CHAPTER IV

ASSOCIATION RULE HIDING TECHNIQUES

4.1 INTRODUCTION

In data mining, association rule learning is a popular and well researched method for discovering interesting relations between variables in large databases. Piatetsky-Shapiro [1991] describes analyzing and presenting strong rules discovered in databases using different measures. Based on the concept of strong rules, Agrawal et al. [1993] introduced association rules that are used for discovering regularities between products in large scale transaction data recorded by point-of-sale (POS) systems in supermarkets. For example, the rule \{milk, bread\} \Rightarrow \{butter\} found in the sales data of a supermarket would indicate that if a customer buys milk and bread together, he or she is likely to buy butter also. Such information can be used as the basis for decisions about marketing activities such as, promotional pricing or product placements. Association rules are employed today in many application areas including web mining, intrusion detection and bioinformatics. Recent years have seen tremendous advances in the ability to perform association rule mining effectively. Such rules often encode important target marketing information about a business.

Association rule hiding techniques are used for protecting the sensitive association rules. There are two aspects to the privacy preserving association rule mining problem. When the input to the data is perturbed, it is a challenging problem to determine accurately the association rules on the perturbed data. The other aspect is the output association rule privacy. In this case, none of the association rules in the output result in leakage of sensitive data is ensured. This problem is referred to as association rule hiding [Verykios et.al, 2004] by the database community, and that of contingency table privacy-preservation by the statistical community.
4.2 SYSTEM ARCHITECTURE

The main objective of this association rule hiding research work is to protect the sensitive association rules by modifying the sensitive items. Four new hiding techniques are proposed based on modified genetic algorithm (M-GA), tabu search (TS), ant colony optimization (ACO) and dummy items creation (DIC). The system architecture of association rule hiding is given in figure 4.1.

![System Architecture - Association Rule Hiding](image-url)

**Figure 4.1 System Architecture - Association Rule Hiding**
The following performances measures are used for finding the efficiency of the hiding techniques.

- **Hiding Failure:** To verify whether any sensitive rules appear in the modified data set or not.

- **Misses Cost:** To verify whether any non-sensitive rules are hidden during the sanitization process as a side effect.

- **Artifactual Errors:** To verify whether any fake rules are generated.

- **Efficiency:** Time required for modifying the sensitive items.

The performances of the proposed hiding techniques are compared with the existing techniques ISL (Increase Support on Left Handside), DSR (Decrease Support on Right HandSide) and the hiding technique based on genetic algorithm. The system architecture of the proposed methodology is given in the figure 4.1.

**Transactional Database and Association Rule Mining**

Let \( I = \{I_1, I_2, \ldots, I_m\} \) be a set of items in transactional database \( D = \{t_1, t_2, \ldots, t_n\} \), where \( t_i = \{I_{i1}, I_{i2}, \ldots, I_{ik(i)}\} \) and \( I_{ij} \in I \), \( 1 \leq j \leq k(i), 1 \leq i \leq n \), where each transaction \( T \) is an item set such that \( T \subseteq I \). An association rule is an implication of the form \( X \Rightarrow Y \), where \( X \subseteq I \), \( Y \subseteq I \), and \( X \cap Y = \emptyset \). The association analysis produces the “strong” association rules only, i.e. those that are “expressive” and “confident”. Common measures of association rule are support and confidence, and these are calculated on the basis of the support of the corresponding itemset. For the rule \( X \Rightarrow Y \), the corresponding itemset is \( X \cup Y \). An arbitrary set \( X \subseteq I \) is called the itemset. The support of the itemset \( X \), denoted by \( \sigma(X) \), is defined by the following formula:

\[
\sigma(X) = |\{t_i | 1 \leq i \leq n \land X \subseteq t_i \land t_i \in D\}|
\]

The support of the item set \( X \) is the number of transactions that contain \( X \). A transaction \( t_i \) contains an itemset \( X \), if \( X \subseteq t_i \). The support of the association rule \( X \Rightarrow Y \), denoted by \( \sigma(X \Rightarrow Y) \), is the quotient of the number of transactions in a database that contain \( X \cup Y \), and the number of all transactions \( n \). The formula for support calculation is,
\[ \sigma(X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{n} \]

The confidence of the association rule \( X \Rightarrow Y \), denoted by \( \alpha(X \Rightarrow Y) \), is the quotient of the number of transactions that contain \( X \cup Y \), and the number of transactions containing \( X \). i.e.

\[ \sigma(X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \]

The association analysis problem is the discovery of all the association rules \( X \Rightarrow Y \), where \( \sigma(X \Rightarrow Y) \geq \text{minsup} \) and \( \alpha(X \Rightarrow Y) \geq \text{minconf} \), where \( \text{minsup} \) and \( \text{minconf} \) are the input parameters of an algorithm for the association analysis problem. Table 4.1 shows a transactional database \( (D) \) which consists of six transactions and five items. The item which is present in the transaction is represented as 1 and 0 indicates the absence of an item.

### Table 4.1 Sample Transactional Database

<table>
<thead>
<tr>
<th>Transactions/Items</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( t_4 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( t_5 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( t_6 )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Association Rule Generation**

Association rules express regularities that exist in a dataset. Interest is restricted to those that occur often and that predicts with a high confidence, because a vast amount of different association rules can be derived from even a tiny dataset. An association rule is of the form \( A \Rightarrow B \), where \( A \) is called the *left hand side of the rule* and \( B \) is called the *right hand side of the rule*. The set of association rules is derived from a set of transactions in a database.

There are several algorithms used for generating association rules. Some of them are apriori algorithm, partition algorithm, pincer-search algorithm, dynamic item set counting algorithm, fp-tree growth algorithm, Eclat algorithm and Dclat algorithm etc. In
this association rule hiding research work, ECLAT algorithm is used for generating association rules. Generally, the problem of mining association rule algorithms are decomposed into two sub-problems. The first sub problem is finding all sets of items (itemsets) whose frequency is greater than or equal to the user-specified minimum frequency \((m_{fre})\). Such itemsets are called frequent itemsets. The second sub problem is generating the desired rules using the frequent itemsets. Association rule or sensitive rules are generated from the frequent item sets [Atallah et.al,1999].

