Association rules mining using cuckoo search algorithm

Rasha A. Mohammed* and Mehdi G. Duaimi

Department of Computer Science,
College of Sciences,
University of Baghdad,
Baghdad, 10071, Iraq
Email: rasha.abbood@yahoo.com
Email: mehdi_duaimi@ymail.com
*Corresponding author

Abstract: Association rules mining (ARM) is a fundamental and widely used data mining technique to achieve useful information about data. The traditional ARM algorithms are degrading computation efficiency by mining too many association rules which are not appropriate for a given user. Recent research in (ARM) is investigating the use of metaheuristic algorithms which are looking for only a subset of high-quality rules. In this paper, a modified discrete cuckoo search algorithm for association rules mining DCS-ARM is proposed for this purpose. The effectiveness of our algorithm is tested against a set of well-known transactional databases. Results indicate that the proposed algorithm outperforms the existing metaheuristic methods.

Keywords: data mining; ARM; association rules mining; DCS; discrete cuckoo search; metaheuristic algorithm.


Biographical notes: Rasha A. Mohammed received her BSc degree in Computer Science from Computer Science Department in the College of Education for Pure Sciences (Ibn al-Haytham) at the University of Baghdad in 2010. In 2014, she joined the Computer Science Department in the College of Science at the University of Baghdad as an MSc student. She accomplished a master’s thesis project, which involved the use of swarm intelligence algorithms and techniques for association rules mining. Her research interests include data mining and artificial intelligence. Also, she has been involved in database and administrative programs at the University of Baghdad headquarter.

Mehdi G. Duaimi received his BSc, MSc and PhD degrees, all in Computer Science from Nahrain University, College of Sciences, Baghdad, Iraq at 1992, 1995, and 2007, respectively. In 2009, he joined the University of Baghdad, where he is now an instructor in the Department of Computer Sciences. During 1999–2009, he was at the Iraqi Commission for Computers and Informatics – Baghdad where he worked as a database designer and as an instructor. He has some publications related to data mining and information retrieval. His current research interests include areas like data mining, databases, and artificial intelligence.
1 Introduction

Currently, an evolving field of descriptive data mining which was called association rule mining has been confirmed supportive to characterise vital features of data from large databases (Deshpande, 2016). Association rule mining is one of the important research areas in data mining whose primary aim is to discover associations among sets of items in the transactional databases.

The formal description of association rules mining problem can be stated as follow: Let $T$ be a set of transactions $T = \{t_1, t_2, \ldots, t_m\}$ representing a transactional database, And $I = \{i_1, i_2, \ldots, i_n\}$ be a finite set of item. An association rule $X \rightarrow Y$ is a relationship between two sets of items $X, Y$ such that $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \emptyset$. $X$ is called antecedent while $Y$ is called consequent and the rule means $X$ implies $Y$. The support of an itemset $I' \subseteq I$ is denoted by $sup(I')$ is the number of transactions containing $I'$. The support of a rule, say $X \rightarrow Y$ is defined as $sup(X \cup Y) / |T|$. The confidence of the rule $X \rightarrow Y$ is defined as $conf(X \rightarrow Y) = sup(X \cup Y) / sup(X)$. Confidence is the measure of the strength of the association rules. Association rules mining problem comprises of extracting all rules with support $\geq$ Minsup and confidence $\geq$ Minconf from a given database, where Minsup and Minconf are user specified thresholds. Most of the association rules mining algorithms are based on methods proposed by (Agrawal et al., 1993) and (Agrawal and Srikant, 1994) these methods mine association rules in two stages separately: First, they discover frequent itemsets and then extract association rules from them.

Following a massive amount of data available in this era the user no longer interests in all rules which will be difficult to interpret, but only a subset of useful rules. Many variations of association rules mining algorithms have been proposed to improve the efficiency of mining association rules, as they have to mine a larger set of data items. However, the performance and complexity of association rules mining algorithms are still subject to the research community. Recently, swarm intelligence (SI) paradigm has been received lots of attention in research. SI is developed through inspiration from the behaviours of the living creatures in nature by simulating the cooperative performance of simple agents.

