Computer Science Journal of Moldova, vol.**32**, no.1(94), 2024

# A Binary Grey Wolf Optimizer with Mutation for Mining Association Rules

KamelEddine Heraguemi, Nadjet Kamel,
Majdi M. Mafarja

**Abstract**

In this decade, the internet becomes indispensable in companies and people life. Therefore, a huge quantity of data, which can be a source of hidden information such as association rules that help in decision-making, is stored. Association rule mining (ARM) becomes an attractive data mining task to mine hidden correlations between items in sizeable databases. However, this task is a combinatorial hard problem and, in many cases, the classical algorithms generate extremely large number of rules, that are useless and hard to be validated by the final user. In this paper, we proposed a binary version of grey wolf optimizer that is based on sigmoid function and mutation technique to deal with ARM issue, called BGWOARM. It aims to generate a minimal number of useful and reduced number of rules. It is noted from the several carried out experimentations on well-known benchmarks in the field of ARM, that results are promising, and the proposed approach outperforms other nature-inspired algorithms in terms of quality, number of rules, and runtime consumption.

**Keywords:** Association rules mining, ARM, Grey Wolf Optimizer, support, confidence.

**MSC2020:** 62H20.

## 1 Introduction

Nowadays, the huge number of connected devices to INTERNET become a relevant source of data. As a consequence, the saved data needs to be explored and used in many other fields such as Marketing,

doi:10.56415/csjm.v32.06

Engineering, and Medical [1]. Due to this huge amount of data, automated processing becomes an interesting research area for academic researchers. The data mining field includes a huge number of techniques that process data and attempt to collect accurate, relevant, engaging, and comprehensible knowledge from huge databases. One of the most attractive tasks in data mining is Association Rule Mining (ARM) [2]. It seeks to define correlations among items in a transactional dataset.

Agrawal et al. [3] introduced the Association rule (AR) concept in 1993. Since that, ARM has been widely and successfully applied in many hypersensitive domains such as healthcare, market analysis, electric engineering, and web recommendation systems [4]. Basically, ARM aims to identify important dependencies between items in a given dataset in the form of an IF-THEN statement: IF $<$ some conditions are satisfied $>$ THEN $<$ some values of other attributes$>$. Conditions in the IF statement are called Antecedents, and those within the THEN clause are called Consequences. Obviously, numerous relations of this kind can be extracted from a dataset, but only the useful relations in the real life need to be selected.

Indeed, discovering ARs in a wide transactional dataset is an NP-Hard problem [4]. In a database with $n$ items, there exists $2n$ itemsets, which generate a maximum number of $2k-2$ association rules, where $k$ is the length of itemsets. This proves that the time consumption exponentially increases with the increase of the number of items. Moreover, the increase of items affects memory consumption too, especially nowadays, with huge stored data. This case makes traditional algorithms, such as Apriori [3] and FP-Growth [4], require a considerable execution time. In order to overcome this drawback, many studies take direction to evolutionary and bio-inspired algorithms, such as genetic algorithms [5], particle swarm algorithms [6], bat algorithm [7], and recently whale optimization algorithm [8], to select the most useful and interesting ARs within a reasonable time and less hardware consumption. Generally, for intelligent algorithms, the database is considered as a search space, and the algorithm – as an exploration strategy that aims to explore the search space and define the rules that maximize/ minimize an earlier defined fitness function that evaluates the rule quality based on its measures. Moreover, many researches deal with ARM as a multi-objective optimization problem [9], [10]. This idea has been motivated by the huge number of rules' quality measures introduced in various objective functions. Almost all the time, optimization algorithms prove their robustness and efficiency to solve ARM issues within an acceptable runtime and less hardware consumption.

Grey Wolf Optimization algorithm (GWO) is one of the most well-known nature-inspired optimization approaches published recently [11]. It mimics the social hierarchy of the grey wolves in nature. GWO confirmed its efficiency in various real-life applications such as electrical engineering [12] and feature selection [13]. GWO confirmed its competitivity compared to other swarm-inspired metaheuristics, such as Particle swarm optimization (PSO), Bat Algorithm (BA), and Bee swarm optimization algorithm (BSO) in terms of exploitation and exploration. Furthermore, GWO beats other metaheuristics in terms of the number of variables that need to be initialized. In GWO, only one variable has to be initialized. With this in mind, and motivated by the success of GWO in various domains, we propose in this paper a new binary version of GWO based on sigmoid function and mutation technique to deal with ARM issue, namely BGWOARM. We use a new bitmap database representation, and an updated wolf's position updating algorithm is introduced to generate candidate rules. Afterward, a mutation operator is applied to get the fittest rule. To evaluate the efficiency of the proposed approach, deep experimentations are carried out on various famous benchmarks in the field of ARM defers in size and item number. Also, a comparative study in terms of runtime and rule quality is made with recently published method in the domain of ARM. The computational results of BGWOARM are promising and prove its efficiency.

The rest of this paper is organized as follows: Section 2 provides a literature review that shows the recently published works in the field of ARM. The section that immediately follows introduces a general background on association rule mining and the original grey wolf optimizer. Section 4 presents the details of our proposed method to solve ARM issue based on a binary grey wolf optimizer. Furthermore, the results of our proposal are outlined. Finally, we conclude and outline our future work and improvements.

## 2 Related Works

Since its inception in 1993 by Agrawal et al. [2], the problem of ARM got a lot of attention. In literary research, there are a remarkable number of researchers that may be divided into two approaches; exact and optimization. The first approach seeks to retrieve all the relationships between objects that exist across the database, while the second aims to produce essential and relevant rules. ARM has been dominated by two major methods: 1) Apriori, a well-known traditional method, identifies all associations depending on the minimal support specified by the expert [3]; 2) FP-growth, which was established to solve Apriori shortcomings, notably multiple dataset scans, in which the entire dataset is only scanned twice [14]. These techniques now have to deal with a lot of data, which makes them slower and memory eaters.