**Table 4.2 Original Database (D)**

<table>
<thead>
<tr>
<th>Items</th>
<th>Tid-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1 3 4 5</td>
</tr>
<tr>
<td>b</td>
<td>1 2 3 4 5 6</td>
</tr>
<tr>
<td>c</td>
<td>2 4 5 6</td>
</tr>
<tr>
<td>d</td>
<td>1 3 5 6</td>
</tr>
<tr>
<td>e</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

**Sensitive Association Rules**

In this association rule hiding, an Eclat association rule algorithm is used for finding the frequent itemsets. Minimum frequency threshold \(m_{fre}\) is a very essential parameter for finding the frequent itemsets. Assigning the minimum support value is done in two ways. (i) It may be given by the data owner or (ii) To select the minimum support automatically; the following calculation is to be done.

\[
x = \text{number of transactions} / 2 \quad \text{(where n = number of transactions)}
\]

The minimum support value is selected from \(x\) to \(n\). This may be decided by the data owner. For example, if the transactional data set contains 10 transactions then the minimum support is selected from 5 to 10. This condition helps to restrict the items which occur less frequently. With this minimum support value, the Eclat algorithm finds all the frequent items. From these frequent items, the association rules are generated. The association rules whose support and confidence is above the given minimum support threshold \((m_{sup})\) and minimum confidence threshold \((m_{conf})\) then those rules are considered
as sensitive rules. In the next step, the items which are found in the sensitive association rules are identified. If there are \( n \) items found in the sensitive association rules, select only \((n/2)\) items whose individual support is very high. These items are considered as sensitive items. Then the hiding techniques are used for modifying the occurrences of the sensitive items by decreasing its support value. Number of modifications required by each sensitive item is calculated as,

\[ \text{Number of modifications} = \text{support of sensitive item} - (m_{\text{fre}} - 1) \]

Based on this information, the hiding techniques modify the sensitive items for protecting sensitive association rules. Consider the table 4.2 where the minimum frequency value is \( m_{\text{fre}} = 3 \). Then the frequent items generated by the Eclat algorithm are \( a, b, d \) and \( e \). From these frequent items, the association rules are generated. Consider the minimum support=60\% and minimum confidence=80\%, then the following rules are considered as sensitive.

\[ \begin{array}{|c|c|c|}
\hline
\text{Sensitive Association Rules} & \text{Minimum Support (60\%)} & \text{Minimum Confidence (80\%)} \\
\hline
a \Rightarrow b & 66.6 & 100 \\
\hline
a \Rightarrow b,e & 66.6 & 100 \\
\hline
b \Rightarrow e & 100 & 100 \\
\hline
d \Rightarrow b & 66.6 & 100 \\
\hline
d \Rightarrow e & 66.6 & 100 \\
\hline
d \Rightarrow b,e & 66.6 & 100 \\
\hline
e \Rightarrow b & 83.3 & 100 \\
\hline
\end{array} \]

The items found in the sensitive association rules are \( a, b, d \) and \( e \). From this, \( b \) and \( e \) are considered as sensitive items, because the support of \( b \) is 6 and \( e \) is 5. Number of modifications required for an item \( b \) is 4 and \( e \) is 3.

**Preprocessing**

Preprocessing is done in the original dataset and it is the first step of the hiding process. This step is very essential since the sensitive items are limited to some transactions, and so it is unnecessary to modify all the transactions. The main function of
this step is to select the transactions which support any one of the sensitive items. This process improves the sanitization speed which helps to achieve better performance.

Hiding Techniques

In Atallah et. al, [1999] it is proved that solving this association rule hiding problem by decreasing the support of large item sets via removing items from transactions or adding fake item into the transactions are NP-hard problem. Therefore, looking for a special modification of \( D \) (the source dataset) in \( D' \) (sanitized dataset which is going to be released) that maximizes the number of rules in \( AR_{non-Sen} \) (minimizing number of lost rules) that can still be mined. Therefore, this is a specific optimization problem. Primarily, it must conceal the sensitive association rule. Thus, it is necessary to modify the dataset, and on the other side it should keep the utility of modified dataset to extract useful information and rules.

4.3 EXISTING HIDING TECHNIQUES

4.3.1 ISL and DSR Algorithms

In this approach [Wang et al, 2007], in order to hide an association rule, \( X \Rightarrow Y \), either decrease its support value, \( (|X|/N \text{ or } |X \cup Y|/N) \), to be smaller than pre-specified minimum support or its confidence \( (|X \cup Y|/|X|) \) to be smaller than pre-specified minimum confidence value. To decrease the confidence of a rule, two strategies can be considered. The first one is to increase the support count of \( X \), i.e., the left hand side of the rule, but not support count of \( X \cup Y \). The second one is to decrease the support count of the item set \( X \cup Y \). For the second case, in the transactions containing both \( X \) and \( Y \), if the support of \( Y \) is decreased, then the right hand side of the rule would reduce the confidence faster than reducing the support of \( X \). To decrease the support count of an item, one item is removed at a time in a selected transaction by changing the value 1 to 0 and to increase the support count of an item, one item is added by changing the value from 0 to 1. The first algorithm, \textit{Increase Support of LHS (ISL)} tries to increase the support of left hand side of the rule. The second algorithm \textit{Decrease Support of RHS (DSR)}, tries to decrease the support of the right hand side of the rule.

\textit{Algorithm 4.1 Increase Support of LHS (ISL)}
Input:
(1) a source database $D$,
(2) a min_support,
(3) a min_confidence,
(4) a set of predicting items $X$

Output: a transformed database $D'$, where rules containing $X$ on LHS will be hidden

1. Find large 1-item sets from $D$;
2. For each predicting item $x \in X$
3. If $x$ is not a large 1-itemset, then $X := X - \{x\}$;
4. If $X$ is empty, then EXIT; // no rule contains $X$ in LHS
5. Find large 2-itemsets from $D$;
6. For each $x \in X$
7. For each large 2-itemset containing $x$
8. Compute confidence of rule $U$, where $U$ is a rule like $x \rightarrow y$;
9. If $\text{conf}(U) < \text{min}_\text{conf}$, then
   10. Go to next large 2-itemset;
11. Else {//Increase Support of LHS
   12. Find $\text{TL} = \{t \text{ in } D|t \text{ does not support } U\}$;
   13. Sort $\text{TL}$ in ascending order by the number of items;
   14. While $\{\text{conf}(U) \geq \text{min}_\text{conf} \text{ and } \text{TL} \text{ is not empty}\}$
   15. Choose the first transaction $t$ from $\text{TL}$;
   16. Modify $t$ to support $x$, the LHS($U$);
   17. Compute support and confidence of $U$;
   18. Remove and save the first transaction $t$ from $\text{TL}$;
   19. }; // end While
20. }; // end if $\text{conf}(U) < \text{min}_\text{conf}$
21. If $\text{TL}$ is empty, then {
   22. Cannot hide $x \rightarrow y$;
   23. Restore $D$;
   24. Go to next large-2 itemset;
25. } // end if $\text{TL}$ is empty
26. } // end of for each large 2-itemset
27. Remove $x$ from $X$;
28. } // end of for each $x \in X$
29. Output updated $D$, as the transformed $D'$;

