Journal of Intelligent & Fuzzy Systems 32 (2017) 4319–4330
DOI:10.3233/JIFS-16963
IOS Press

Modified binary cuckoo search for association rule mining

Uroš Mlakar*, Milan Zorman, Iztok Fister Jr. and Iztok Fister
Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia

Abstract. This paper proposes a modified single-objective binary cuckoo search for association rule mining (MBCS-ARM). The proposed algorithm includes a novel representations of individuals, which tackles the problems of large dimensionality with an increasing number of attributes. The MBCS-ARM also supports the mining of rules, where intervals of attributes can either be negative or positive. It uses an objective function composed of support and confidence weighted by two parameters, which control the importance of each measure in the found rules. It is tested on eight publicly available databases, while also compared to several single-objective evolutionary algorithms, and traditional algorithms, all found in the KEEL software tool. The experiments show promising results of the MBCS-ARM, compared to other algorithms, by producing rules, which are interesting, simple, and also easy to understand, which is of great importance in domains like medicine.

Keywords: Data mining, association rule mining, Apriori, swarm intelligence, modified binary cuckoo search

1. Introduction

In the modern world, there is a huge amount of data stored in real-world databases and this trend continues to grow daily. There are different kinds of databases such as medical, scientific, financial, and marketing transaction data. Many of them hold large amounts of data. Therefore, it has become an issue how to effectively analyze these data and find the interesting information hidden beneath. In the past decade, the most successful method for solving this kind of problems has been data mining, or to be more specific: classification, clustering, regression, and association rule mining [6].

Recently association rule mining has again gained a lot of attention within the scientific community. This is a popular method for discovering relations between attributes in large databases. The goal of the method is to find those rules having the high level of interestingness based on various measures. The pioneer of association rule mining is Agrawal [1, 2] who presented the idea of discovering regularities between products in large-scale transaction databases. An Apriori algorithm was proposed, which has become a standard approach in association rule mining. Normally, the algorithm selects the mined association rules according to interestingness measures, like support and confidence identifying the most important relationship of attributes in transaction databases.

Since the datasets are getting larger and more complex [16], the traditional algorithms, like Apriori, usually face problems of high computational complexity when generating association rules. To overcome this problem, researchers have proposed new stochastic population-based nature-inspired algorithms that treat the rule mining process as an optimization process by applying search heuristics to the underlying optimization problem.

The heuristics for association rule mining normally involve one or more interestingness measures

*Corresponding author. Uroš Mlakar, Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia. Tel.: +386 41 364 960; E-mail: uros.mlakar@um.si.

depending on whether we approach the as a single-objective or multi-objective problem. The measures are disguised as fitness values guiding the optimization process towards the more promising regions of the search space, where more association rules of better quality can be found.

Roughly speaking, we can divide the stochastic nature-inspired algorithms into two large groups: evolutionary algorithms (EAs) and swarm intelligence (SI) based algorithms. Both families are population-based algorithms, which generate new solutions by applying appropriate variation operators (e.g., crossover, mutation, etc.).

Many single and multi-objective methods have been developed based on EAs, as follows: genetic algorithms (GA) [11], genetic programming (GP) [20], differential evolution (DE) [31]; and SI: particle swarm optimization (PSO) [19], artificial bee colony (ABC) [18], bat algorithm (BA) [33], and cuckoo search (CS) [34].

Luna et al. [22] developed a GP based algorithm for mining rare class association rules that cannot be found using the traditional data mining algorithms. Song et al. [30] discuss the effectiveness of a multi-objective based binary BA, comparing it to single objective binary PSO and BA, and the Apriori algorithm. They conclude that the proposed algorithm is feasible and highly effective. Yan et al. [32] designed a genetic algorithm-based strategy for identifying association rules without specifying the minimum support. Confidence is used as a fitness function, while the FP-Tree algorithm [12] is implemented to improve the algorithm efficiency. In [10] Ghosh et al. presented a Pareto based genetic algorithm for extraction of interestingness rules on large market-basket type dataset. They use three commonly used measures to evaluate individuals: support, comprehensiveness, and interestingness. Other interesting approaches are described in the following works [5, 7, 8, 14, 28].

Another multi-objective genetic algorithm is described in [25], where the problem of dealing with numerical data is tackled. Same rule measures are used as in [10]. Heraguemi et al. [13] proposed a bat algorithm, which produced better results when compared to the FP-Growth algorithm [12]. Their method relies on the minimum support and confidence that must be supplied by the user. Sarath et al. [28] proposed a binary particle swarm optimization algorithm (BPSO), for operating on transactional databases. With a product of support and confidence as the fitness function, they evaluate their method on three transactional datasets, and a real life bank dataset. Based on the results, they conclude that BPSO is a good alternative to Apriori and FP-growth algorithms.

Algorithms based on binary encoding of individuals often suffer on the dimensionality of the problem when confronted with datasets introducing attributes with many possible attribute values. These methods are also mainly used for working on transactional databases, while not appropriate for dealing with datasets that include categorical or even numerical values. EA or SI-based algorithms also focus mainly on discovering positive association rules, while ignoring the potential rules that involve negation of attributes (an exception is the method in [3]).