Afterward, studies have been made to deal with data mining problems as optimization problems, even for ARM which is considered as an NP-hard problem. Thus, several researches started in applying genetic algorithms (GA) to extract ARs from transactional databases [15]. GAs are evolutionary algorithms based on the natural reproduction of DNA's. Mainly, the application of such algorithms to tackle ARM has three main tasks, which are: rule encoding, fitness function definition, and generating new rules from the dataset. Yang et al. in [5] proposed an approach based on GA for identifying the ARs. This method didn't use any user specified minimum thresholds. Whereas, the authors utilized a relative minimum confidence as objective function to pick the best rules. In 2014, Drias in [16] declared that most of the optimization algorithms for ARM have two disadvantages: they generate false rules and extract low support and confidence rules as a high-quality rule. To cope with these drawbacks, the authors proposed two GA-based approaches, the first one named IARMGA and the other based on a Memetic algorithm, named IARMMA. In this work, the authors described a new technique called delete and decomposition strategy, that aimed to obtain rules with higher fitness. Their test results demonstrate that IARMMA offers greater solution quality. Whereas, IARMMA has increased processing time relative to IARMGA, especially when the data growth.

Recently, in [17] a modified GA was proposed with the aim to extract interesting and non-redundant relationships between items in a dataset. In order to concretize their objective, the authors consider four different quality measures, support, confidence, comprehensibility, and interestingness, to evaluate the rules. Also, they controlled the redundant rules by designing a novel rules filter method. The results obtained by the experimental study proves the efficiency of their idea.

With the development of bio-inspired techniques, numerous swarm-based algorithms, such as the PSO algorithm, Bat algorithm (BA), and Firefly algorithm (FA), etc., are recommended to cope with ARM problem. In the work presented in [18], PSO has been used to discover ARs in a transactional database. In this work, there were two phases: preprocessing and mining phase. The first one was to evaluate the objective function, while, the other one was to generate the rules based on PSO algorithm. An enhanced technique based on PSO was designed in [19]. This study proposed a Boolean variant of PSO to extract ARs named (BPSO), whereby it obtains the best rules without imposing any measurement criteria.

More recently, the authors in [6] proposed a technique, based on a new binary PSO for detecting unseen correlations among both machine abilities and product characteristics without specified minimum limits. At the same time, they provided a unique overlapping measure indicator to remove less quality regulations. Derouiche et al., presented in [20] an application of Chemical Reaction Optimization metaheuristic (CRO) for solving ARM problem, namely CRO-ARM. Many experiments were carried out on two datasets and compared to Apriori, FP-growth, and BSPO. The outcomes were promising; however, this approach needs to be tested on larger datasets.

There are several further papers in the literature that focus on Bee swarm optimization techniques and provide an approach called BSO-ARM to mine ARs [21]. Tests showed that BSO-ARM enjoys much better results than genetic algorithms. As well, an additional study was published using three processes to determine each bee’s study area (Modulo, next, syntactic). Moreover, and based on the Penguin Search Engine Optimization (Pe-ARM) method, the authors suggested an association rule miner [22]. This approach is distinguished by thorough exploration of the search space. The efficiency of this proposal is proved by numerous tests carried out on different biological data-sets.

As the first investigation of the ARM problem with the BAT algorithm, an algorithm called BAT-ARM was proposed in [23]. The bat algorithm mimics the echolocation behavior of microbats, where they move toward the prey based on the processing of the echolocation. In BAT-ARM, a new formulation of bat movements was introduced according to ARM problem. In order to prove the efficiency of their proposal, the authors carried out several tests and compared the results to those of Apriori and FP-Growth algorithms. The major drawback of this approach was those bats in the populations don't share their information about the preys. Consequently, the search space for exploration is reduced. In the continuity of their work, the same authors proposed an improvement for BAT-ARM by introducing communication strategies, namely: master/slave [24], ring, and Hybrid [7]. All strategies proved their superiority against BAT-ARM and other recently published works in terms of runtime and rules quality [7]. Along with our work on the bat algorithm, in [9] we suggested a multi-objective BA to optimize 4 quality measurements in the field of ARs which are: interestingness, comprehensibility, support, and confidence. Results show the superiority of multi-objective approach to extract the best and useful rules to the final user.

The approach proposed in [25] uses a sigmoid function binary cuckoo search, and it was applied to extract categorical ARs. Recently, Whale optimization algorithm (WOA) was adopted to extract relationships between items in a dataset [8]. The researchers look into the excellent trade-off between intensity and diversity that characterized the classic whale optimizer, which was founded on an encircling methodology, a spiral-shaped pathway, and a hunt strategy. In terms of execution speed, excellence, and memorial consumption, WOA-ARM technique outperformed other works. A new review, that summarizes several evolutionary-based computation methods used for solving an ARM problem, was presented in [4].

A review of the works conducted on ARM detection from a large-scale dataset shows the key role optimization algorithms to accelerate the mining process. In most cases, SI algorithms suffer from the large number of parameters, which makes the process to choose the best ones hard for a final user. The main difference between the proposed method of this paper and the existing methods is updating a binary version of GWO to deal with ARM, which has only one parameter that needs to be chosen by the final user. Also, a mutation technique is introduced in order to improve the generated rules.

## 3 Background

In this section, some fundamental specifications of the proposed methodology are described. First, we explain the basics of ARM. Then, we present the GWO algorithm.