Algorithm 4.2 Decrease Support of RHS (DSR)
Input:
(1) a source database D,
(2) a min_support,
(3) a min_confidence,
(4) a set of predicting items X
Output: a transformed database D’, where rules containing X on LHS will be hidden
1. Find large 1-item sets from D;
2. For each predicting item $x \in X$
3. If $x$ is not a large 1-itemset, then $X := X \setminus \{x\}$;
4. If $X$ is empty, then EXIT; // no rule contains $X$ in LHS
5. Find large 2-itemsets from D;
6. For each $x \in X$
   7. For each large 2-itemset containing $x$
      8. Compute confidence of rule $U$, where $U$ is a rule like $x \rightarrow y$;
      9. If $\text{conf}(U) < \text{min}_\text{conf}$, then
         10. Go to next large 2-itemset;
      11. Else //Decrease Support of RHS
         12. Find $\text{TR} = \{t \in D | t \text{ fully support } U\}$;
         13. Sort $\text{TR}$ in ascending order by the number of items;
         14. While $\{\text{conf}(U) \geq \text{min}_\text{conf} \text{ and } \text{TR} \text{ is not empty}\}$ {
            15. Choose the first transaction $t$ from $\text{TR}$;
            16. Modify $t$ so that $y$ is not supported;
            17. Compute support and confidence of $U$;
            18. Remove and save the first transaction $t$ from $\text{TR}$;
         19. }; // end While
         20. }; // end if $\text{conf}(U) < \text{min}_\text{conf}$
21. If $\text{TR}$ is empty, then {
   22. Cannot hide $x \rightarrow y$;
   23. Restore $D$;
24. Go to next large-2 itemset;
25. }; // end if $\text{TR}$ is empty
26. }; // end of for each large 2-itemset
27. Remove $x$ from $X$;
28. }; // end of for each $x \in X$
29. Output updated $D$, as the transformed $D’$;
4.3.2  Hiding Technique based on Genetic Algorithm

A genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. It was developed by John Holland, University of Michigan (1970’s) to understand the adaptive processes of natural systems and to design artificial systems software that retains the robustness of natural systems. It is an efficient, effective technique for optimization and machine learning applications, [David L, 1991], [Goldberg D E, 1989] [Abdullah Konaka et al, 2005]. 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 are widely used today in business, scientific and engineering circles. The components of genetic algorithms are [Colin R.Reeves, 2002] as follows:

- Encoding technique (gene, chromosome)
- Initialization procedure (creation)
- Evaluation function (environment)
- Selection of parents (reproduction)
- Genetic operators (mutation, recombination)
- Parameter settings (practice and art)

Algorithm 4.3 Simple Genetic Algorithm

```
{
    initialize population;
    evaluate population;
    while TerminationCriteriaNotSatisfied    {
        select parents for reproduction;
        perform recombination and mutation;
        evaluate population;  }
}
```
Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible [Goldberg, 1989]. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, and multiple individuals are selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population 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. Genetic algorithms are different from more normal optimization and search procedures in four ways:

- GAs work with a coding of the parameter set, not the parameters themselves.
- GAs search from a population of points, not a single point.
- GAs use payoff (objective function) information, not derivatives or other auxiliary knowledge.
- GAs use probabilistic transition rules, not deterministic rules. [Darrell Whitley, 1994]

A typical genetic algorithm requires:

- A genetic representation of the solution domain.
- A fitness function to evaluate the solution domain.

**Existing Hiding Technique Based on Genetic Algorithm**

The most important steps in the existing genetic algorithm based hiding technique are preprocessing phase and the specification of fitness function in genetic algorithm method [Mohammed Naderi et al, 2009]. Dataset pre-sanitization process is the first step which preprocesses the original database. The reason behind this is that the sensitive items are limited to some transactions, and therefore there is no need to modify all of the transactions. The preprocessing phase of this technique selects all the transactions that support sensitive items. The second step of this technique is genetic algorithm technique which is used for protecting sensitive rules. The genetic algorithm initializes the population and individual chromosomes. Each transaction is represented as a chromosome
and the presence of an item in transaction is shown by 1 and the absence of the item by 0. The fitness of a chromosome is determined by several factors and different strategies.

Each population consists of several chromosomes and the best chromosome is used to generate the next population. For the initial population, a large number of random transactions are chosen. Based on the survival of fitness, the population will transform into the future generation. The next step is to calculate the fitness function of every individual. Fitness strategy relies on both hiding all sensitive rules and minimum number of modification in original dataset. After evaluation of population’s fitness, the next step is chromosome selection. Selection embodies the principle of the survival of the fittest. Chromosomes that satisfy the fitness are selected for reproduction. Tournament selection is used when two chromosomes are chosen randomly from the population. First, for a predefined probability $p$, the chromosomes with more fitness of the two are selected and for the probability $(1-p)$ the other chromosome with less fitness is selected. The next step of this technique is the Crossover. The main function of the crossover operation in genetic algorithm is the combination of the two chromosomes, to generate a new offspring (child). Crossover occurs only with some probability (crossover probability). Chromosomes are not subject to crossover but remain unmodified.

**Algorithm 4.4 Hiding Technique based on Genetic Algorithm**

1. Initialization
   1.1 Consider a transactional database $D$ which consists of $T$ transactions where $D=\{T_1,T_2,\ldots,T_n\}$
   1.2 Each transaction $T_i$ contains a set of items $I$ where $I = \{I_1,I_2,I_3,\ldots,I_m\}$
      Where $T_i \in I$
   1.3 Initializing the sensitive items $S_i$ where $S_i \in I$
   1.4 Initializing the number of modifications required for each sensitive item
2. Preprocessing
   2.1 Select the transaction which contain a sensitive item where $Y_i=(S_i \in T_i)$
   2.2 Consider these transactions $Y_i$ are sensitive transactions ($ST_i$)
3. Fitness function
   3.1 $F(ST_i) = W_1 \times \text{Itemsets Hiding Distances} + W_2 \times \text{Number of Modifications}$
      // $W_1 + W_2$ is the necessary condition for weighted sum optimization
      problem and their values specified based on their costs
// Itemset, Hiding Distance = \begin{align*} & 0 & \text{if Support(Itemset)} \leq \text{MST} \\ & \text{Support(Itemset)} - \text{MST} & \text{otherwise} \end{align*} \\
// Itemsets Hiding Distances = \sum_{i=1}^{\text{Number of sensitive Itemsets}} \text{Itemset Hiding Distance}

4. Selection
   4.1 Select the transactions based on the fitness value

5. Crossover // modification of the sensitive items
   5.1 Perform crossover operation in the selected transactions

6. Mutation // after crossover if any Si needs modification
   6.1 Perform mutation operation

7. Termination
   7.1 Ensure all the sensitive items are modified
   7.2 Number of modification becomes 0 then the process is completed

8. Exit

The intuition behind crossover is exploration of new solutions and exploitation of old solutions. Better fitness value chromosomes have more prospects to be selected than the less fitness value chromosome, so that a good solution is always alive to the next generation. After performing crossover operation, the new introduced generation will only have the character of the parents. This behavior can lead to a problem where no new genetic material is introduced in the offspring and finding better population has been stopped. Mutation operator permits new genetic patterns to be introduced in the new chromosomes (random changed in random gene on chromosome). Mutation introduces a new sequence of genes into a chromosome but there is no guarantee that mutation will produce desirable features in the new chromosome. The selection process will keep it if the fitness of the mutated chromosome is better than the general population. Otherwise, selection will ensure that the chromosome does not live to mate in future.

