Algorithms for Balancing Privacy and Knowledge Discovery in Association Rule Mining

Stanley R. M. Oliveira[1,2]
[1]Embrapa Informática Agropecuária
Av. André Tosello, 209
13083-886 - Campinas, SP, Brasil
oliveira@cs.ualberta.ca

Osmar R. Zaïane[2]
[2]Department of Computing Science
University of Alberta
Edmonton, AB, Canada T6G 2E8
zaiane@cs.ualberta.ca

Abstract

The discovery of association rules from large databases has proven beneficial for companies since such rules can be very effective in revealing actionable knowledge that leads to strategic decisions. In tandem with this benefit, association rule mining can also pose a threat to privacy protection. The main problem is that from non-sensitive information or unclassified data, one is able to infer sensitive information, including personal information, facts, or even patterns that are not supposed to be disclosed. This scenario reveals a pressing need for techniques that ensure privacy protection, while facilitating proper information accuracy and mining. In this paper, we introduce new algorithms for balancing privacy and knowledge discovery in association rule mining. We show that our algorithms require only two scans, regardless of the database size and the number of restrictive association rules that must be protected. Our performance study compares the effectiveness and scalability of the proposed algorithms and analyzes the fraction of association rules which are preserved after sanitizing a database. We also report the main results of our performance evaluation and discuss some open research issues.

1 Introduction

The recent advance of data mining technology to analyze vast amount of data has played an important role in marketing, business, medical analysis, and other applications where pattern discovery is paramount for strategic decision making. Despite its benefits in such areas, data mining also opens new threats to privacy and information security if not done or used properly. Recent advances in data mining and machine learning algorithms have introduced new problems in privacy protection [6, 2]. The main problem is that from non-sensitive data, one is able to infer sensitive information, including personal information, facts, or even patterns that are not supposed to be disclosed.

The current status in data mining research reveals that one of the current technical challenges is the development of techniques that incorporate security and privacy issues. The main reason is that the increasingly popular use of data mining tools has triggered great opportunities in several application areas, which also requires special attention regarding privacy protection.

In this paper, we focus on privacy preserving association rule mining. We start by considering a motivating example discussed in [3, 4]. Suppose a situation exists in which one supplier offers products in reduced prices to some consumers and, in turn, this supplier receives permission to access the database of the consumers' customer purchases. The threat becomes real whenever the supplier is allowed to derive highly sensitive knowledge from unclassified data that is not even known to the database owners (consumers). In this case, the consumers benefit from reduced prices, whereas the supplier is provided with enough information to predict inventory needs and negotiate other products to obtain a better deal for his consumers. This implies that the competitors of this supplier start losing business.

The simplistic solution to address the problem of our motivating example is to implement a filter after the mining phase to weed out/hide the restricted discovered association rules. However, in the context of our research, the users are provided with the data and not the association rules and are free to use their own tools, and thus the restriction for privacy has to be applied before the mining phase on the data itself. For this reason, to address this particular problem, we need to develop mechanisms that will enable data owners to choose an appropriate balance between privacy and precision in discovered association rules. Such mechanisms can lead to new privacy control systems to convert a given database into a new one in such a way to preserve the general rules mined from the original database. The released database is called sanitized database.

The procedure of converting an original database into a sanitized one is called the sanitization process and it was initially introduced in [1]. To do so, a small number of transactions have to be modified by deleting one or more items from them or even adding noise to the data by turning some items from 0 to 1 in some transactions. This approach relies on boolean association rules. On one hand, this approach slightly modifies some data, but this is perfectly acceptable in some real applications [3, 4, 9]. On the other hand, such an approach must hold the following restrictions: (1) the impact on the non-restricted data has to be minimal and (2) an appropriate balance between a need for privacy and knowledge discovery must be guaranteed.

To accomplish these restrictions, we introduce new algorithms for balancing privacy and knowledge discovery in association rule mining. Our sanitizing algorithms require only two scans regardless of the database size and the number of restrictive association rules that must be protected. The first scan is required to build the index (inverted file) for speeding up the sanitization process, while the second scan is used to sanitize the original database. This represents a significant improvement over the previous algorithms presented in the literature [4, 9], which require various scans depending on the number of association rules to be hidden. One major novelty with our approach is that we take into account the impact of our sanitization not only on hiding the association rules that should be hidden but also on hiding legitimate rules that should not be hidden. Other approaches presented in the literature focus on the hiding of restrictive rules but do not study the effect of their sanitization on accidentally concealing legitimate rules or even generating artifact rules (i.e. rules that do not exist in the original database).

Our algorithms are integrated with the framework for enforcing privacy in association rule mining presented in [7, 8]. The framework is composed of a transaction retrieval engine relying on an inverted file and Boolean queries for retrieving transaction IDs from a database, a set of sanitizing algorithms, and performance measures that quantify the fraction of association rules which are preserved after sanitizing a database. Our experiments demonstrate that our algorithms are effective and achieve reasonable results when compared with the other approaches presented in [4, 9].

This paper is organized as follows. In Section 2, we provide the basic concepts to understand the issues addressed in this paper. In addition, the problem definition is given. We present the idea behind our framework in Section 3. In Section 4, we introduce our sanitizing algorithms. In Section 5, we present the experimental results and discussion. Related work is reviewed in Section 6. Finally, Section 7 presents our conclusions and a discussion of future work.

2 Basic Concepts

In this section, we briefly review the idea behind transactional databases and association rules. After that, we present the formulation of the research problem.