Swarm algorithms is a nature-inspired, population-based algorithms that are capable of creating low-cost, fast, and robust solutions for many complex problems (Blum and Merkle, 2008; Panigrahi et al., 2011). Many swarm intelligence algorithms based on different natural swarm systems have been presented and successfully applied in several real-life applications. Particle swarm optimisation (Eberhart and Kennedy, 1995), Ant colony optimisation (ACO) (Dorigo and Stützle, 2004) and cuckoo search algorithm (CS) (Yang and Deb, 2009) are some well-known swarm intelligence algorithms.

Many metaheuristic algorithms have been utilised to solve association rules mining problem. The main reason for employing metaheuristic algorithms for mining such rules is that these algorithms unlike classical methods looking only for the best rules. For this purpose, a new algorithm based on discrete CS metaheuristic is proposed for exploring high-level association rules.

The rest of this paper is organised as follows: In section two, related works are discussed. Section three presents a brief explanation of CS algorithm. The DCS-ARM algorithm is proposed in section four. In section five, our work is evaluated experimentally. Finally, we conclude with a summary in Section 6.
2 Related work

One of the most well-known algorithms for mining association rules is Apriori algorithm (Agrawal et al., 1993). Apriori operates in two stages. The first stage is the generation of frequent itemsets. Frequent itemsets are discovered from all possible item-sets by using support count measure and a user-defined threshold (Minsup). The second stage consists of generation the rules from those frequent itemsets. However, Apriori algorithm has mainly two limits. Firstly it is very time consuming since it does multiple scans over the database to find frequent itemsets. Furthermore, it extracts association rules from them. Secondly, it mines too many association rules which are not appropriate for a given user.

Another popular algorithm for association rule mining is the FP-growth algorithm (Han et al., 2004). FP-growth uses a divide and conquer strategy. It requires two scans on the database. During its first database scan, a list of frequent items is computed and sorted by frequency in descending order (F-List). The database is compressed into a compact data called FP-tree. In the second scan, FP-growth starts to mine the FP-tree for each item whose support is larger than the minimum support by recursively building its conditional FP-tree.

It was Mata et al., (2001, 2002), who first proposed the use of the evolutionary algorithm in association rule mining. They proposed new algorithms for ARM based on genetic algorithm named GENAR and GAR. In GENAR an evolutionary methodology was used to prevent the individuals from tending to the same solution (rule). GAR algorithm is developed to find the best frequent itemsets that are, frequent itemsets with the best support based on evolutionary algorithm principle, but requires another algorithm to extract the rules depart from the frequent itemsets.

In Yan and Zhang (2009) the authors developed a new algorithm called ARMGA where relative confidence considered as the fitness criteria. The primary limit of ARMGA algorithm is the generation of many rules with high fitness quality without regard to minimum support and minimum confidence constraints. In Wang et al. (2011) an adaptive genetic algorithm named AGA for extracting association rules is proposed. The two most significant differences between ARMGA and AGA are the mutation and crossover operators.

Won and McLeod (2012) proposed hierarchical association rule categorisation (HARC) system. HARC works by combining a large dataset from different sources. First this approach generates association rules using the Apriori, then, by applying domain ontology, the original rules are merged and generalised to multi-level association rules. Finally by using relevance measure original rules are categorised so that the user can search or analyse the data by giving various conditions that fit his needs. In Sarath and Ravi (2013) binary particle swarm optimisation (BPSO) based association rule miner is proposed. The algorithm generates the top M rules from the dataset that is the best rules with high fitness quality, where M is the user defined parameter without specifying the minimum support and confidence threshold. In Djenouri et al. (2012), two new algorithms based on genetic metaheuristic and bees swarm optimisation are proposed named IARMGA and BSO-ARM respectively. These two algorithms have been successful in solving the problem of not admissible solutions. After that, a new version of the BSO - ARM with new encoding method and three different strategies for the determination of Search Area is proposed in Djenouri et al. (2014).
3 CS algorithm

