Privacy Preserving Frequent Itemset Mining by Reducing宋
Sensitive Items Frequency using GA

B.Janakiramaiah a, Dr A.Rama Mohan Reddy b, G.Kalyani c
a,c Department of Computer Science and Engineering,DVR &Dr HS MIC College of Technology, Vijayawada, India.
b Department of Computer Science and Engineering, S.V. University, Tirupathi, India.

Abstract
Frequent Itemset mining extracts novel and useful knowledge from large repositories of data and this knowledge is useful for effective analysis and decision making in telecommunication networks, marketing, medical analysis, website linkages, financial transactions, advertising and other applications. The misuse of these techniques may lead to disclosure of sensitive information. Motivated by the multiple conflicting requirements of data sharing, privacy preserving and knowledge discovery, privacy preserving data mining has become a research hotspot in data mining and database security fields. To address this challenging problem, different data distortion approaches were projected to protect sensitive data in transactional databases. We address this problem of privacy preserving frequent itemset mining by applying data sanitization to prevent the disclosure of sensitive item sets while maintaining data utility. To identify the most sensitive transactions for hiding given sensitive frequent item sets, we are using Genetic Algorithm optimization technique. In the proposed work, we focused on finding the sensitive transactions for hiding sensitive frequent item sets using Genetic Algorithm. The Genetic Algorithm is used due to its inbuilt characteristics like robustness with respect to local maxima/minima and area–autonomous nature for huge space exploration technique to find exact or estimated solutions for optimization and exploration problems. The Genetic Algorithm approach identifies the sensitive transactions to hide all the sensitive frequent item sets while minimizing the effect on non sensitive item sets. The performance of the algorithm is validated against representative synthetic and real datasets with some performance measures. Finally our experimental results show that the proposed work yields good performance for hiding sensitive knowledge.

Keywords: Frequent Itemset Mining, Sensitive Knowledge, Genetic Algorithm, Data Sanitization, Privacy, Optimization, Data Distortion

1. Introduction
Frequent Itemset mining extracts novel, hidden and useful patterns from huge repositories of data. These patterns are useful for effective analysis and decision making in telecommunication network, marketing, business, medical analysis, website linkages, financial transactions, advertising and other applications. The sharing of Frequent Itemsets can bring lot of advantages in industry, research and business collaboration. At the same time, a huge repository of data contains private data and sensitive itemsets that must be protected before sharing. On demand to various mismatched requirements of data sharing, privacy preserving and knowledge discovery, privacy preserving data mining (PPDM) has become a research hotspot in data mining. Simply, the Frequent Itemset hiding problem is to hide secret, sensitive patterns contained in data from being discovered, while without losing non-sensitive at the same time. The problem of Frequent Itemset hiding motivated many authors [1, 2], and proposed different approaches. The majority of the proposed approaches can be classified along two principal research directions: (i) Data hiding approaches and (ii) Knowledge hiding approaches.

1.1. Data hiding approaches
Data hiding methods [3, 4] collect methodologies that explore how the privacy of raw data, or information, can be maintained before the course of mining the data. The approaches of this category aim at the removal of confidential or private information from the original data prior to its discloser and operate by applying techniques such as transformation, generalization, perturbation and sampling, etc.
1.2. Knowledge hiding approaches

These approaches involve methodologies that aim to protect the sensitive data produced by the raw data itself, which were produced by the application of data mining tools on the original database. These can be further classified into two subcategories: Data Distortion techniques and Data Blocking techniques. Data Distortion [1,5] is implemented by deleting or adding items to reduce the support of the sensitive itemset, while data blocking [2] is implemented by replacing certain items with a question mark (?) to make the support of the sensitive itemsets uncertain.