2.1 Transactional Databases

A transactional database is a relation consisting of transactions in which each transaction t is characterized by an ordered pair, defined as t = $\langle$*TID*, *list_of_elements*$\rangle$, where *TID* is a unique transaction identifier number and *list_of_items* represents a list of items making up the transactions. For instance, in market basket data, a transactional database is composed of business transactions in which the list of elements represents items purchased in a store.

2.2 The Basics of Association Rules

Association rules provide a very simple but useful form of rule patterns for data mining. A rule consists of a left-hand side proposition (the antecedent or condition) and a right-hand side (the consequent). Both the left and right-hand side consist of Boolean statements (or propositions). The rules state that if the left-hand side is true, then the right-hand side is also true.

Formally, association rules are defined as follows: Let I = $\{i_1,...,i_n\}$ be a set of literals, called items. Let D be a database of transactions, where each transaction t is an itemset such that $t \subseteq I$. A unique identifier, called *TID*, is associated with each transaction. A transaction t supports X, a set of items in I, if $X \subset t$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$ and $X \cap Y = \emptyset$. Thus, we say that a rule $X \Rightarrow Y$ holds in the database D with *confidence* φ if $\frac{|X \cup Y|}{|X|} \geq \varphi$, where $|A|$ is the number of occurrences of the set of items A in the set of transactions D. Similarly, we say that a rule $X \Rightarrow Y$ holds in the database D with *support* σ if $\frac{|X \cup Y|}{|N|} \geq \sigma$, where N is the number of transactions in D.

Association rule mining algorithms rely on support and confidence and mainly have two major phases: (1) based on a support σ set by the user, frequent itemsets are determined through consecutive scans of the database; (2) strong association rules are derived from the frequent item sets and constrained by a minimum confidence φ also set by the user.

2.3 Privacy Preservation: Problem Definition

The scenario we address in this paper is one which deals with two parties A and B, A owning a transactional database and B wanting to mine it for association rules. The problem is how can A make some restrictive rules hidden regardless of which minimum support threshold B would use. Note that A does not know which association rule mining algorithm or support threshold B would use.

In this context, the database owner A needs to look for some sensitive association rules in order to prevent them from being disclosed. So A, the owner of the transactional database, has full access to the database and would know what should be restricted based on the application and the database content, whether these rules to restrict exist in the database or not. A only knows that if these rules exist they should not be disclosed to B. The user B has no knowledge that some rules were hidden. Once B gets access to the sanitized database, B can mine any available rule. The restricted rules, if they existed in the original database, are supposedly removed by the sanitization process by changing some transactions in the database. In other words, the user B does not have to know about the rules, and A only needs to know which rules (existing or not) should not be disclosed.

Given these facts, the specific problem addressed in this paper can be stated as follows: If D is the source database of transactions and R is a set of relevant association rules that could be mined from D, the goal is to transform D into a database D' so that the most association rules in R can still be mined from D' while others, representing restricted knowledge, will be hidden. In this case, D' becomes the released database.

3 The Framework for Privacy Preservation

As depicted in Figure 1, our framework encompasses a transactional database (modeled into a text database), an inverted file, a set of sanitizing algorithms used for hiding restrictive association rules from the database, a transaction retrieval engine for fast retrieval of transactions, and performance measures that quantify the fraction of association rules which are preserved after sanitizing a database. We describe the inverted file, the transaction retrieval engine, and the performance measures in this section, and the new algorithms in Section 4.

3.1 The Inverted File Index

Sanitizing a transactional database consists of identifying the sensitive transactions and adjusting them. To speed up this process, we model transactions into documents in which the items simply become terms. This model preserves all the information and provides the basis for our indexing (inverted file), borrowing from the information retrieval domain. We index the transactional database with the purpose of speeding up the sanitization process.

In our framework, the inverted file's vocabulary is composed of all different items in the transactional database,

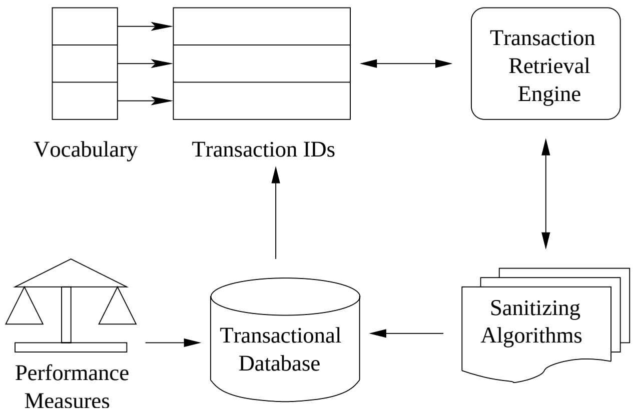
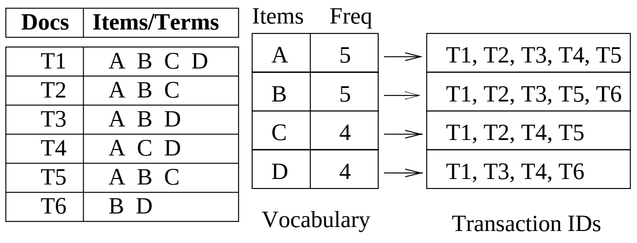


Figure 1. Privacy Preservation Framework

and for each item there is a corresponding list of transaction IDs in which the item is present. Figure 2 shows an example of an inverted file corresponding to the sample transactional database shown in the figure.



