Expert Systems With Applications 64 (2016) 340–351

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Association rule hiding using cuckoo optimization algorithm

Mahtab Hossein Afshari*, Mohammad Naderi Dehkordi, Mehdi Akbari

Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Isfahan, Iran

ARTICLE INFO

Article history:
Received 25 December 2015
Revised 27 May 2016
Accepted 1 August 2016
Available online 2 August 2016

Keywords:
Data mining
Privacy preserving data mining
Association rule hiding
Cuckoo optimization algorithm

ABSTRACT

Privacy preserving data mining is a new research field that aims to protect the private information and avoid the leakage of this information during the data mining process. One of the techniques in this field is the Privacy Preserving Association Rule Mining which aims to hide sensitive association rules. Many different algorithms with particular approaches have so far been developed to reach this purpose. In this paper, a new and efficient approach has been introduced which benefits from the cuckoo optimization algorithm for the sensitive association rules hiding (COA4ARH). In this method the act of hiding is performed using the distortion technique. Further in this study three fitness functions are defined which makes it possible to achieve a solution with the fewest side effects. Introducing an efficient immigration function in this approach has improved its ability to escape from any local optimum. The efficiency of proposed approach was evaluated by conducting some experiments on different databases. The results of the execution of the proposed algorithm and three of the previous algorithms on different databases indicate that this algorithm has superior performance compared to other algorithms.

1. Introduction

As the use of data mining process to extract useful patterns from massive data is increasing, concerns about the disclosure of private information during this process has also risen. Trying to preserve the privacy of sensitive information while extracting useful patterns led to the formation of a new field in data mining known as privacy preserving data mining (PPDM). PPDM is applied in all data mining techniques such as clustering, classification, association rule. The algorithms of this field prevent the disclosure of private information, while preserving the utility of non-sensitive information as much as possible by modification and distortion of the database. However, manipulating the database in order to protect the confidentiality of sensitive information is not flawless and has some side effects. The side effects include:

- Hiding failure: It occurs when the PPDM algorithms cannot act completely successful and even after data base sanitization, part of the sensitive information could be extracted by data mining algorithms.
- Lost rule or misses cost: sanitization process not only hides sensitive information but also ruins some parts of the non-sensitive data which cannot be extracted any more.
- Ghost rule or artificial pattern: Because of the sanitization process, new and unrealistic patterns may form in database that did not exist before.

So far, many algorithms have been introduced in this area, each of which has different strengths and weaknesses. The conducted activities in privacy preserving association rule mining until the present time can be divided into three main categories: border-based (Moustakides & Verykios, 2006, 2008; Sun & Philip, 2007; Sun & Yu, 2005), exact (Gkoulalas-Divanis & Verykios, 2006, 2009a, 2009b; Menon & Sarkar, 2007; Menon, Sarkar, & Mukherjee, 2005) and heuristic approach (Dasseni, Verykios, Elmagarmid, & Bertino, 2001; Oliveira & Zaïane, 2002, 2003a, 2003b, 2006; Saygin, Verykios, & Clifton, 2001; Saygin, Verykios, & Elmagarmid, 2002; Verykios, Pontikakis, Theodoridis, & Chang, 2007; Wu, Chiang, & Chen, 2007). In recent years, meta heuristic algorithms have been employed for privacy preserving association rule mining. One of the newest meta heuristic algorithms is cuckoo optimization algorithm (Walton, Hassan, Morgan, & Brown, 2011; Yang & Deb, 2013).

Considering the above-mentioned facts, the main concern and objective of this study is to use cuckoo optimization algorithm to propose a new algorithm in privacy preserving association rule mining area able to hide the sensitive information completely successful and with minimal side effects by

- Not having Hiding failure
- Minimizing ghost rule amount
- Decreasing lost rule impressively

* Corresponding author.
E-mail addresses: mahtab.h.afshar@gmail.com, Afshari@sco.iaun.ac.ir (M.H. Afshari), Naderi@iaun.ac.ir (M.N. Dehkordi), mehdi_akbari@hotmail.com, mehdi_akbari@pco.iaun.ac.ir (M. Akbari).

http://dx.doi.org/10.1016/j.eswa.2016.08.005
0957-4174/

The rest of the article is organized as follows: Section 2 contains a review of literatures conducted in the field of privacy preserving association rule mining and a brief explanation about the Cuckoo Optimization Algorithm (COA). In Section 3, the problem of hiding sensitive association rules are clearly explained. Section 4 then describes the proposed algorithm for hiding association rules and explains it in details by providing an example. In Section 5, the results of conducted experiments on Real and Synthetic data sets are provided and the efficiency of proposed approach will be compared with some existing algorithms. The effects of sparse data are described in Section 6. Finally, in Section 7 results and future works will be discussed.

2. Background and related work

There have been so far many studies conducted in privacy preserving association rule mining, some of which are briefly reviewed in this section.

The subject of association rules hiding from frequent item sets through decrease of support was first suggested by Attallah et al. that used a lattice-like graph for hiding. Appropriate time complexity was one of the advantages of the proposed method (Atallah, Bertino, Elmagarmid, Ibrahim, & Verykios, 1999). Five algorithms namely 1.a, 1.b, 2.a, 2.b and 2.c are provided in Verykios, Elmagarmid, Bertino, Saygin, & Dasseni, (2004) by Verykios et al. The three first algorithms are Rule-Oriented and the next two are Itemset-Oriented. Removing just one sensitive rule at a time is one of the disadvantages of aforementioned algorithms. By removing the right hand side of the sensitive rule, algorithm 1.b decreases the rule support and hides it. In algorithm 1.b, along with an increase in the number of sensitive rules, the number of lost rules is also increased. Multiple rule hiding was first proposed by Oliveira &Zaïane. Their suggested algorithms were Naïve, MinFIA, MaxFIA and IGA that hide sensitive rules based on the conflict level and were not efficient with regard to implementation time (Oliveira & Zaïane, 2002). The two algorithms ISL and DSR were introduced by Wang in Wang, Parikh, & Jafari, (2007). The former conducts the hiding process by increasing the support of the left hand side item of sensitive rules while the latter conducts the hiding process by decreasing the support of the right hand side item of sensitive rules. ISL and DSR algorithms are respectively weak in the hiding failure rate and the number of lost rules. The genetic algorithm for hiding sensitive rules is suggested by Dehkordi and et al. in Dehkordi, Badie, & Zadeh, (2009). Besides hiding sensitive data, the main purpose of this algorithm is to decrease modifications in the databases. A preprocess phase for selection of sensitive transactions of database and making changes in them is defined in this algorithm. Four different strategies are also provided in order to calculate Fitness. By introducing a new Fitness function, Azam Khan et al. in Khan, Qureshi, & Hussain, (2014) have improved the proposed algorithm in Dehkordi et al., (2009).

Cuckoo Algorithm was first developed by Yang & Deb, (2009) and then was investigated in details by Rajabioun, (2011). Interesting and different lifestyle as well as egg-laying of cuckoo has been the main influence for development of this algorithm. This bird is able to brilliantly deceive other birds and make them participate in its own survival. Unlike other birds, cuckoo never builds nests and never protects its eggs, but puts its eggs in the nests of other birds to be protected along with other eggs.