2. Genetic Algorithm

In the computer science field of artificial intelligence, the Genetic Algorithm (GA) is a search heuristic that mimics the process of natural fruition. This heuristic is usually used to generate useful solutions to optimization and exploration problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. Genetic algorithms find applications in bioinformatics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics and other fields. In a genetic algorithm, a population of candidate solutions to an optimization problem is evolved towards better solution. Each candidate solution has a set of properties which can be mutated and distorted; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual’s genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population [14]. The simple function representation of Genetic Algorithm is as follows:

```c
Function SGA ()
{
    Initialize the population;
    Calculate the fitness function;
    While (fitness value! = terminating criteria)
    {
        Apply genetic operations;
        Calculate fitness function;
    }
    The process of representing a solution in the form of a string conveys the necessary information. Just as in a chromosome, each gene controls a particular characteristic of the individual; similarly, each bit in the string represents a characteristic of the solution. Most common method of encoding is binary coded. A fitness function value quantifies the optimality of a solution. The value is used to rank a particular solution against all the other solutions. A fitness value is assigned to each solution depending on how close it is actually to the optimal solution of the problem. The primary objective of the selection operator is to emphasize the good solutions and eliminate the bad solutions in a population while keeping the population size constant. Identify the good solutions in a population. Make multiple copies of the good solutions. Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population. There are different techniques to implement selection in Genetic Algorithms. They are: Tournament selection, Roulette wheel selection, Rank selection, Steady state selection, etc [6]. The crossover operator is used to create new solutions from the existing solutions. The most popular crossover selects any two solutions strings randomly from the mating pool and some portion of the strings is exchanged between the strings. The selection point is selected randomly. Mutation is the occasional introduction of new features in to the solution strings of the population pool to maintain diversity in the population [7]. Mutation operator changes a 1 to 0 or vice-versa. The mutation probability is generally kept low for steady convergence. Sometimes it really takes a huge amount of time. This is understood because the genetic algorithm is not a mathematical way to the solution of the problem, but it is more or less a stochastic, discrete, non-linear, and a high-dimensional seeking algorithm. The genetic algorithm is parallel-adjusted by itself, so it does not take much work to reform it into a parallel algorithm. There are many parallelization based methods for increasing the genetic algorithm execution speed. Basically, they can be divided into three groups: global, migration, and dispersed methods [8]. At the end of the discussion, we will list some of the advantages of genetic algorithms:

- Solves all optimization problems which can be described with binary encoding.
- Solves problems with numerous solutions.
- Solves multi-dimensional, non-differential, non continuous and even non-parametrical problems.
- Genetic algorithm is a method which is very easy to understand and it practically doesn’t insist the knowledge on mathematics.

3. Related Works

Distortion-based approaches operate by selecting specific items to include to (or exclude from) selected trans- actions of the original database in order to facilitate the hiding of the sensitive frequent itemsets. Two of the
most commonly employed strategies for data distortion involve the swapping of values between transactions [1], as well as the deletion of specific items from the database.

Dasseni, et al. [1] generalize the hiding problem in the sense that they consider the hiding of both sensitive frequent itemsets and sensitive association rules. The authors propose three single rule heuristic hiding algorithms that are based on the reduction of either the support or the confidence of the sensitive rules, but not both. In all three approaches, the goal is to hide the sensitive rules while minimally affecting the support of the non-sensitive itemsets.


Oliveira and Zaane [2] were the first to introduce multiple rule hiding approaches. The proposed algorithms are efficient and require two scans of the database, regardless of the number of sensitive itemsets to hide. During the first scan, an index file is created to speed up the process of finding the sensitive transactions and to allow for an efficient retrieval of the data. In the second scan, the algorithms sanitize the database by selectively removing the least amount of individual items that accommodate the hiding of the sensitive knowledge. Three item restriction-based algorithms (known as MinFIA, MaxFIA, and IGA) are proposed that selectively remove items from transactions that support the sensitive rules. A more efficient approach than that of [2] and the work of [1], was introduced by Oliveira and Zaane in [10]. The proposed algorithm, called SWA, is an efficient, scalable, one-scan heuristic which aims at providing a balance between the needs for privacy and knowledge discovery in association rule hiding.

Amiri [11] proposes three effective, multiple association rule hiding heuristics that outperform SWA by offering higher data utility and lower distortion, at the expense of increased computational speed. Although similar in philosophy to the previous approaches, the three proposed methodologies do a better job in modelling the overall objective of a rule hiding algorithm.

Pontikakis, et al. [5] propose two distortion-based heuristics to selectively hide the sensitive association rules. The proposed schemes use efficient data structures for the representation of the association rules and effectively prioritize the selection of transactions for sanitization. However, in both algorithms the proposed hiding process may introduce a number of side effects, either by generating rules which were previously unknown, or by eliminating existing non-sensitive rules.