Docs	Items/Terms
T1	A B C D
T2	A B C
T3	A B D
T4	A C D
T5	A B C
T6	B D

Items	Freq	Transaction IDs
A	5	T1, T2, T3, T4, T5
B	5	T1, T2, T3, T5, T6
C	4	T1, T2, T4, T5
D	4	T1, T3, T4, T6

Vocabulary
Transaction IDs

Figure 2. An example of transactions modeled by documents and the corresponding inverted file

We implemented the vocabulary based on a perfect hash table [5], with no collision, insertion, or deletion. For a given item, one access suffices to find the list of all transaction IDs that contain the item.

3.2 The Transaction Retrieval Engine

To search for sensitive transactions in the transactional database, it is necessary to access, manipulate, and query transaction IDs. The transaction retrieval engine performs these tasks. It accepts requests for transactions from a sanitizing algorithm, determines how these requests can be filled (consulting the inverted file), processes the queries using a query language based on Boolean model, and returns the results to the sanitizing algorithm. The process of searching for sensitive transactions through the transactional database works on the inverted file. In general, this process follows three steps: (1) *Vocabulary search*: each restrictive association rule is split into single items. Isolated items are transformed into basic queries to the inverted index; (2) *Retrieval of transactions*: The lists of all transaction IDs of transactions containing each individual item respectively are retrieved; and (3) *Intersections of transaction lists*: The lists of transactions of all individual items in each restrictive association rule are intersected using a conjunctive Boolean operator on the query tree to find the sensitive transactions containing a given restrictive association rule.

3.3 Performance Measures

In this section, we introduce our privacy performance measures related to the problems illustrated in Figure 3.

Problem 1 occurs when some restrictive association rules are discovered. We call this problem ***Hiding Failure***, and it is measured in terms of the percentage of restrictive association rules that are discovered from D'. Ideally, the hiding failure should be 0%. The hiding failure is measured by $HF = \frac{\#R_R(D')}{\#R_R(D)}$ where $\#R_R(X)$ denotes the number of restrictive association rules discovered from database X. In our framework, the proportion of restrictive association rules that are nevertheless discovered from the sanitized database can be controlled with the disclosure threshold ψ, and this proportion ranges from 0% to 100%. Note that ψ does not control the *hiding failure* directly, but indirectly by controlling the proportion of sensitive transactions to be sanitized for each restrictive association rule.

Problem 2 occurs when some legitimate association rules are hidden by accident. This happens when some non-restrictive association rules lose support in the database due to the sanitization process. We call this problem ***Misses Cost***, and it is measured in terms of the percentage of legitimate association rules that are not discovered from D'. In the best case, this should also be 0%. The misses cost is calculated as follows: $MC = \frac{\#\sim R_R(D) - \#\sim R_R(D')}{\#\sim R_R(D)}$ where $\# \sim R_R(X)$ denotes the number of non-restrictive association rules discovered from database X. Notice that there is a compromise between the misses cost and the hiding failure. The more association rules we hide, the more legitimate association rules we miss.

Problem 3 occurs when some artificial association rules are generated from D' as a product of the sanitization process. We call this problem ***Artifactual Patterns***, and it is measured in terms of the percentage of the discovered association rules that are artifacts. This is measured as: $AP = \frac{|R'| - |R \cap R'|}{|R'|}$ where $|X|$ denotes the cardinality of X.

4 Sanitizing Algorithms

In this section, before we introduce our sanitizing algorithms, we present our heuristic approach to sanitize a transactional database.

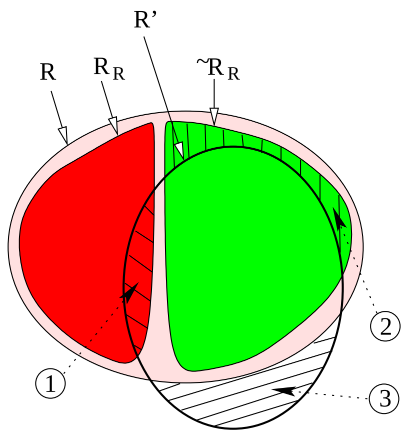


Figure 3. (A): Visual representation of restrictive and non-restrictive association rules and the rules effectively discovered after transaction sanitization.

4.1 Heuristic Approach

The goal of our heuristic is to facilitate proper information accuracy and mining, while protecting a group of association rules which contains highly sensitive knowledge. We refer to these rules as restrictive association rules and define them as follows:

Definition 1 (Restrictive Association Rules): Let D be a transactional database, σ the minimum support threshold, R be a set of all association rules that can be mined from D based on a minimum support σ, and $Rules_H$ be a set of decision support rules that need to be hidden according to some security policies. A set of association rules, denoted by R_R, is said to be restrictive if $R_R \subset R$ and if and only if R_R would derive the set $Rules_H$. $\tilde{}R_R$ is the set of non-restrictive association rules such that $\tilde{}R_R \cup R_R = R$.

Figure 3 illustrates the relationship between the set R of all association rules in the database D, the restrictive and non-restrictive association rules, as well as the set R' of patterns discovered from the sanitized database D'. 1, 2, and 3 are potential problems that respectively represent the restrictive association rules that were failed to be hidden, the legitimate rules accidentally missed, and the artificial association rules created by the sanitization process. We provide performance measures for these potential problems in Section 3.3.