### 3.1 Preliminaries on Association rule mining

ARM problem was presented by Agrawal et al. in 1993, with the goal of helping in decision-making and assisting grocery administrators in designing discounts and placing products in the store to achieve maximized profitability. These decisions depend on connections that have been created from a massive portion of previous transactions gathered by the sellers [3].

**Definition 1.** *"Formally, the association rule problem is defined as follows: Let $I = i_1, i_2, ..., i_n$ be a set of literals called items; let $Dt_1, t_2, ..., t_m$ be a transactional database, where each transaction $t$ contains a set of items. An association rule is an implication like $X \Rightarrow Y$, where $X, Y \in I$ and $X \bigcap Y = \Phi$" [7], where $X, Y$ are called antecedent (If statement) and consequent (then statement), respectively."*

To calculate the quality of the generated patterns from whichever datasets collected and in order to determine the highest notable instances for the decision maker, several objective and subjective [26] measurements are created and published over the time, that can be used to judge ARs. Objective measures were utilized to examine the produced rules in this research. Because of the huge number of frequent item sets collected from a large scale dataset, a discovered pattern is allowed as an AR only if its support and its confidence are equal or greater than the minimal limit imposed by the user, and disallowed if they are not. Support and confidence are two measures that aim to determine rules quality, which is defined as follows:

**Definition 2.** *"Support is the proportion of transactions in D that contains X, to the total of records in database. Support of item X is calculated using equation 1 and the support of an association rule $X \Rightarrow Y$ is the support of $X \cup Y$" [7].*

$$support(X) = \frac{(Number\ of\ transactions\ containing\ X)}{(Total\ Number\ of\ transactions)}. \tag{1}$$

**Definition 3.** *"Confidence is the proportion of transactions covering X and Y, to the total of records containing X. When the percentage exceeds a threshold of confidence, an interesting association rule can be generated" [7].*

For instance, a rule $X \Rightarrow Y$ with a confidence level of 0.8 states that 80 percent of the transactions containing $X$ also include $Y$. The confidence can be formulated as follows:

$$confidence(X \Rightarrow Y) = \frac{support(X \bigcup Y)}{support(X)}. \tag{2}$$

## 3.2 Grey Wolf Optimization algorithm (GWO)

GWO algorithm is one of the well-known nature-inspired optimization approaches that were published recently [11]. GWO was differentiated from other swarm intelligence (SI) algorithms by a number of features. There is just one variable that may be adjusted in this method. Furthermore, with GWO, a suitable balance between diversity and intensity may be established. Consequently, this proposed method has shown promising convergence in dealing with a wide range of engineering challenges. Additionally, GWO is a basic approach which may be simply applied and implemented. It mimics the hierarchical structure of wolves' pack and its collective hunting strategy in wildlife. Usually, wolves desire being in a community with something like a consistent hierarchy.

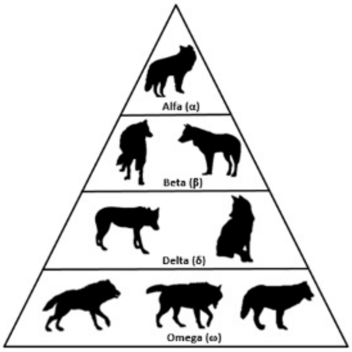


Figure 1. A graphic illustrating the hierarchical structure of wolves' pack

Figure 1 provides an illustration of wolf hierarchy, in which $\alpha$ wolves always considered as the most significant hunters and $\gamma$ wolves have the lowest strength in face to other classes. Mathematically the wolves' hierarchy structure in the pack is presented by Mirjalili et al. [11]. The remaining search agents have been declared as $\omega$, which is driven and guided by $\alpha, \beta$, and $\gamma$ to the search space which is full of promising solutions in hope to find the best optimal solution. Essentially, the mathematical formalism of GWO is stated in 3 key stages defined as follows: surrounding the victim, hunting the prey, and attacking the prey [12], and they are stated as follows:

### 3.2.1 Surrounding the victim

When a prey is found, the iteration begins ($t = 1$). Hence, $\alpha, \beta$, and $\gamma$ wolves drive the $\omega$ group to chase and ultimately surround the victim, this Gray wolf's strategy can be formulated mathematically as:

$$X(t \vec{+} 1) = X_p\vec{(}t) + \vec{A}.\vec{D}, \tag{3}$$

where $\vec{X}$ is the wolves' actual location, $\vec{X_p}$ referes to the prey's locality, $t$ presents the actual iteration, and $\vec{A}$ is an array of coefficients. Whereas, $\vec{D}$ is defined as follows:

$$\vec{D} =\mid \vec{D}.X_p\vec{(}t) - X\vec{(}t) \mid . \tag{4}$$

The parameters $\vec{A}$ and $\vec{C}$ are combinations of controlling parameters which can be calculated as follows:

$$\vec{A} = 2\vec{a}.\vec{r_1} - \vec{a}, \tag{5}$$

$$\vec{C} = 2.\vec{r_2}, \tag{6}$$

where $\vec{a}$ are elements gradually reduced from 2 to 0 throughout the optimization process, and $\vec{r_1}, \vec{r_2}$ are random arrays in [0,1].