4. Problem Definition

We focus on the knowledge hiding thread of PPDM and study on specific class of approaches which are collectively known as frequent itemset hiding approaches. In the context of privacy preserving frequent itemset mining, we do not concentrate on privacy of individuals rather, we concentrate on the problem of protecting sensitive knowledge mined from databases. The sensitive knowledge is represented by a special group of association rules called sensitive association rules. These rules are most important for strategic decision making and must remain private (i.e., the frequent itemsets are private to the owner of the data). The problem of protecting sensitive knowledge in transactional databases draw the assumption that Data owners have to know in advance some knowledge ( frequent itemsets and/or rules) that they want to protect. Such rules are fundamental in decision making, so they must not be discovered. The problem of protecting sensitive knowledge in association rule mining can be stated as, given a data set D to be released, a set of frequent itemsets R mined from D, and a set of sensitive itemsets RS ⊆ R to be hidden. How can we get a new data set D1, such that the itemsets in RS cannot be mined from D1, while the itemsets in R−RS can still be mined as many as possible. In this case, D1 becomes the released database.

5. Proposed Framework

In the proposed framework, initially the frequent itemsets, R will be mined from the database D by using any association rule mining algorithm (AR). Then the user will specify the sensitive itemsets, RS which need to be hidden from mining. By considering sensitive itemsets and original dataset as input our proposed algorithm RSIF-GS will release a sanitized dataset D1. Then by applying any association rule mining algorithm on the sanitized dataset D1 we can mine all frequent itemsets which are mined from original dataset D except the sensitive itemsets. The proposed framework is shown in figure 1.

![Proposed Framework](image)

**Fig. 1.** Proposed Framework for frequent itemset hiding.

6. The Proposed Algorithm

The proposed algorithm assumes the sensitive itemsets to be hidden are predefined by data owners. For achieving this purpose, transactions may be deleted or modified. Here, the deletion of items in PPDM is thus used for hiding sensitive item sets or knowledge which reduces the support
of the item sets below the user specified threshold. It evaluates the degrees of transactions associated with given sensitive item sets. The measure for the Fitness Function (FF) value of a transaction Ti is defined as follows:

\[ FF = \sum_{i=1}^{n} \frac{F_{ij}}{F_j} \times \frac{1}{\sum_{i=1}^{n} \log_{MDC-support} n} \]

where \( F_{ij} \) is the number of sensitive items contained in the \( i \)th sensitive item set in \( T_j \), and \( F_j \) is the number of items in the offspring, \( n \) is the number offspring’s, support is the frequency count of each item, and MDC is the maximum decreased count of each item. The above formula consists of two components. One is the sensitive items frequency (SIF) and the other is the database frequency (DF). The sensitive items frequency (SIF) value is measured for each sensitive item set \( F_{ij} \) in a offspring \( T_j \). It is calculated as the number (Fij) of sensitive items in \( T_j \) which are included in an assigned sensitive item set \( F_{ij} \) divided by the number of all the items in \( T_j \). On the contrary, the DF value shows the influence degree of the sensitive item sets within offspring by considering all the offspring’s. The proposed GA based approach first selects the random offspring’s. Then it applies crossover and mutation to generate the new offspring. For each newly generated offspring SIF of each sensitive item set will be calculated. After that for each item decreased count (DC) value will be evaluated as \( C_i - MST \times n + 1 \), where \( C_i \) is occurrence frequency of \( i \)th sensitive item in new offspring’s, Minimum support threshold(MST) and \( n \) is number of offspring’s. MDC will be evaluated for each item by considering DC’s of the item with respect to each sensitive item set. DF value of each item will be evaluated by using MDC and Support count of that item as \( \sum_{i=1}^{n} \log_{MDC-support} n \). After that FF value will be evaluated for each offspring. The offspring with the lowest FF value will be discarded and in that place the offspring with the highest FF value will be placed and the whole procedure will be repeated until the fitness function value decreases or the same offspring will have again the highest FF value. Then identify the transaction from the database corresponding to that offspring. From that transaction delete the item with highest count among the sensitive item sets. Update the support of sensitive item sets and repeat again until all the sensitive item sets frequency is less than the MST. The proposed algorithm uses and modifies the concept from SIF-ID [12] and TF-IDF [13] in text mining to evaluate the degrees of transactions associated with given sensitive item sets.