With this in mind, the paper proposes a modified binary cuckoo search for association rule mining (MBCS-ARM) based on a binary representation of individuals, where each individual encodes the corresponding association rule. Thus, all attributes in a database are encoded as binary strings with additional three control bits determining the presence/absence of the attribute in the corresponding association rule, when the attribute is part of antecedent/consequent part of the rule, and when the attribute values are taken from positive/negative domain of attribute values. The MBCS-ARM was tested on eight publicly available datasets, where seven of them are available in the KEEL tool [4] or the UCI machine learning repository [21], and one dataset supplied by a professional cyclist (produced for our research purpose) [17]. The performance of the proposed algorithm was compared to the four single-objective EAs for association rule mining, and three traditional algorithms for solving the same problem.

In summary, the proposed MBCS-ARM algorithm brings the following two key novelties: (1) the novel (i.e., binary) representation of individuals, and (2) the application of a single-objective modified binary CS algorithm to association rule mining.

The remainder of the paper is organized as follows. In Section 2, a formal definition of association rule mining is given, along with some descriptions of traditional algorithms. Section 3 presents the original and the proposed modified binary CS algorithms. Section 4 deals with describing the experimental environment (i.e. parameter settings, and used databases), and a brief review of EAs used for association rule mining. In Section 5, the results are depicted, while in Section 6, the paper is concluded outlining the directions for the future work.

2. Association rule mining

Mathematically, the association rule mining is defined as follows. Let us suppose a set of attributes $I = \{a_1, \ldots, a_m\}$ called items and a set of transactions $D = \{t_1, \ldots, t_n\}$ called a dataset, where each transaction has a unique identification and contains a subset of items $t_i \subset I$ called an item-set, and m denotes the number of items and n the number of transactions. Then, the association rule is an implication $X \rightarrow Y$, where X and Y are two item-sets and it holds $X \cap Y = \emptyset$. The criteria to identify the most important relationships in the specific rule are estimated according to two interestingness measures as follows:

1. Minimum support that is defined as proportion of transactions containing X and Y, and the total number of transactions in database, as follows:

$$support(X \rightarrow Y) = \frac{|\{t_i | t_i \in X \wedge t_i \in Y\}|}{\sum_{i=0}^{n} t_i}, \quad (1)$$

2. Minimum confidence that is defined as proportion of transactions that contains X also contains Y, as follows:

$$confidence(X \rightarrow Y) = \frac{support(X \cup Y)}{support(X)} \quad (2)$$

There are many traditional association rule mining algorithms, but the most used are Apriori, Eclat, and FP-Growth. Therefore, a brief description of the mentioned algorithms follows in the next sections.

2.1. Apriori algorithm

The Apriori algorithm proposed by Agrawal [2] is the most representative association rule mining algorithm. It works in two steps, where the first step is dedicated for generating the most frequently arisen item-sets in a transaction database. They are selected from all possible item-sets using the parameter minimum support as defined by Equation (1). After the most frequently arisen item-sets are selected, those occurring less than the specified minimum support are removed. From these item-sets, the rules are selected in the second step using the parameter minimum confidence as defined by Equation (2). Thus, all rules having the lower confidence than the one provided are removed.

2.2. Eclat

First introduced by Zaki et al. [35] in 1997, the Eclat algorithm finds the frequently arisen item-sets by a depth first search on the subset lattice and determines the support of these item-sets by intersecting transaction lists. It is suitable for parallel processing with local enhancing properties. For more information readers are referred to [35].

2.3. FP-Growth

FP-Growth algorithm was proposed by Han et al. [12] in 2000 with an intention to overcome the bottlenecks of Apriori. It uses an extended prefix-tree structure for storing crucial information about frequently arisen patterns found in the transaction database. All item-sets are generated by only two passes over the whole database. More information can be found in [12].

3. Cuckoo search algorithm for association rule mining

3.1. Original cuckoo search

Cuckoo search (CS) is a stochastic population-based nature-inspired algorithm that has been introduced by Yang and Deb in 2009. CS belongs to the SI-based algorithms [9], and it is inspired by natural behavior of some cuckoo species and their brood parasitism. Typically, some cuckoo species lay their eggs in the nests of other birds in order to care for them as if they were their own. To trap the behavior of cuckoos in nature on the one hand and to adapt it to be suitable for using as the computer algorithm on the other hand, authors idealized three rules [34]:

- Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest.
- The best nests with high-quality eggs will be carried over to the next generations.
- The number of available host nests is fixed and the egg laid by a cuckoo may be discovered by the host bird with a probability $p_a \in (0, 1)$. In this case, the host bird can either get rid of the egg, or simply abandon the nest and build a completely new nest.

Initially, a solution corresponding to a cuckoo egg placed inside the nest is randomly generated within

the search space. Mathematically, the position of a nest is defined as:

$$\mathbf{x}_i^{(t)} = \{x_{i,j}^{(t)}\}, \text{ for } i = 1, \ldots, NP \text{ and } j = 1, \ldots, D, \tag{3}$$

where NP denotes the number of cuckoo nests, D the dimensionality of the problem, and t the generation number. In each iteration, the nest is updated using the the following equation:

$$\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} + \alpha L(s, \lambda), \tag{4}$$

where

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0). \tag{5}$$

Equation (4) describes the local random walk that is mainly intended for exploitation of the current solutions. This random walk is governed by Lévy flight distribution expressed as $L(s, \lambda)$ and scaled by the scaling factor $\alpha > 0$ of the step size s.

The pseudocode of the CS is presented in Algorithm 1 (see Fig. 1 for the flowchart).