### 3.2.2 Hunting the prey

The grey wolf is hunting by shifting the location of every wolf in the pack by moving toward the prey; this habit is theoretically expressed in the form: $\alpha$ is the leader with best position, $\beta$ and $\gamma$ are supposed to have extra details regarding prey's possible places. Thus, the $\omega$ group will follow them and be forced to move in light of the leaders within the next iterations. The location changing or hunting behavior is stated as follows:

$$\vec{D_\alpha} =\mid \vec{C_1^t}.\vec{X_\alpha^t} - X^t \mid,\ \vec{D_\beta} =\mid \vec{C_1^t}.\vec{X_\beta^t} - X^t \mid,\ \vec{D_\gamma} =\mid \vec{C_1^t}.\vec{X_\gamma^t} - X^t \mid, \tag{7}$$

$$\vec{X_1^t} = \vec{X_\alpha^t} - A_1^t.\vec{D_\alpha^t},\ \vec{X_2^t} = \vec{X_\beta^t} - A_2^t.\vec{D_\beta^t},\ \vec{X_3^t} = \vec{X_\gamma^t} - A_3^t.\vec{D_\gamma^t}, \tag{8}$$

$$X^{t+1} = \frac{X_1^t + X_2^t + X_3^t}{3}. \tag{9}$$

### 3.2.3 Attacking the prey

The parameter $\vec{a}$ governs the attacking procedure, updates the values of $\vec{A}$, and guides the $\omega$ group to pursue / leave the victim (solution). Theoretically, if $\mid \vec{A} \mid > 1$, wolves are on the lookout for a new strategy to expand their search area. Otherwise, they go toward their dominants, implying that omega wolves would follow the leaders who take advantage of the limited search area. $\vec{a}$ are carried on:

$$\vec{a} = 2(1 - t/N), \tag{10}$$

where $N$ is the maximum iteration number, and $t$ refers to the actual iteration. The bounds of $\vec{A}$ will be inside $[-2a, 2a]$. Hence, the search agents can touch any region between their position and the location of the quarry when the $\vec{A}$ is in the interval of $[-1, 1]$ by decreasing the weighting values of $\vec{a}$ from 2 to 0. The motion rule when $|\vec{A}| < 1$ or $|\vec{A}| > 1$ is vividly shown in Fig. 2.

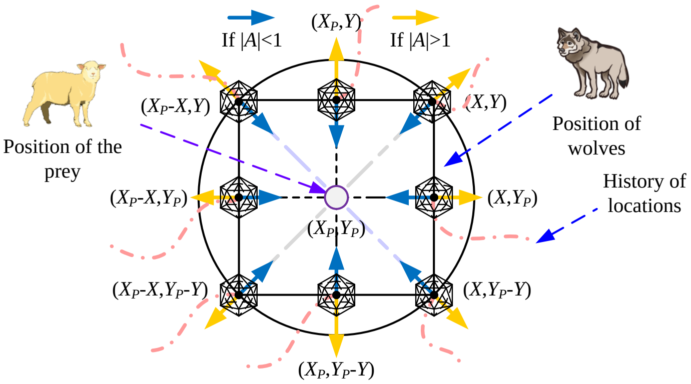
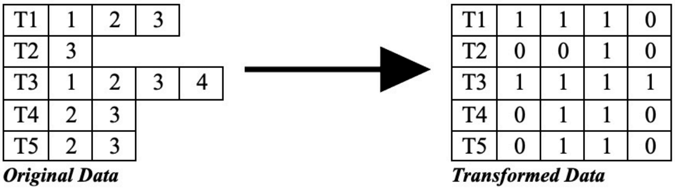


Figure 2. Impact of $\vec{A}$ on the direction of motion in GWO

## 4 Proposed Algorithm

### 4.1 Dataset and rule representation

In data mining, the data preprocessing task is one of the most irrelevant tasks. This task can influence directly on the model results. With this in mind, as well as to avoid the multi-database scans that also affect calculation time and memory consumption, datasets are transformed to bitmap representation [2] which simplifies the process of support and confidence computing. Let us consider, as shown in Figure 3, that there are 5 transactions ***T1*** to ***T5*** in transactional database which contains 4 Items. All transactions are transformed to binary form. For more illustration, consider ***T4*** in which the consumer purchased 2 products ***(I2, I3)***. Therefore, for ***B4***, the rows under ***I2*** and ***I3*** will contain the value '1'. Whereas, ***I1*** and ***I4*** will contain the value '0' because these two items don't exist in the transaction.



|    |   |   |   |   |
|----|---|---|---|---|
| T1 | 1 | 2 | 3 |   |
| T2 | 3 |   |   |   |
| T3 | 1 | 2 | 3 | 4 |
| T4 | 2 | 3 |   |   |
| T5 | 2 | 3 |   |   |

*Original Data*

|    |   |   |   |   |
|----|---|---|---|---|
| T1 | 1 | 1 | 1 | 0 |
| T2 | 0 | 0 | 1 | 0 |
| T3 | 1 | 1 | 1 | 1 |
| T4 | 0 | 1 | 1 | 0 |
| T5 | 0 | 1 | 1 | 0 |

*Transformed Data*

Figure 3. Database representation

Besides, in order to use nature-inspired algorithms, which are mainly designed for continuous optimization problems, to solve an ARM problem, rules need to be represented in a structural form that can be used by the proposed algorithms. Indeed, the literature contains two main representations for ARM which are the Pittsburgh method and Michigan method [27]. The first supposed that a set of rules is considered as a single individual; however, the other considers every rule as one individual in the population. Our proposal opts for the second one, where each rule (solution) is presented by an array of ***2k*** items, where ***k*** is the number of items in the database. The vector is coded as follows:

- $R[]i] = 1$ if the $i^{th}$ item exists in the rule, and 0 otherwise.
- $R[i+1] = 0$ if the $i^{th}$ item exists in antecedent of the rule, and 1 if it appears in the consequence part.

**For instance:** let $I = i_1, i_2, i_3$ be a set of items: the rule $i_2 \rightarrow i_1, i_3$. It is coded as $X1 = 1, 1, 1, 0, 1, 1$. Figure 4 shows a rule encrypted.

