ISSN: 0067-2904

Modified Bees Swarm Optimization Algorithm for Association Rules Mining

Rasha A. Mohammed*[1], Mehdi G. Duaimi[1], Ahmed T. Sadiq[2]

[1]Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq.
[2]Department of Computer Science, University of Technology, Baghdad, Iraq.

Abstract
Mining association rules is a popular and well-studied method of data mining tasks whose primary aim is the discovers of the correlation among sets of items in the transactional databases. However, generating high- quality association rules in a reasonable time from a given database has been considered as an important and challenging problem, especially with the fast increasing in database's size. Many algorithms for association rules mining have been already proposed with promosing results. In this paper, a new association rules mining algorithm based on Bees Swarm Optimization metaheuristic named Modified Bees Swarm Optimization for Association Rules Mining (MBSO-ARM) algorithm is proposed. Results show that the proposed algorithm can be used as an alternative to the traditional methods.
Keywords: Bees Swarm Optimization, Association Rules Mining.

خوارزمية سرب النحل ألامثل المعدلة للتنقيب عن قواعد الارتباط

رشا عبود محمد*[1]، مهدي كزار دعيمي[1]، احمد طارق صادق[2]

[1]قسم علوم الحاسبات، كلية العلوم، جامعة بغداد، بغداد، العراق.
[2] قسم علوم الحاسوب، كلية العلوم، الجامعة التكنولوجية، بغداد، العراق.

الخلاصة
التنقيب عن قواعد الارتباط هو من الطرق الشائعة والمهمة من مهام التنقيب عن البيانات والتي تهدف بشكل رئيسي الى ايجاد العلائقية بين مجموعة من العناصر في قاعدة البيانات . ومع ذلك فان توليد قواعد ارتباط ذات جودة عالية في وقت مناسب من قاعدة بيانات معينة يعتبر تحدي مهم و صعب خصوصا" مع الازدياد السريع في احجام قواعد البيانات .العديد من خوارميات التنقيب عن قواعد الارتباط قد اقترحت مع نتائج واعدة . هذا البحث يقدم خوارزمية جديدة للتنقيب عن قواعد الارتباط باستخدام خوارزمية سرب النحل الامثل . النتائج اظهرت امكانية استخدام الخوارزمية المقترحة كبديل للطرق التقليدية.

1. Introduction
Since it's has been introduced in (Agrawal et al. 1993) [1] the task of association rules, mining has received a great deal of attention in research. Currently, the mining of such rules is still one of the richest pattern discovery methods in data mining.
Association rules mining (ARM) problem is formally stated as follows: Let T be a set of transactions

*Email:rasha.abbood@yahoo.com

T= {t_1, t_2, …,t_m} representing a transactional database, and A= {a_1, a_2, … ,a_n} be a finite set of items. An association rule is an implication in the form of X→Y, where X, Y are sets of items ,$X \subset A$, $Y \subset A$ and $X \cap Y = \phi$.

X is called antecedent while Y is called consequent and the rule means X implies Y. The support of an itemset $A' \subseteq A$ is the number of transactions containing A'. Two main parameters are commonly used for determining the utility of association rule, namely the support of a rule and the confidence of a rule. Rule support is the relative occurrence of the detected association rules within the entire database. To determine the rule support of an association rule, say X →Y, divide the number of transactions supporting the rule by the total number of transactions sup (X→Y) = sup (X∪ Y)/|T|. The confidence of an association rule is the probability that a transaction contains Y given that the transaction contains X. The confidence of a rule X→Y is defined as conf (X→Y) = sup (X∪ Y) /sup(X). Confidence is the measure of the strength of the association rules. Association rules X → Y with a confidence of 90% means that 90% of the transactions that include X also include Y together.
Association rules mining problem consist of extracting all rules with support ≥ Minsup and confidence ≥ Minconf from a given database, where Minsup and Minconf are user specified constraints.
Most of the association rules mining algorithms are based on methods proposed by Agrawal et al. 1993 [1] and Agrawal & Srikant 1994 [2] such as Eclat [3] and FP-Growth [4]. These methods mine association rules in two steps:

1. Finding all itemsets with supports greater than, or equal to, the user-specified minimum support (Minsup).
2. Generating all the rules which satisfy the minimum confidence (Minconf) constraint.

However, since the databases are growth rapidly, the classical methods have become inefficient and the extracting of best rules in one execution of an algorithm has become a necessity. In addition, this explosive growth in the amount of data accompanied with the fact that user no longer interests in all rules but only a subset of useful rules, led to the idea of using metaheuristic algorithms for association rules mining. In contrast to classical methods these algorithms are looking for only a part of good quality rules rather than all valid rules.

In recent years, the swarm intelligence algorithms received widespread attention in research and considered as a powerful nature-inspired metaheuristic algorithms. Swarm Intelligence (SI) can be defined as a relatively new branch of artificial intelligence using models that simulate the social behavior which can be observed in nature, such as ant colonies, flocks of birds, fish schools and bee hives where a number of individuals with limited capabilities are able to find intelligent solutions for complex problems of very diverse nature [5]such problems are search of food, building of the nest, distribution of work and lieu the tasks among the individuals,.., etc [6].