7. Algorithm: RSIF-GA (Reducing Sensitive Item Frequency using Genetic Algorithm)

Input:
1. A Transactional Dataset D
2. Minimum Support Threshold MST
3. A set of Sensitive item sets

Output:
A Sanitized Database with no sensitive item sets \( D' \)

Method:
1. Randomly select the initial population of required size from transactional database \( D \).
2. Perform the single point crossover to generate new offspring’s.
3. Calculate the SIF of each sensitive item with respect to new offspring’s as follows:

\[ SIF_{ij} = \frac{F_{ij}}{F_j} \]

Where \( F_{ij} \) = no. of items of \( i \)th sensitive item that are present in \( j \)th transaction, \( F_j \) = No of items in \( j \)th transaction.
4. Calculate the DF of each item with the following steps.
   4.1 Calculate the DC of each item with respect to the sensitive item as \( C_i - MST \times n + 1 \), where \( C_i \) is occurrence frequency of \( i \)th sensitive item in new offspring’s, and \( n \) is no of offspring’s.
   4.2 Calculate the MDC of each Item.
   4.3 Calculate the DF of each item as:

\[ DF = \log_{MDC-support} n \]

Where support is no of times the item is in the offspring’s.
5. Find the initial offspring’s randomly from the transactional database \( D \).
6. Perform the single point crossover to generate new offspring’s.
7. Find the initial offspring’s randomly from the transactional database \( D \).
8. Perform the single point crossover to generate new offspring’s.
9. Calculate the SIF of each sensitive item with respect to new offspring’s as follows:

\[ SIF_{ij} = \frac{F_{ij}}{F_j} \]

Where \( F_{ij} \) = no. of items of \( i \)th sensitive item that are present in \( j \)th transaction, \( F_j \) = No of items in \( j \)th transaction.
10. Calculate the DF of each item with the following steps.
    10.1 Calculate the DC of each item with respect to the sensitive item as \( C_i - MST \times n + 1 \), where \( C_i \) is occurrence frequency of \( i \)th sensitive item in new offspring’s, and \( n \) is no of offspring’s.
    10.2 Calculate the MDC of each Item.
    10.3 Calculate the DF of each item as:

\[ DF = \log_{MDC-support} n \]

Where support is no of times the item is in the offspring’s.
11. Calculate the DF of each offspring with respect to each sensitive item as :

\[ \sum_{i=1}^{p} \log_{MDC-support} n \]

Where \( p \) is the number of items of the sensitive itemsets that are present in the offspring.
12. Calculate the FF of each offspring as:

\[ FF = \sum_{i=1}^{m} \frac{F_{ij}}{F_j} \times \sum_{i=1}^{n} \log_{MDC-support} n \]

where \( m \) is the no of sensitive item sets.
13. Repeat step 2 to step 6 until either of the following condition satisfies,1. Highest fitness function values decreases in the next iteration,2. Same offspring will have highest fitness function value again.
14. Find the offspring with highest FF value and identify its corresponding transaction in the transactional database.
15. Process the transaction to prune the appropriate items by the following steps:
   15.1 Sort the items in descending order of their occurrence frequencies within the Sensitive item sets.
   15.2 Sort the items in descending order of their occurrence frequencies within the Sensitive item sets.
   15.3 Delete the item from the transaction.
16. Update the occurrence frequencies of the sensitive item sets.
17. Repeat step 1 to 10 until the set of sensitive item sets is null which indicates that the support of all the sensitive item sets are below the user specific MST.

The flow chart of the algorithm is as shown in Figure 2.

8. Illustrative Example

In this section an example is given to demonstrate the proposed RSIF-GA algorithm. Assume a database shown in Table 1.

<table>
<thead>
<tr>
<th>Tid</th>
<th>List of Items</th>
<th>Tid</th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item 4</th>
<th>Item 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>I1, I2, I3</td>
<td>100</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>200</td>
<td>I2, I4</td>
<td>200</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>300</td>
<td>I2, I3</td>
<td>300</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>400</td>
<td>I2, I2, I4</td>
<td>400</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>500</td>
<td>I1, I3</td>
<td>500</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>600</td>
<td>I2, I3</td>
<td>600</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>700</td>
<td>I1, I3</td>
<td>700</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>800</td>
<td>I1, I2, I3, I5</td>
<td>800</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>900</td>
<td>I1, I2, I3</td>
<td>900</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

It consists of 9 transactions and 5 items denoted I1 to I5. Assume the set of user specified sensitive item sets are \{(I1, I3), (I1, I2, I3)\}. Also Assume the User specified MST is set at 22% which indicates that the minimum count is 0.22*9 which is approximately 2. The proposed approach to hide the sensitive item sets proceeds as follows.