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 |

Figure 4. Rule Encoding

## 4.2 Fitness function

The fitness function examines the quality of solutions in nature-inspired algorithms. To get the optimum results, it must be maximized. Indeed, to formulate a good objective function, that rewards the proper kind of solutions, is important. As previously stated, an association rule is approved if its support and confidence levels meet the user's requirements. The fitness function of a rule R may be formulated as follows:

$$fit(R) = \begin{cases} \alpha.Support(R) + \beta.Confidence(R) & \text{if R is accepted} \\ -1 & \text{otherwise} \end{cases} \tag{11}$$

## 4.3 Binary Grey Wolf Optimizer for ARM

This section describes the whole process of our binary GWO algorithm to deal with the ARM problem. Our approach preserves the same philosophy and process as in the original GWO, from which the described algorithm has inherited its performance and advantages. The modifications take the three main steps of GWO, Encircling, Hunting the prey, and attacking, in relation to our application problem (ARM).

As mentioned above, our proposal is based on Pittsburgh encoding which means that any individual in the population is a solution (rule). Thus, the wolves in the pack in the BGWOARM algorithm represent the generated rules. The initialization step consists of creating and assigning to each wolf a random rule from the search space [11], evaluating the initial fitness values, and selecting the leaders, $(X_\alpha = best\ agent, X_\beta = Second\ best\ agent, X_\gamma = Third\ best\ agent)$, that will lead the pack through the search space and guide the pack to the prey. **Algorithm 1** shows the Binary Grey Wolf Optimizer for ARM.

When the prey is detected, the encircling task starts by $i = 1$. The leader leads the rest of the agents toward the prey based on the new proposed rule generation algorithm presented in **Algorithm 2**. The algorithm consists of calculating the distance of the new rule based on two steps according to rule encoding (Item existence and Item Positions). In order to evaluate each distance, equation 4 is used. After-

**Algorithm 1** Binary Grey Wolf Optimizer for ARM

```
Input: Number of MaxIte, number of wolves in Population, minSup, MinConf
Output: Set of valid Association rules (ValidRules)
Initialize the swarm Xi (i = 1, 2, . . ., n),
Evaluate the initial fitness for all agents,
Select $X_\alpha, X_\beta, X_\gamma$                         ▷ The Fittest wolves
$i \leftarrow 1$
while $i <= MaxIte$ do
    for each wolf in the pack do
                                        ▷ Update positions by Algorithm 2
        X1 = Generation of new rule ($X_\alpha$, $X_i, C_i$);
        X2 = Generation of new rule ($X_\beta$, $X_i, C_i$);
        X3 = Generation of new rule ($X_\gamma$, $X_i, C_i$);
      ▷ Generate new position (Rule) based on Mutation Algorithm 3
        NewRule= Mutation ($X_1, X_2, X_3$);
    end for
    Update a, A and C,
    if NewRule is accepted then
        Evaluate the fitness function Eq. (11)
        Add NewRule to the set of rules ValidRules
        Update $X_\alpha, X_\beta, X_\gamma$
    end if
    $i \leftarrow i + 1$
end while
Return ValidRules
```

ward, a sigmoid function, equation 12, is applied to convert the results to binary, either for the item's existence or appearance position. When the sigmoid of the distance, in relation to the item, is less or equal to a random value, the item exists and not otherwise. Whereas, if the sigmoid of the distance, in relation to the position, is less or equal to a random value, the item appears in the consequence, and in the antecedent otherwise.

$$sigmoid(x) = \frac{1}{1 + e^{-10(A*D-0.5)}}, \tag{12}$$

where $A$ is the actual coefficient for the actual wolf and $D$ is the distance between the agent and the prey, which is divided into two values that are $D_{item}, D_{Position}$. The first refers to the item that exists or is not in the rule, whereas, the second defines where the item appears, in the antecedent or consequence of the rule.

Afterward, the hunting behavior is the process in which each wolf in the pack has to change its position with the aim of approaching the prey. In the mathematical formulation of the original GWO which is appropriate for continuous optimizations, the hunting is presented by the equations 7,8, and 9, in which the $\omega$ pack, that can move continuously in the search space, is obliged to pursuit the leaders ($\alpha$, $\beta$, and $\gamma$). Thus, it is impossible to use the same hunting process for solving the ARM problem. With this in mind, a crossover algorithm is proposed to make the $\omega$ pack obliged to pursue the leaders in the hunting process. To resume, three new rules are generated in relation to $\alpha$, $\beta$, and $\gamma$ actual positions. Afterward, the crossover is applied between them to generate the new position. **Algorithm 3** shows the crossover algorithm.

Moreover, the generated rule is evaluated against the user thresholds (MinSup, MinConf); when the rule is accepted, it is added to a set of valid rules, and its fitness is evaluated. Finally, the leaders are updated based on the new fitness values. This search will be repeated until the maximum number of iterations is reached.

**Algorithm 2** Generation of new rule

```
Input: $Rule_x$, $Rule_i$, $C : coefficient$
Output: Rule
$i \leftarrow 1$
while $i <= 2k$ do
                                        ▷ Calculate the prey Distance
    $D_{item} =\mid C * Rule_x(i) - Rule(i) \mid$
    $D_{Position} =\mid C * Rule_x(i+1) - Rule(i+1) \mid$
                                        ▷ Identifying prey's position
    if sigmoid $(D_{item}) <= RAND$ then
        Rule(i) = 1
    else
        Rule(i)=0
    end if
    if sigmoid $(D_{Position}) <= RAND$ then
        Rule(i+1) = 1
    else
        Rule(i+1)=0
    end if
    $i \leftarrow i + 1$
end while
Return $Rule$
```

**Algorithm 3** Mutation Algorithm

```
Input: Rules (X1, X2, X3)
Output: NewRule (X)
$i \leftarrow 1$
while $i <= 2k$ do
                                                    ▷ Mutate the rules
    if $(rand < 0.33)$ then
        X(i) = X1(i)
        X(i+2) = X1(i+2)
    else
        if $(rand < 0.66)$ then
            X(i) = X2(i)
            X(i+2) = X2(i+2)
        else
            X(i) = X3(i)
            X(i+2) = X3(i+2)
        end if
    end if
    $i \leftarrow i + 2$
end while
Return NewRule
```

## 5 Results and discussions

In order to show the efficiency of the presented proposal, several experiments were carried out on different and well-known datasets in the field, which are described in the next section. After that, we present a comparative study in-face-of recently developed methods. To make the comparison totally fair, all algorithms are written in Java and executed on Intel Core I5 machine with 4 GB of memory running under Linux Ubuntu. Also, each algorithm is used with its best parameters recommended in the original paper.