Algorithm 1 Cuckoo search algorithm

Input: Population of nests $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,D})^T$ for $i = 1 \ldots NP$, MAX_FE.
Output: The best solution $\mathbf{x}_{best}$ and its corresponding value $f_{min} = \min(f(\mathbf{x}))$.

```
generate_initial_host_nest_locations;
FE = 0;
while termination_condition_not_meet do
    for i = 1 to NP do
        u_i = generate_new_solution(x_i);
        f_trial = evaluate_the_new_solution(u_i);
        FE = FE + 1;
        j = ⌊rand(0, 1) * NP + 1⌋;
        if f_trial < f_j then
            x_j = u_i; f_j = f_trial; // replace the j-th random selected solution
        end if
        if rand(0, 1) < p_a then
            init_nest(x_worst);
        end if
        if f_trial < f_min then
            x_best = u_i; f_min = f_trial; // replace the global best solution
        end if
    end for
end while
```

3.2. Modified binary cuckoo search algorithm

Since the CS was applied to association rule mining, it seems that a binary representation of solutions is more promising to achieve the better results. In line with this, the modified binary CS (MBCS) is proposed, where the solution is represented as a n dimensional boolean lattice, in which the solutions are updated across the corners of a hypercube [27].

In order to prepare the MBCS for association rule mining, the following five steps need to be preformed:

- identifying an attribute domain,
- rule representation,
- candidate solution generation,
- fitness evaluation,
- association rule mining using MBCS.

In the remainder of the paper the mentioned steps are discussed in detail.

3.2.1. Identifying an attribute domain

Attributes in MBCS are represented as binary strings. The size of the strings is determined with the number of attributes. It is well known that the maximum 2^n different attributes can be represented using a binary string of length n.

In order to identify the number of attributes, an analysis of a transaction database D needs to be performed. As a result, sets of attribute values $a_i = \{\alpha_{i,1}, \ldots, \alpha_{i,m_i}\}$ are defined after performing the analysis that contain the unique attribute values $\alpha_{i,j}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, m_i$, where m is the number of attributes and m_i the number of attribute values for attribute a_i. Let us notice that data in a database can either be categorical, numerical or transactional. However, if data are continuous, a discretization must be performed.

Algorithm 2 Generating new solutions

Input: A solution $\mathbf{x}^{(t)} = (x_{i,1}, \ldots, x_{i,D})^T$.
Output: Binary vector $\mathbf{x}^{(t+1)}$.

```
for i = 1 to D do
    u_i = x_i^(t) + αL(s, λ)
    S_b = 1 / (1 + e^(−u_i))
    if rand(0, 1) < S_b then
        x_i^(t+1) = 1
    else
        x_i^(t+1) = 0
    end if
end for
```

3.2.2. Rule representation

As found in literature, there are two well established encodings for representing association rules in EA domain. The first is the Pittsburgh approach [15],

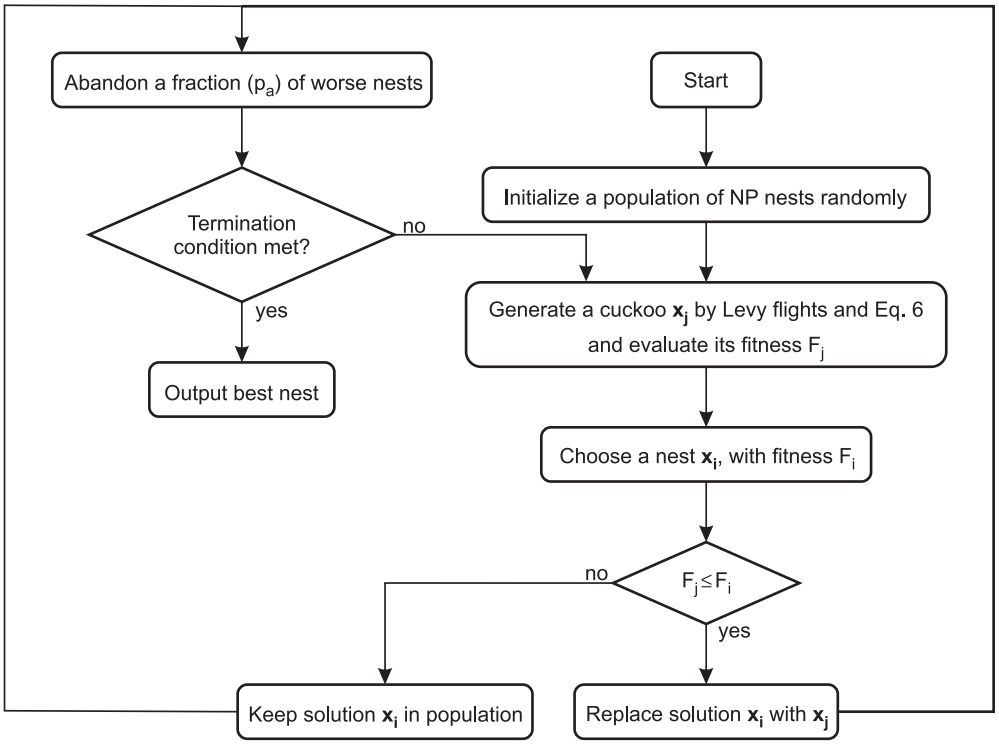

Fig. 1. Flowchart of the CS algorithm.

where each individual represents a set of association rules, while by the second so called Michigan approach [15], each individual represents the separate association rule. The main downfall of both approaches is that the search space of the EA grows quite big by increasing the number of attributes as well as the number of attribute values. In this work we present a novel representation of individuals for association rule mining, which tries to overcome this problem.