Step 1: Select 4 transactions randomly from the database as T1, T2, T6 and T8 which are initial offspring’s.
Step 2: Perform single point crossover on the result of step 1.
Step 3: Calculate SIF. Take the first offspring as an example i.e 1 1 0 0 1. SIF of (I1, I3) is 1/3 because among I1, I3 only I1 is existed in the offspring and total 3 items are present in that offspring. In the similar manner SIF of (I1, I2, I3) is 2/3 because I1 and I2 are existed in the first offspring. Perform the same procedure to all the offspring’s with respect to (I1, I3) and (I1, I2, I3). The results of steps 1 to 3 are shown in Table 2.

Figure 2: Flow chart of RSIF-GA
Table 2: Result of Crossover Operation and SIF Values of iteration 1

<table>
<thead>
<tr>
<th>Initial off springs</th>
<th>After single crossover</th>
<th>SIF (I1,I3)</th>
<th>SIF (I1,I2,I3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001</td>
<td>11010</td>
<td>1/3</td>
<td>2/3</td>
</tr>
<tr>
<td>01010</td>
<td>01001</td>
<td>0</td>
<td>½</td>
</tr>
<tr>
<td>01100</td>
<td>01101</td>
<td>1/3</td>
<td>2/3</td>
</tr>
<tr>
<td>11101</td>
<td>11100</td>
<td>2/3</td>
<td>1</td>
</tr>
</tbody>
</table>

Step 4: Calculating DF. Consider first item I1. Because I1 is existing in sensitive item (I1,I3), DC of I1 with respect to (I1,I3) is 1-2+1((I1,I3) that exists only once in the offspring’s) which is 0. In the similar manner II also existed in the sensitive item set (I1,I2,I3), DC of II with respect to (I1,I2,I3) is 1-2+1 (I1,I2,I3) exists only once in the offspring’s) which is 0. If the item does not exist in the sensitive itemset then DC will be 0 for that. After evaluating DC of each item with respect to each sensitive itemset then identify Maximum DC (MDC) as maximum of all DC’s.

Step 5: Calculate DF of each item. For item I1 the MDC is 0. Support of I1 in the offspring’s is 2 and we are considering 4 offspring’s. So DF(I1)=log 4/2-0=log 2=0.301. In the similar manner calculate the DF of each item. The results of step 4 and 5 are shown in the Table 3.

Table 3: Calculating Data Frequency Values in iteration 1

<table>
<thead>
<tr>
<th>Item Name</th>
<th>DC (I1,I3)</th>
<th>DC (I1,I2,I3)</th>
<th>MDC</th>
<th>Support Data Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0.301</td>
</tr>
<tr>
<td>I2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>I3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0.301</td>
</tr>
<tr>
<td>I4</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0.602</td>
</tr>
<tr>
<td>I5</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0.602</td>
</tr>
</tbody>
</table>

Step 6: Calculating the FF value for each offspring. For this first calculate the DF of each offspring with respect to each sensitive item. Consider the first offspring 11010. For this SIF(I1,I3) is 1/3 ; DF(I1,I3) is 0.301 and SIF(I1,I2,I3) is 2/3 and DF(I1,I2,I3) is 0.301. So the FF value will become 1/3*0.301+2/3*0.301 which is 0.3009. In the similar manner FF value will be calculated for every offspring. The results of step 6 are shown in the Table 4.