### 5.1 Benchmark and setup description

With the aim to test and compare our proposed algorithm BGWORM, we use seven well-known benchmarks, that are frequently used in data mining community, from numerous and well-known sources in data mining field, such as Frequent and mining dataset repository [28] and Bilkent University function approximation repository [29]. Table 1 shows the datasets utilized in our experiments. Moreover, we observe from Table 1 that benchmarks vary in terms of transactions number and elements in each one. For example, connect dataset has 100,000 records with 999 items, whereas BMS-WebView-1 has fewer transactions and items.

Table 1. Benchmarks description

| Dataset | Transactions size | Item size |
|---|---|---|
| IBM-Stand | 1 000 | 20 |
| Quack | 2 178 | 4 |
| Chess | 3 196 | 37 |
| Mushroom | 8 124 | 119 |
| Pumbs-star | 40 385 | 7 116 |
| BMS-WebView-1 | 59 602 | 497 |
| Connect | 100 000 | 999 |

## 5.2 Stability study

As mentioned above, one of the most known advantages of Grey Wolf Optimizer is the minimum parameters number which are mainly, number of wolves in the pack and number of iterations. In order to choose the best parameters, the comparison study is presented in the section, which will prove the efficiency of our proposal. With this in mind and to analyze the behavior of our method as a stochastic evolutionary algorithm, in this section, we mainly aim to look into our algorithm (BGWOARM) stability and how the algorithm deals with the objective function and CPU time when we vary numbers of wolves in the pack and iterations. In these tests, we utilize five datasets (IBM-Stand, Quack, Chess, Mushroom, Connect).

Table 2 presents the results attained by the execution of BGWOARM with varying wolves' numbers in packs regularly from 10 to 50. Results in Table 2 were obtained with a fixed number of iterations equal to 200. It is observed that the best results are achieved with 10 wolves in a pack in most datasets. Whereas, with Mushroom and Connect datasets, the best fitness becomes acceptable starting from 30 wolves. This observation can be clarified by the transactions number and items in these datasets. On the other hand, we can note that CPU time grows with the iterations increment, which is a natural behavior of each swarm-based algorithm.

Table 2. Evaluation of the GWOARM with several numbers of Wolves

| **Dataset** | **Wolves** | **10** | **15** | **20** | **25** | **30** | **35** | **40** | **45** | **50** |
|---|---|---|---|---|---|---|---|---|---|---|
| **IBM-Stand** | **Fitness** | 0,96 | 0,96 | 0,96 | 0,96 | 0,96 | 0,96 | 0,96 | 0,96 | 0,96 |
| | **CPU time** | 0,48 | 0,65 | 0,85 | 1.11 | 1,30 | 1,52 | 1,70 | 1,91 | 2,11 |
| **Quack** | **Fitness** | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| | **CPU time** | 0,37 | 0,46 | 0,60 | 0,69 | 0,84 | 0,96 | 1,12 | 1,25 | 1,37 |
| **Chess** | **Fitness** | 0,99 | 0,99 | 0,99 | 0,99 | 0,99 | 0,99 | 0,99 | 0,99 | 0,99 |
| | **CPU time** | 1,02 | 1,42 | 1,86 | 2,24 | 2,81 | 3,21 | 3,84 | 4,53 | 4,78 |
| **Mushroom** | **Fitness** | 0,76 | 0,80 | 0,82 | 0,86 | 0,92 | 0,90 | 0,83 | 0,97 | 0,92 |
| | **CPU time** | 1,54 | 2,35 | 3,46 | 4,17 | 5,33 | 5,45 | 6,24 | 6,94 | 8,48 |
| **Connect** | **Fitness** | 0,72 | 0,81 | 0,85 | 0,80 | 0,96 | 0,95 | 0,93 | 0,96 | 0,94 |
| | **CPU time** | 27 | 42 | 57 | 72 | 85 | 103 | 111 | 126 | 155 |

On the other hand, the number of iterations is a substantial parameter for Grey Wolf Optimizer, which has an impact on algorithm stability and execution time. On this basis, we repeated our tests by fixing the number of wolves to 30 and varying the number of iterations from 100 to 900, regularly. Table 3 illustrates the results achieved by our algorithm. From the outcomes, the best values of the objective function were obtained starting from 300 iterations with all the datasets. This can be a sign that the best parameters for our algorithm are 30 and 300 for the number of wolves and iterations, respectively.

