensitive itemset and computing the number of sensitive transactions to be sanitized, (4) sorting the sensitive transactions by size, (5) sanitizing the transaction by removing the victim item. Two main characteristics of the algorithm are being scalable to large databases and allowing different sensitivity threshold, i.e. rather than a common threshold, for each sensitive itemset.

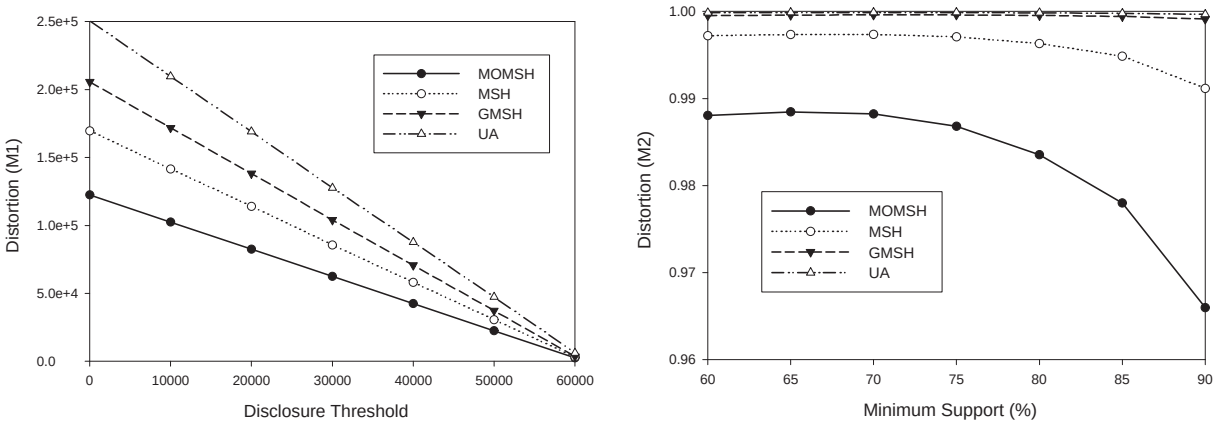
Lee et al. [16] proposes a sanitization matrix based approach. The sanitization matrix has dimensions $|\mathcal{I}| \times |\mathcal{I}|$ where entries are restricted to be $\{-1, 0, 1\}$. The database $\mathcal{D}$ is transformed to the matrix (with the size $|\mathcal{D}| \times |\mathcal{I}|$) representation in which the entries are assigned to be either 0 (non-existence of the item in the transaction) or 1 (otherwise). The matrix representation of $\mathcal{D}'$ is then obtained by multiplying the data matrix of $\mathcal{D}$ with the sanitization matrix. To give 0 or 1 entries, the multiplication results are truncated to zero or one. Clearly, if the sanitization matrix is the identity matrix then $\mathcal{D}'$ equals to $\mathcal{D}$. The challenge is to selectively turn some non-diagonal elements into -1 so that support of sensitive itemsets reduce while the support of others are affected as less as possible. Authors present three greedy algorithms to do so. The main heuristic is selecting the entry (j, k), to be replaced with -1, such that itemset $\{i_j, i_k\}$ exists in some sensitive itemset but does not exist in nonsensitive itemsets. The algorithms do not guarantee hiding failure, i.e. some sensitive itemsets can still be mined.

A support/confidence framework is introduced in [13] where authors present three strategies: (i) increasing the support of rule antecedent, (ii) decreasing the support of rule consequent, (iii) decreasing the rule support. Clearly, first two strategies are to reduce the rule confidence while the last one is to reduce rule support. One of the important limitation of the strategies is the assumption of the disjointness, i.e. sensitive rules do not share items, of the sensitive rules. An extension of [13] is presented in [25]. Another support/confidence framework is presented in [22]. The authors introduce unknowns (symbol ?)in the data matrix representation where the symbol ? has the interpretation of being either 0 or 1. The notion brings a minimum and maximum values for both of the support and confidence of rules. The objective is then to reduce the



(a) (b)

Figure 1: *Toy* dataset distortion results: (a) M1 (b) M2



(a) (b)

Figure 2: *Connect* dataset distortion results: (a) M1 (b) M2

Table 3: Hitting sets for *Toy* dataset

Algorithm	hitting set	hitting set cardinality
MOMSH	{b c, c e}	2
MSH	{a b d, c e, e}	3
GMSH3	{a b d, c e, e}	3
UA	{a b d, b c, c d, c e, e}	5

Table 4: Hitting sets for *Connect* dataset

Algorithm	hitting set	hitting set card.
MOMSH	{106 34 52 88, 13 16 37 91}	2
MSH	{106 34 52 88, 127 31 37 75, 13 16 37 91}	3
GMSH	{106 109 127 16 19 72 85 88, 106 34 52 88, 127 31 37 75, 13 16 37 91}	4
UA	{106 109 121 127 19 55 75 91, 106 109 127 16 19 72 85 88, 106 34 52 88, 109 19 34 55 72 75 88, 127 31 37 75, 13 16 37 91, 19 34 37 72 75 88}	7

support and/or confidence of all rules below the disclosure parameters. Authors give three algorithms for sanitization. Since both zeros and ones can be turned to unknowns, the reconstruction of the source database is shown to be impossible.