CS is a metaheuristic search algorithm that was proposed by Xin-She Yang and Suash Deb in 2009 for solving optimisation problems. This metaheuristic algorithm is inspired by the obligate brood parasitic behaviour of some cuckoo species in combination with the Lévy flight behaviour of some birds and fruit flies (Yang and Deb, 2009).

At the most basic level, female cuckoos lay their eggs in the nests of another species to let host birds to hatch and brood young cuckoo chicks. If the host bird discovers that the eggs are not its own, it will either destroy the egg or abandon the nest altogether. So, to minimise the probability of leaving eggs by the host birds and raising the likelihood of having a new cuckoo, cuckoos (female, male, and young) use several strategies (Payne et al., 2005).

To apply this as an optimisation tool, (Yang and Deb, 2009) used the following idealised rules:

- Each cuckoo lays one egg at a time and dumps it in a random nest.
- A fraction of the nests with high-quality eggs will carry over to the next generation.
- The nest’s number is fixed, and the host bird can discover an alien egg with probability \( p_a \in (0, 1) \). If this happens, the host bird can either get rid of the egg or relinquish the nest, thereby building a new nest in a new location.

3.1 Basic CS

For simplicity, this last presumption could be approximated by the fraction \( p_a \) of the \( n \) nests which are replaced by new nests (with new random solutions). The use of Lévy flights in CS algorithm is considered the most important component of algorithm since it employs Lévy flights for both local and global searching) (Yang and Deb, 2009). Based on these three rules, the basic steps of the CS can be summarised as shown in Algorithm 1 (Yang and Deb, 2009).

In the standard CS algorithm, new solutions are generated for a cuckoo via Lévy flights equation (1):

\[
x_i^{(t+1)} = x_i^{(0)} + \alpha \odot \text{Lévy}
\]

where \( \alpha (\alpha > 0) \) is the step size. Generally, \( \alpha \) should be related to the scales of the problem of interest. In most cases, we can use \( \alpha = 1 \). The product \( \odot \) means entry-wise multiplications. This entry-wise product is similar to those used in PSO, but here the random walk via Lévy flight is more efficient in exploring the search space as its step length is much longer in the long run (Yang and Deb, 2009).

The step length follows the Lévy distribution:

\[
\text{Lévy}(t, \beta) \sim t^{-\beta} \quad 1 < \beta \leq 3
\]

Lévy flights, named by the French mathematician Paul Lévy, essentially provide a random walk that is characterised by a series of instantaneous steps drawn from a probability density function which has a power-law step length distribution with a heavy tail.
Algorithm 1 CS via Lévy flights

1 Begin
2 Objective function \( f(x), x = (x_1, ..., x_d)^T \)
3 Generate initial population of \( n \) host nests \( x_i \) \((i = 1, 2, ..., n)\)
4 While \((t < \text{MaxGeneration}) \) or \((\text{stop criterion})\)
5 Get a cuckoo randomly by Lévy flights
6 Evaluate its quality/fitness \( F_i \)
7 Choose a nest among \( n \) (say, \( j \)) randomly
8 If \((F_i > F_j)\),
9 Replace \( j \) by the new solution;
10 End
11 Fraction \((p_a)\) of worse nests is abandoned, and new ones are built;
12 Keep the best solutions (or nests with quality solutions);
13 Rank the solutions and find the current best
14 End while
15 Postprocess results and visualisation
16 End