Like other evolutional algorithms, COA also begins its work from a random initial population which is called "*habitat*" and it is formed by *cuckoos*. A habitat is expressed by an array of $1\times Nvar$, showing current living position of cuckoo. This array is identified by Eq. (1)

$$habitat = [x_1, x_2, \ldots, x_{N_{var}}], \tag{1}$$

where N_{var} is the dimension of the problem.

Each cuckoo is allocated some random eggs that will be put in the nests of a number of host birds. The nests of host birds exist within a certain range that is called maximum Egg Laying Radius (*ELR*) which is identified by Eq. (2)

$$ELR = \left[\propto \times \frac{Number\ of\ current\ cuckoo's\ eggs}{Total\ number\ of\ eggs}\right] \times (Var_{hi} - Var_{low}), \tag{2}$$

where $\propto$ is a number, intended to handle the maximum value of *ELR*. Var_{hi} and Var_{low} are respectively high and low limit of each variable in optimization problems.

A number of cuckoo's eggs with more similarity to the host eggs will have more chance to turn into a mature cuckoo. Other eggs (about 10%) are identified by the host bird and removed. The more the number of survived eggs in one region, the more benefit allocated to that region. Therefore, a situation in which the largest number of eggs survives will be a parameter that COA aims to optimize. Survived chicks are grown in the host's nest and will turn into mature cuckoos. At the time of egg-laying, cuckoos migrate to better and more beneficial habitats with high chance of egg survival. When migrating, cuckoos do not traverse the whole course toward the target point and pass a part of it (λ% of total path) toward the target and then have a ϕ deviation. This parameter helps them to search for more areas. After arriving to the target place, each cuckoo owns a number of eggs according to which it's *ELR* is determined and the egg-laying is started. Considering the fact that there are always such factors as food shortage, hunting by hunters, etc. in the nature that causes balance in the number of birds, there is also a number like N_{max} in Cuckoo Algorithm that can control the maximum number of cuckoos. After several iterations, cuckoos reach a point with maximum similarity of eggs with the host eggs and with maximum food resources. It is a place with the highest total benefit and lowest number of eggs being destroyed. For more detail on COA, refer to Rajabioun, (2011).

3. Problem definition

Assume that $I = \{i_1, i_2, i_3, \ldots, i_m\}$ is a complex of items and D is a database including several transactions. Each transaction is a subset of I. The extracted association rules set from D is R and sensitive association rules set is R_s, where $R_s \subset R$. Each association rule is expressed as $A \rightarrow B$ where A is antecedent or left hand side and B is consequent or right hand side, so that $A, B \subset I$ and $A \cap B \neq \emptyset$. The purpose is to change D into D', so that the existing rules in R_s cannot be mined from D'while the existing rules in $R - R_s$ can be possibly mined. Two criteria are considered in the association rule mining; the first is called item support that indicates frequency of one rule in database and can be calculated by Eq. (3):

$$Sup(A \rightarrow B) = \frac{|A \cup B|}{|D|}, \tag{3}$$

where $Sup(A \rightarrow B)$ is the support of $A \rightarrow B$, $|A \cup B|$ is the number of transactions that include both item sets A and B and $|D|$ is the total number of transactions.

The second criterion is called the rule confidence that indicates the rule strength in database and can be calculated by Eq. (4):

$$Conf(A \rightarrow B) = \frac{|A \cup B|}{|A|}, \tag{4}$$

where $Conf(A \rightarrow B)$ is the confidence of $A \rightarrow B$, $|A \cup B|$ is the number of transactions that include both item sets A and B and $|A|$ is the number of transactions of database that include item set A.

For each association rule, one Minimum Support Threshold (*MST*) and one Minimum Confidence Threshold (*MCT*) are defined.

Table 1
Definitions of notations.

Notation	Definition
I	An itemset
D	An original dataset
D'	A sanitized dataset
R	The extracted association rules set from D
R'	The extracted association rules set from D'
R_s	A set of sensitive association rules
$\sim R_s$	A set of non-sensitive association rules
r	An association rule
$A \rightarrow B$	An association rule
$Sup(A \rightarrow B)$	The support of $A \rightarrow B$
$Conf(A \rightarrow B)$	The confidence of $A \rightarrow B$
MST	The minimum support threshold
MCT	The minimum confidence threshold
Var_{hi}	The high limit of each variable
Var_{low}	The low limit of each variable
α	The number of total iterations of right hand side items of sensitive rules
K	The number of new solutions
N	The number of current solution's items that should be modified
N_{max}	The maximum number of survived solutions with better fitness values
N_{var}	The dimension of problem
N_{pop}	The size of initial population
$MaxNNS$	The max number of new solutions
$MinNNS$	The min number of new solutions
MR	The modifications radius
HF	The hiding failure
LR	The lost rules
GR	The ghost rules
HD	The hiding distance
LD	The lost distance
RHD	The rules hiding distances
RLD	The rules lost distances

A rule is minable when its support is more than *MST* and its confidence is higher than *MCT*. Therefore, in order to hide a rule it is required to reduce its support or confidence to a level under the thresholds. Table 1 shows a list of notations and their definitions used in this paper.

4. Proposed algorithm

The main steps of the proposed algorithm named Cuckoo Optimization Algorithm for Association Rule Hiding (COA4ARH) are shown in Algorithm 1 and Fig. 1 shows the flowchart of the algorithm. Fitness function, function to find the best solution and immigration function are shown in Algorithms 2, 3 and 4 and these algorithms will be discussed in detail in the next sections.

4.1. Preprocess of original dataset

The first step of the proposed algorithm includes a preprocess operation on the original dataset which is one of the most important and influential phases of the proposed approach. Since the sensitive items exist only in some transactions of the database, changing and manipulating all transactions is a time-consuming and useless task that not only increases the size of habitats, but also causes the production of useless and unrelated solutions and an increase in the time consumed for obtaining more effective solutions; hence increasing the sanitization time. To avoid such negative effects, a preprocess operation have been considered that includes two phases. In the first phase, the original database is processed in such a way that only critical transactions of the database are chosen. In the present study, critical transaction is a transaction that fully supports one or more sensitive rules. In the second phase of the preprocess operation only those sensitive items with critical role in sanitization, are addressed for change. Sensitive item is defined as follows:

1 If a sensitive rule has only one item in its right hand side, the same item will be selected as the sensitive item.
2 If a sensitive rule has more than one item in its right hand side, from among them an item will be chosen as a sensitive one that has respectively the most frequency in the right hand side of sensitive rules and the least frequency in non-sensitive rules.

Therefore, conducting preprocess operation on the original database will lead to decrease in size of habitats, reduction of unnecessary changes in database, prevention of producing irrelevant solutions and speedier access to an optimal solution.