Each individual represents a separate rule as in the Michigan encoding approach, where each attribute of the rule is encoded with three control bits as follows:

- the first control bit indicates presence/absence of this attribute in the corresponding association rule,
- the second control bit denotes, whether the attribute belongs to the antecedent (bit one) or the consequent of the rule (bit zero),
- the third control bit defines, whether the attribute values are taken from a positive/negative attribute domain, and
- the actual value of the corresponding attribute.

It is obvious that, when the first control bit of an attribute is set then it is present in the corresponding association rule. Contrary, when this bit is not set means that the attribute does not belong to the rule. The similar is true for the second bit. Third control bit is reserved for defining the positive or negative attribute domain, where the positive domain means, that the actual value of the attribute is used and negative whether all other values except the actual value are used. Let us assume an attribute $a = \{1, 2, 3\}$. Then, the positive domain for an attribute value $\{2\}$ is the same value, i.e., $\mathcal{D}_a^P = \{2\}$, while the negative domain is expressed as $\mathcal{D}_a^N = \{1, 3\}$.

Finally, the attribute value is represented as a binary value, where the number of bits depends on the minimum value of bits m_i required to represent the maximum number of different attribute values 2^{m_i} for an attribute a_i. However, when using this individual representation, a problem of redundant values can occur when an invalid attribute value is represented.

For instance, the attribute a_2 has five different attribute values. That means that it needs three bits for representing the attribute values, in other words $\{0, 1, 2, 3, 4, 5, 6, 7\}$. Consequently, only first five attribute values are needed for representing attribute values, while other three values $\{5, 6, 7\}$ are redundant. However, these redundant values are simply

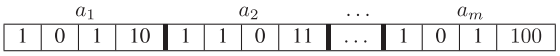

Fig. 2. Representation of the i-th individual, where each attribute is presented using 3 control bits, and the actual attribute value. For example the control bits of attribute a_1 are 1-0-1, which means that the attribute is present in the rule, it is part of the antecedent, and the interval of the attribute is negative. The whole example can be interpreted as follows: the first and last attributes are present in the rule and are a part of the antecedents, while the second attribute is part of the consequent. $((a_1 \neq 2) \wedge (a_m = 4) \wedge \ldots \rightarrow (a_2 = 3) \wedge \ldots)$.

fixed by reinitializing the invalid value to the correct domain in the study.

Each attribute is thus encoded with as little information as possible. Figure 2 depicts this process for easier understanding.

3.2.3. Candidate solution generation

The candidate solution is generated by the original CS according to Equation (4) and this enters in the selection process with randomly selected solution in the current population. If the candidate solution is better than the existing one, it becomes the new population member. However, the generation in MBCS slightly differs from those in the original CS.

The generation of candidate solution in MBCS algorithm is accomplished using Algorithm 2, where the candidate solution is generated using the same Equation (4) as the original CS. However, this equation is conducted on the binary vector in the case of MBCS, where only two values 0/1 are allowed. This values are preserved by calculating the threshold using the following function [27]:

$$S_b = \frac{1}{1 + e^{(-u_i)}}, \tag{6}$$

that determines the corresponding value of the candidate solution randomly. The parameter u_i in Equation (6) is calculated according to the Algorithm 2 (line 2).

3.2.4. Fitness evaluation

Since the proposed method for association rule mining is applied using the MBCS algorithm, a metric for identifying the quality of rules (i.e. a fitness function) must be determined. There are several different fitness functions applied in the literature, varying from simple to very complex. Our method relies on the already mentioned measures of support (Equation (1)) and confidence (Equation (2)) for the mined association rule $X \rightarrow Y$ as follows [13]:

$$f(\mathbf{x}) = \frac{\beta * confidence(X \rightarrow Y) + \gamma * support(X \rightarrow Y)}{\beta + \gamma}, \tag{7}$$

Both confidence and support are weighted by factors β and γ, which control the importance of both measures. Increasing the value β, would result in the algorithm finding rules with an increased confidence. In contrary, increasing the value γ would result in rules favoring the increased support. Depending on the application of rule mining, the user can control the importance of both measures.

3.2.5. Association rule mining using MBCS

A detailed explanation of the proposed algorithm for association rule mining (MBSC-ARM) is given in this subsection. As described in the beginning of Section 3.2, the algorithm consists of two parts. In the first part, the data are mapped from real-coded elements of the CS to binary encoded strings of the MBSC, while in the second part the MBCS is employed for finding the optimal association rules. The pseudo-code of the proposed algorithm can be seen in Algorithm 3.

Algorithm 3 Association rule mining using BCS

Input: maximum number of restarts K
Output: archive of best rules R

```
1:  k = 0
2:  R = {∅}
3:  while k < K do
4:      rule = BCS()
5:      if rule ⊄ R then
6:          R = R ∪ rule
7:          k = 0
8:      else
9:          k = k + 1
10:     end if
11: end while
```

The algorithm is implemented in such a way, that when the termination condition is reached by the algorithm, the best association rule found is put into an archive of solutions. If the same solution is already in the archive, a counter k is increased. When this counter reaches the maximum number of K, the archive becomes the final result of different solutions. The K parameter is supplied by the user.

4. Experimental environment

This section describes the experiment settings, the used databases and PC configuration on which the experiments were conducted.