As mentioned above some of the new solutions are created by Lévy flight around the best solution gained so far, this will clearly speed up the local search (Yang and Deb, 2009). To avoid the system from being restricted in a local optimum, the algorithm generates a substantial fraction of the new solutions by far field randomisation, and their sites must be away enough from the current best solution. Although from quick look CS algorithm sounds to be similar to hill-climbing in combination with some big scale randomisation. However, there are some considerable differences. Firstly, CS is a population-based algorithm, similar to genetic algorithm GA and particle swarm optimisation PSO, but it employs sort of elitism and/or selection similar to that of harmony search. Secondly, the randomisation is more effective as its step length is much longer in the long run (Bacanin, 2011). However, an advantage of CS over PSO is that the number of parameters to be adjusted is less. This benefit makes CS much easier to be adapted for a wider class of optimisation problems. Obviously, this parameter is the fraction of nests to abandon \( p_a \). (Yang and Deb, 2009) indicated that \( p_a \) did not highly affect the convergence rate. Hence, they suggested setting \( p_a = 0.25 \) (Walton et al., 2011).

### 3.2 Discrete CS

In the above discussion, CS is limited in real number space. However, discrete binary encoding of CS algorithm (DBCS) is recently proposed in Zheng et al. (2012). Two main differences between basic CS algorithm and DBCS algorithm can be referred. First, the nests are represented by binary variables; second, the Lévy flights are transformed into the change of probability. This operation can be prepared by using sigmoid limiting transformation and the random value \((\text{rand})\) drawn from a uniform distribution \( U(0,1) \). Equations (3) and (4) describe this method more formally:
In basic CS, each cuckoo adjusts its nest according to the Lévy flight in DBCS algorithm, since each nest’s position is either 0 or 1. The cuckoo adjusts its nest according to the S(Lévy) and a random value of each bit in a particle. The higher value is more likely to choose 1. Moreover, lower value favours the 0 choices as equation (4) suggested.

4 Solving association rules mining with CS

The process of adapting CS to solve association rules mining (ARM) problem focuses fundamentally on the following parts:

4.1 Rule representation

The first task is to represent an association rule as individual (nest). There are two approaches for representing a rule. One is Pittsburgh approach, where each one represents a set of rules and the second approach is the Michigan approach, where each individual represents a separate rule. In this work, Michigan approach is followed.

Assume there is N number of items in the dataset. Each nest (X) is a vector of N items, and their locations are defined as follows (Djenouri et al., 2014)

1. If the item i is not in the nest then \( X[i] = 0 \).
2. If the item i belongs to the antecedent part of the nest then \( X[i] = 1 \).
3. If the item i belongs to the consequent part of the nest then \( X[i] = 2 \).

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

\[
X_1 = \{0, 1, 0, 0, 1, 2, 0\}
\]

represents the rule

\[
\text{Rule 1: } t_2, t_5 \rightarrow t_6.
\]

This representation has the advantage of easy separating between the antecedent and the consequent part of the rule consequently; the fitness function can be calculated more practically.

4.2 Initial population

The initial population is generated from initial nest \( X_i \) by adding 1 to two random bits of \( X_i \). However, if any bit value exceeds 2, the target bit value will replace by 0. Based on this simple operation, \( N \) of host nests is created.
4.3 Fitness function

The goal of association rule mining is to discover all association rules having support and confidence not less than the given value of the threshold. Let a and b are two empirical parameters; the nest X will be evaluated by the following fitness function (Djenouri et al., 2014):

\[
\text{Fitness}(X) = a \times \text{confidence}(X) + b \times \text{support}(X)
\]

if \( \text{Confidence}(X) \geq \text{MinConf} \) and \( \text{Support}(X) \geq \text{Minsup} \)

\[
\text{Fitness} = -1 \text{ otherwise}
\]