Table 4: Calculating the Fitness Function Value for Every Transaction in iteration 1

<table>
<thead>
<tr>
<th>Trans</th>
<th>SIF (I1,I3)</th>
<th>DF (I1,I3)</th>
<th>SIF (I1,I2,I3)</th>
<th>DF (I1,I2,I3)</th>
<th>FF</th>
<th>RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>11010</td>
<td>1/3</td>
<td>0.301</td>
<td>2/3</td>
<td>0.301</td>
<td>0.3</td>
<td>1</td>
</tr>
<tr>
<td>01001</td>
<td>0</td>
<td>0</td>
<td>½</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01101</td>
<td>1/3</td>
<td>0.301</td>
<td>2/3</td>
<td>0.301</td>
<td>0.3</td>
<td>1</td>
</tr>
<tr>
<td>11100</td>
<td>2/3</td>
<td>0.602</td>
<td>1</td>
<td>0.602</td>
<td>1.0</td>
<td>2</td>
</tr>
</tbody>
</table>

Step 7: The offspring with lowest FF value will be discarded which is indicated with rank 0 in Table 5. The offspring with highest FF value will be repeated 2 times which is indicated with rank 2 in Table 5. Here the offspring 11100 will have highest FF value. With the modified new offspring’s again repeat step 1 to step 6. The corresponding results are shown in Table 6, 7 and 8. After the second iteration also the offspring 11100 is having the highest FF value so the iterations are stopped.

Table 5: Result of Crossover Operation and SIF Values of iteration 2

<table>
<thead>
<tr>
<th>Initial off springs</th>
<th>After Single Crossover</th>
<th>SIF (I1,I3)</th>
<th>SIF (I1,I2,I3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001</td>
<td>11100</td>
<td>1/3</td>
<td>2/3</td>
</tr>
<tr>
<td>01010</td>
<td>01001</td>
<td>0</td>
<td>½</td>
</tr>
<tr>
<td>01100</td>
<td>01101</td>
<td>1/3</td>
<td>2/3</td>
</tr>
<tr>
<td>11101</td>
<td>11100</td>
<td>2/3</td>
<td>1</td>
</tr>
</tbody>
</table>

Step 8: The corresponding Transaction of the offspring 11100 is T9.

Step 9: The sorted list of the items based on the sensitive item sets is I1:2, I3:2, I2:1. The first sensitive item in the list is I1. So delete I1 from T9. The result database is shown in Table 8.

Table 7: Calculating the Fitness Function Value for Every Transaction in iteration 2

<table>
<thead>
<tr>
<th>Trans</th>
<th>SIF (I1,I3)</th>
<th>DF (I1,I3)</th>
<th>SIF (I1,I2,I3)</th>
<th>DF (I1,I2,I3)</th>
<th>FF</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>11100</td>
<td>2/3</td>
<td>0.602</td>
<td>3/3</td>
<td>0.727</td>
<td>1.12</td>
<td>8</td>
</tr>
<tr>
<td>11010</td>
<td>1/3</td>
<td>0.301</td>
<td>2/3</td>
<td>0.426</td>
<td>0.38</td>
<td>4</td>
</tr>
<tr>
<td>01100</td>
<td>1/2</td>
<td>0.301</td>
<td>2/2</td>
<td>0.426</td>
<td>0.57</td>
<td>6</td>
</tr>
<tr>
<td>11101</td>
<td>2/4</td>
<td>0.602</td>
<td>¾</td>
<td>0.727</td>
<td>0.84</td>
<td>6</td>
</tr>
</tbody>
</table>

Step 10: Before deletion the support counts of (I1, I3) and (I1, I2, I3) are 4 and 2 respectively. After deletion the support counts are updated as 3 and 1 respectively. The Sensitive item set (I1, I2, I3) support count is now less than the minimum support threshold which indicates that it is hidden. Repeat Steps 1 to 10 again until all the sensitive item sets were hidden (support count less than the minimum support threshold). The final sanitized database is shown in Table 9.
Table 9: Final Sanitized Database

<table>
<thead>
<tr>
<th>TID</th>
<th>I1</th>
<th>I2</th>
<th>I3</th>
<th>I4</th>
<th>I5</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>200</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>300</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>400</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>500</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>600</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>700</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>800</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>900</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

9. Performance Measures

9.1. Hiding Failure (HF)

When some restrictive patterns are discovered from \( D^1 \), we call this problem as Hiding Failure, and it is measured in terms of the percentage of restrictive patterns that are discovered from \( D^1 \). The hiding failure is measured by

\[
HF = \frac{\text{#Rp}(D^1)}{\text{#Rp}(D)}
\]

where \( \text{#Rp}(D^1) \) denotes the number of restrictive patterns discovered from sanitized database \( (D^1) \), and \( \text{#Rp}(D) \) denotes the number of restrictive patterns discovered from original database \( D \).