Swarm algorithm is a nature-inspired, population-based algorithm that is capable of creating low-cost, fast, and robust solutions to many complex problems [7, 8]. Many swarm intelligence algorithms based on different natural swarm systems have been presented and successfully applied in several real-life applications. Particle Swarm optimization [9], Ant Colony Optimization (ACO) [10], Bees Swarm Optimization(BSO) [11] and Cuckoo Search algorithm (CS) [12] are examples of swarm intelligence algorithms. Several versions of bees swarm algorithms have been already proposed for solving association rules mining problem and optimization problems [11, 13, 14].

In this paper, a new algorithm for mining association rules based on Bees Swarm Optimization (BSO) is proposed to extract the maximum number of valid rules with high support and confidence values.

The rest of this paper is organized as follows. In section 2 related works are discussed. In section3, a brief explanation of Bees Swarm Optimization algorithm and its principles is presented. In section 4 the Modified Bees Swarm Optimization for Association Rules Mining (MBSO-ARM) algorithm is introduced. Experimental setup and results are stated in section 5. Finally, we conclude with a summary in section 6.

2. Related works

The idea of using metaheuristic algorithm for mining association rules was first applied by Mata et al. 2001 [15]. The authors proposed a new algorithm for association rule mining based on genetic algorithm named GENetic Association Rules (GENAR). This algorithm is designed to discover quantitative association rules, based on the distribution of values of the quantitative attributes and used a special evolutionary methodology to prevent the individuals from tending to the same solution (rule).

Mata et al.2002 [16] then proposed a new algorithm based on an evolutionary algorithm called Genetic Association Rules(GAR) .GAR is an extension to GENAR algorithm, and it was designed for finding frequent itemsets only where each individual has been considered as k-itemset. However, GAR is used for the first stage of association rules mining, another algorithm is required for extraction the rules. Guo & Zhou 2009 [17] proposed another GA for mining association rules. In this algorithm, an adaptive mutation rate is used to avoid immoderate variations in fitness at earlier generation thereby avoiding non-convergence. However, because the mutation probability is computed at each iteration, the computational time is increasing as a result. Yan et al. 2009, [18], developed a new algorithm for ARM based on a genetic algorithm called ARMGA. Relative confidence considered as the fitness function in ARMGA. Therefore ARMGA generates many rules with high fitness quality, but, without regard to minimum support and minimum confidence constraints. In addition, only rules with fixed length are extracted.

All of the above-mentioned algorithms have a fundamental limit in their operations where the generated solutions may not be admissible that are solutions have the same item in antecedent part and consequents part and moreover, there is no specific treatment to deal with this issue. Djenouri et al. 2012 [19], proposed a new algorithm named BSO-ARM based on bees swarm behavior. BSO-ARM has been successful in solving the problem of non- admissible solutions using "delete and decompose strategy." After that Djenouri et al. 2014 [20], proposed a new version of the BSO-ARM with new encoding method and three different strategies for the determination of Search Area.

The existing BSO-ARM [20] yields good results in terms of the fitness function and time complexity; however, there is always a possibility of improvement in the existing algorithms. Our study indicates that there is some possibility to create an improved algorithm than the existing one; therefore a new possible solution is proposed which will be a better performer than the existing ones in terms of the number of valid rules and their quality.

3. Bees Swarm Optimization (BSO) Algorithm

The “Bees Swarm optimization” metaheuristic is a population-based search algorithm inspired by the natural foraging behavior of bees first proposed in [11]. It is based on a group of artificial bees cooperating with one another to solve a problem. In its basic version, the algorithmstarts by one bee called InitBee settle for finding a solution submitting some good features. From this solution named Sref a collection of other solutions of the search space are determined. This collection of the solution is called Search Area. Then, every bee will take one solution as its starting place in the search. When each bee completes its search, the best-visited solution of every bee is transported to all its neighbors through a table named Dance. In the next iteration, one of the solutions saved in this table will be the new reference solution. So as to avert cycles, the reference solution is saved each time in a taboo list. The reference solution is selected first according to the quality criterion. But, if the swarm noticed that the solution is not improved any more, it introduces a criterion of diversification forbidding it from being restricted in a local optimum. The diversification mechanism consists of choosing between the solutions saved in the taboo list, the most distant one. Algorithm 1 describes the general steps of BSO metaheuristic [11].

Algorithm 1: The general BSO algorithm

```
1.  Begin
2.  Let Sref  be the solution found by Init Bee
3.  While i < Max-Iter and not stop do
4.  Begin
5.  Insert Sref in taboo list
6.  Determine SearchArea from Sref
7.  Allocate a solution from SearchArea to each bee
8.  for each Bee k do
9.  Begin
10. Search starting with the solution assigned to it
11. Store the result in the table Dance
12. End for
13. Choose the new solution of reference Sref
14. End while
15. End
```

4. MBSO-ARM Algorithm

In this section, the Modified Bees Swarm Optimization for Association Rules Mining (MBSO-ARM) algorithm is proposed. The following of this section are some important parts of the algorithm which are explained: rule representation, initialization, Search Area determination strategy, the fitness function and the neighborhood search.

4.1 Rule Representation

Basically, there are two approaches for representing a rule which is Pittsburgh approach, and Michigan approach. Although those approaches have been taken from the GA community they can be generalized for all population-based algorithms [21]:

- Pittsburgh approach

In this approach, each individual of the population represents a set of rules.