4.4 Decomposition strategy

In discrete binary encoding of CS (DBCS) algorithm, each cuckoo adjusts its nest according to equation (4). However; the nest’s position in our representation method can take one of three possible values while the strategy of (DBCS) is defined exclusively for binary encoding with zero and one only. Hence; a new method named Decomposition strategy is suggested as an alternative approach which instead of adjusting nest’s positions it creates two new nests from nest X by decomposing it. Assuming that N is the length of the nest (X), Rn is an integer number chosen randomly between \((N/2, N/4)\) and R is a random number between 0 and 1 inclusive by 0.05 increments. Then; the decomposition point (DP) of X is chosen by comparing the sigmoid limiting transformation of Lévy flight S (Lévy) equation (3) with R as follow:

\[
\text{DP} = \begin{cases} 
\frac{N}{2} & \text{if } S(\text{Lévy}) \geq R \\
R_n & \text{Otherwise}
\end{cases}
\]

After the decomposition point is determined the process of creation new nests with the same length of the original nest X; say, Xa and Xb are accomplished by performing the two steps below:

1. The first nest Xa is created from the original nest X by copying the items of X located before the decomposition point (DP), while the rest \(N-DP\) positions of Xa is padding by zeros. After that; the entire Xa items are shuffled.

2. The second nest Xb is created from the original nest X by copying the items of X located beyond the decomposition point (DP), while Xb positions equal to \(Dp\) is padding by zeros. Finally, the entire Xb items are shuffled.

The last step of this strategy includes comparing the fitness values of the original nest with that of the two new nests so that the nest with high fitness value is selected otherwise it returns Xa. Figure 1 introduces an example that illustrates the scenario of creating the nests using this strategy by considering the original nest \(X = 12112022\) and \(DP = 3\).
**Figure 1** The creation of new nests by the decomposition strategy

![Diagram of nest creation](image)

**Algorithm 2** DCS-ARM algorithm

1. **Input:** β, pa, MaxGeneration, a, b, Transactional Dataset, Minsup, Minconf
2. **Output:** Set of Asso. Rules
3. Xi ←, the initial nest, generated randomly
4. While (t < MaxGeneration) or (stop criterion)
5. Begin
6. Generate the initial population of n host nests from Xi
7. For each X in population
8. Evaluate its fitness Fx
9. Get Xa and Xb from X by decomposition strategy
10. Evaluate fitness of Xa (Fxa) and Xb (Fxb)
11. If Fx is greater than Fxa and Fxb
12. X is not replaced
13. Else If Fxb greater than Fxa
14. Replace X by Xb
15. Else
16. Replace X by Xa
17. End if
18. End for
19. Fraction (pa) of worse nests are deserted, and new ones are built // where pa refers to
20. Save the best nest in the List of Best Nests;
21. Xi← the best nest in List of Best Nests // best of the best
22. t ← t + 1
23. End while
24. For each nest X in List of Best Nests do
25. Generate the rule from S
26. End for
4.5 DCS-ARM algorithm

The initial nest (Xi) is generated randomly. From the current Xi nest, the initial population of n nests is produced as mentioned in Section 5.2. After that, the decomposition strategy is performed on each nest X in the initial population to produce a new population. The next step is the sorting of the nests in the population according to their fitness function. The fraction pa of the worst nests is substituted by new nests which are created from the best pa nests in the population. This process is made by changing two bits in a random way for a given nest X. At this point, the current best nest is selected to be the next Xi nest and kept in a list named the List of Best Nests. 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 List of Best Nests and visualised to the user. Algorithm 2 describes more formally the proposed method.

Table 1 The specifications of datasets

<table>
<thead>
<tr>
<th></th>
<th>Zoo</th>
<th>Primary tumour</th>
<th>German credit</th>
<th>Chess</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item size</td>
<td>36</td>
<td>31</td>
<td>112</td>
<td>75</td>
</tr>
<tr>
<td>Transaction size</td>
<td>101</td>
<td>336</td>
<td>1,000</td>
<td>3,196</td>
</tr>
</tbody>
</table>

5 Experiments and results

For the sake of testing the proposed algorithm and comparing it with BSO-ARM algorithm (Djenouri et al., 2014), four datasets were utilised from Java open-source data mining found at http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php and constraint programming for itemset mining at https://dtai.cs.kuleuven.be/CP4IM/datasets. The attributes of both primary tumour and chess datasets are categorical while zoo and German credit datasets contain integer and categorical attributes. However, those datasets are in annotated transaction format where each line represents only a single transaction. A transaction is a space-separated list of items. The specifications of those data sets are revealed in Table 1.