9.2. Misses Cost (MC)

Some non-sensitive patterns can be hidden by mining algorithms accidentally. This happens when some non-sensitive patterns lose support in the database due to the sanitization process. We call this problem as Misses Cost, and it is measured in terms of the percentage of legitimate patterns that are not discovered from \( D^1 \). The misses cost is calculated as follows:

\[
MC = \frac{\#~\text{Rp}(D) - \#~\text{Rp}(D^1)}{\#~\text{Rp}(D)}
\]

where \( \#~\text{Rp}(D) \) denotes the number of non-sensitive patterns discovered from original database \( D \), and \( \#~\text{Rp}(D^1) \) denotes the number of non-sensitive patterns discovered from sanitized database \( D^1 \).

9.3. Artifactual Patterns (AP)

Some artificial patterns are going to be generated from \( D^1 \) as a product of the sanitization process. We call this problem as Artifactual Patterns, and it is measured in terms of the percentage of the discovered patterns that are artifacts.

9.4. \( \text{diff}(D, D^1) \)

We could measure the dissimilarity between original and sanitized database by simply comparing their histograms.

10. Experiment Results

All the experiments were conducted on PC, Intel i5 CPU @ 2.50 GHz and 4 GB of RAM running on a windows 7, 64-bit operating system. To measure the effectiveness of the algorithm, we used a dataset generated by the IBM synthetic data generator and FIMI Repository [14]. The Data set characteristics are shown in Table 10.

Table 10: Data Set Characteristics

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>No. of items</th>
<th>Avg Length</th>
<th>No. of Transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>bms1</td>
<td>497</td>
<td>2.5</td>
<td>59,602</td>
</tr>
<tr>
<td>Retail</td>
<td>16,469</td>
<td>103</td>
<td>88,162</td>
</tr>
<tr>
<td>T10I4D10K</td>
<td>1000</td>
<td>10</td>
<td>10000</td>
</tr>
</tbody>
</table>

When some sensitive itemsets are discovered from \( D^1 \), we call this problem as Hiding Failure, and it is measured in terms of the percentage of sensitive itemsets that are discovered from \( D^1 \).

Fig. 3. Percentage of Sensitive rules produced by RSIF-GA.

Fig. 3 shows efficiency of the proposed algorithm in the Hiding Failure. Accordingly, the RSIF-GA algorithm will not produce any sensitive itemsets from \( D^1 \), when hiding any number sensitive rules.

Some non-sensitive itemsets can be hidden by mining algorithms accidentally. We call this problem as misses cost and is measured in terms of percentage of non-sensitive itemsets that are not generated from \( D^1 \). This happens when some non-sensitive itemsets lose support in the database due to the sanitization process.

Fig. 4. Percentage of non-sensitive rules produced by RSIF-GA.
Fig. 4 shows the efficiency of the proposed algorithms in the Misses cost minimization. Accordingly, the RSIF-GA algorithm achieved better results in reducing Misses cost.

Some artificial itemsets are going to be generated from D' as a product of the sanitization process. We call this problem as Artifactual itemsets and it is measured in terms of the percentage of the discovered itemsets that are artifacts.

![Artifactual Itemsets](image)

Fig. 5. Percentage of Artifactual rules produced by RSIF-GA.

Fig. 5 shows efficiency of the proposed algorithm in the Artifactual patterns. Accordingly, the RSIF-GA algorithm did not produce any aritfactual itemsets, when hiding any number sensitive rules.

The time needed by sanitization algorithm, increases proportional to |D|, |RS| and also depends on MST, MCT.

11. Conclusion

In this paper a novel framework has been implemented for privacy preserving frequent itemsets mining. We have introduced an efficient implementation of Reducing Sensitive Item Frequency using Genetic Algorithm (RSIF-GA) for hiding sensitive itemsets from transactions and generating a sanitized database D'. This Sanitization algorithm preserves privacy for sensitive itemsets and the non-sensitive itemsets that are found when mining this original database can still be mined from its sanitized database. Our further research will focus on integrating other soft computing techniques to get better performance of the proposed approach.

References