- Michigan approach

In this approach, each individual of the population represents a single rule.

In our work, the Michigan approach is followed as rules representation's method. While for the encoding of the rule into solution (individual), the encoding technique proposed in [20] is used which is a combination of two well- known encoding techniques namely binary encoding and integer encoding. In binary encoding, a vector S of n items is used to represent the solution (rule) where n is the numbers of items. If S [i] = 1 the item will be in the rule and 0 otherwise. On the other hand, the integer encoding represents the solution (rule) by a vector S of k + 1 items where k is the rule size. The first element is the separator index between antecedent and consequent parts of the solution. For all others items i in S, if S[i] = j then the item j appears in the ith position of the rule.

According to encoding technique proposed in [20] the solution, S is a vector of n items where n is the number of all items in dataset. The locations of the items in S are defined as follows:

1. If S[i] = 0 then the item i is not in the solution S.

2. If S[i] =1 then the item i belongs to the antecedent part of the solution S.

3. If S[i] =2 then the item i belongs to the consequent part of the solution S.

Example 1: Let T = {t_1, t_2,..., t_7} be a set of items

S1= {0, 1, 0, 0, 1, 2, 0} V1 represents the rule R1:

$t_2, t_5 \rightarrow t_6$.

S2= {1, 0, 2, 0, 0, 0, 1} V2 represents the rule R2:

$t_1, t_7 \rightarrow t_3$.

In this representation the separating between the antecedent and the consequent part of the rule become simpler than that of other representations, thereby the fitness function can be calculated more practically.

4.2 Initialization

In MBSO-ARM algorithm, the choosing of the initial solution as pure random is avoided because such random initialization can result in rules that will cover no data instance consequently having a large percentage of invalid rules. On another hand, the utilizing of non–random initialization can lead to an enhancement in the quality of the solutions and minimizing runtime as presented in [22]. This non–random initialization, however, requires prior information about search space that might be absent in some cases. To overcome this issue the following initialization method is suggested:

Initially, a seed solution S of size n where n is the number of items in dataset is chosen as follows:

1. Two items (1, 2) placed randomly in S while the rest positions of S are set to zero.

2. The initial solution Sref is chosen by permutation of the items of the seed solution S over n times. Then we evaluate the fitness function of the solutions generated from this permutation process so that the solution with high fitness quality will be the first Serf.

4.3 Search Area Determination Strategy

A Search Area is generated from the reference solution Sref by adding 1 to two bits of Sref, one of them is chosen in a random way. This operation is repeated n times where n is the length of Sref. Algorithm 2 describes formally the suggested strategy.

Algorithm 2: Search Area Determination algorithm

```
1.  Input: Sref, n // the initial solution, n the number of items in Sref
2.  Output: Search- Area: Array [1...n [1...n]
3.  m← random integer with $0 \le m \le n$
4.  For i=0, i<n, i= i+ +do
5.      Copy (Sref, Search- Area[i])
6.      For j =0, j <n, j = j + +do
7.          If j= i Then
8.              Search- Area [j] = Search- Area [j] +1
9.              If Search- Area [j] > 2 Then
10.                 Search- Area [j] =0
11.             End if
12.         End if
13.         Search- Area [m] = Search- Area [m] +1
14.         If Search- Area [m] > 2 Then
15.             Search- Area [m] =0
16.         End if
17.     End for
18. End for
19. Return Search- Area
```

Example1: Consider Sref = {0, 1, 0, 1, 2, 0, 0}

1) add 1 to the first bit and to another bit selected randomly in Sref :

$$S1=\{\underset{\uparrow}{1}, 1, 0, \underset{\uparrow}{2}, 2, 0, 0\}$$

2) add 1 to the second bit and to another bit selected randomly in Sref:

$$S2=\{0, \underset{\uparrow}{2}, 0, 1, 2, 0, \underset{\uparrow}{1}\}$$

4.4 The Neighborhood Space

The neighborhood space is calculated from each solution S by adding or subtracting one from a random bit of S. An adaptive value named Adp is used to determine either adding or subtracting the value of a target bit as shown in algorithm 3. The reason for utilizing an adaptive rate is that we observed that fixed Adp value may led to one of the following non-desirable cases:

The first case occurs when Adp value is too small; then most solutions tend to have small fitness function .This attributed to the large increasing in size of rules (solutions) and thereby having rules with very small support and with time, the most solutions (rules) of the population will be invalid. On the other hand using large Adp value may result in having some solutions closing to the optimal, which will always be eliminated, not convergent in the process of evolution. Consequently the possibility of trapping in local optimum will be high. This is due to the fact that we repeatedly perform on solutions of a population (having small rules with relatively high support). However, this case can reach the situation where certain values for certain positions in the solution eliminate in the early stage of the algorithm. Therefore we design the following equation to calculate the value of Adp as follow:

$$\text{Adp} = \left(\frac{i}{\text{MaxIter}}\right)^{\text{EXP}^{-1}} \qquad (1)$$

Where i is, the number of current iteration and MaxIter is the maximum number of iterations.
According to Eq. (1) the value of Adp ∈ [0, 1] increases with increasing i so that we avoid the large growth in the size of rules with time. In addition, it enables the algorithm to escape from local optimum by preventing the elimination of certain items in the early iterations.