4.4 PROPOSED HIDING TECHNIQUES

4.4.1 Hiding technique based on Modified Genetic Algorithm

The methodology used for developing the genetic algorithm based hiding technique is given in figure 4.2. The first step of this technique has initialized all the sensitive items and the number of modifications required for each sensitive item. The
transactions which contain sensitive items are sensitive transactions and these are considered as population. Each transaction is treated as an individual or chromosome of a population and the presence of an \( i^{th} \) item in the transaction i.e. shown by 1 and absence of the item by 0 in \( i^{th} \) bit of transaction. The fitness function is calculated for each individual using,

\[
f(ST_i) = \frac{(X_i + Y_i)}{2}
\]

where \( X_i \) represents the number of items present in an individual and \( Y_i \) the represents number of sensitive items found in an individual transactions. After the population’s fitness evaluation, the next step is to select the chromosome. Selection represents the principle of survival of the fittest. There are several selection methods, such as Roulette-Wheel selection, Rank selection and Tournament selection [Artem Sokolov, 2007]. Rank selection, which is used in this association rule hiding research work, the highest rank (fitness value) chromosomes are chosen for mutation. In this modified genetic algorithm, crossover operation is not performed. The main function of the crossover is generating new chromosomes by exchange of bit patterns. In this hiding technique, there is no need for generating new chromosomes (transactions). The modification is applied to the existing transactions only. This can be done through mutation operation. In mutation process, it verifies whether the first sensitive item is found in the transaction or not. If the sensitive item is found then it modifies the value as 0 from 1. After modification, the value of the number of modifications of the corresponding sensitive item is decreased by 1. Again, the new fitness value is calculated for the modified transaction and rank selection is applied to select the chromosome and again the mutation process is carried out. The sensitive items are modified in a round robin method. This process is repeated until the termination condition has occurred. Here the termination condition is fulfilled when the number of modifications of each sensitive item has become 0.
Figure 4.2 Flowchart - Hiding Technique Based on Modified Genetic Algorithm

Algorithm 4.5 Hiding Technique based on Modified Genetic Algorithm

1. Initialization
   1.1 Consider a transactional database \( D \) which consists of \( T \) transactions where \( D = \{ T_1, T_2, \ldots, T_n \} \)
   1.2 Each transaction \( T_i \) contains a set of items \( I \) where \( I = \{ I_1, I_2, I_3, \ldots, I_m \} \) where \( T_i \subseteq I \)
   1.3 Initializing the sensitive items \( S_i \) where \( S_i \subseteq I \)
   1.4 Initializing the number of modifications required for each sensitive item
2. Preprocessing
2.1 Select the transaction which contain a sensitive item where $Y_i=(S_i \text{ in } T_i)$

2.2 Consider these transactions $Y_i$ are sensitive transactions ($ST_i$)

3. Fitness function  
   //fitness function is computed for all the sensitive transactions

   3.1 For each sensitive transaction $ST_i$ in $D$ where $ST_i \in D$

   3.2 Calculate $f(ST_i) = \frac{(X_i + Y_i)}{2}$

      // $X_i$ represents number of items present in $ST_i$
      // $Y_i$ represents number of sensitive items present in $ST_i$

4. Selection

   4.1 Choose the highest rank transaction $ST_i$ based on $f(ST_i)$

5. Mutation  
   // modify the sensitive items in a round robin method $S_i = 1, 2, ..., n$

   5.1 Verify (if $S_i$ found in $ST_i$) and ($ST_i$ has more than one item) Then {

   5.2 Modify the sensitive item $S_i$ from ‘1’ to ‘0’

   5.3 Calculate the new fitness value of $ST_i$

   5.4 Decrease the number of modification of $S_i$ by 1

   5.5 Go to step 4 }

   5.6 Else { ‘no modification’

   5.7 Go to step 4 }

6. Terminating

   6.1 Ensure all the sensitive items are modified

   6.2 Number of modification becomes 0 then the process is completed

7. Exit

The sensitive items of the original database are sanitized by using the modified genetic algorithm. The modified database ($D'$) is given in Table 4.4.

### Table 4.4 Sanitized Database ($D'$) - Modified Genetic Algorithm

<table>
<thead>
<tr>
<th>Items</th>
<th>Tid-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1 3 4 5</td>
</tr>
<tr>
<td>B</td>
<td>4 6</td>
</tr>
<tr>
<td>C</td>
<td>2 4 5 6</td>
</tr>
<tr>
<td>D</td>
<td>1 3 5 6</td>
</tr>
<tr>
<td>E</td>
<td>2 3</td>
</tr>
</tbody>
</table>
4.4.2 Ant Colony Optimization

Ant Colony Optimization (ACO) is a paradigm for designing meta-heuristic algorithms for combinatorial optimization problems. [Marco, 2004]. The essential trait of ACO algorithms is the combination of a priori information about the structure of a promising solution with a posteriori information about the structure of previously obtained good solutions. ACO is a class of algorithms, whose first member, called Ant System, was initially proposed by Colorni, Dorigo and Maniezzo. The ant colony optimization algorithm is a probabilistic technique for solving computational problems which can be reduced for finding good paths through graphs. The algorithm is based on the behavior of ants seeking a path between their colony and a source of food. Ant is supposed to find the path based on the pheromone update and trail update. The main underlying idea, loosely inspired by the behavior of real ants, is that of a parallel search over several constructive computational threads based on local problem data and on a dynamic memory structure containing information on the quality of previously obtained results. The concept behind the ant colony optimization technique is,

- Ants (blind) navigate from nest to food source
- Shortest path is discovered via pheromone trails
  - each ant moves at random
  - pheromone is deposited on path
  - ants detect lead ant’s path, inclined to follow
  - more pheromone on path increases probability of path being followed

The following steps are carried out in the ant colony optimization process.

- Starting node selected at random
- Path selected at random
  - based on amount of “trail” present on possible paths from starting node
  - higher probability for paths with more “trail”
- Ant reaches next node, selects next path
- Continues until reaches starting node
- Finished “tour” is a solution
- A completed tour is analyzed for optimality
- “Trail” amount adjusted to favor better solutions
• Better solutions receive more trail
• Worse solutions receive less trail
• Higher probability of ant selecting path that is part of a better-performing tour
• New cycle is performed
• Repeated until most ants select the same tour on every cycle (convergence to solution) [Nada, 2009]

Algorithm 4.6 Simple Ant Colony Optimization Algorithm

Initialize Trail

Do While (Stopping Criteria Not Satisfied) – Cycle Loop
  • Do Until (Each Ant Completes a Tour) – Tour Loop
  • Local Trail Update
  • End Do
  • Analyze Tours
  • Global Trail Update

End Do [Vittorio, 2004]

Hiding Technique based on Ant Colony Optimization

The methodology used for developing the ant colony optimization based hiding technique is given in the figure 4.3.
Initialization of sensitive items (ants), number of modifications required, sensitive transactions

Calculate the initial (pheromone value) cost for all the sensitive transactions
Cost(ST_i) = (X_i + Y_i)/2

Select the sensitive item which needs more modification

Choose the highest cost transaction (probability) for modifying the sensitive item S_i from 1 to 0

If S_i is found in the transaction?

// Trail Update
Modify S_i as 1 to 0. Decrease the number of modification of S_i by 1

If Stopping Criterion met?