4.2. Generating initial population

In order to solve an N_{var} dimension problem with the size of N_{pop}, the Cuckoo Optimization Algorithm randomly generates an initial population in the form of a habitat matrix with the size of $N_{pop} \times N_{var}$. Each member of this population shows the current habitat of cuckoos. Since each habitat is allocated to one cuckoo, hereinafter for comfort, *"cuckoo's habitat"* is replaced with contracted form of *"cuckoo"*. In the present problem, each cuckoo in initial population is indicative of a solution (sanitized database) which is shown with a sequence of 0 s and 1 s. In this sequence, the presence and absence of one item in transaction are respectively presented with 1 and 0. The original database is considered as the first cuckoo or solution in initial population. Indeed, the first solution of initial population is a sequence of critical transactions of original database that were selected in the preprocess operation. The random generation of the other $(N_{pop}-1)$ solutions can be conducted as such: only those sensitive items that had been addressed in preprocess operations are randomly quantified (0 or 1) and other items are left unchanged from the first solution (original database) and are transferred to other solutions. Thus, an initial population with N_{pop} number of solutions is generated. Each

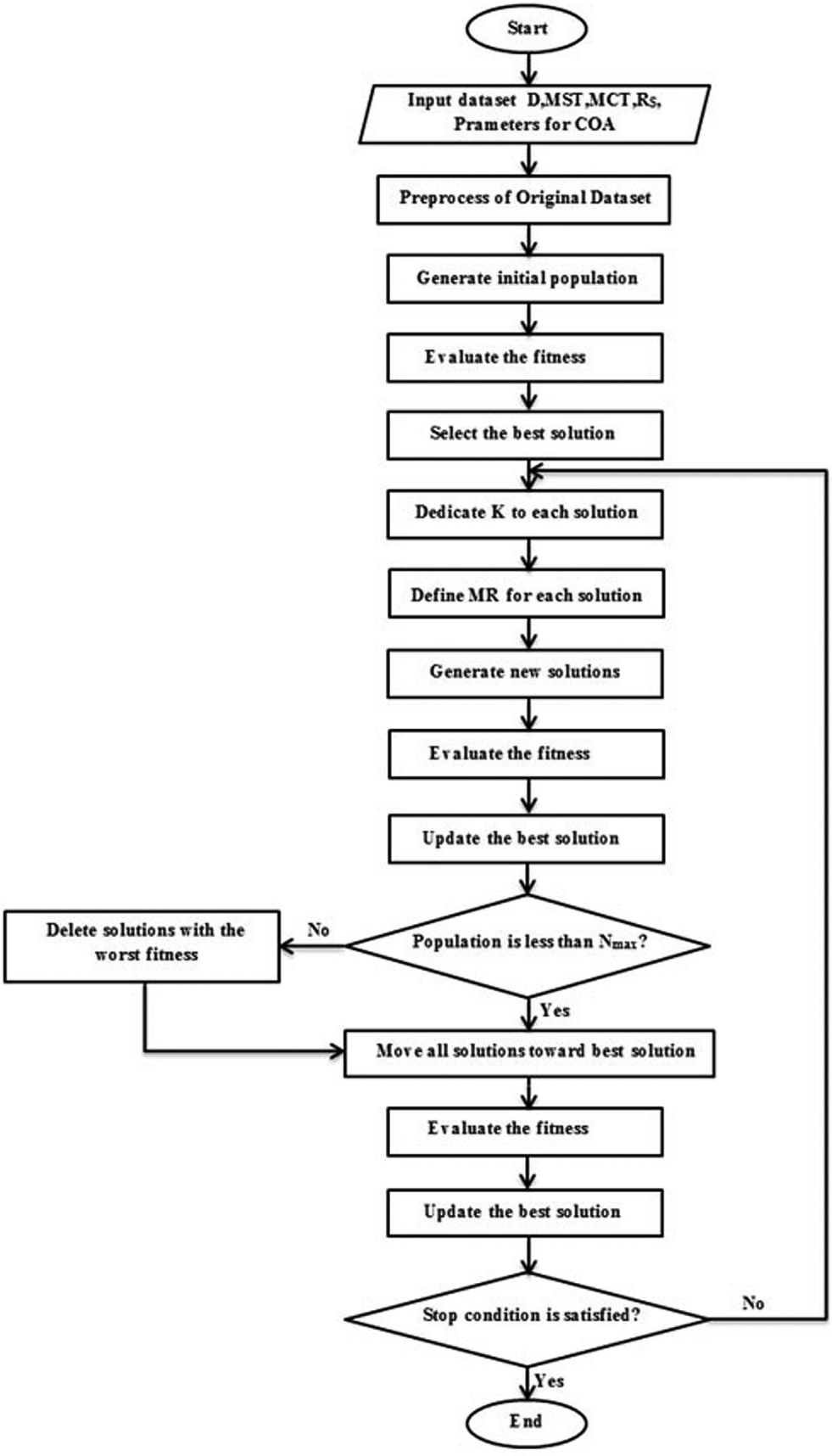


Fig. 1. Flowchart of COA4ARH algorithm.

Algorithm 1
COA4ARH algorithm.

```
Input:
    Original Dataset D, R_s, MST and MCT;
    α, N_pop, N_max, MinNNS, MaxNNS, MaxIteration;
Output:
    A Sanitized Dataset D';
begin
    Preprocess of Original Dataset;
    Generate an initial population;
    call FitnessFunction;
    call BestSolutionFunction;
    repeat
      // Generate new solutions
      for each solution in population
        K = (Max NNS − Min NNS ) × Rand[0, 1] + Min NNS; //Dedicate K
        MR = [∝ × Current solution's K / Totall of all solution's K] × (Var_hi − Var_low); //Define MR
        for K times do
          for MR times do
            Select an item randomly from among the sensitive items;
            Set Rand[0,1] to the selected item;
          end for
        end for
      end for
      call FitnessFunction;
      call BestSolutionFunction;
      Limit number of solutions to N_max;
      for each solution in population
        call ImmigrationFunction; // Move all solutions toward the best solution
      end for
      call FitnessFunction;
      call BestSolutionFunction;
    until the termination condition is satisfied;
end.
```

solution is an array the length of which equals the total length of critical transactions.

4.3. Evaluate the fitness

In this part, fitness values are calculated for each existing solution in the initial population. To do so, it is necessary that each solution is merged with that part of the database which was separated in the preprocess (non-critical transactions) so that sanitization effect is considered for the whole database, and acquired fitness values would be representative of overall state of the database. It should be mentioned that calculation of fitness values is not necessary for the first solution in the initial population; for, as noted in the previous section, the first solution is the original database which was provided (unchanged) in the initial population. The fitness values for other solutions are calculated by Eqs. (5), (8) and (11) which come as below:

$$minfit1 = |HF|, \tag{5}$$

where $|HF|$ is the number of hiding failure which is identified by Eq. (6)

$$|HF| = \sum_{i=1}^{|R_s|} r_i state, \tag{6}$$

where $|R_s|$ is the number of sensitive rules and r_i *state* is calculated by Eq. (7)

$$r_i\ state = \begin{cases} 0 & if\ Sup(I_i) < MST\ or\ Conf(r_i) < MCT \\ 1 & otherwise \end{cases}, \tag{7}$$

where I_i is the itemset of r_i.