A group of restrictive association rules is mined from a database D based on a special group of transactions. We refer to these transactions as sensitive transactions and define them as follows:

Definition 2 (Sensitive Transactions): Let T be a set of all transactions in a transactional database D and R_R be a set of restrictive association rules mined from D. A set of transactions is said to be sensitive, as denoted by S_T, if $S_T \subset T$ and if and only if all restrictive association rules can be mined from S_T and only transactions in S_T contain items involved in the restrictive association rules.

In most cases, a sensitive transaction derives more than one restrictive association rule. We refer to such transactions as conflicting transactions, since modifying one of them causes an impact on other restrictive transactions or even on non-restrictive ones. The *degree of conflict* of a sensitive transaction is defined as follows:

Definition 3 (Degree of Conflict of a Sensitive Transaction): Let D be a transactional database and S_T be a set of all sensitive transactions in D. The degree of a sensitive transaction t, denoted by degree(t), such that $t \in S_T$, is defined as the number of restrictive association rules that have items contained in t.

To illustrate the presented concepts, let us consider the sample transactional database in Figure 2. Suppose that we have a set of restrictive association rules R_R = {A,B→D; A,C→D}. This example yields the following results: the sensitive transactions S_T containing the restrictive association rules are {T1, T3, T4}. The degrees of conflict for the transactions T1, T3 and T4 are 2, 1 and 1 respectively. Thus, the only conflicting transaction is T1, which covers both restrictive association rules at the same time. An important observation here is that any association rule that contains a restrictive association rule is also restrictive. Hence, if A,B→D is a restrictive association rule but not A,C→D as above, any association rule derived from the itemset ABCD will also be restrictive since it contains ABD. This is because if ABCD is discovered to be a frequent itemset, it is straightforward to conclude that ABD is also frequent, which should not be disclosed. In other words, any superset containing ABD should not be allowed to be frequent.

Our sanitizing algorithms, presented in Section 4.2, act on the original database taking into account the degree of conflict of sensitive transactions.

4.2 Sanitizing Algorithms

Unlike algorithms that hide restrictive rules by modifying existing information in the database, our algorithms solely remove information by reducing the support of some items. This creates a smaller impact on the database since they do not generate artifacts such as illegal association rules that would not exist had the sanitizing not happened. These artifactual rules are generated by a noise addition approach, i.e., by adding some items in certain transaction. Such algorithms create the possibility of discovering some association rules that are not supposed to exist.

For our hiding strategies: Round Robin and Random algorithms, the inputs are a transactional database D, a set of restrictive association rules R_R, and a disclosure threshold ψ, while the output is the sanitized database D'. To sanitize a database, each sanitizing algorithm requires only two scans of the original database: one initial scan to build the inverted index, and an additional scan to alter some sensitive transactions, while keeping the other transactions intact.

All our sanitizing algorithms have essentially four major steps: (1) Identify sensitive transactions for each restrictive association rule; (2) For each restrictive association rule, identify a candidate item that should be eliminated from the sensitive transactions. This candidate item is called the *victim item*; (3) Based on the disclosure threshold ψ, calculate for each restrictive association rule the number of sensitive transactions that should be sanitized; and (4) Based on the number found in step 3, identify for each restrictive association rule the sensitive transactions that have to be sanitized and remove the victim item from them.

Our sanitizing algorithms mainly differ in step 2 in the way they identify a victim item to remove from the sensitive transactions for each restrictive rule, and in step 4 where the sensitive transactions to be sanitized are selected. Steps 1 and 3 remain essentially the same for all approaches.

The complexity of our sanitization algorithms in main memory is $O(n \times NlogN)$, where n is the number of restrictive association rules and N the number of transactions in the database. The proof of this is given in [7].

In section 5, we compare the effectiveness and scalability of Round Robin and Random algorithms with those ones proposed in [4, 9], and with the Item Grouping Algorithm, our best algorithm so far published and presented in [8]. The main idea behind the Item Grouping Algorithm, denoted by IGA, is to group restricted association rules in groups of rules sharing the same itemsets. If two restrictive rules intersect, by sanitizing the conflicting sensitive transactions containing both restrictive rules, one would take care of hiding these two restrictive rules in one step and consequently reduce the impact on the released database. However, clustering the restrictive rules based on the intersections between rules leads to groups that overlap since the intersection of itemsets is not transitive. By solving the overlap between clusters and thus isolating the groups, we can use a representative of the itemset linking the restrictive rules in the same group as a victim item for all rules in the group. By removing the victim item from the sensitive transactions related to the rules in the group, all sensitive rules in the group would be hidden in one step [8]. This again minimizes the impact on the database and reduces the potential accidental hiding of legitimate rules.