Yes

Modified Data Set D'

No

Pheromone Update
Algorithm 4.7 Hiding Technique based on Ant Colony Optimization

1. Initialization
   1.1 Consider a transactional database $D$ which consists of $T$ transactions where $D = \{T_1, T_2, \ldots, T_n\}$
   1.2 Each transaction $T_i$ contains a set of items $I$ where $I = \{I_1, I_2, I_3, \ldots, I_m\}$ where $T_i \in I$
   1.3 Initializing the sensitive items $S_i$ where $S_i \in I$
   1.4 Initializing the number of modifications required for each sensitive item

2. Preprocessing
   2.1 Select the transaction which contains a sensitive item where $Y_i = (S_i \text{ in } T_i)$
   2.2 Consider these transactions $Y_i$ are sensitive transactions $(ST_i)$

3. Construction // cost function is computed for all the sensitive transactions
   // pheromone value
   3.1 $\text{Cost}(ST_i) = (X_i + Y_i)/2$
      // $X_i$ represents number of items present in $ST_i$
      // $Y_i$ represents number of sensitive items present in $ST_i$

4. Selection
   4.1 Choose the highest cost transaction $ST_i$ based on $\text{Cost}(ST_i)$
   4.2 Choose the sensitive item $S_i$ which needs more modification
      // The sensitive item which needs more modification is selected
      // All the modification of this sensitive item is completed then only the next sensitive
      // item is selected for modification

5. Modification
   5.1 Verify (if $S_i$ found in $ST_i$) and ($ST_i$ has more than one item)
   5.2 Then modify the sensitive item $S_i$ from ‘1’ to ‘0’
   5.3 {Update}
   5.4 Calculate new cost value for the modified transaction
   5.5 Decrease the number of modification of $S_i$ by 1
   5.6 if (number of modifications of $S_i = 0$) then
   5.7 Select the next sensitive item for modification
5.8 Go to step 5
5.9 Else Go to step 5
5.10 Else ‘no modification’
5.11 goto step 4

6. Terminating
6.1 Ensure all the sensitive items are modified
6.2 Number of modification becomes 0 then the process is completed

7. Exit

Ant Colony optimization based association rule hiding technique first initializes all the sensitive items and the number of modifications required for each sensitive item. In this hiding technique, the sensitive items are considered as ants. During the construction step, the pheromone value is calculated, i.e. calculating the cost of all the sensitive transactions.

$$\text{Cost(ST}_i) = \frac{X_i + Y_i}{2}$$

where $X_i$ represents number of items present in the transaction and $Y_i$ represents the number of sensitive items present in the transaction. The item which requires more number of modifications is selected first and the transaction which has the highest cost is selected for modification. Now the algorithm verifies whether the sensitive item is present in the transaction or not. If the item is present, then it performs the modification from 1 to 0. Now, the new cost of the modified transaction is calculated.

After modification, trail update occurs in the form of decreasing the number of modifications of the sensitive item by 1. Again, the same sensitive item is selected for modification till its number of modification becomes 0. Only after modifying the first sensitive item the next item is selected. This process is repeated until the termination condition is met. After modifying the sensitive items using ant colony optimization algorithm, the modified database ($D'$) is given in Table 4.5.
4.4.3 Tabu Search Algorithm

Tabu search is a mathematical optimization method that belongs to the class of local search. Tabu search enhances the performance of a local search method by using memory structures. Once a potential solution has been determined, it is marked as “taboo” so that the algorithm does not visit that possibility repeatedly. Tabu search is attributed to Fred Glover. Tabu search is a metaheuristic algorithm that can be used for solving combinatorial optimization problems, such as the travelling salesman problem (TSP). [Glover F, 1993]. Tabu search uses a local or neighbourhood search procedure to iteratively move from a solution $x$ to a solution $x'$ in the neighbourhood of $x$, until some stopping criterion has been satisfied. In order to explore the regions of the search space that would be left unexplored by the local search procedure, tabu search modifies the neighbourhood structure of each solution as the search progresses. The solutions admitted to $N^*(x)$, the new neighbourhood, are determined through the use of memory structures. The search then progresses by iteratively moving from a solution $x$ to a solution $x'$ in $N^*(x)$. Perhaps the most important type of memory structure used to determine the solutions admitted to $N^*(x)$ is the tabu list. In its simplest form, a tabu list is a short-term memory which contains the solutions that have been visited in the recent past. Tabu search excludes solutions in the tabu list from $N^*(x)$. A variation of a tabu list prohibits solutions that have certain attributes or prevent certain moves. Selected attributes in solutions recently visited are labeled tabu-active. Solutions that contain tabu-active elements are tabu. This type of short-term memory is also called recency-based memory.
**Algorithm 4.8 Basic Tabu Search Algorithm**

Choose an initial solution $i$ in $S$.

Set $i^* = i$ and $k=0$.

Set $k=k+1$ and generate a subset $V^*$ of solution in $N(i,k)$ such that either one of the Tabu conditions is violated or at least one of the aspiration conditions holds.

Choose best $j$ in $V^*$ and set $i=j$.

If $f(i) < f(i^*)$ then set $i^* = i$.

Update Tabu and aspiration conditions.

If a stopping condition is met then stops. Else go to Step 2.

**Hiding Technique based on Tabu Search**

The methodology used for developing the tabu search based hiding technique is given in the figure 4.4.

In the initialization step, all the sensitive items and the number of modifications required for each sensitive item is initialized. The transactions which contain the sensitive items are considered as sensitive transactions and these are selected for modification. The next step is, to calculate the cost of each transaction.

$$\text{Cost (ST}_i) = (P_i - Q_i)$$

where $P_i$ is the number of items present in the transaction and $Q_i$ is the number of items not present in the transaction. The algorithm randomly selects the transaction and assigns the selected transaction is the best and now this transaction (current) is compared with its neighborhood transaction. In order to find the best between these two, the costs of two transactions are verified. If the cost of the neighborhood transaction is greater than the initial solution, then it assigns the neighborhood transaction as the best one and now this transaction is selected for modification. During modification, the sensitive items are modified in the order of their occurrence. If the particular sensitive item is not present in the selected transaction then it will choose the next item which needs modification. At each modification, the modified transactions are stored in the tabu list. The sensitive items selected for modification are done in a round robin method. The process is repeated until all the sensitive items are modified i.e. until the number of modifications becomes 0. The modified database $D'$ is shown in table 4.6.
Figure 4.4 Flowchart – Hiding Technique based on Tabu Search
Algorithm 4.9 Hiding Technique based on Tabu Search

1. Initialization
   1.1 Initialize all the sensitive items, sensitive transactions, number of modifications required for each sensitive item

2. Construction
   2.1 Generate initial solutions by calculating the cost value of sensitive transactions
   2.2 \[ \text{Cost(ST}_i\text{)} = P_i - Q_i \]
      // \( P_i \) represents number of items present in the transaction
      // \( Q_i \) represents number of items not present in the transaction
   2.3 Consider the current solution is a best solution \( \text{Cost(Best)} \)

3. Search
   3.1 While termination condition not satisfied
   3.2 do
   3.3 Select the neighborhood solution \( S \), \( \text{Cost(S)} \)
   3.4 if \( \text{Cost(Best)} < \text{Cost(S)} \) Then
   3.5 \[ \text{Cost(Best)} = \text{Cost(S)} \]
   3.6 Select the transaction for modification
   3.7 Select the sensitive items one by one
   3.8 if (\( S_i \) is found in the transaction) and (\( \text{ST}_i \) has more than one item)
   3.9 Then (i) Modify the sensitive item \( S_i \) as 1 to 0
   3.10 (ii) Update these modifications in the tabu list
   3.11 Else go to step 3.7
   3.12 Else go to 3.3
   3.13 Repeat the steps until all the modifications become 0