Table 1
Parameters of BCS for association rule mining

Parameter	Value
NP	30
G	200
p_a	0.25
β	1
γ	1
K	dataset dependent

4.1. Parameter settings

During testing the parameters of MBCS-ARM were as described in Table 1. As can be seen from Table 1, NP denotes the population size, G the maximum number of generations, and p_a is the switching probability. The parameters β and γ are used to weigh the importance of either support or confidence, and lastly K is the number of restarts in a row, which is dataset dependent, i.e. the bigger datasets with more attributes require a smaller number of restarts, because to many rules would be produced otherwise.

4.2. PC configuration

All runs were made on a desktop computer, with the following configuration:

- Processor: Intel(R) Core(TM) i5-4570 CPU @ 3.20 GHz,
- RAM: 16 GB,
- Operating system: Linux Mint 17 Qiana.

The MBCS-ARM was implemented in the C++ programming language.

4.3. Test databases

The datasets listed in Table 2 were used for evaluating the performance of the MBCS-ARM. The characteristics of each database is also presented in terms of the number of transactions, attributes and attribute types. The latter can be one of the following: categorical, integer, and real. All quantitative attributes have been partitioned into four categories (as suggested by [3]), by equal frequency sampling.

4.4. Comparing algorithms

In the experimental part of this work, our algorithm was compared to all single-objective evolutionary, and three traditional association rule mining algorithms found in the KEEL software tool [4]. The evolutionary algorithms are:

Table 2
Datasets used in the study. The abbreviations for attribute types are: C (categorical), I (integer), R (real)

Dataset	No. of instances	Attributes	Attribute type (C/I/R)
Basketball	95	5	0/2/3
Car	1728	7	7/0/0
Cyclist	100	8	8/0/0
House	22784	17	0/7/10
Nursery	12960	9	9/0/0
Quake	2178	4	0/1/3
Stock	950	10	0/0/10
Wine	178	14	0/3/11

- Alatasetal [3],
- EARMGA [32],
- GENAR [23], and
- GAR [24],

while the traditional algorithms are:

- Apriori [2],
- Eclat [35], and
- FP-Growth [12].

All listed evolutionary algorithms (Alatasetal, EARMGA, GENAR, and GAR) are GAs. The Alatasetal algorithm is able to mine both positive and negative intervals for attribute values, without specifying the minimum support and confidence. Another interesting fact is that the initialization is not done randomly, but close to final solutions. The rules are obtained from the last generation of a single run. The EARMGA is also independent of minimum support and confidence. It uses a generalized FP-Tree to improve the algorithm efficiency. Another characteristic of this algorithm is the specification of rule length, which must be supplied by the user. GENAR is able to find rules in numeric datasets by incorporating the lower and upper bounds of each attribute in the solution. Rule length is always the same as the number of attributes, where the last attribute forms the consequent. The used objective function penalizes those solutions, which cover the same records in the dataset. The GAR algorithm is an extension of GENAR, where frequent item sets are firstly discovered, on which the rules are later produced. Like GENAR this algorithm also searches for rules in numeric dataset with no need for discretization.

All traditional algorithms have previously been described in Section 2.

Let us notice, that all algorithms were run with their default parameters, as set in the KEEL tool. Each

algorithm in the comparison was executed 10 times, so the reported results are an average of those 10 runs.

5. Results

In this section, the results of the MBCS-ARM are analyzed and compared to the algorithms in the KEEL tool. Let us notice that two variants of the MBCS-ARM are reported, denoted as MBCS-ARM$^{\pm}$ and MBCS-ARM^{+}, where the first includes the possibility of mining positive an negative intervals of a attribute in association rules, while the latter searches only for positive intervals.

The following experiments have been performed:

- comparison of MBCS-ARM$^{\pm}$ and MBCS-ARM^{+},
- comparison of the best MBCS-ARM variant with evolutionary algorithms in the KEEL tool,
- comparison of the best MBCS-ARM variant with traditional algorithms.

Firstly, both variants of the proposed MBCS-ARM will be compared against each other, then the better one will be compared to the evolutionary algorithms, which were presented in Section 4. The comparison will be made on the basis of the following measures: number of rules produced, support, confidence, coverage (defined as the support of the antecedent), and antecedent length. Secondly the best MBCS-ARM variant will also be compared to traditional association rule mining algorithms, also using the same measures as with the EAs. Then lastly some interesting rules will be presented, obtained with the MBCS-ARM. All reported values are averages of 10 runs.

All experiments are discussed in detail in the following subsections.

5.1. Comparison of MBCS-ARM$^{\pm}$ and MBCS-ARM^{+}

Let us analyze both variants of the MBCS-ARM from Table 3. The following conclusions can be made: the MBCS-ARM^{+} almost always produces more rules than MBCS-ARM$^{\pm}$, and these rules are longer, based on the average number of antecedents. Longer association rules are often harder to understand than those of shorter length. It is also obvious that the MBCS-ARM^{+} produces rules, which in general have lower support than MBCS-ARM$^{\pm}$, which implies those rules may be uninteresting. Also the coverage of the records in the database is way lower in all cases for the MBCS-ARM^{+}, so it is obvious that the MBCS-ARM$^{\pm}$ is the better performing algo-

Table 3
Comparison MBCS-ARM$^{\pm}$ and MBCS-ARM^{+}. The reported results are averages of 10 individuals runs