The second fitness value is computed by Eq. (8)

$$minfit2 = |LR|, \tag{8}$$

Algorithm 2
Fitness function.

```
Begin
    for each solution in population
      Merge with noncritical transactions;
      Calculate minfit1; // Eq. (5)
      Calculate minfit2; // Eq. (8)
      Calculate minfit3; // Eq. (11)
    end for
end.
```

where $|LR|$ is the number of lost rules which is defined by Eq. (9)

$$|LR| = \sum_{i=1}^{|\sim R_s|} r_i state, \tag{9}$$

where $|\sim R_s|$ is the number of non-sensitive rules and r_i *state* is calculated by Eq. (10)

$$r_i\ state = \begin{cases} 0 & if\ Sup(I_i) \geq MST\ and\ Conf(r_i) \geq MCT \\ 1 & otherwise \end{cases}. \tag{10}$$

The third fitness value is identified by Eq. (11)

$$minfit3 = RHD + RLD, \tag{11}$$

where *RHD* is Rules Hiding Distances, *RLD* Rules Lost Distances and *RHD* is calculated by Eq. (12)

$$RHD = \sum_{i=1}^{|R_s|} r_i HD, \tag{12}$$

r_i *HD* is defined by Eq. (13)

$$r_i\ HD = \begin{cases} 0 & if\ Sup(I_i) < MST \\ 0 & else\ if\ Conf(r_i) < MCT \\ Conf(r_i) - MCT + 1 & otherwise \end{cases}, \tag{13}$$

RLD is computed by Eq. (14)

$$RLD = \sum_{i=1}^{|\sim R_s|} r_i LD, \tag{14}$$

and r_i *LD* is calculated by Eq. (15)

$$r_i\ LD = \begin{cases} 0 & if\ Sup(I_i) \geq MST\ and\ Conf(r_i) \geq MCT \\ MCT - Conf(r_i) & else\ if\ Conf(r_i) < MCT \\ MST - Sup(I_i) & otherwise \end{cases}. \tag{15}$$

The process of mining of all existing solutions in a population, in each iteration is a very time consuming operation. Hence the functions *minfit*1, *minfit*2 and *minfit*3 have been defined in such a manner where the mining of the solutions is not necessary. In fact, these functions are able to determine the number of hiding failures and lost rules for each solution without the need of data mining. As a result in COA4ARH, data mining is performed only twice. The first time, the original database is mined in order to determine the sensitive and non-sensitive rules. The second time the final sanitized database is mined so that except for hiding failures and lost rules, the number of ghost rules can also be defined. Fitness function is shown in Algorithm 2.

4.4. Select the best solution

After calculating fitness of each solution it is necessary to compare all fitness values in order to select the best solution. The first solution in the initial population does not have any fitness value and hence will not be participated in the comparison. In order to select the best solution, all of the solutions are first sorted in

Algorithm 3
Best solution function.

```
Begin
    Sort the solutions in increasing order of their minfit1;
    if two or more solutions have same minfit1 then
      Sort those in increasing order of their minfit2;
      if two or more solutions have same minfit2 then
        Sort those in increasing order of their minfit3;
      end if
    end if
    set the first member of sorted list as the best solution;
    if (the fitness values of the best solution ≤ the fitness values of the
  previous best solution)
      return best solution;
    else
      return previous best solution;
end.
```

Algorithm 4
Immigration function.

```
Input:
    Current Solution, Best Solution;
    // Current Solution refers to every and each one of the existing
  solutions in a population;
    // Best Solution is a solution with the best fitness value so far;
Output:
    Next Solution;
    // Next Solution is a solution close to the best solution;
begin
    Number of Different Items= Hamming distance (Current Solution, Best
  Solution);
    N= Rand (0, 1)× Number of Different Items;
    // N is the number of Current Solution's items that should be modified;
    Selected Items= Select randomly N items from Different Items of Current
  Solution;
    Next Solution= 1 XOR each Selected Items in Current Solution;//Reduce
  hamming distance;
end
```

rising order based on the value of *minfit*1 function. Then, those with equal values of *minfit*1 will be sorted in rising order based on *minfit*2 and finally, if two solutions are equal in the value of *minfit*2, they will be sorted in rising order based on *minfit*3. The result would be a sorted list of solutions, the first member of which will be selected as the best solution. Then, the best gained solution in this step and the best solution of the previous one will be compared in order for the global best solution to be determined. Best solution function is shown in Algorithm 3.

4.5. Generating new solutions

Each existing solution should generate a number of new solutions. Therefore, according to Eq. (16), a random K number is dedicated to each solution. It is representative of the number of new solutions that should be generated by each parent solution.

$$K = (Max\ NNS - Min\ NNS\ ) \times Rand[0,1] + Min\ NNS, \tag{16}$$

where K is the number of new solutions that should be generated by each parent solution, *Max NNS* is Maximum Number of New Solutions (determined by user) and *Min NNS* is Minimum Number of New Solutions (determined by user).

For generating new solutions, each parent solution is only allowed to change a limited number of its items or bits. This number is called Modifications Radius which is shown with *MR*.The value of *MR* for each parent solution is calculated by Eq. (17).

$$MR = \left[\propto \times \frac{Current\ solution's\ K}{Totall\ of\ all\ solution's\ K} \right] \times (Var_{hi} - Var_{low}), \tag{17}$$

where *MR* is the number of each parent solution's items that could be changed and $\propto$ shows the number of total iterations of right hand side items of sensitive rules in critical transactions, received as algorithm input. Var_{hi} and Var_{low} are respectively high and low limit of each variable in optimization problems. Considering the fact that in the present problem high and low limits of the variables are respectively determined with 1 and 0, the $(Var_{hi} - Var_{low})$ can be removed from the above equation.

Given the value of K as well as the specified *MR* for each parent solution, new solutions are being developed. The process of generating each new solution is as follows:

- For each parent solution, a number (equal to it's *MR*) of items are randomly selected from among the sensitive items and randomly quantified (0 or 1).
- Other items are transmitted from the parent solution into the new solution without any change.
- These steps are iterated K times for each parent solution to which the value K is allocated.

4.6. Limit solutions' maximum number

The total number of the existing solutions should always be controlled in order not to exceed the N_{max} threshold. For this reason, in each iteration, a number of worst solutions must be removed from the end of the sorted list, so that the number of the existing solutions would remain equal to N_{max}.

4.7. Immigration

In order to improve the remaining solutions, with the purpose of improving the ability of the proposed algorithm to escape from local optimums and finding better solutions compared to what is already gained, all solutions are changed in a way to be more similar and closer to the best solution. This change can lead to the generation of better solutions compared to the best current solution. Nevertheless, the best current solution will be considered as the best global solution. Algorithm 4 shows this process.

In order to clarify the functionality of this algorithm, an example has been offered in Fig. 2.

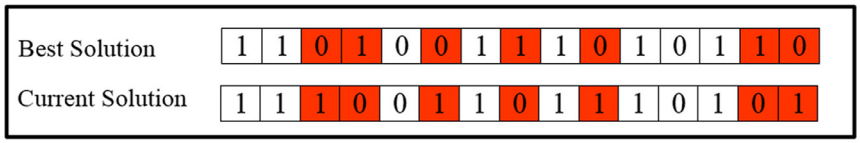
The Hamming distance is defined as the number of positions at which the corresponding symbols are different between two strings of equal length. For binary strings ***a*** and ***b*** the Hamming distance is equal to the number of ones in ***a*** XOR ***b*** (Hamming, 1950).

Based on this definition, the number of different items between the current solution and the best solution is equal to 7 which is illustrated in red as shown in Fig. 2(a). The value of N can now be determined which is equal to a random number of different items, for example 3. This means that 3 items of different items of the current solution should be modified. Hence from those 7 items, 3 items will be chosen randomly. These items have been illustrated in blue in Fig. 2(b).