4. Termination
   4.1 Ensure all the sensitive items are modified
   4.2 Number of modification becomes 0 then the process Completed

5. Exit

Table 4.6 - Modified Dataset (\( D' \)) - Tabu Search

<table>
<thead>
<tr>
<th>Items</th>
<th>Tid-list</th>
</tr>
</thead>
</table>
4.4.4 Dummy Items Creation

Dummy items creation is a non-heuristic algorithm which selects the transactions for modifications based on the cost of the items which was given by the data owner. The dummy items creation technique protects sensitive association rules. In addition to this, after modification of the sensitive data items, the algorithm creates dummy items for maintaining the total cost of the database. In the initialization step, all the sensitive items, number of modifications required for each sensitive item, the sensitive transactions and the cost of each item are initialized. Based on the presence of an item the cost of the individual transaction is calculated. The sums of the cost of the individual transactions are added to get the total transaction cost. After this calculation, the sensitive items are selected one by one for modification. The transactions which contain the sensitive item are selected for modification. These transactions are arranged in the decreasing order based on their costs. Consider the first transaction and verify if it has more than one item. Then the sensitive item is modified from 1 to 0. After all the modifications of the sensitive item are carried out, the new cost of the transactions are calculated. The same process is repeated for all the sensitive items and to get the modified database.

The first step of dummy items creation hiding technique is to calculate the cost of individual transactions and the total cost of all the transactions in the modified database. Next step is, to find the difference between the original transaction cost and the modified transaction cost and arrange the differences in decreasing order. Based on the threshold support value (it fixes the number of occurrences as frequent and infrequent) the differences are grouped into several clusters. The average values of these clusters are computed. In order to create dummy items, the clusters are named $d_1, d_2, \ldots, d_n$ ($n$-number of clusters). Now the average value of each cluster is assigned as the cost of the dummy items.

These dummy items are placed in the corresponding transactions. After inserting the dummy items, the cost of the individual transaction and the cost of the total

<table>
<thead>
<tr>
<th></th>
<th>1 3 4 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>2 4</td>
</tr>
<tr>
<td>c</td>
<td>2 4 5 6</td>
</tr>
<tr>
<td>d</td>
<td>1 3 5 6</td>
</tr>
<tr>
<td>e</td>
<td>1 3</td>
</tr>
</tbody>
</table>
transactions are calculated in the modified database (D') to ensure that the total transaction cost is the same as the original dataset (D). The proposed methodology for this technique is represented in the figure 4.5.

![Figure 4.5 Flowchart - Dummy Items Creation Technique](image)

Dummy Items creation technique consists of four important steps.

(i) Sensitive Item Modification
(ii) Calculating the cost for modified data set
(iii) Dummy items creation
(iv) Assigning cost to dummy items

The algorithms used for implementing these techniques are given below.

**Algorithm 4.10 Dummy Items Creation – Main**

1. **Consider a transactional database D which consists of T transactions where**
   
   \[ D = \{T_1, T_2, \ldots, T_n\} \]

2. **Each transaction \( T_i \) contains a set of items I where**
   
   \[ I = \{ I_1, I_2, I_3, \ldots, I_m\} \]
   
   \( T_i \in I \)

3. **Initializing the sensitive items \( S_i \) where**
   
   \( S_i \in I \)
4. Initializing the number of modifications required for each sensitive item
5. Assign the cost of an item \( CI_i = \{ CI_{i1}, CI_{i2}, \ldots, CI_{im} \} \)
6. Calculate the cost of each transaction
   \[ CT_i = \sum_{j=1}^{m} CI_j \]  
   (where \( j = 1 \) to \( m \) and \( i = 1 \) to \( n \))
7. Calculate the total cost of all the transactions
   \[ \text{Total cost} = \sum_{i=1}^{n} CT_i \]
8. Algorithm: sensitive_item_modification()
9. Algorithm: modified_database_cost_cal()
10. Algorithm: dummy_items_creation()
11. Algorithm: new_database()
12. Algorithm: find_sensitive_rule1()
13. End

The algorithm 4.10 considers the transactional data base which consists of set of transactions and items. The cost of each item is assigned by the data owner. With this information, the cost of each transaction is calculated and then the sum of the cost of all the transaction value is calculated and this value is assigned to the total cost of the original transaction database.
Algorithm 4.11 Dummy Items Creation – Sensitive item modification

\[ I_p : l=\text{no. of sensitive items, no. of modifications required for each sensitive items}, \]
\[ n=\text{no. of transactions} \]

\[ O/p : \text{Sensitive items modified, new modified data base } D' \]

1. Consider the sensitive items one by one (sen_items, where \( i = 1 \) to \( l \))
2. // select the transactions for modification
3. Check if (the sensitive item is found in the transaction)
4. Then (i) select the transaction
5. Else (ii) ignore the transaction
6. Consider the selected transactions and arrange these transactions in decreasing order based on their cost
7. Repeat
8. {
9. Check if the transaction contains only a single item
10. Then { no modification}
11. Else
12. {
13. Replace sen_item as 0.
14. Decrease the value of no. of modifications by 1.
15. Calculate the new cost of the transaction
16. }
17. } until (for all the transactions)
18. Verify no.of modification becomes 0 otherwise modify the single item transaction also
19. Repeat the steps 2 to 18 for all the sensitive items

The sensitive item modification algorithm is used for modifying the sensitive items. The sensitive items are considered one by one. First, the transactions which contain sensitive item 1 are selected for modification. Based on the cost, the transactions are arranged in a decreasing order. In order to perform modification, the algorithm verifies whether the transaction contains only a single item or more items. If it contains a single item, then no modification is performed. Otherwise, the sensitive item is modified from 1
to 0. The number of modifications required for sensitive item 1 is decremented by 1. Now the new cost of the transaction is calculated. Then it selects the next transaction. This process is repeated until the number of modification becomes 0. The same process is repeated for all the sensitive items.

Algorithm 4.12 Dummy Item Creation – Cost calculation for modified database

I/p : Transactions, cost of an item, tot-trans-cost
O/p: Modified database cost, diffi, tot-diff
1. Calculate the cost of each transaction in a modified database
   \[ Mod_{CT_i} = \sum_{j=1}^{m} C_{ij} \] (where j=1 to m and i= 1 to n)

2. Calculate the total-transaction cost of the modified database
   \[ Mod_{tot-cost} = \sum_{i=1}^{n} Mod_{CT_i} \] (where j=1 to m and i= 1 to n)

3. Find out the difference between the original transaction cost and the modified transaction cost for all the transactions
   \[ diff_i = CT_i - (Mod_{CT_i}) \]

4. Find out the difference between the original total transaction cost with the modified total transaction cost
   \[ Tot_{diff} = (total_{cost}) - (mod_{tot}_{cost}) \]

After modifying the sensitive items, there is a need to calculate the cost of the modified database. The costs of the items present in the transaction are added to get the cost of the transaction. The costs of all the transactions are added to get the total cost of the modified data base. For each transaction, the difference between the original transaction cost and the modified transaction cost is found. And also, the difference between the original total transaction cost and the modified total transaction cost is found.