4.2.1 The Round Robin Algorithm

The main idea behind the Round Robin Algorithm, denoted by RRA, is: rather than selecting a unique victim item per given restrictive association rule, we select different victim items in turns starting from the first item, then the second and so on in each sensitive transaction. The process starts again at the first item of the restrictive rule as a victim item each time the last item is reached. The rationale behind this selection is that by removing one item at a time from the sensitive transactions it would alleviate the impact on the sanitized database and the legitimate association rules to be discovered, since this strategy tries to balance the decreasing of the support of the items in restrictive association rules. Selecting the sensitive transactions to sanitize is simply based on their degree of conflict. Given the number of sensitive transactions to alter, based on ψ, this approach selects for each restrictive rule the sensitive transactions whose degree of conflict is sorted in descending order. The rationale is that by sanitizing the conflict sensitive transactions that share a common item with more than one restrictive rule, this optimizes the hiding strategy of such rules in one step and, consequently, minimizes the impact of the sanitization on the discovery of the legitimate association rules. The sketch of the Round Robin Algorithm is given as follows:

```
Round_Robin_Algorithm
Input: $D$, $R_R$, $\psi$
Output: $D'$
Step 1. For each association rule $rr_i \in R_R$ do
            1. $T[rr_i]$ ←Find_Sensitive_Transactions($rr_i$, $D$);
Step 2. For each association rule $rr_i \in R_R$ do
            1. $Victim_{rr_i} \leftarrow item_v$ such that $item_v \in rr_i$ and
                  if there are $k$ items in $rr_i$, the i$th$ item is
                  assigned to $item_v$ mod $k$ in round robin fashion
Step 3. For each association rule $rr_i \in R_R$ do
            // $|T[rr_i]|$ is the number of sensitive transactions for $rr_i$
            1. $NumbTrans_{rr_i} \leftarrow |T[rr_i]| \times (1 - \psi)$
Step 4. $D' \leftarrow D$
        For each association rule $rr_i \in R_R$ do
            1. Sort_Transactions($T[rr_i]$); //in descending order of
                  degree of conflict
            2. $TransToSanitize \leftarrow$ Select first $NumbTrans_{rr_i}$
                  transactions from $T[rr_i]$
            3. in $D'$ foreach transaction $t \in TransToSanitize$ do
               3.1. $t \leftarrow (t - Victim_{rr_i})$
End
```

The four steps of this algorithm correspond to the four steps described above in the beginning of this section. The first step builds an inverted index of the items in D in one scan of the database. In step 2, the victim item $Victim_{rr_i}$ is selected in a round robin fashion, for each restrictive association rule. Line 1 in step 3 shows that ψ is used to compute the number $NumbTrans_{rr_i}$ of transactions to sanitize. This means that the threshold ψ is actually a measure on the impact of the sanitization rather than a direct measure on the restricted association rules to hide or disclose. Indirectly, ψ does have an influence on the hiding or disclosure of restricted association rules. There is actually only one scan of the database in the implementation of step 4. Transactions that do not need sanitization are directly copied from D to D', while the others are sanitized before copied to D'. In our implementation, the sensitive transactions to be cleansed are first marked before the database scan for copying. The selection of the sensitive transactions to sanitize, $TransToSanitize$ is based on their degree of conflict, hence the sort in line 1 of step 4. When a transaction is selected for sanitization, only the victim items are removed from it (line 3.1 in step 4).

4.2.2 The Random Algorithm

The intuition behind the Random Algorithm, denoted by RA, is to select as a victim item, for a given restrictive association rule, one item of such rule randomly. Like the Round Robin Algorithm, the rationale behind this selection is that removing different items from the sensitive transactions would slightly minimize the support of legitimate association rules that would be available for being mined in the sanitized database. Selecting the sensitive transactions to sanitize is simply based on their degree of conflict. We evaluated the sanitization through the Random Algorithm by selecting sensitive transactions sorted in ascending and descending order. The approach based on descending order, in general, yielded the best results. That is why we have adopted such an approach for our algorithm. The sketch of the Random Algorithm is given as follows:

```
Random_Algorithm
Input: $D$, $R_R$, $\psi$
Output: $D'$
Step 1. For each association rule $rr_i \in R_R$ do
            1. $T[rr_i]$ ←Find_Sensitive_Transactions($rr_i$, $D$);
Step 2. For each association rule $rr_i \in R_R$ do
            1. $Victim_{rr_i} \leftarrow item_v$ such that $item_v \in rr_i$ and
                  if there are $k$ items in $rr_i$, the item assigned to
                  $item_v$ is random(k)
Step 3. For each association rule $rr_i \in R_R$ do
            // $|T[rr_i]|$ is the number of sensitive transactions for $rr_i$
            1. $NumbTrans_{rr_i} \leftarrow |T[rr_i]| \times (1 - \psi)$
Step 4. $D' \leftarrow D$
        For each association rule $rr_i \in R_R$ do
            1. Sort_Transactions($T[rr_i]$); //in descending order of
                  degree of conflict
            2. $TransToSanitize \leftarrow$ Select first $NumbTrans_{rr_i}$
                  transactions from $T[rr_i]$
            3. in $D'$ foreach transaction $t \in TransToSanitize$ do
               3.1. $t \leftarrow (t - Victim_{rr_i})$
End
```

The four steps of this algorithms correspond to those in the Round Robin Algorithm. The only difference is that the Random Algorithm selects the victim item randomly, while the Round Robin Algorithm selects the victim item taking turns.

5 Experimental Results

We performed two series of experiments: the first to measure the effectiveness of our sanitization algorithms and the second to measure the efficiency and scalability of the algorithms. All the experiments were conducted on a PC, AMD Athlon 1900/1600 (SPEC CFP2000 588), with 1.2 GB of RAM running a Linux operating system. To measure the effectiveness of the algorithms, we used a dataset generated by the IBM synthetic data generator to generate a dataset containing 500 different items, with 100K transactions in which the average size per transaction is 40 items. The effectiveness is measured in terms of the number of restrictive association rules effectively hidden, as well as the proportion of legitimate rules accidentally hidden due to the sanitization. We selected for our experiments a set of ten restrictive association rules from the dataset ranging from two to five items in length with support ranging from 20% to 42% and confidence ranging from 80% to 100% in the database.