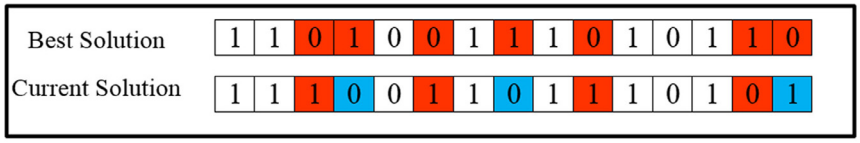
The value of each selected item should then be modified (the value of selected item XOR 1). In this way, the next solution is generated which is more similar to the best solution where the number of non-equal items between them have decreased. The next solution is illustrated in Fig. 2(c).

4.8. Case study

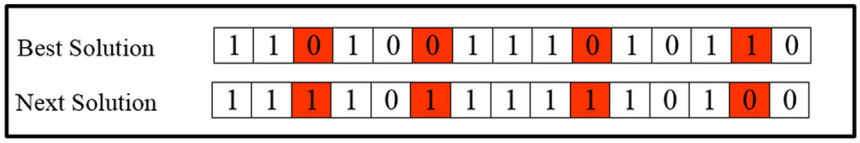
In this part an example is provided for a better understanding of the proposed algorithm. Assume an original database including 10 transactions that is shown in Table 2. Considering MST = 40 and MCT = 70, the mined rules from the database are shown in Table 3. Also assume that the set of sensitive rules is defined as $\{b \rightarrow c, b \rightarrow ce\}$ to be hidden.



(a) Different Items of the Best Solution and a Current Solution



(b) Randomly selection of N items from Different Items of Current Solution



(c) Modification of Selected Items and generation of Next Solution

Fig. 2. Moving a solution toward the best solution.

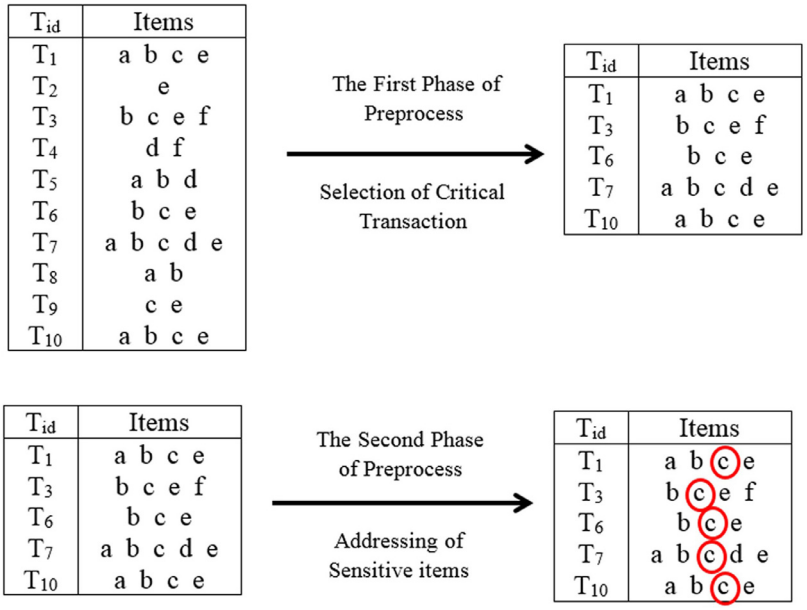


Fig. 3. Preprocess of original dataset.

Table 2
Original dataset.

T_{id}	Items
T_1	a, b, c, e
T_2	e
T_3	b, c, e, f
T_4	d, f
T_5	a, b, d
T_6	b, c, e
T_7	a, b, c, d, e
T_8	a, b
T_9	c, e
T_{10}	a, b, c, e

The preprocess operation on the original database is shown in Fig. 3.

Assume that the user has determined the N_{pop} as 4 cuckoos. Therefore, a random initial population can be shown as Fig. 4.

As seen in Fig. 4, the first solution is the critical transactions of the original database. The fitness values for other solutions after merging with non-critical transactions, can be calculated by Eqs. (5), (8) and (11). For example the fitness values for C2 are shown as follow:

$$\text{minfit1}(C2) = 0 \text{minfit2}(C2) = 8 \text{minfit3}(C2) = 2.56$$

After fitness calculation, the sorted list of solutions is like the following.

$$\text{SortedList} = \{C4, C2, C3\}$$

The first member of the list (C4) is determined as the best solution. Next stages are continued with determination of *MR* and allocation of *K* value to each solution and also generation of new solutions.

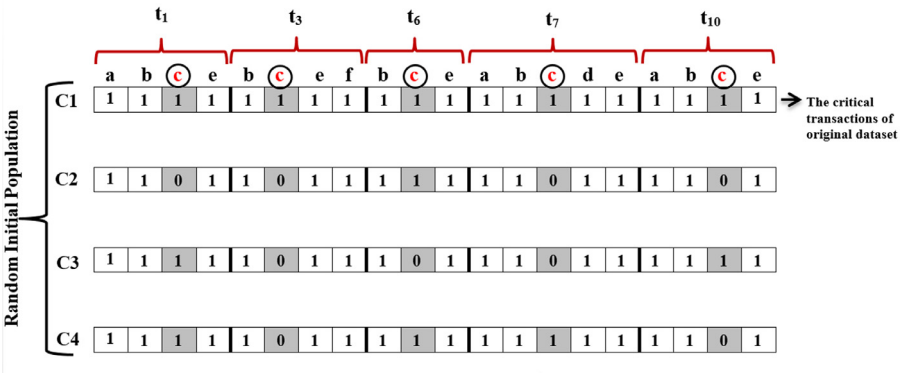


Fig. 4. Random initial population.

Table 3
Association rules extracted from original dataset with MST = 40% and MCT = 70%.

Rule	Support	Confidence
$a \rightarrow b$	50	100
$b \rightarrow a$	50	71.43
$b \rightarrow c$	50	71.43
$c \rightarrow b$	50	83.33
$b \rightarrow e$	50	71.43
$e \rightarrow b$	50	71.43
$c \rightarrow e$	60	100
$e \rightarrow c$	60	85.71
$b \rightarrow ce$	50	71.43
$c \rightarrow be$	50	83.33
$e \rightarrow bc$	50	71.43
$ce \rightarrow b$	50	83.33
$be \rightarrow c$	50	100
$bc \rightarrow e$	50	100

Assume that the user has determined the *MaxNNS* and *MinNNS* as follow:

$$MaxNNS = 3,$$
$$MinNNS = 1,$$

so the random values of K will be calculated as follow:

$$K(C1) = (3-1) \times rand[0,1] + 1 = 2,$$
$$K(C2) = (3-1) \times rand[0,1] + 1 = 1,$$
$$K(C3) = (3-1) \times rand[0,1] + 1 = 1,$$
$$K(C4) = (3-1) \times rand[0,1] + 1 = 2,$$

and the value of MR for each parent solution is calculated as follow:

$$\alpha = 10,$$
$$MR(C1) = \left[10 \times \tfrac{2}{6}\right] = 3,$$
$$MR(C2) = \left[10 \times \tfrac{1}{6}\right] = 2,$$
$$MR(C3) = \left[10 \times \tfrac{1}{6}\right] = 2,$$
$$MR(C4) = \left[10 \times \tfrac{2}{6}\right] = 3,$$

then the new solutions are generated as shown in Fig. 5. The items which are randomly changed are highlighted with black.