Algorithm 4.13 Dummy Items Creation

I/p : Diffi, threshold values
O/p: dummy items
1. Arrange the differences in decreasing order
2. Find out the number of occurrence of an item to become infrequent based on the
given minimum frequency threshold

3. Group the differences according to the occurrence value

4. Create a dummy items for each group i.e. \( d_1, d_2, \ldots, d_k \)

5. Calculate the average value of differences found in each group

6. Assign the average value of each group is the cost of the dummy items

6. Use a dummy item to the relevant transactions based on the group

This dummy items creation algorithm is used for creating dummy items. First step is the differences are arranged in the decreasing order. Then, to find the number of occurrences required for an item to become non-sensitive in the same threshold minimum frequency. The differences are grouped according to the occurrence value. For each group a dummy item i.e. \( d_1, d_2, \ldots, d_k \) is assigned. The costs of dummy items are calculated based on the average value of the differences which are found in each group. The dummy item is created for the corresponding transaction and the value of the dummy item is assigned.

**Algorithm 4.14 Dummy Items Creation - Modified Database**

| I/p : dummy items, cost, modified_database |
| O/p: Cost |
| 1. Add dummy items in the modified database |
| 2. Calculate the transaction cost of the new data base |
| \[ \text{New}_i \text{_} CT = \sum_{j=1}^{m} CI_j \] |
| 3. Calculate the total transaction cost of the new data base |
| \[ \text{New}_i \text{_} \text{total} \_\text{cost} = \sum_{i=1}^{n} \text{CT}_i \] |
| 4. Verify the original database total cost = new data base total cost. i.e |
| \[ \text{Total} \_\text{cost} = \text{New}_i \text{_} \text{total} \_\text{cost} \] |

After creating dummy items, to ensure that the original total transaction cost is equal to the modified total transaction cost. After inserting dummy items, the cost of the new data base transactions are calculated. Then, the total costs of the new data base transactions are also calculated. It is verified whether the total cost of the original data base is equal to the total cost of the new data base. An Eclat algorithm is applied to the
modified data set for generating the association rules for the same support and confidence value. Now, all the performance measures are verified in the modified data set.

**Table 4.7 Modified Dataset (D’) – Dummy Items Creation**

<table>
<thead>
<tr>
<th>Items</th>
<th>Tid-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1 3 4 5</td>
</tr>
<tr>
<td>b</td>
<td>2 6</td>
</tr>
<tr>
<td>c</td>
<td>2 4 5 6</td>
</tr>
<tr>
<td>d</td>
<td>1 3 5 6</td>
</tr>
<tr>
<td>e</td>
<td>2 3</td>
</tr>
<tr>
<td>d₁</td>
<td>1 4</td>
</tr>
<tr>
<td>d₂</td>
<td>3 5</td>
</tr>
</tbody>
</table>

4.5 PERFORMANCE ANALYSIS

The efficiency of the proposed and the existing hiding techniques are analyzed by using various performance measures. The following performance factors are considered for this analysis.

- Hiding failure
- Misses Cost
- Artifactual Errors
- Efficiency

These techniques are implemented in VB.Net 2005. VB.Net is the front end and SQL Server 2005 is the back end. Dataset is collected from the website [www.fimi.ua.ac.be/data/](http://www.fimi.ua.ac.be/data/). Various types of datasets such as mushroom, chess, connect etc. are available in this website. Connect dataset is used for this research work. It consists of 67557 instances and 127 attributes. From this connect data set 1K, 2K, 3K and 5K, instances are used with different sizes of attributes with different thresholds. The experiments are performed on a PC with Intel Core I3 processor with a CPU clock rate of 2.4GHz, 500GB Hard disk and 4 GB RAM running on Windows operating system.

4.5.1 Hiding Failure (HF)

This measure quantifies the percentage of the sensitive patterns that remain exposed in the sanitized dataset. It is defined as the fraction of the restrictive association rules that appear in the sanitized database divided by the ones that appeared in the original dataset. Formally,

$$HF = \frac{|AR_{sen}(D')|}{|AR_{sen}(D)|}$$
where $AR_{sen}(D')$ corresponds to the sensitive rules discovered in the sanitized dataset $D'$, $AR_{sen}(D)$ to the sensitive rules appearing in the original dataset $D$ and $|X|$ is the size of set $X$. Ideally, the hiding failure should be 0%.

**Table 4.8 Hiding Failure Measure - Existing Hiding Techniques**

<table>
<thead>
<tr>
<th>Data Set Size</th>
<th>Thresholds</th>
<th>Existing Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ISL Algorithm</td>
</tr>
<tr>
<td>1000 T 23 I</td>
<td>$\sigma_{10}$ C30</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{20}$ C40</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{30}$ C50</td>
<td>3</td>
</tr>
<tr>
<td>2000 T 30 I</td>
<td>$\sigma_{10}$ C30</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{20}$ C40</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{30}$ C50</td>
<td>2</td>
</tr>
<tr>
<td>3000 T 40 I</td>
<td>$\sigma_{10}$ C30</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{20}$ C40</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{30}$ C50</td>
<td>1</td>
</tr>
<tr>
<td>5000 T 50 I</td>
<td>$\sigma_{10}$ C30</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{20}$ C40</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{30}$ C50</td>
<td>2</td>
</tr>
</tbody>
</table>

The table 4.8 shows the performance of the hiding failure measure for the existing hiding techniques ISL algorithm, DSR algorithm and genetic algorithm. Analyzing these results, the rate of hiding failure is very less in genetic algorithm compared to ISL and DSR algorithms. Existing hiding techniques, hiding failure performance is depicted in figure 4.6.
Figure 4.6 Hiding Failure – Existing Hiding Techniques

Table 4.9 Hiding Failure - Proposed Hiding Techniques

<table>
<thead>
<tr>
<th>Data Set Size</th>
<th>Thresholds</th>
<th>Proposed Hiding Techniques</th>
<th>Modified Genetic Algorithm</th>
<th>Tabu Search</th>
<th>Ant Colony Optimization</th>
<th>Dummy Items Creation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 T 23 I</td>
<td>σ10 C30</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2000 T 30 I</td>
<td>σ10 C30</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3000 T 40 I</td>
<td>σ10 C30</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5000 T 50 I</td>
<td>σ10 C30</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The figure 4.7 depicts the hiding failure measure for the proposed hiding techniques. From the above experimental results, the hiding technique based on tabu search is found to be more reliable than the other hiding techniques.

4.5.2 Misses Cost (MC)

This measure quantifies the percentage of the nonrestrictive patterns that are hidden as a side-effect of the sanitization process. It is computed as follows:

\[
MC = \frac{|AR_{sen}(D)| - |AR_{sen}(D')|}{AR_{sen}(D)}
\]

where \(AR_{sen}(D)\) is the set of all non-sensitive rules in the original database \(D\) and \(AR_{sen}(D')\) is the set of all non-sensitive rules in the sanitized database \(D'\).