We ran the Apriori algorithm to select such association rules. The time required to build the inverted file in main memory was 4.05 seconds. Based on this inverted file, we retrieved all the sensitive transactions in 1.02 seconds. With our ten original restrictive association rules, 94701 rules became restricted in the database since any association rule that contains restrictive rules should also be restricted.

5.1 Measuring effectiveness

In this section, we measure the effectiveness of our algorithms taking into account the performance measures introduced in Section 3.3. We compare our algorithms with a similar one proposed in [4] to hide rules by reducing support, called Algo2a. The algorithm GIH designed by Saygin et al. [9] is similar to Algo2a. The basic difference is that in Algo2a some items are removed from sensitive transactions, while in GIH a mark ? (unknowns) is placed instead of item deletions.

Figure 4 shows a special case in which the disclosure threshold ψ is set to 0%, that is no restrictive rule is allowed to be mined from the sanitized database. In this situation, 30.16% of the legitimate association rules in the case of RRA and RA, 24.76% in the case of Algo2a, and 20.08% in the case of IGA are accidentally hidden.

While the algorithms proposed in [4, 9] hide rules reducing their absolute support in the database, in our framework the process of modifying transactions satisfies a disclosure threshold ψ controlled by the database owner. This threshold basically expresses how relaxed the privacy preserving mechanisms should be. When $\psi = 0\%$, no restrictive association rules are allowed to be discovered. When $\psi = 100\%$, there are no restrictions on the restrictive association rules. The advantage of having this threshold is that it enables a compromise to be found between hiding association rules while missing legitimate ones and finding all legitimate association rules but uncovering restrictive ones.

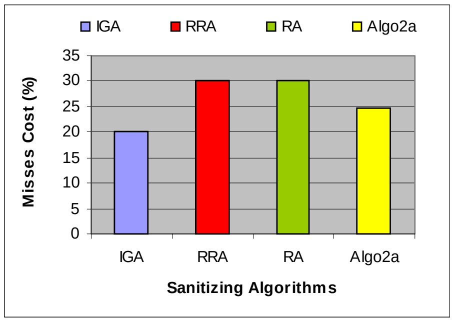
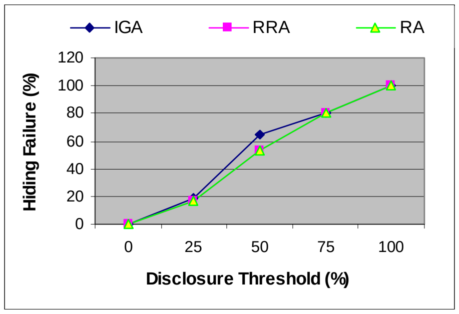


Figure 4. Effect of ψ on misses cost

Figure 5 shows the effect of the disclosure threshold ψ on the hiding failure and the misses cost for all three algorithms, considering the minimum support threshold $\sigma = 5\%$. Notice that RRA and RA yielded basically the same results. That is why their curves are very identical at the scale of the figure. As can be observed, when ψ is 0%, no restrictive association rule is disclosed for all three algorithms. However, 30.16% of the legitimate association rules in the case of RRA and RA, and 20.08% in the case of IGA are accidentally hidden. When ψ is equal to 100%, all restrictive association rules are disclosed and no misses are recorded for legitimate rules. What can also be observed is that the hiding failure for RA is slightly better than that for the other approaches. On the other hand, the impact of IGA on the database is smaller and the misses cost of IGA is the lowest among all approaches before $\psi = 75\%$. After this value, all the algorithms yield similar results.

Regarding the third performance measure, artifactual patterns, one may claim that when we decrease the frequencies of some items, the relative frequencies in the database may be modified by the sanitization process, and new rules may emerge. However, in our experiments, the problem artifactual pattern AP was always 0% with all algorithms regardless of the values of ψ. Our sanitization, indeed, does not remove any transaction. The same results can be observed for the algorithms presented in [4, 9].

We could measure the dissimilarity between the original and sanitized databases by computing the difference between their sizes in bytes. However, we believe that this



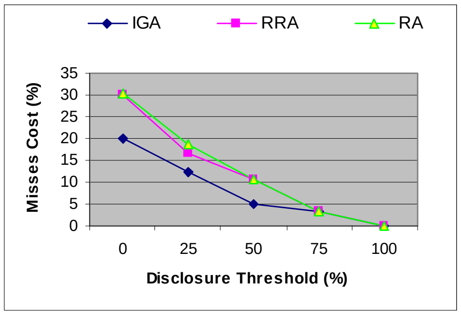


Figure 5. Effect of ψ on the hiding failure and misses cost

dissimilarity should be measured comparing their contents, instead of their sizes. Comparing their contents is more intuitive and gouges more accurately the modifications made to the transactions in the database.

To measure the dissimilarity between the original and the sanitized datasets we simply compare the difference of their histograms. In this case, the horizontal axis of a histogram contains all items in the dataset, while the vertical axis corresponds to their frequencies. The sum of the frequencies of all items gives the total of the histogram. So the dissimilarity between D and D', denoted by $dif(D, D')$, is given by:

$$dif(D, D') = \frac{1}{\sum_{i=1}^{n} f_D(i)} \times \sum_{i=1}^{n} [f_D(i) - f_{D'}(i)]$$

where $f_X(i)$ represents the frequency of the ith item in the dataset X.