Algorithm	#Rules	avgSupp	avgConf	coverage(%)	avgAntLen
			Basketball		
MBCS-ARM$^{\pm}$	9.8	0.672	0.891	95.21	1.02
MBCS-ARM^{+}	17.9	0.039	1.000	39.375	2.644
			Car		
MBCS-ARM$^{\pm}$	14.9	0.720	0.967	96.25	1
MBCS-ARM^{+}	24.5	0.118	0.990	72.911	1.895
			Cyclist		
MBCS-ARM$^{\pm}$	32	0.453	0.882	100	1.35
MBCS-ARM^{+}	12.5	0.124	0.985	51.200	2.174
			House		
MBCS-ARM$^{\pm}$	357.7	0.355	0.824	99.99	2.91
MBCS-ARM^{+}	1545.2	0.002	0.999	86.451	6.399
			Nursery		
MBCS-ARM$^{\pm}$	85	0.654	0.961	100	1.28
MBCS-ARM^{+}	148.2	0.055	1.000	41.671	2.773
			Quake		
MBCS-ARM$^{\pm}$	3.2	0.630	0.841	56.97	1
MBCS-ARM^{+}	8.2	0.002	0.993	2.273	2.978
			Stock		
MBCS-ARM$^{\pm}$	326.2	0.598	0.893	100	1.55
MBCS-ARM^{+}	533.3	0.058	0.990	99.811	2.853
			Wine		
MBCS-ARM$^{\pm}$	114.5	0.496	0.833	99.83	1.70
MBCS-ARM^{+}	136.4	0.030	0.996	91.124	3.344

rithm, and is going to be the base for comparing to other EAs in this study.

5.2. Comparison of MBCS-ARM$^{\pm}$ with Evolutionary algorithms

A similar analysis can be made when comparing the MBCS-ARM$^{\pm}$ to other algorithms. The results of this experiment are collated in Table 4. The proposed algorithm obtains higher support averages than others on 5 datasets, while the confidence obtains fairly good results. The balance between these two measures can be easily regulated by the β and γ parameters, which weigh the values of support and confidence in the objective value. Although no attention was payed to those parameters, good confidence

Table 4
Comparison with single-objective evolutionary algorithms. The reported results are averages of 10 individuals runs

Algorithm	#Rules	avgSupp	avgConf	coverage(%)	avgAntLen
			Basketball		
Alatasetal-A	11.7	0.977	0.999	99.89	3.33
EARMGA-A	100	0.314	1.000	100	2.00
GAR-A	1.8	0.680	0.793	87.09	1.80
GENAR-A	30	0.296	0.969	90.32	5.00
MBCS-ARM$^{\pm}$	9.8	0.672	0.891	95.21	1.02
			Car		
Alatasetal-A	89	0.027	1.000	85.07	5.27
EARMGA-A	100	0.376	1.000	100	2
GAR-A	–	–	–	–	–
GENAR-A	–	–	–	–	–
MBCS-ARM$^{\pm}$	14.9	0.720	0.967	96.25	1
			Cyclist		
Alatasetal-A	88.5	0.154	1.000	82.93	5.06
EARMGA-A	100	0.514	1.000	100	2
GAR-A	–	–	–	–	–
GENAR-A	30	0.020	0.970	29.81	8
MBCS-ARM$^{\pm}$	32	0.453	0.882	100	1.35
			House		
Alatasetal-A	37.5	0.068	0.399	37.21	4.04
EARMGA-A	77	0.357	1.000	100	2
GAR-A	110.1	0.766	0.900	99.99	2.02
GENAR-A	30	0.434	0.990	87.39	17
MBCS-ARM$^{\pm}$	357.7	0.355	0.824	99.99	2.91
			Nursery		
Alatasetal-A	57.5	0.026	0.700	58.98	4.10
EARMGA-A	98.3	0.413	1.000	100	2
GAR-A	–	–	–	–	–
GENAR-A	–	–	–	–	–
MBCS-ARM$^{\pm}$	85	0.654	0.961	100	1.28
			Quake		
Alatasetal-A	3.1	0.496	0.787	67.96	1.84
EARMGA-A	100	0.327	1.000	100	2
GAR-A	1	0.388	0.766	46.15	1.90
GENAR-A	30	0.550	0.950	81.84	4
MBCS-ARM$^{\pm}$	3.2	0.630	0.841	56.97	1
			Stock		
Alatasetal-A	25.7	0.265	0.989	69.78	3.47
EARMGA-A	100	0.321	1.000	100	2
GAR-A	1.7	0.423	0.792	57.15	1.8
GENAR-A	30	0.292	0.922	87.35	10
MBCS-ARM$^{\pm}$	326.2	0.598	0.893	100	1.55
			Wine		
Alatasetal-A	94.4	0.374	1.000	68.65	7.48
EARMGA-A	100	0.384	1.000	100	2
GAR-A	7	0.219	0.928	80.178	2.62
GENAR-A	30	0.010	1.000	16.95	14
MBCS-ARM$^{\pm}$	114.5	0.496	0.833	99.83	1.70

was obtained on all datasets. When comparing the coverage of records, the EARMGA was best with 100% on all datasets, but support is low on some datasets. On the other hand, the lowest coverage of records was obtained by GENAR, but due to the fact that the algorithm involves all attributes in rule generation. It is also worth noting that both GAR and GENAR had trouble finding any rules at all on some datasets. The rules mined by Alatasetal are the closest to MBCS-ARM$^{\pm}$ in quality based on results in Table 4, since both algorithms include the possibility for finding positive and negative intervals of attributes in association rules. Let us emphasize that the rules found by the proposed algorithm are short in length, thus ensuring an easier understanding from the user's perspective.