<table>
<thead>
<tr>
<th>Data Set Size</th>
<th>Thresholds</th>
<th>ISL Algorithm</th>
<th>DSR Algorithm</th>
<th>Genetic Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 T 23 I</td>
<td>σ10 C30</td>
<td>3</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{10} C_{30}$</td>
<td>3</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>$\sigma_{20} C_{40}$</td>
<td></td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$\sigma_{30} C_{50}$</td>
<td></td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td><strong>3000 T 40 I</strong></td>
<td>$\sigma_{10} C_{30}$</td>
<td>2</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{20} C_{40}$</td>
<td>0</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{30} C_{50}$</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td><strong>5000 T 50 I</strong></td>
<td>$\sigma_{10} C_{30}$</td>
<td>2</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{20} C_{40}$</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>$\sigma_{30} C_{50}$</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

The table 4.10 shows the misses cost measure for the existing hiding techniques ISL algorithm, DSR algorithm and genetic algorithm. The ISL algorithm gives more number of non sensitive rules which is same as the original dataset. Hence, the misses cost measure is very low in ISL.

Figure 4.8 Misses Cost – Existing Techniques
Table 4.11 Misses Cost Performance - Proposed Hiding Techniques

<table>
<thead>
<tr>
<th>Data Set Size</th>
<th>Thresholds</th>
<th>Proposed Hiding Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Modified Genetic Algorithm</td>
</tr>
<tr>
<td>1000 T 23 I</td>
<td>σ10 C30</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>3</td>
</tr>
<tr>
<td>2000 T 30 I</td>
<td>σ10 C30</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>3</td>
</tr>
<tr>
<td>3000 T 40 I</td>
<td>σ10 C30</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>2</td>
</tr>
<tr>
<td>5000 T 50 I</td>
<td>σ10 C30</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4.9 Misses Cost – Proposed Techniques
The figure 4.9 shows the misses cost measure for the proposed hiding techniques. The hiding technique based on Tabu Search gives more number of non sensitive rules which is same as the original dataset. Hence, the misses cost measure is very low in Tabu Search.

4.5.3 Artifactual Patterns (AP)

This measure quantifies the percentage of the discovered patterns that are artifacts. It is computed as follows:

\[
AP = \frac{|P'| - |P \cap P'|}{|P'|}
\]

where \( P \) is the set of association rules discovered in the original database \( D \) and \( P' \) is the set of association rules discovered in \( D' \).

Table 4.12 Artifactual Pattern - Existing Hiding Techniques

<table>
<thead>
<tr>
<th>Data Set Size</th>
<th>Thresholds</th>
<th>Existing Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ISL Algorithm</td>
</tr>
<tr>
<td>1000 T 23 I</td>
<td>σ10 C30</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>4</td>
</tr>
<tr>
<td>2000 T 30 I</td>
<td>σ10 C30</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>6</td>
</tr>
<tr>
<td>3000 T 40 I</td>
<td>σ10 C30</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>7</td>
</tr>
<tr>
<td>5000 T 50 I</td>
<td>σ10 C30</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>7</td>
</tr>
</tbody>
</table>
The figure 4.10 shows the side effect factor artifactual pattern which means fake rule generation measure for the existing hiding techniques. The percentage of fake rules generated by the *hiding technique based on genetic algorithm* is very less compared to ISL and DSR.

**Table 4.13 Artifactual Pattern - Proposed Hiding Techniques**

<table>
<thead>
<tr>
<th>Data Set Size</th>
<th>Thresholds</th>
<th>Proposed Hiding Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Modified Genetic Algorithm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000 T 23 I</td>
<td>σ10 C30</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>2</td>
</tr>
<tr>
<td>2000 T 30 I</td>
<td>σ10 C30</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>2</td>
</tr>
<tr>
<td>3000 T 40 I</td>
<td>σ10 C30</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>3</td>
</tr>
<tr>
<td>5000 T 50 I</td>
<td>σ10 C30</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>3</td>
</tr>
</tbody>
</table>
The table 4.13 shows the side effect factor artifactual pattern which means fake rule generation measure for the proposed hiding techniques. The percentage of fake rules generated by the hiding technique based on Tabu Search is very less compared to other proposed hiding techniques.

![Figure 4.11 Artifactual Pattern – Proposed Techniques](image)

4.5.4 Efficiency

This efficiency category consists of measures that quantify the ability of a privacy preserving algorithm to use efficiently the available resources and execute with good performance. Efficiency is measured in terms of CPU-time needed for modifying the sensitive items in the sensitive transactions. The table 4.14 shows the time required to perform the hiding process of all the existing techniques.
Table 4.14 Efficiency - Existing Hiding Techniques

<table>
<thead>
<tr>
<th>Data Set Size</th>
<th>Thresholds</th>
<th>Existing Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ISL Algorithm</td>
</tr>
<tr>
<td>1000 T 23 I</td>
<td>σ10 C30</td>
<td>7896</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>4129</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>784</td>
</tr>
<tr>
<td>2000 T 30 I</td>
<td>σ10 C30</td>
<td>9265</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>6351</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>927</td>
</tr>
<tr>
<td>3000 T 40 I</td>
<td>σ10 C30</td>
<td>10241</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>5415</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>824</td>
</tr>
<tr>
<td>5000 T 50 I</td>
<td>σ10 C30</td>
<td>11514</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>7895</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>2347</td>
</tr>
</tbody>
</table>

![Figure 4.12 Efficiency – Existing Techniques](http://www.novapdf.com/)
The figure 4.12 shows the efficiency performance factor for the existing hiding techniques. From these results, the **DSR algorithm** requires minimum time for performing modification of the sensitive items.

**Table 4.15 Efficiency - Proposed Hiding Techniques**

<table>
<thead>
<tr>
<th>Data Set Size</th>
<th>Thresholds</th>
<th>Proposed Hiding Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Modified Genetic Algorithm</td>
</tr>
<tr>
<td>1000 T 23 I</td>
<td>σ10 C30</td>
<td>12310</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>7624</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>1346</td>
</tr>
<tr>
<td>2000 T 30 I</td>
<td>σ10 C30</td>
<td>25631</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>21965</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>3251</td>
</tr>
<tr>
<td>3000 T 40 I</td>
<td>σ10 C30</td>
<td>18125</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>7965</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>2453</td>
</tr>
<tr>
<td>5000 T 50 I</td>
<td>σ10 C30</td>
<td>19258</td>
</tr>
<tr>
<td></td>
<td>σ20 C40</td>
<td>10425</td>
</tr>
<tr>
<td></td>
<td>σ30 C50</td>
<td>5610</td>
</tr>
</tbody>
</table>
The figure 4.13 shows the efficiency of the proposed hiding techniques. Analyzing these results, it is observed that the time required for modification is very low in *Tabu Search* based hiding technique.

### 4.6 SUMMARY

In this chapter, the methodology and algorithms of existing hiding techniques such as ISL, DSR and hiding technique based on genetic algorithm and the proposed association rule hiding techniques based on modified genetic algorithm, tabu search, ant colony and dummy items creation are discussed. It also describes various performance measures used for finding the accuracy of each hiding technique.