Algorithm 3: Neighborhood Space Search

```
1. Input: S, K, n //S the solution assigning to bee, K bee's number, n is the number of items in S.
2. 2 Output: Neighborhood _Space: Array [1... K*n][1...n]
3. a← 0
4. While a< K do
5. For j =0, j <n, j = j + +do
6. Copy (S, Neighborhood _Space [a])
7. r← random decimal with 0 ≤ r ≤ 1
8. j ← random integer with 0 ≤ h≤ n
9. If r > Adp
10. Neighborhood _Space [h] = Neighborhood _Space [h] +1
11. If Neighborhood _Space [h]>2 then
12. Neighborhood _Space [h] =0
13. 13           End if
14. Else
15. Neighborhood _Space [h] = Neighborhood _Space [h] -1
16. If Neighborhood _Space [h] <0 then
17. Neighborhood _Space [h] =0
18. End if
19. End if
20. End for
21. a← a+ 1
22. End while
23. Return neighborhood _Space
```

4.5 Fitness Function

The goal of association rules mining is to discover all association rules having support and confidence not less than a given threshold value. Let a and b are two empirical parameters where a+b=1; the solution s will be evaluated by the following fitness function [20]:
Fitness(s) = a× Confidence(s) +b × Support(s)
if Confidence(s) ≥ Minconf and Support(s) ≥ Minsup
Fitness= -1 otherwise.

4.6 The Algorithm

A reference solution (Sref) is generated as mentioned in section 4.2, from the current Sref. The Search Area regions are explored by Search Area determination strategy and then selecting k of bees with the highest fitness qualities from Search Area. At this point, every bee explores its region through neighborhood search operation. After that, bees communicate each other so as to choose the best neighborhood through table called dance table. The best neighborhoods (solutions) in dance table becomes the next reference solution (Sref), a list named best list is used to save the best solutions from each pass. This operation will be repeated until the maximum number of iterations is reached. At the termination of the algorithm, the rules are generated from the best list and visualized to the user. Algorithm 4 describes more formally the proposed method.

Algorithm 4 MBSO-ARM

```
1. Input: k (number of Bee), MaxIter, a, b, Transactional Dataset,   Minsup, Minconf
2. Output: Set of Asso. Rules
3. i ← 0
4. Sref ← an initial solution
5. While i < MaxIter do
6. Generate Search Area from Sref via Search Area Determination strategy
7. Evaluate each solution in Search Area
8. Choose k solutions from Search Area
9. For each solutions J do
10. Generate (k) neighborhoods from solution J
```

```
11. Put the best neighborhood in dance table
12. End for
13. Add the best solutions to the Best List
14. Sref← the best solution in dance table
15. i ← i + 1
16. End while
17. For each solution S in the Best List do
18. Generate the rule from S
19. End for
```

Table1- The specifications of datasets.

Dataset name	Transactions size	Item size(n)
Zoo	101	36
Primary tumor	336	31
German credit	1,000	112
chess	3,196	75

5. Experimental setup and results

For verifying the performance of suggested algorithm and comparing with BSO-ARM algorithm [20], four categorical datasets are considered from Java Open-Source Data Mining http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php [23] and Constraint programming for Itemset mining https://dtai.cs.kuleuven.be/CP4IM/datasets [24]. The specifications of them are implied in Table-1. All the experiments were conducted in an environment of Microsoft Windows 7 using a laptop computer with Intel core Intel(R) Celeron, the clock speed of 1.70 GHz and 4GB RAM. All programs are scripted in Microsoft visual basic 4.0 Client Profile using visual basic.net.

Table 2-The comparison of obtained results from the MBSO-ARM and BSO-ARM in terms of average fitness

		Bee number=5		
Average fitness	Zoo	Primary tumor	German credit	Chess
MBSO-ARM	0.73	0.79	0.73	0.94
BSO-ARM (Next strategy)	0.71	0.65	0.64	0.71
		Bee number=7		
Average fitness	Zoo	Primary tumor	German credit	Chess
MBSO-ARM	0.73	0.78	0.73	0.93
BSO-ARM (Modulo strategy)	0.66	0.62	0.65	0.63

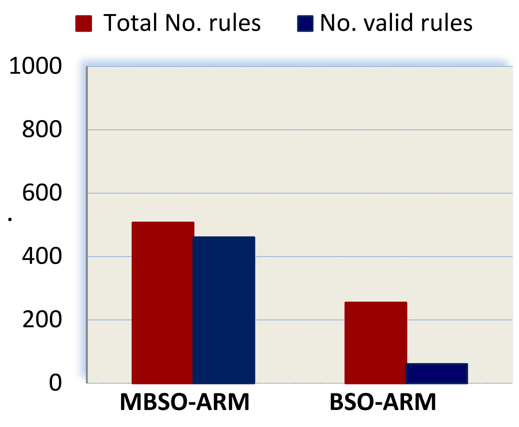
5.1 parameters settings

In all executions the following parameters are fixed: the coefficient a = 0.6, b =0.4, Minsup =0.1, Minconf =0.7and number of iterations =100. Each reported result corresponds to the average of 10 executions.