Recently, knowledge hiding is extended to hide sequential patterns [2] from sequence databases and spatio-temporal patterns [1] from trajectory databases against sequential pattern mining [6] and spatio-temporal pattern mining [11], respectively.

In all of the works mentioned above, sensitive knowledge have the same format as the knowledge in the respective data mining task. In other words, the result of data mining is considered a list of knowledge and some of them are attributed sensitive. With this work, we extend the sensitivity specification beyond this restriction to meta-knowledge level. This extension allows several new directions of sensitivity specification at meta-knowledge level like co-occurrence, conditionals and exceptions. We study one of the directions, co-occurring frequent itemsets, in this work.

7. CONCLUSION

Privacy and related issues are becoming widespread with the ease of powerful information processing and transmission. The problem is becoming more versatile with development of sophisticated analysis software like data mining tools. This made privacy preserving and privacy aware data mining a very hot topic for researchers and practitioners. Knowledge hiding is aimed at hiding knowledge attributed sensitive from the datasets to be released. In most non trivial settings, it reduces to the study of the tradeoff between sensitivity and utility protection. It is traditionally studied in the context of frequent itemsets and associations rules and many algorithmic solutions are proposed in the literature.

With this work, we extend sensitive knowledge hiding to meta-knowledge hiding and present a version of it in the context of frequent itemset mining. Namely, co-occurring frequent itemset hiding problem is introduced and some theoretical properties are analyzed. As a solution, a two-stage sanitization framework is provided, where both stages are generic. The framework obtains an instance of the standard frequent itemset hiding problem in the first stage, and to solves the instance in the second stage. We have designed heuristic algorithms for the first stage as the second stage is a well-established field. Experimental evaluations are carried out on two datasets to understand the performances.

The results indicate that the proposed heuristics, especially MOMSH, performs very well on the defined metrics, M1/M2 and hitting set cardinality, in comparison to the most straightforward approach.

Our future work will include following;

- New heuristics: Currently, as an initial attempt we have developed three heuristics to be used in the first stage. We believe that there is room for more sophisticated heuristics like weighted hitting set.
- New experiments: Currently, we experiment with a toy dataset and a real dataset, *Connect*, from the game domain as initial tests. Clearly, more datasets can be tried with more realistic hiding scenarios.
- More direct algorithms: Currently, our solution involves transforming the instance of co-occurring frequent itemset hiding problem to an instance of frequent itemset hiding. More direct algorithms may attain smaller distortion.
- Other meta-knowledge: Currently, we only define co-occurring frequent itemsets as the meta-knowledge, but other meta knowledge that can find real world applications, e.g. correlations and exceptions, can be investigated.
- Other knowledge formats: Currently, we only address the meta-knowledge in the context of frequent itemset mining, but they can be defined similarly on other contexts such as association rules mining, sequential pattern mining etc.

8. REFERENCES

[1] O. Abul, M. Atzori, F. Bonchi, and F. Giannotti. Hiding sensitive trajectory patterns. In *6th International Workshop on Privacy Aspects of Data Mining (PADM'07), in conjunction with ICDM'07*.

[2] O. Abul, M. Atzori, F. Bonchi, and F. Giannotti. Hiding sequences. In *Third ICDE International Workshop on Privacy Data Management (PDM'07), in conjunction with ICDE'07*.

[3] R. Agarwal, C. Aggarwal, and V. Prasad. A tree projection algorithm for generation of frequent itemsets. *Journal of Parallel and Distributed Computing*, 61:350–371, 2000.

[4] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings SIGMOD'93*, pages 207–216, 1993.

[5] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB'94)*, pages 487–499, 1994.

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

[7] 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.

[8] 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.

[9] 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.

[10] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi. Geopkdd: Alignment report on privacy-preserving data mining. Technical report, Jan. 2006. Pisa KDD laboratory, ISTI-CNR and University of Pisa.

[11] 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*.

[12] 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.

[13] 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.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W. H. Freeman, Jan. 1979.

[15] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, 2004.

[16] 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.

[17] G. Moustakides and V. Verykios. A maxmin approach for hiding frequent itemsets. *IEEE Transactions on Knowledge and Data Engineering*, 65:75–89, 2008.

[18] 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.

[19] S. R. M. Oliveira and O. R. Zaiane. A framework for enforcing privacy in mining frequent patterns. Technical report, Computer Science Department, University of Alberta, Canada, June 2002.

[20] S. R. M. Oliveira and O. R. Zaïane. Protecting sensitive knowledge by data sanitization. In *Proceedings ICDM 2003*, pages 211–218, 2003.

[21] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. pages 432–444, 1995.

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

[23] X. Sun and P. S. Yu. A border-based approach for hiding sensitive frequent itemsets. In *Proceedings ICDM 2005*, pages 426–433, 2005.

[24] 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.

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

[26] X. Wu, Y. Wu, Y. Wang, and Y. Li. Privacy aware market basket data set generation: A feasible approach for inverse frequent set mining. In *Proceedings of the 2005 SIAM International Conference on Data Mining (SDM 2005)*, 2005.