Table 3. Evaluation of the BGWOARM with several numbers of Iterations

| Dataset | Iterations | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 |
|---|---|---|---|---|---|---|---|---|---|---|
| **IBM-Stand** | **Fitness** | 0,77 | 0,98 | 0,98 | 0,98 | 0,98 | 0,98 | 0,98 | 0,98 | 0,98 |
| | **CPU time** | 0,83 | 1,57 | 2,28 | 2,96 | 3,67 | 4,42 | 5,26 | 5,73 | 6,55 |
| **Quack** | **Fitness** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | **CPU time** | 0,42 | 0,47 | 1,08 | 1,42 | 180 | 2,16 | 2,57 | 288 | 3,22 |
| **Chess** | **Fitness** | 0,84 | 0,92 | 0,96 | 0,96 | 0,97 | 0,98 | 0,96 | 0,99 | 0,98 |
| | **CPU time** | 1,18 | 2,27 | 3,52 | 4,68 | 5,97 | 7,11 | 8,43 | 9,34 | 10,65 |
| **Mushroom** | **Fitness** | 0,55 | 0,76 | 0,69 | 0,73 | 0,77 | 0,64 | 0,73 | 0,78 | 0,76 |
| | **CPU time** | 2 | 5 | 7 | 10 | 12 | 14 | 17 | 19 | 22 |
| **Connect** | **Fitness** | 0,62 | 0,75 | 0,95 | 0,89 | 0,97 | 0,97 | 0,98 | 0,95 | 0,97 |
| | **CPU time** | 44 | 91 | 144 | 178 | 223 | 268 | 308 | 353 | 402 |

## 5.3 Comparison against similar approaches

In order to prove the effectiveness of our approach, a series of comparisons were carried out with recently developed algorithms in the field of rule mining by fixing the algorithm parameters to the best values detected from the stability study, 30 and 300 for the number of wolves and iterations, respectively.

The outcomes from the binary gray wolf optimizer for ARM were compared against the following algorithms: Whale Optimization Algorithm for ARM (WO-ARM) [8], Bat algorithm for ARM (BAT-ARM) [23], Bees swarm optimization algorithm for ARM (BSO-ARM) [30], Penguins Search Optimization Algorithm for ARM (Pe-ARM) [22], and multi-swarm bat algorithm for ARM (MSB-ARM) [7].

The outcomes illustrate the overage obtained by 20 executions for every algorithm. Table 4 presents the results obtained in our tests by each algorithm in terms of CPU time consumption with four middle size benchmarks. It is clearly noted that BGWOARM outclasses the other algorithms. The exception is with IBM-Stand datasets where the Whale Optimization algorithm outguesses our proposal with 0.3 seconds, which is negligible. From the previous section, we observe that the number of iterations can affect on the CPU time in direct proportion, which is the case for all swarm-based algorithms. Based on this, a comparison study for our proposal in the face of BAT-ARM and MSB-ARM was carried out. Results are presented in Figure 5, in which the evolution of CPU time in terms of iteration number is shown. Results prove the efficiency of BGWOARM. Also, we can note the reduced time consumed by the proposed algorithm over all datasets.

Table 4. Comparing our approach to existing approaches w.r.t Time (sec)

| **Dataset** | **Pe-ARM** | **BSO-ARM** | **MSB-ARM** | **BAT-ARM** | **WO-ARM** | **BGWO-ARM** |
|---|---|---|---|---|---|---|
| IBM-Stand | 1.68 | 1.92 | 13 | 19 | **1.2** | 1.57 |
| Quack | 3.35 | 4.5 | 40 | 76 | 2.3 | **1.08** |
| Chess | 4.92 | 5.1 | 13 | 141 | 7.7 | **3.52** |
| Mushroom | 10.68 | 9.1 | 144 | 341 | 10 | **7** |

Actually, CPU time is an important fact to judge an evolutionary algorithm, but it's not enough. The optimal value of the fitness function is also critical which describes the solution quality, in our tests, the fitness function aims to maximize two main measures in the ARM field which are support and confidence. With this in mind, we compare our results in the face of other swarm-based algorithms in terms of maximum fitness function values. The outcomes illustrate the superiority of BGWOARM against other algorithms with all the datasets.

Table 5 presents the outcomes of our comparison in terms of fitness function values. On another side, swarm-based algorithm needs to explore the search space to extract the best rules, which needs to extract the maximum number of rules from the dataset that satisfied the minimum threshold support and minimum threshold confidence introduced by the final user. So, another comparison is accomplished.

Figure 6 and Figure 7 summarize the evolution of the number of generated rules from five different datasets in terms of minimum sup-

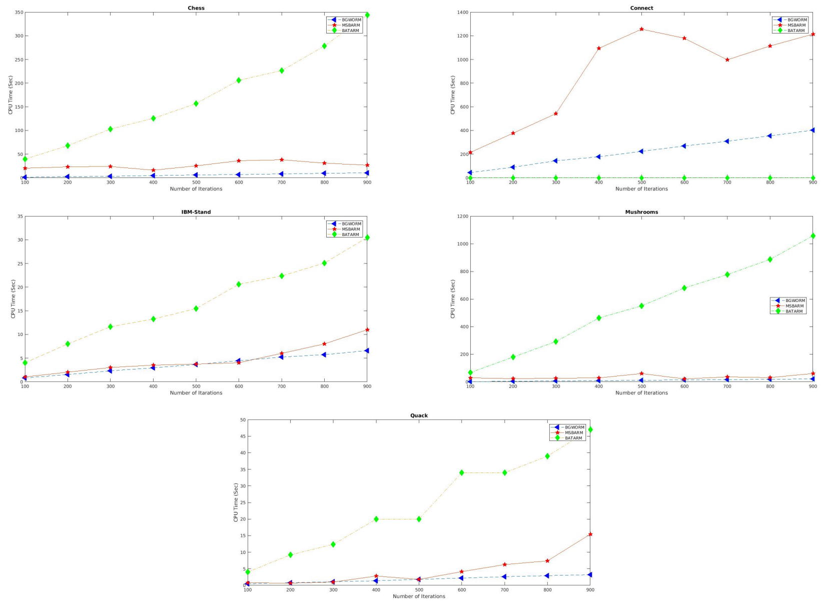


Figure 5. CPU time performance in terms of the iterations number.

port and minimum confidence, respectively. It can be noticed that the proposed algorithm outperforms the other algorithms in the majority of cases. This superiority can be argued by the great exploration of the search space extended from the original gray wolf algorithm.