The fitness values for all new solutions are calculated (after merging with non-critical transactions) and sorted by the fitness values. The sorted list represented as follow:

$$\text{SortedList} = \{C11, C31, C12, C21, C41, C42\}$$

The first member of the list (C11) is determined as the best solution and compared with the previous best solution (C4). Since both of them have equal fitness values, the best solution still remains C11. Assume that the user has determined the N_{max} as 4, hence 2 number of the worst solutions are deleted from the end of the sorted list.

$$\text{SortedList} = \{C11, C31, C12, C21\}.$$

In the next step, the Number of Deferent Items and N are calculated as follow:

Number of Diferent Items = Hamming Distance (C31, C11) = 4,
N = Rand (0, 1) × 4 = 1
Number of Diferent Items = Hamming Distance (C12, C11) = 3,
N = Rand (0, 1) × 3 = 2
Number of Diferent Items = Hamming Distance (C21, C11) = 2,
N = Rand (0, 1) × 2 = 1

Immigrated solutions toward the best solution (C11) is shown in Fig. 6 . The changed items are illustrated in black.

In this stage again each immigrated solution is merged with non-critical transactions and the fitness values are calculated for each of them, eventually the sorted list of solutions is like the following.

$$\text{SortedList} = \{C'12, C11, C'21, C'31\}$$

The first member of the list (C'12) will be selected as the best solution and compared whit C11 (the previous best solution), finally C'12 is determined as the best solution in the first iteration of the algorithm.

Next iteration is started with determination of MR and allocation of K value to each solution and also generation of new solutions. The sanitized dataset after 2 iterations is shown in Table 4.

Table 4
Final sanitized dataset.

T_{id}	Items
T_1	a, b, c, e
T_2	e
T_3	b, c, e, f
T_4	d, f
T_5	a, b, d
T_6	b, c, e
T_7	a, b, c, d, e
T_8	a, b
T_9	c, e
T_{10}	a, b, e

5. Performance evaluation

In order to evaluate the efficiency of proposed algorithm in sanitization of Real and Synthetic databases, some experiments were

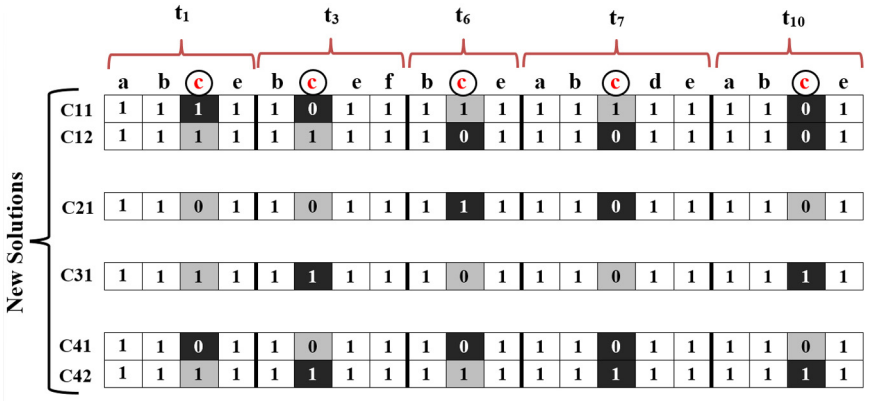


Fig. 5. Generate new solutions.

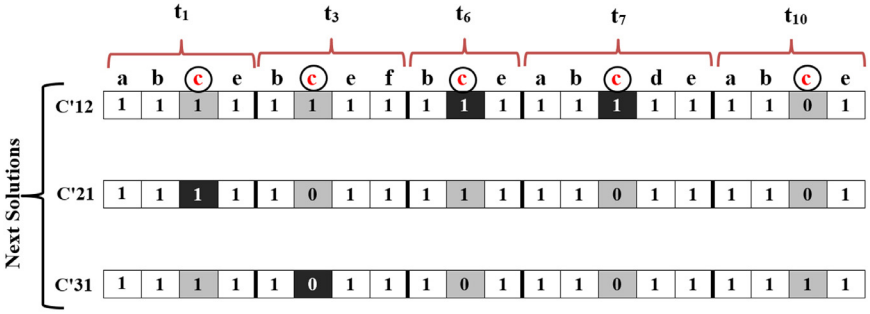


Fig. 6. Immigrate toward the best solution.

Table 5
Database characteristics.

Database	No. of transactions	Avg. trans. length	No. of items
Mushroom	8124	23	119
Chess	3196	37	75
T100	100	19	37

conducted on the proposed algorithm and three others namely 1.b(Verykios et al., 2004), DSR(Wang et al., 2007), and the Improved Genetic (Khan et al., 2014) algorithms, which are discussed here.

5.1. Datasets

Three databases were used in the experiments. Chess and Mushroom databases were used as real data and T100 database was used as synthetic data. Characteristics of the specified datasets are presented in Table 5.

5.2. Performance measures

The efficiency criteria that will be used in comparison of the proposed algorithm with 1.b, DSR and Improved Genetic algorithms include:

- ***Hiding Failure* (*HF*)**: indicates the number of sensitive rules which sanitization algorithm could not hide and are still mined from the sanitized database D'. It is calculated by Eq. (18).

$$HF = \frac{|R_S(D')|}{|R_S(D)|}, \tag{18}$$

where, $|R_S(D')|$ is the number of sensitive rules explored in the sanitized database D' and $|R_S(D)|$ is the number of sensitive rules explored in the original database D.

- ***Lost Rules* (*LR*)**: indicates the number of non-sensitive rules that are lost due to the act of sanitization and will not be mined from the sanitized database D'. It is calculated by Eq. (19).

$$LR = \frac{|\sim R_S(D)| - |\sim R_S(D')|}{|\sim R_S(D)|}, \tag{19}$$

where $|\sim R_S(D)|$ is the number of non-sensitive rules explored in the original database D and $|\sim R_S(D')|$ is the number of non-sensitive rules explored in the sanitized database D'.

- ***Ghost Rules* (*GR*):** indicates the number of artificial rules which were not within the original database and are generated due to the act of sanitization and are mined from database D'.It is calculated by Eq. (20).

$$GR = \frac{|R'| - |R \cap R'|}{|R'|}, \tag{20}$$

where $|R'|$ is the number of mined rules from D' and $|R|$ is he number of mined rules from D.

- ***Number of Iterations:*** the number of iterations required to attain the optimal solution.

5.3. Experimental results

In conducting the experiments, all the algorithms were implemented with C# in a same platform. The experiments were performed on a PC with 2.20 GHz @Intel core i7 CPU and 6 GB of RAM under the Windows 8 (64-bit). The results are shown in Figs. 7, 8 and 9.

As it is illustrated in Figs. 7, 8 and 9 in all three data bases, the value of the *HF* in the proposed algorithm is equal to zero. This is due to an appropriate definition of the fitness function and also high prioritizing of the *minfit*1 function. Further by not using the

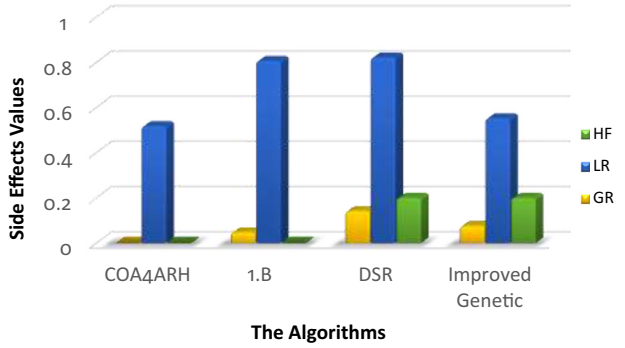


Fig. 7. Comparative evaluation of the algorithms on Mushroom dataset.