Table 2 The parameters for run DCS – ARM

<table>
<thead>
<tr>
<th></th>
<th>Zoo</th>
<th>Primary tumour</th>
<th>German credit</th>
<th>Chess</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop size</td>
<td>30</td>
<td>30</td>
<td>100</td>
<td>70</td>
</tr>
<tr>
<td>pa</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>β</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 3 The mean number of valid rules of two algorithms

<table>
<thead>
<tr>
<th></th>
<th>Zoo</th>
<th>Primary tumour</th>
<th>German credit</th>
<th>Chess</th>
</tr>
</thead>
<tbody>
<tr>
<td>N.rule DCS – ARM</td>
<td>93</td>
<td>91</td>
<td>93</td>
<td>97</td>
</tr>
<tr>
<td>N.rule BSO-ARM (next strategy)</td>
<td>60</td>
<td>25</td>
<td>14</td>
<td>48</td>
</tr>
<tr>
<td>N.rule BSO-ARM (Modulo strategy)</td>
<td>85</td>
<td>67</td>
<td>2</td>
<td>89</td>
</tr>
</tbody>
</table>
The current experimental study was conducted in an environment of Microsoft Windows 7 using a laptop PC with Intel core–Intel(R) Celeron, the clock speed of 1.70 GHz and 4GB RAM. The programs were coded in Microsoft Visual Basic 4.0 Client Profile using Visual Basic.net. The comparison with BSO-ARM algorithm was done by using the two strategies of the determination of search area in BSO-ARM which are the next and modulo strategies, where bee number is set to 5 in both cases.

In Table 2 the parameters of running the DCS-ARM algorithms are presented. However, the fitness function parameters in two algorithms are established as follows: the coefficients are $a = 0.6$, $b = 0.4$; the minimum support threshold = 0.1; minimum confidence threshold = 0.7 in the four datasets.

The reported results are an average of 120 executions where the number of iterations is set to 100. Table 3 shows the mean number of valid rules obtained from DCS – ARM and BSO-ARM. In this table, we express that the numbers of valid rules obtained by the DCS-ARM algorithm from the four datasets are greater than those obtained by BSO-ARM (next and Modulo strategies).
Association rules mining using cuckoo search algorithm

Figure 3  The average support and confidence of the obtained rules from primary tumour dataset, (a) BSO-ARM (next strategy) (b) BSO-ARM (Modulo strategy) (c) DCS-ARM (see online version for colours)

Figures 2–5 illustrates the average support and confidence of the final population rules for every 25 repetitions of the two algorithms. These figures show that regarding to confidence; the results of the two algorithms are fairly equal upon the four datasets.

However with regard to support, Figure 2 shows that the support of rules obtained by both algorithms in zoo dataset are relatively close. On other hand DCS-ARM found higher support rules in the three other datasets than those found by BSO-ARM algorithm as exposed in Figures 3, 4 and 5 respectively. The fitness average of the final population without repetitive rules for every 25 repetitions of both BSO-ARM (with Next strategy and Modulo strategy) algorithm and the DCS-ARM algorithm is illustrated in Figure 6. Although this figure shows that DCS-ARM and BSO-ARM (Next strategy) are not trapped in local optimum in all datasets; DCS-ARM yields high fitness values while BSO-ARM (Modulo strategy) strategy stays on the local mode in all datasets and sometimes the evolution in solutions is deteriorating dramatically.
Table 4  The mean size of the final population rules of two algorithms