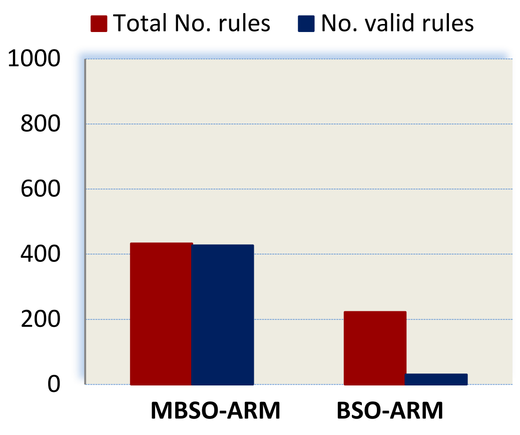
5.2 Experiments and Comparison

Table- 2 presents the average fitness quality of MBSO-ARM and BSO-ARM (Next strategy, modulo strategy). These tables show that the proposed algorithm outperforms BSO-ARM with regard to the quality of solutions .Furthermore the experiences also find that the number of valid rules obtained by our algorithm is more than those returned from BSO-ARM. Figures-1 and 2 illustrates the scenario of two algorithms with regard to the total numbers of unrepeated rules and the number of valid rules when bee's number is set to 6 and 7 using next strategy and modulo strategy, respectively.
These results can be explained by:
- The initialization method of MBSO-ARM algorithm supplies the algorithm with a good preliminary solution (Sref).

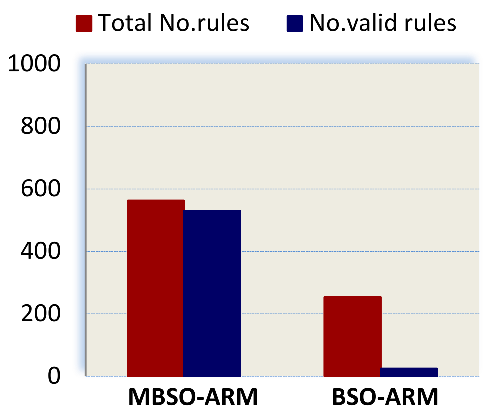
- high diversification of proposed algorithm thanks to efficient search area determination strategy which employs very simple randomization technique that helps the algorithm in generating different rules and preventing the solutions of group from tending to the same rule. In fact when the search area is explored without this randomization the number of repeated rules in certain cases increased to more than 50%. Moreover; the neighborhood search operation in MBSO-ARM with Adp reinforces the intensification strategy and thereby prevents the algorithm from stuck in a local mode.



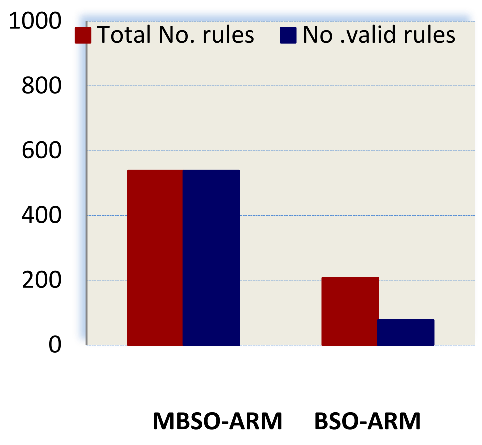
(a) Zoo dataset



(b) Primary Tumor dataset



(c) German Credit dataset

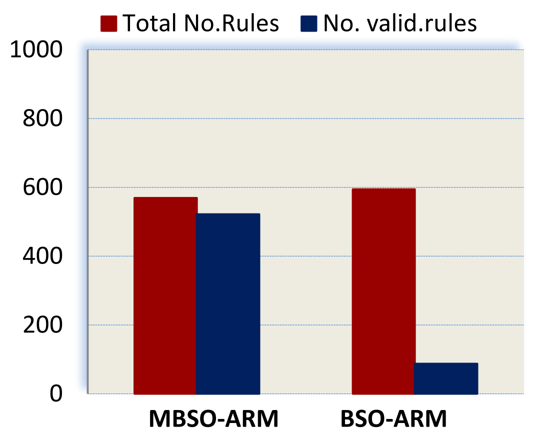


(d) Chess dataset

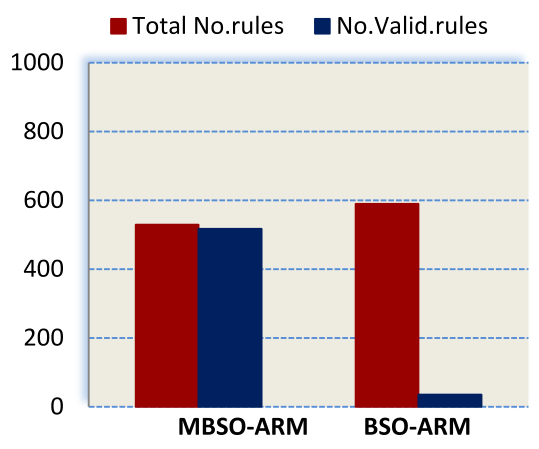
Figure 1- the total number of rules and the numbers of valid rules obtained by BSO-ARM (Next strategy) and MBSO-ARM in datasets.