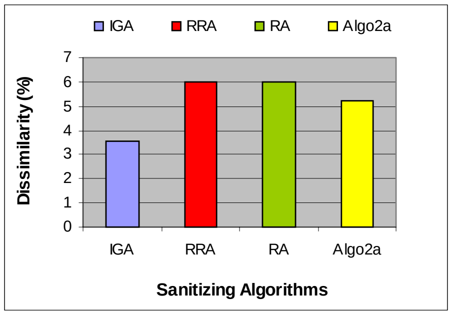


Figure 6. Difference in size between D and D'

Figure 6 shows the differential between the initial size of the database and the size of the sanitized database when the disclosure threshold $\psi = 0\%$. To have the smallest impact possible on the database, the sanitization algorithm should not reduce the size of the database significantly. As can be seen, IGA is the one that impacts the least on the database. In this particular case, 3.55% of the database is lost in the case of IGA, 6% in the case of RRA and RA, and 5.24% in the case of Algo2a.

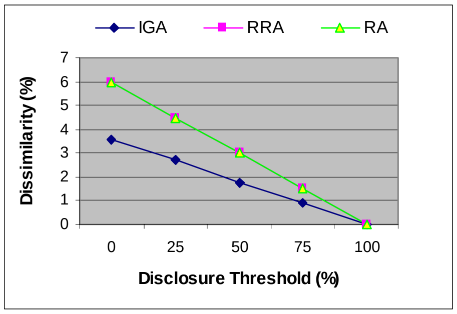


Figure 7. Difference in size between D and D'

Figure 7 shows the differential between the initial size of the database and the size of the sanitized database for our three algorithms with respect to the disclosure threshold ψ. Again, IGA is the one that impacts the least on the database for all values of the disclosure threshold ψ. Thus, as can be seen, the three algorithms slightly alter the data in the original database, while enabling flexibility for someone to tune them.

5.2 CPU Time for the Sanitization Process

We tested the scalability of our sanitization algorithms vis-à-vis the size of the database as well as the number of rules to hide. Our comparison study also includes the algorithm Algo2a.

We varied the size of the original database D from 20K transactions to 100K transactions, while fixing the disclosure threshold ψ and the support threshold to 0%, and keeping the set of restrictive rules constant (10 original patterns). Figure 8A shows that IGA, RRA, and RA increase CPU time linearly with the size of the database, while the CPU time in Algo2a grows fast. This is due the fact that Algo2a requires various scans over the original database, while our algorithms require only two. Note that our algorithms yield almost the same CPU time since they are very similar. Although IGA sanitizes less sensitive transactions, it has an overhead to group restrictive association rules that share the same items and optimizes this process.

We also varied the number of restrictive rules to hide from approximately 6000 to 29500, while fixing the size of the database to 100K transactions and fixing the support and disclosure thresholds to ψ = 0%. Figure 8B shows that our algorithms scale well with the number of rules to hide. The figure reports the size of the original set of restricted rules, which varied from 2 to 10. This makes the set of all restricted rules range from approximately 6097 to 29558. This scalability is mainly due to the inverted files we use in our approaches for indexing the transactions per item and indexing the sensitive transactions per restrictive rule. There is no need to scan the database again whenever we want to access a transaction for sanitization purposes. The inverted file gives direct access with pointers to the relevant transactions. The CPU time for Algo2a is more expensive due the number of scans over the database.

6 Related Work

Some effort has been made to address the problem of privacy preservation in association rule mining. The class of solutions for this problem has been restricted basically to randomization, data partition, and data sanitization. In this work, we focus on the latter category.

The idea behind data sanitization was introduced in [1]. Atallah et al. considered the problem of limiting disclosure of sensitive rules, aiming at selectively hiding some frequent itemsets from large databases with as little impact on other non-sensitive frequent itemsets as possible. Specifically, the authors dealt with the problem of modifying a given database so that the support of a given set of sensitive rules, mined from the database, decreases below the minimum support value. The authors focused on the theoretical approach and showed that the optimal sanitization is an NP-hard problem.

In [4], the authors investigated confidentiality issues of a broad category of association rules and proposed some algorithms to preserve privacy of such rules above a given privacy threshold. Although these algorithms ensure privacy preservation, they are CPU-intensive since they require multiple scans over a transactional database. In addition, such algorithms, in some way, modifies true data values and relationships by turning some items from 0 to 1 in some transactions.

In the same direction, Saygin et al. [9] introduced a method for selectively removing individual values from a database to prevent the discovery of a set of rules, while preserving the data for other applications. They proposed some algorithms to obscure a given set of sensitive rules by replacing known values with unknowns, while minimizing the side effects on non-sensitive rules. These algorithms also require various scans to sanitize a database depending on the number of association rules to be hidden.

Oliveira and Zaïane [8] introduced a unified framework that combines techniques for efficiently hiding restrictive patterns: a transaction retrieval engine relying on an inverted file and Boolean queries; and a set of algorithms to “sanitize” a database. In this framework, the sanitizing algorithms require two scans regardless of the database size and the number of restrictive patterns that must be protected.