<table>
<thead>
<tr>
<th></th>
<th>Zoo</th>
<th>Primary tumour</th>
<th>German credit</th>
<th>Chess</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of rule DCS – ARM</td>
<td>4.7</td>
<td>3.9</td>
<td>6</td>
<td>6.6</td>
</tr>
<tr>
<td>Size of rule BSO-ARM (Next strategy)</td>
<td>4.2</td>
<td>5.2</td>
<td>7</td>
<td>6.2</td>
</tr>
<tr>
<td>Size of rule BSO-ARM (Modulo strategy)</td>
<td>8.4</td>
<td>8.2</td>
<td>11.9</td>
<td>9</td>
</tr>
</tbody>
</table>

Figure 4 the average support and confidence of the obtained rules from German credit dataset, (a) BSO-ARM (next strategy) (b) BSO-ARM (Modulo strategy) (c) DCS-ARM (see online version for colours)

All these interested results can be explained as follows:

- In case that BSO-ARM is exploring the search area by Next strategy, the algorithm will have a limited capability to inspect search area in large scale. Thereby the evolution in the quality of the solutions is limited as well.
- In case that BSO-ARM is 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 iterations, the number of items in solution will increase. As a consequence, the fitness quality and the number of valid rules are decreasing.

Table 4 offers the average size of final population rules.
BSO-ARM has utilised a certain strategy to upgrade the level of diversification. However, this strategy raises the risk of choosing a bad solution to be the reference solution. Hence, the number of valid rules and the fitness can be influenced improperly.

Figure 5  The average support and confidence of the obtained rules from chess dataset, (a) BSO-ARM (next strategy) (b) BSO-ARM (Modulo strategy) (c) DCS-ARM (see online version for colours)

On another hand, the proposed algorithm can overcome the problems above by the accomplishing the two following setups:

- The good diversification of DCS-ARM algorithm that results from both the decomposition strategy which can be considered as the core step of the proposed algorithm and the operation of generating the population which enables the algorithm to explore the search space on a global scale.

- The Decomposition strategy also plays a key role in avoiding the accumulation of items which can be caused by the previous operation of the algorithm. Furthermore, by using the shuffling mechanism to create new solutions this strategy is contributing toward the focus on the local region where the optimality may be close to another word promote the intensification level.
The step of building the new nests instead of abandoned ones from the best nests helps the algorithms in improving the quality of the new population.

Table 5 demonstrates that BSO-ARM is more expensive in term of CPU-time because of the neighborhood search operation and the diversification strategy. Of course, the CPU time varies w.r.t to different datasets.

Table 5 The execution time in second of two algorithms

<table>
<thead>
<tr>
<th></th>
<th>Zoo</th>
<th>Primary tumour</th>
<th>German credit</th>
<th>Chess</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCS – ARM</td>
<td>11</td>
<td>53</td>
<td>756</td>
<td>1,684</td>
</tr>
<tr>
<td>BSO-ARM (next strategy)</td>
<td>50</td>
<td>247</td>
<td>3,071</td>
<td>5,224</td>
</tr>
<tr>
<td>BSO-ARM (Modulo strategy)</td>
<td>53</td>
<td>203</td>
<td>3,134</td>
<td>7,068</td>
</tr>
</tbody>
</table>

Figure 6 The average fitness of the final population for every 25 repetitions of the run of DCS-ARM and BSO-ARM over datasets, (a) zoo dataset (b) primary tumour dataset (c) German credit dataset (d) chess dataset (see online version for colours)
6 Conclusions

In this work, an algorithm for mining association rules based on CS algorithm named DCS-ARM is applied for the extraction of high-quality association rules. The conducted results indicate that DSC-ARM outperforms BSO-ARM in terms of the number of valid rules and their quality. Moreover, DCS-ARM is less time consumption than BSO-ARM. Indeed, when the number of transactions and the number of items becomes so large, the CPU time increases dramatically, for this reason, the future trend is a parallel DCS-ARM to be investigated.

References