- On another hand, BSO-ARM employs either next strategy or modulo strategy to scout the search area. The main drawback of next strategy is its working in such a way that each bee explores a limited region of search space which is reflected negatively on both the number of valid rules and their qualities.
- In the case that BSO-ARM are exploring search area by modulo strategy the ability of the algorithm to reach to further regions of search space will get high. But unfortunately, after some iteration, the number of items in solutions will increase so that the fitness and the number of valid rules are decreasing as a result. Tables-3 and 4 show the mean number of items contained in the antecedent and consequent parts of rules generated by two metaheuristics MBSO-ARM and BSO-ARM (next and modulo strategies). It's clearly that the size of rules obtained by the proposed algorithm are smaller

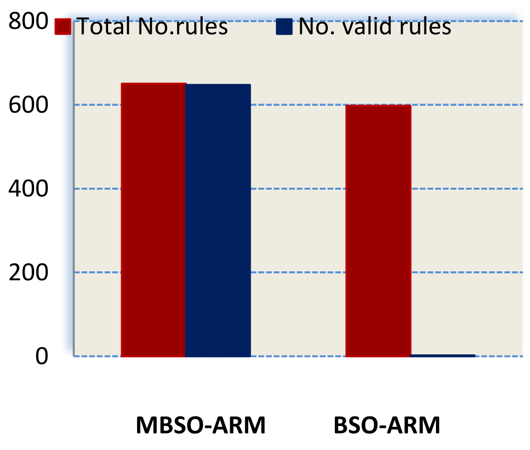
than those obtained by BSO-ARM (next strategy) in two out of four datasets and they are smaller than the values obtained from the BSO-ARM (modulo strategy) in four datasets.



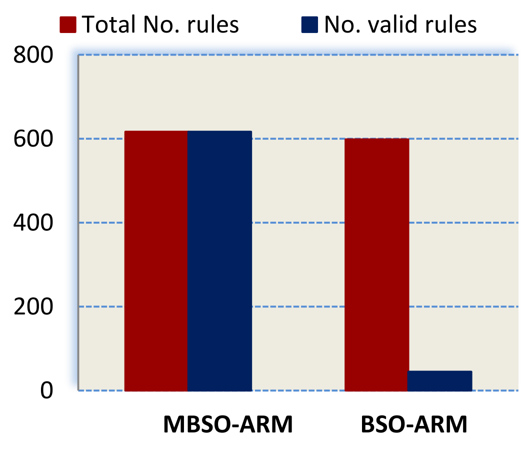
(a) Zoo dataset



(b) Primary Tumor dataset



(c) German Credit dataset



(d) Chess dataset

Figure 2- the total number of rules and the numbers of valid rules obtained by BSO-ARM (Modulo strategy) and MBSO-ARM in datasets.

However, the rules obtained by MBSO-ARM from four datasets have smaller consequent size than those obtained by BSO-ARM (next and modulo strategies). The experiments also show that the proposed algorithm generated the large size rules in the early iterations and then the rule's size will decreases with time.

- The diversification strategy of BSO-ARM raises the risk of choosing a bad solution to be reference solution; hence the number of valid rules and fitness will be decreased as a result. Also, BSO-ARM uses a very deterministic neighborhood search operation without any randomization, so the chances of searching around the current best are limited.

Table 3-The comparison between the size of rules obtained by MBSO-ARM and BSO-ARM (Next Strategy).

Bee number 6				
Antecedent size	Zoo	Primary tumor	German credit	Chess
MBSO-ARM	3.8	3	5.8	3.4
BSO-ARM	2.4	2.9	3.8	2.6
Consequent size	Zoo	Primary tumor	German credit	Chess
MBSO-ARM	1.6	1.3	1.7	1.5
BSO-ARM	1.9	2.8	3.4	2.6

Table 4-The comparison between the size of rules obtained by MBSO-ARM and BSO-ARM (Modulo Strategy).

Bee number 7				
Antecedent size	Zoo	Primary tumor	German credit	Chess
MBSO-ARM	3.7	3	5.5	3.4
BSO-ARM	6.7	6.3	6.4	8.8
Consequent size	Zoo	Primary tumor	German credit	Chess
MBSO-ARM	1.5	1.2	1.6	1.5
BSO-ARM	5.3	5	6.4	6

Table 5-The comparison of obtained results from the MBSO-ARM and BSO-ARM (Next strategy) in term of CPU time.

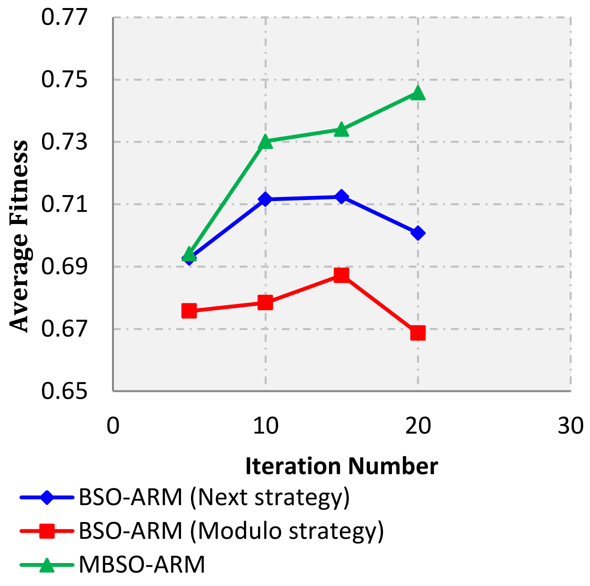
Execution Time(sec)		
Bee number=5		
Dataset	MBSO-ARM	BSO-ARM
Zoo	35	50
Primary Tumor	175	247
German Credit	2354	3071
Chess	4839	5224
Bee number=6		
Dataset	MBSO-ARM	BSO-ARM
Zoo	38	64
Primary Tumor	197	277
German Credit	2656	3204
Chess	5352	6319

Table 6- The comparison of obtained results from the MBSO-ARM and BSO-ARM (Modulo strategy) in term of CPU time.