5.3. Comparison of MBCS-ARM$^{\pm}$ with traditional algorithms

In this section the proposed algorithm will be compared with three traditional association rule mining algorithms, all presented in Section 2. A mandatory discretization of datasets, which include quantitative attributes was performed in order to effectively mine association rules with traditional algorithms. The used discretization was equal frequency sampling, which introduces new attributes with intervals. Many other discretization algorithms exist in the literature, but they often require some knowledge of the data being processed. To keep things simple the mentioned sampling into four intervals was used for each quantitative attribute (as used in [3]).

The results of the last experiment are collated in Table 5. The proposed MBCS-ARM$^{\pm}$ achieved the highest support for all datasets, with a high value of confidence. The reason lies in the possibility of finding negative attribute intervals. The obtained rules also cover the datasets better in 7/8 cases, and these rules are on average shorter, thus easier to understand.

5.4. Examples of mined rules

In this section we present and analyze some rules mined by the MBCS-ARM$^{\pm}$. Table 6 shows some interesting rules obtained from the Car, Quake, and Stock datasets. All rules have their interesting measures reported alongside.

The rules in Table 6 can be interpreted as follows:

– Car: if the car safety is proven to be low, then the car is not accepted very good by the consumers (it is tend to sell less).

Table 5
Comparison with traditional algorithms (Apriori, Eclat, and FP-Growth)

Algorithm	#Rules	avgSupp	avgConf	coverage	avgAntLen
		Basketball			
Apriori, Eclat, FP-Growth	4	0.15	0.87	33.34	2.75
BCSRM$^{\pm}$	9.8	0.672	0.891	95.21	1.02
		Car			
Apriori, Eclat, FP-Growth	15	0.16	0.98	66.65	2.74
BCSRM$^{\pm}$	14.9	0.720	0.967	96.25	1
		Cyclist			
Apriori, Eclat, FP-Growth	60	0.12	0.89	94.95	3.46
BCSRM$^{\pm}$	32	0.453	0.882	100	1.35
		House			
Apriori, Eclat, FP-Growth	207	0.14	0.92	92.55	3.64
BCSRM$^{\pm}$	357.7	0.355	0.824	99.99	2.91
		Nursery			
Apriori, Eclat, FP-Growth	24	0.14	1.0	33.34	2.92
BCSRM$^{\pm}$	85	0.654	0.961	100	1.28
		Quake			
Apriori, Eclat, FP-Growth	18	0.25	0.91	90.54	2.55
BCSRM$^{\pm}$	3.2	0.630	0.841	56.97	1
		Stock			
Apriori, Eclat, FP-Growth	129	0.14	0.92	95.9	3.23
BCSRM$^{\pm}$	326.2	0.598	0.893	100	1.55
		Wine			
Apriori, Eclat, FP-Growth	244	0.14	0.92	98.31	3.41
BCSRM$^{\pm}$	114.5	0.496	0.833	99.83	1.70

Table 6
Some examples of interesting rules found by the MBCS-ARM$^{\pm}$

Dataset	Rule	Support	Confidence	Coverage
Car	*if* (safety is *low*) *then* (acceptability is not *v-good*)	0.333	1	33.3%
Quake	*if* (longitude is not [−180.0, −67.8)) *then* (richter is not [6.2, 6.9])	0.625	0.834	75%
Stock	*if* (company3 is [12.8, 16.2)) *then* (company2 is not [49.1, 53.4))	0.268	1	26.8%

– Quake: if the longitude is not in the range [−180.0,−67.8), then the Richter scale of the quake is not in the range [6.2,6.9].
– Stock: if the stock price of company3 is between 12.8 and 16.2, then the stock price of company2 is not between 49.1 and 53.4.

6. Conclusion

A single-objective binary cuckoo search using a novel individual representation was proposed in this paper. The representation tackles the problems of large dimensionality, while also support the mining of both positive and negative intervals of attributes in association rules. The proposed MBCS-ARM algorithm produces rules, which are interesting, simple, easy to understand, and offer good coverage of the dataset. It uses an objective function composed of the support and confidence, weighted by two parameters. These parameters control the importance of each measure, and give the user the control for finding rules with either greater support or confidence.

MBCS-ARM was tested on many publicly available databases, and compared to several evolutionary and traditional algorithms, all available in the KEEL tool. The experiments show promising result compared to other algorithms, based on the interesting rules, and also provide rules which include a lower number of attributes.

For future work we would like to test the proposed method on larger databases (with millions of records), perform a large scale study of parameters of the algorithm, while introducing new objective functions. There is also an interesting growth in hybridizing SI algorithms [26], so a step in that direction could definitely improve the results.

References

[1] R. Agrawal, T. Imieliński and A. Swami, Mining association rules between sets of items in large databases, *ACM SIGMOD Record* **22**(2) (1993), 207–216.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A.I. Verkamo, Fast discovery of association rules, *Advances in Knowledge Discovery and Data Mining* **12**(1) (1996), 307–328.

[3] B. Alataş and E. Akin, An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules, *Soft Computing* **10**(3) (2006), 230–237.

[4] J. Alcala-Fdez, L. Sanchez, S. Garcia, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit and V.M. Rivas, et al., Keel: A software tool to assess evolutionary algorithms for data mining problems, *Soft Computing* **13**(3) (2009), 307–318.