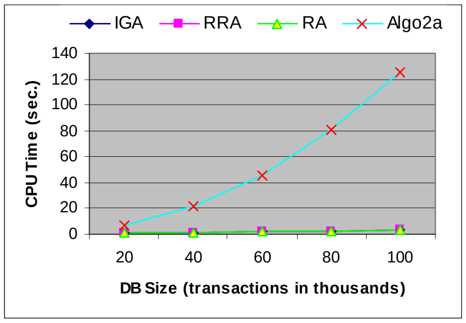
The work presented here differs from the related work in some aspects, as follows: First, we extended our previous work presented in [8] by adding two new algorithms (Round Robin and Random) to the set of sanitizing algorithms. Second, the hiding strategies behind our algorithms deal with the problem 1 and 2 in Figure 3, and most importantly, they do not introduce the problem 3 since we do not add noise to the original data. Third, we study the impact of our hiding strategies in the original database by quantifying how much information is preserved after sanitizing a database. So, our focus is not only on hiding restrictive association rules but also on maximizing the discovery of rules after sanitizing a database. Another difference of our algorithms from the related work is that our algorithms require only two scans over the original database, while the algorithms presented in [4, 9] require various scans depending on the number of association rules to be hidden. This is due the fact that our sanitizing algorithms are built on indexes and, consequently, they achieve a reasonable performance.

7 Conclusions

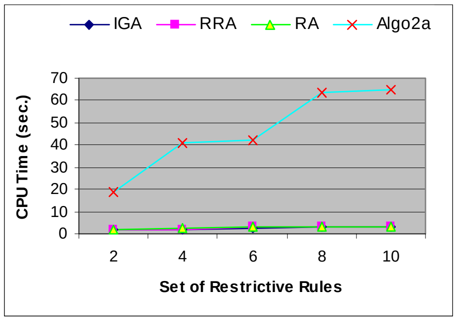
In this paper, we have introduced two algorithms for balancing privacy and knowledge discovery in association rule mining. Our sanitizing algorithms require only two scans regardless of the database size and the number of restrictive association rules that must be protected. This first scan is required to build the index (inverted file) for speeding up the sanitization process, while the second scan is used to sanitize the original database. This represents a significant improvement over the previous algorithms presented in the literature [4, 9].

Our algorithms are integrated to the framework presented in [8], which combines three advances for efficiently hiding restrictive rules: inverted files, one for indexing the transactions per item and a second for indexing the sensitive transactions per restrictive association rule; a transaction retrieval engine relying on Boolean queries for retrieving transaction IDs from the inverted file and combining the resulted lists; and a set of sanitizing algorithms.

The experimental results revealed that our algorithms for sanitizing a transactional database can achieve reasonable



A



B

Figure 8. Results of CPU time for the sanitization process

results when compared with the other approaches in the literature. Such algorithms slightly alter the data while enabling flexibility for someone to tune them. In particular, the IGA algorithm reached the best performance, in terms of dissimilarity and in terms of preservation of legitimate association rules. On the other hand, the results suggested that RA is slightly better than the other algorithms for hiding failure.

Although our algorithms guarantee privacy and do not introduce false drops to the data, an extra cost is payed because some rules would be removed accidentally since there are functional dependencies between restricted and non-restricted rules. The rationale behind this is that privacy preserving association rule mining deals with a trade-off: privacy and accuracy, which are contradictory, i.e., improving one usually incurs a cost for the other.

It is important to note that our sanitization methods are robust in the sense that there is no de-sanitization possible. The alterations to the original database are not saved anywhere since the owner of the database still keeps an original copy of the database intact while distributing the sanitized database. Moreover, there is no encryption involved. There is no possible way to reproduce the original database from the sanitized one.

Currently, we are investigating new optimal sanitization algorithms that minimize the impact in the sanitized database, while facilitating proper information accuracy and mining. In addition, we are working on the optimization of the algorithms RRA and RA, specially in terms of preservation of legitimate association rules, since their results revealed they are promising.

8 Acknowledgments

Stanley Oliveira was partially supported by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) of Ministry for Science and Technology of Brazil, under Grant No. 200077/00-7. Osmar Zaïane was partially supported by a Research Grant from NSERC, Canada. We would like to thank Yücel Saygin and Elena Dasseni for providing us the code of their respective algorithms for our comparison study.

References

[1] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure Limitation of Sensitive Rules. In *Proc. of IEEE Knowledge and Data Engineering Workshop*, pages 45–52, Chicago, Illinois, November 1999.

[2] C. Clifton. Using Sample Size to Limit Exposure to Data Mining. *Journal of Computer Security*, 8(4):281–307, November 2000.

[3] C. Clifton and D. Marks. Security and Privacy Implications of Data Mining. In *Workshop on Data Mining and Knowledge Discovery*, pages 15–19, Montreal, Canada, February 1996.

[4] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, and E. Bertino. Hiding Association Rules by Using Confidence and Support. In *Proc. of the 4th Information Hiding Workshop*, pages 369–383, Pittsburg, PA, April 2001.

[5] M. Dietzfelbinger, A. R. Karlin, K. Mehlhorn, F. M. auf der Heide, H. Rohnert, and R. E. Tarjan. Dynamic Perfect Hashing: Upper and Lower Bounds. *SIAM Journal on Computing*, 23(4):738–761, 1994.

[6] 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. AAAI/MIT Press, pages 507-516, Menlo Park, CA, 1991.

[7] S. R. M. Oliveira and O. R. Zaïane. A Framework for Enforcing Privacy in Mining Frequent Patterns. Technical report, TR02-13, Computer Science Department, University of Alberta, Canada, June 2002.

[8] S. R. M. Oliveira and O. R. Zaïane. Privacy Preserving Frequent Itemset Mining. In *Proc. of the IEEE ICDM Workshop on Privacy, Security, and Data Mining*, pages 43–54, Maebashi City, Japan, December 2002.

[9] Y. Saygin, V. S. Verykios, and C. Clifton. Using Unknowns to Prevent Discovery of Association Rules. *SIGMOD Record*, 30(4):45–54, December 2001.