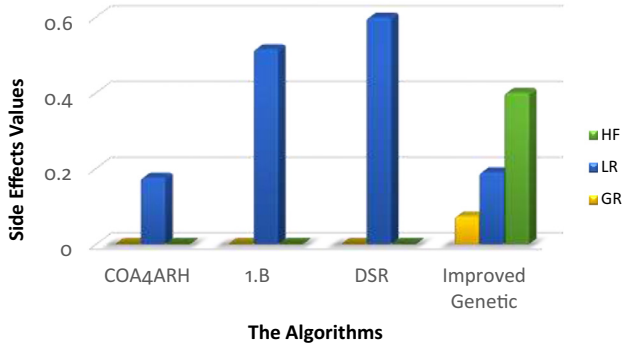


Fig. 8. Comparative evaluation of the algorithms on Chess dataset.

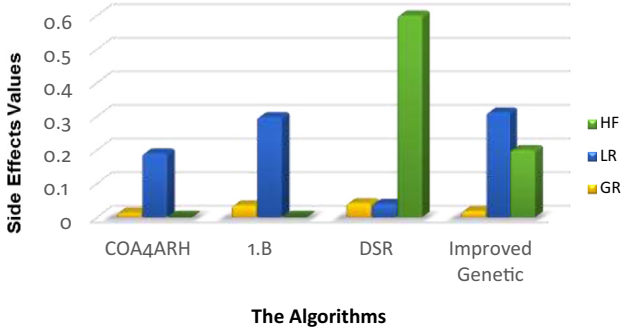


Fig. 9. Comparative evaluation of the algorithms on Synthetic dataset.

insertion technique in the proposed algorithm and selecting items with minimum frequency in non-sensitive rules for deletion, the value of *GR* in this algorithm will be zero or only a small amount. Furthermore as it is shown in Figs. 7, 8 and 9, there is an improvement in the value of *LR* in the proposed algorithm compared to the other algorithms, which is the result of selecting items with the minimum frequency in non-sensitive rules for deletion. As shown in Fig. 9, the proposed algorithm has a higher value of *LR* compared to the DSR algorithm, this is since the proposed algorithm needs to apply more deletion to decrease the *HF* to zero, hence compared to DSR with higher *HF*, the proposed algorithm will have a higher value of *LR*.

5.3.1. Comparing the number of iteration

One of the most important evaluation criteria in meta-heuristic algorithms is the number of iterations required to attain the optimal solution. Since measures have been taken to minimize the number of iterations in COA4ARH algorithm, the comparison between this algorithm and Improved Genetic algorithm is done on three databases in Table 5. The results of these comparisons are seen in Figs. 10 to 15.

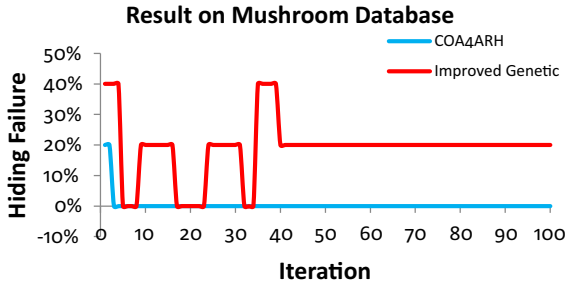


Fig. 10. Comparing the number of hiding failures in each iteration on Mushroom database.

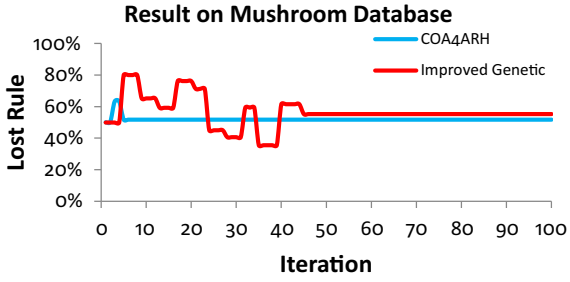


Fig. 11. Comparing the number of lost rules in each iteration on Mushroom database.

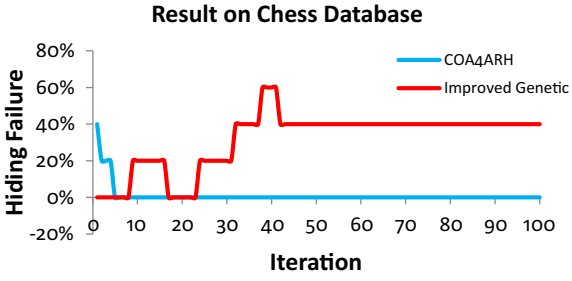


Fig. 12. Comparing the number of hiding failures in each iteration on Chess database.

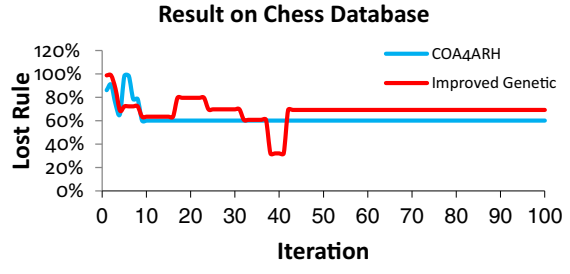


Fig. 13. Comparing the number of lost rules in each iteration on Chess database.

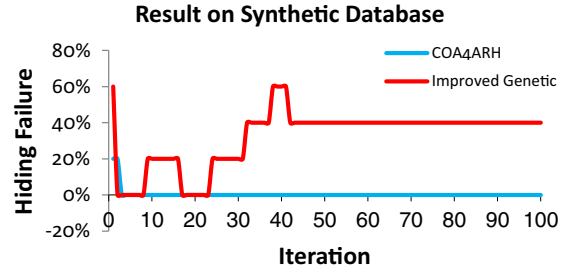


Fig. 14. Comparing the number of hiding failures in each iteration on synthetic database.

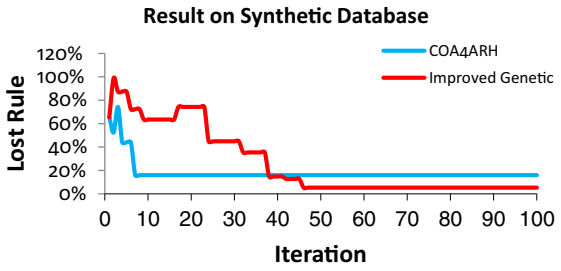


Fig. 15. Comparing the number of lost rules in each iteration on synthetic database.

As it can be seen in these figures, COA4ARH algorithm has been converged with a high speed and in the number of iteration less than 10 can attain the optimal solution. Furthermore, this algorithm showed better results compared to the other algorithm even in the very low number of iteration. Due to the way preprocessing is defined, significant decline occurred in the number of required iteration in COA4ARH algorithm. The operation is defined in a way that only the items involved in hiding sensitive rules are chosen to be removed. As a result, first, this prevents the production of useless solution and increases the chance of suitable solution production; second, the number of possible modes to produce a solution reduces significantly. Because in a string length of L the placement modes of 0 and 1 is 2^L. In fact, preprocessing reduces L size significantly by choosing sensitive items to remove; as a result, the number of cases for attaining the result will be reduced.