[5] S. Ankita, A. Shikha, A. Jitendra and S. Sanjeev, A review on application of particle swarm optimization in association rule mining. In *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)*, 2013, pp. 405–414. Springer.

[6] E. Cantú-Paz and C. Kamath, On the use of evolutionary algorithms in data mining, *Data Mining: A Heuristic Approach* (2001), 48–71.

[7] Y. Djenouri, H. Drias and Z. Habbas, Bees swarm optimisation using multiple strategies for association rule mining, *International Journal of Bio-Inspired Computation* **6**(4) (2014), 239–249.

[8] Y. Djenouri, H. Drias, Z. Habbas and H. Mosteghanemi, Bees swarm optimization for web association rule mining. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 03*, 2012, pp. 142–146. IEEE Computer Society.

[9] I. Fister Jr, X.-S. Yang, I. Fister, J. Brest and D. Fister, A brief review of nature-inspired algorithms for optimization, *Elektrotehniški Vestnik* **80**(3) (2013), 116–122.

[10] A. Ghosh and B. Nath, Multi-objective rule mining using genetic algorithms, *Information Sciences* **163**(1) (2004), 123–133.

[11] D.E. Goldberg, Genetic algorithms in search optimization and machine learning, volume 412. Addison-wesley Reading Menlo Park, 1989.

[12] J. Han, J. Pei and Y. Yin, Mining frequent patterns without candidate generation. In *ACM Sigmod Record* **29** (2000), 1–12. ACM.

[13] K.E. Heraguemi, N. Kamel and H. Drias, Association rule mining based on bat algorithm. In *Bio-Inspired Computing-Theories and Applications*, 2014, pp. 182–186. Springer.

[14] K.E. Heraguemi, N. Kamel and H. Drias, Multi-population cooperative bat algorithm for association rule mining. In *Computational Collective Intelligence*, 2015, pp. 265–274. Springer.

[15] J.H. Holland, Adaptation*. In Robert Rosen and Fred M. Snell, editors, Progress in Theoretical Biology, 1976, pp. 263–293. Academic Press.

[16] G. Hrovat, I. Fister Jr, K. Yermak, G. Stiglic and I. Fister, Interestingness measure for mining sequential patterns in sports, *Journal of Intelligent & Fuzzy Systems* **29**(Preprint) (2015), 1981–1994.

[17] I. Fister Jr, Umetni športni trener, *Predavanje na Mednarodnem sejmu IFAM 2016*, 2016.

[18] D. Karaboga and B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm, *Journal of Global Optimization* **39**(3) (2007), 459–471.

[19] J. Kennedy and R. Eberhart, Particle swarm optimization. In *Neural Networks, 1995 Proceedings, IEEE International Conference on*, volume 4, 1995, pp. 1942–1948.

[20] J.R. Koza, *Genetic programming ii: Automatic discovery of reusable subprograms*. Cambridge, MA, USA, 1994.

[21] M. Lichman, UCI machine learning repository, 2013.

[22] J.M. Luna, J.R. Romero, C. Romero and S. Ventura, Reducing gaps in quantitative association rules: A genetic programming free-parameter algorithm, *Integrated Computer-Aided Engineering* **21**(4) (2014), 321–337.

[23] J. Mata, J.-L. Alvarez and J.-C. Riquelme, Mining numeric association rules with genetic algorithms. In *Artificial Neural Nets and Genetic Algorithms*, 2001, pages 264–267. Springer.

[24] J. Mata, J.-L. Alvarez and J.-C. Riquelme, An evolutionary algorithm to discover numeric association rules. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, 2002, pp. 590–594. ACM.

[25] B. Minaei-Bidgoli, R. Barmaki and M. Nasiri, Mining numerical association rules via multi-objective genetic algorithms, *Information Sciences* **233** (2013), 15–24.

[26] U. Mlakar, I. Fister Jr and I. Fister, Hybrid self-adaptive cuckoo search for global optimization, *Swarm and Evolutionary Computation* **29** (2016), 47–72.

[27] D. Rodrigues, L.A.M. Pereira, T.N.S. Almeida, J.P. Papa, A.N. Souza, C.C.O. Ramos and X.-S. Yang, Bcs: A binary cuckoo search algorithm for feature selection. In *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, 2013, pp. 465–468. IEEE.

[28] K.N.V.D. Sarath and V. Ravi, Association rule mining using binary particle swarm optimization, *Engineering Applications of Artificial Intelligence* **26**(8) (2013), 1832–1840.

[29] S.F. Smith, A learning system based on genetic algorithms, PhD thesis, 1980.

[30] A. Song, X. Ding, J. Chen, M. Li, W. Cao and K. Pu, Multi-objective association rule mining with binary bat algorithm, *Intelligent Data Analysis* **20**(1) (2016), 105–128.

[31] R. Storn and K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization* **11**(4) (1997), 341–359.

[32] X. Yan, C. Zhang and S. Zhang, Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support, *Expert Systems with Applications* **36**(2) (2009), 3066–3076.

[33] X.-S. Yang, A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, 2010, pp. 65–74. Springer.

[34] X.-S. Yang and S. Deb, Cuckoo search via lévy flights. In *Nature & Biologically Inspired Computing, 2009 NaBIC 2009 World Congress on*, 2009, pp. 210–214. IEEE.

[35] M.J. Zaki, Scalable algorithms for association mining, *Knowledge and Data Engineering, IEEE Transactions on* **12**(3) (2000), 372–390.