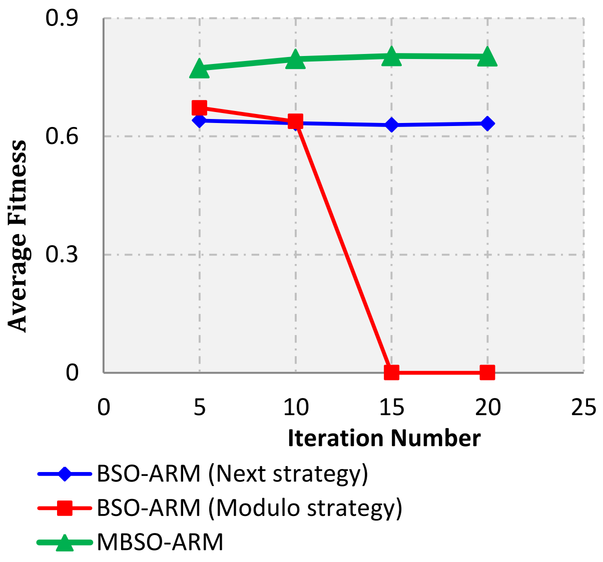
Bee number=7		
Execution Time(sec)		
Dataset	MBSO-ARM	BSO-ARM
Zoo	43	64
Primary Tumor	229	264
German Credit	3112	4181
Chess	6253	10822

Table-5 and Table-6 present the execution time of the two metaheuristics. It indicates clearly that BSO-ARM is more expensive in term of CPU-time because of its diversification strategy. Definitely, the CPU time increases when the number of transactions and the number of items becomes large.
Because our goal is to find the best set of rules, not the best rule. Figure-3 illustrates the fitness average of the final population without repetitive rules at each 25 repetitions of BSO-ARM (Next, Modulo) strategy algorithm and the MBSO-ARM algorithm. This figure shows that MBSO-ARM isn't

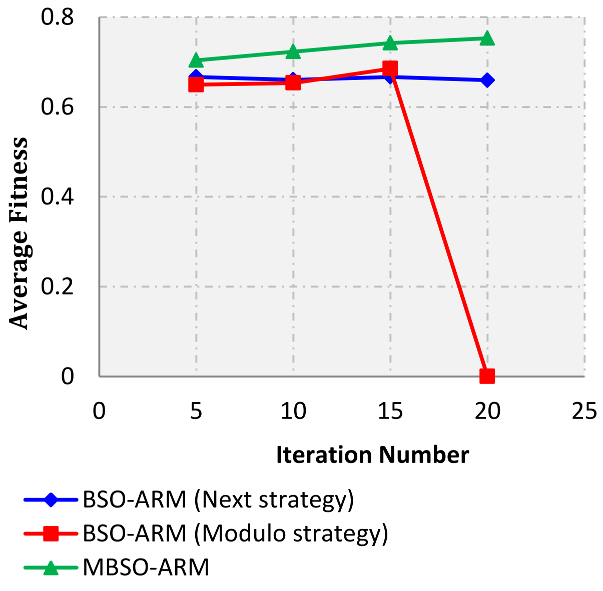
trapped in local optimum in all datasets while BSO-ARM (Next strategy) stays on the local mode in zoo datasets



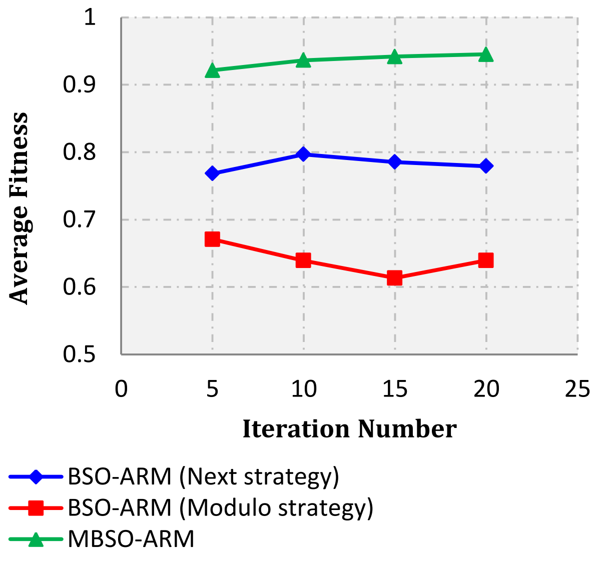
(a) Zoo dataset



(b) Primary Tumor dataset



(c) German Credit dataset



(d) Chess dataset

Figure 3- The average fitness of final population at each 25 repetitions of the run of MBSO-ARM and BSO-ARM in datasets (six bees).