Due to the manipulation of all the items of the database in Improved genetic algorithm, useless solutions are produced that decrease the chance of producing more effective solutions and increase the time needed to attain optimal solution. As it is seen in Figs 10–15, this algorithm has been converged in the number of iterations over 40.

6. The effects of sparse dataset

The existence of sparse datasets as input data in proposed algorithm make extracted association rules to have a lower degree of overlap. Therefore, sensitive rules elected for deletion will also have little overlap. The low degree of overlap suggests that the number of frequent items between sensitive rules is low so the number of sensitive items of the algorithm chosen to be deleted increase to zero the hiding failure which has the highest priority in side effects. This factor will lead to an increase in the number of lost rules. Consequently, it can generally be concluded that the presence of sparse datasets in the proposed algorithm input data increase the number of lost rules.

7. Conclusions and future works

The present article has proposed a new meta heuristic method for hiding sensitive association rules using cuckoo optimization algorithm. The proposed algorithm is capable of simultaneously hiding several sensitive association rules. The most important and influential feature of the present study is defining a preprocess operation which includes two phases in the beginning of proposed algorithm. This preprocess operation causes a remarkable decrease in the number of iterations and speedy access to the optimal solution. In this study, three fitness functions have also been introduced which can find the solution with minimum side effects. Further an immigration algorithm has been defined which improved the ability of the proposed algorithm to escape from local optimums. The proposed approach was compared to the three algorithms 1.b, DSR and Improved Genetic. For efficiency assessment, all of the algorithms were examined on both Real and Synthetic data. The results were analyzed based on the four criteria *HF, LR, GR* and the number of iteration. The results show a higher efficiency for the proposed algorithm compared to the others. Since, *HF* was 0, *GR* was close to 0 and *LR* had a remarkable decrease compared to other algorithms. Moreover the proposed algorithm is converged with a high speed and in the lower number of iteration can attain the better solution compared to Improved Genetic algorithm.

Defining a new fitness function that can decrease the amount of Lost Rules and preserve the algorithm's capability of hiding sensitive rules and avoiding generation of ghost rules will be the purpose of future studies. Also, through some computations, the number of sensitive items that should be deleted for hiding sensitive rules can be calculated; only this number of sensitive items should be deleted to decrease the number of lost rules. Moreover, a self-adaptive mechanism will be added to the proposed algorithm. In this mechanism, algorithm input parameters will be changed by each iteration and will be moved toward being optimized.

References

Atallah, M., Bertino, E., Elmagarmid, A., Ibrahim, M., & Verykios, V. (1999). Disclosure limitation of sensitive rules. In *IEEE 1999 knowledge and data engineering exchange* (pp. 45–52). IEEE.

Dasseni, E., Verykios, V. S., Elmagarmid, A. K., & Bertino, E. (2001). Hiding association rules by using confidence and support. In *Information hiding: 2137* (pp. 369–383). Pittsburgh, PA, USA: Springer Berlin Heidelberg.

Dehkordi, M. N., Badie, K., & Zadeh, A. K. (2009). A novel method for privacy preserving in association rule mining based on genetic algorithms. *Journal of Software, 4*, 555–562.

Gkoulalas-Divanis, A., & Verykios, V. S. (2006). An integer programming approach for frequent itemset hiding. In *Proceedings of the 15th ACM international conference on information and knowledge management* (pp. 748–757). ACM.

Gkoulalas-Divanis, A., & Verykios, V. S. (2009a). Exact knowledge hiding through database extension. *IEEE Transactions on Knowledge and Data Engineering, 21*, 699–713.

Gkoulalas-Divanis, A., & Verykios, V. S. (2009b). Hiding sensitive knowledge without side effects. *Knowledge and Information Systems, 20*, 263–299.

Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System Technical Journal, 29*, 147–160.

Khan, A., Qureshi, M. S., & Hussain, A. (2014). Improved genetic algorithm approach for sensitive association rules hiding. *World Applied Sciences Journal, 31*, 2087–2092.

Menon, S., & Sarkar, S. (2007). Minimizing information loss and preserving privacy. *Management Science, 53*, 101–116.

Menon, S., Sarkar, S., & Mukherjee, S. (2005). Maximizing accuracy of shared databases when concealing sensitive patterns. *Information Systems Research, 16*, 256–270.

Moustakides, G. V., & Verykios, V. S. (2006). A max-min approach for hiding frequent itemsets. In *Proceedings of the sixth IEEE international conference on data mining-workshop* (pp. 502–506).

Moustakides, G. V., & Verykios, V. S. (2008). A max-min approach for hiding frequent itemsets. *Data & Knowledge Engineering, 65*, 75–89.

Oliveira, S. R., & Zaïane, O. R. (2002). Privacy preserving frequent itemset mining. In *IEEE 2002 ICDM workshop on privacy, security and data mining: vol. 14* (pp. 43–54). Maebashi, Japan: Australian Computer Society, Inc.

Oliveira, S. R., & Zaïane, O. R. (2003a). Algorithms for balancing privacy and knowledge discovery in association rule mining. In *IEEE 2003 7th international database engineering and applications symposium* (pp. 54–63). IEEE.

Oliveira, S. R., & Zaïane, O. R. (2003b). Protecting sensitive knowledge by data sanitization. In *IEEE 2003 3th international conference data mining* (pp. 613–616). IEEE.

Oliveira, S. R., & Zaïane, O. R. (2006). A unified framework for protecting sensitive association rules in business collaboration. *International Journal of Business Intelligence and Data Mining, 1*, 247–287.

Rajabioun, R. (2011). Cuckoo optimization algorithm. *Applied Soft Computing, 11*, 5508–5518.

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

Saygin, Y., Verykios, V. S., & Elmagarmid, A. K. (2002). Privacy preserving association rule mining. In *IEEE 2002 12th international workshop on research issues in data engineering: engineering e-commerce/e-business systems* (pp. 151–158). IEEE.

Sun, X., & Philip, S. Y. (2007). Hiding sensitive frequent itemsets by a border-based approach. *Journal of Computing Science and Engineering, 1*, 74–94.

Sun, X., & Yu, P. S. (2005). A border-based approach for hiding sensitive frequent itemsets. In *IEEE 2005 5th international data mining conference* (pp. 426–433). IEEE.

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

Verykios, V. S., Pontikakis, E. D., Theodoridis, Y., & Chang, L. (2007). Efficient algorithms for distortion and blocking techniques in association rule hiding. *Distributed and Parallel Databases, 22*, 85–104.

Walton, S., Hassan, O., Morgan, K., & Brown, M. (2011). Modified cuckoo search: A new gradient free optimisation algorithm. *Chaos, Solitons & Fractals, 44*, 710–718.

Wang, S.-L., Parikh, B., & Jafari, A. (2007). Hiding informative association rule sets. *Expert Systems with Applications, 33*, 316–323.

Wu, Y.-H., Chiang, C.-M., & Chen, A. L. (2007). Hiding sensitive association rules with limited side effects. *IEEE Transactions on Knowledge and Data Engineering, 19*, 29–42.

Yang, X.-S., & Deb, S. (2009). Cuckoo search via Lévy flights. In *Nature & biologically inspired computing, 2009. NaBIC 2009. World congress on* (pp. 210–214). IEEE.

Yang, X.-S., & Deb, S. (2013). Multiobjective cuckoo search for design optimization. *Computers & Operations Research, 40*, 1616–1624.