On another hand, in BSO-ARM (Modulo strategy) the fitness decreases with the increase of iterations for all the datasets.

6. Conclusion

In this paper we have proposed a new algorithm named MBSO-ARM based on Bees Swarm Optimization metaheuristic (BSO) for extracting high quality association rules. Our algorithm outperformed the existing BSO-ARM algorithms in terms of the number of valid rules and fitness value. Furthermore, the computational time for MBSO-ARM algorithm is better than that of BSO-

ARM algorithm. The experimental results showed that our approach can be used as an alternative to existing association rules mining algorithms. Future trends and suggestions include proposing of new fitness functions.

References

1. Agrawal, R., Imielinski, T. and Swami, A. **1993**. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD international conference on Management of data. pp: 207–216.
2. Agrawal, R. and Srikant, R. **1994**. Fast Algorithms for Mining Association Rules in Large Databases. Proceedings of the 20th International Conference on Very Large Data Bases (VLDB ’94), pp: 487–499.
3. Zaki, M J. **2000**.Scalable algorithms for association mining, *IEEE Transactions on Knowledge and Data Engineering*, **12**(3): 372-390.
4. Han, J., Pe, J., Yin, J.and Ma, R. **2004**. Mining frequent patterns withoutcandidate generatio*n*, in *Data Knowledge and Knowledge discovery*, **8:** 53-87.
5. Martens, D., Baesens, B. and Fawcett, T., **2010**. Editorial survey: Swarm intelligence for data mining. *Machine Learning*, **82**(1): 1–42.
6. Ahmed, H., Glasgow, J. and Kln, C. **2012**. Swarm Intelligence : Concepts , Models and Applications Technical Report , pp: 2012-585. , (February).
7. B. K. Panigrahi, Y. Shi and M.-H. Lim, **2011** (eds.): *Handbook of Swarm Intelligence. Series Adaptation, Learning, and Optimizatio,* **7**, Springer-Verlag Berlin HeidelbergISBN 978-3-642-17389-9.
8. Blum, C. and Merkle, D. (eds.) **2008**. *Swarm Intelligence – Introduction and Applications. Natural Computing.* Springer, Berlin.
9. Kennedy, J. and Eberhart, R. **1995**. Particle swarm optimization. Neural Networks, 1995. Proceedings of IEEE International Conference on, **4**: 1942–1948.
10. Dorigo, M. and Stützle, T., **2004**. *Ant Colony Optimization,* MIT Press, Cambridge, ISBN: 978-0-262-04219-2.
11. Drias, H., Sadeg, S. and Yahi, S. **2005**. Cooperative Bees Swarm for Solving the Maximum Weighted Satisfiability Problem. *In Computational Intelligence and Bioinspired Systems* , Springer Berlin Heidelberg. pp: 318-325
12. Yang, X.S. and Deb, S. **2009**. Cuckoo search via Levy flights. In Proceeding of 2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 , pp: 210–214.
13. Sadiq, Ah.T., Duaimi M. G. and Shaker S. A. **2012**. Data Missing Solution Using Rough Set Theory and Swarm Intelligence, In Proceeding of the International Conference on Advanced Computer Science Applications and Technologies (ACSAT).
14. Sadiq, Ah. T. and Hamad A.G. **2010**. BSA: A Hybrid Bees’ Simulated Annealing Algorithm To Solve Optimization & NP-Complete Problems, *Engineering & Technology Journal,* **28(**2): 271-281.
15. Mata, J., Alvarez, J. and Riquelme, J. **2001**. Mining numeric association rules with genetic algorithms, In Proceedings of the International Conference ICANNGA, pp: 264–267.
16. Mata, J., Alvarez, J. and Riquelme, J. **2002** .An evolutionary algorithm to discover numeric association rules,In Proceedings of the ACM Symposium on Applied ComputingSAC, pp: 590–594.
17. Guo, H. and Zhou, Y. **2009**. An Algorithm for Mining Association Rules Based on Improved Genetic Algorithm and its Application. 2009 Third International Conference on Genetic and Evolutionary Computing, pp: 117–120.
18. Yan, X., Zhang, C. and Zhang, S. **2009**. Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support. *Expert Systems With Applications,* **36**(2): 3066–3076.
19. Djenouri, Y. et al. **2012**. Bees swarm optimization for web association rule mining. Proceedings of the 2012 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops, WI-IAT 2012, pp: 142–146.
20. Djenouri, Y., Drias, H. and bbas, Z. **2014**. Bees swarm optimisation using multiple strategies for association rule mining, *Int. J. Bio-Inspired Computation*, **6**(4): 239–249.

21. Cervantes, A., Galvan, I. and Isasi, P. **2005**. A comparison between the Pittsburgh and Michigan approaches for the binary PSO algorithm. In Proceeding of 2005 IEEE Congress on Evolutionary Computation, 1(January), pp: 290–297.

22. Surry, P., and Radcliffe, N. **1996**. Inoculation to initialize evolutionary search. In Proceeding of Evolutionary Computing: AISB Workshop, Springer, Brighton, U.K.

23. Fournier-Viger, P. **2008** . SPMF: A java open-source data mining library. Available at: http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php.

24. CP4IM: Constraint programming for Itemset mining. Available at: https://dtai.cs.kuleuven.be/CP4IM/datasets.