Table 5. Comparing our approach to existing approaches, w.r.t Fitness

| **Dataset** | **Pe-ARM** | **BSO-ARM** | **MSB-ARM** | **BAT-ARM** | **WO-ARM** | **BGWO-ARM** |
|---|---|---|---|---|---|---|
| IBM-Stand | 0.92 | 0.93 | 0.84 | 0.41 | 0.94 | **0.98** |
| Quack | 0.91 | 1 | 1 | 0.52 | 1 | 1 |
| Chess | 0.89 | 0.88 | 0.97 | 0.92 | 0.99 | **0.99** |
| Mushroom | 0.88 | 0.75 | 0.68 | 0.93 | 0.93 | **0.97** |

Therefore, as it is shown in these results, the binary gray wolf optimizer for ARM generated rules with competitive measures in terms of support and confidence compared to similar algorithms. Also, the results prove the superiority of the proposed algorithm in terms of CPU time against the other algorithms. These results were obtained thanks

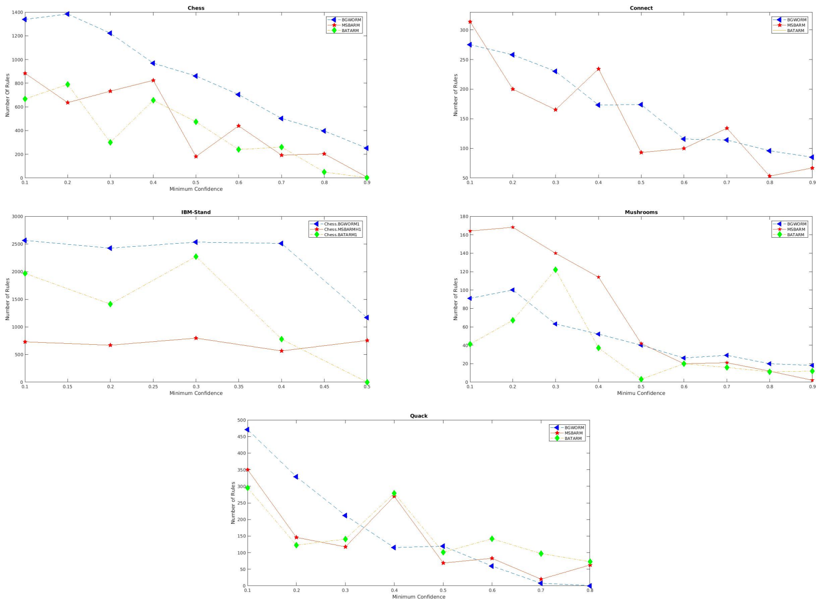


Figure 6. Number of generated rules performance in terms of minimum support.

to the newly introduced binary encoding of the rules, which reduces the algorithm complexity. Moreover, the original gray wolf optimizer algorithm simplicity, power exploitation, and exploration have been bequeathed to our algorithm.

## 5.4 Comparison against exact approaches

With the aim to discover the coverage rate of our proposal on the datasets, we calculate the proportion of valid rules generated (PVR). This proportion is calculated based on the full number of valid rules generated by the exact exhaustive algorithms (Apriori [3], FP-Growth [14]). PVR is defined by the following rule:

$$PVR = 100 * \frac{Number\ of\ generated\ rules}{Total\ number\ of\ Valid\ rules}. \tag{13}$$

The obtained results are the overage of twenty executions on

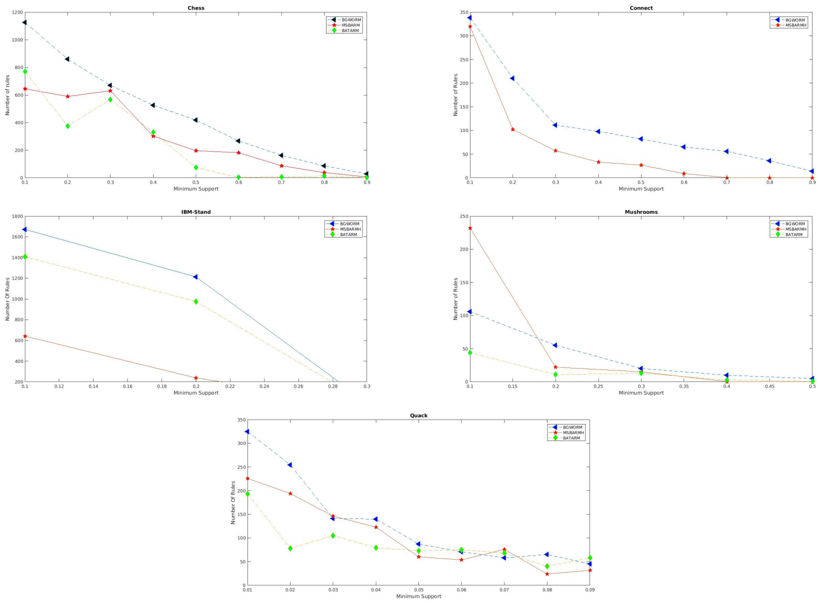


Figure 7. Number of generated rules performance in terms of minimum confidence

medium size datasets that have more than 40 000 transactions. Table 6 presents how the CPU time varies w.r.t different datasets. Again, it is clearly observed that BGWOARM surpasses the exact algorithms in terms of CPU time, thanks again to the fast search mechanism of the gray wolf algorithm and the new binary encoding proposed for the rules. Table 7 illustrates the percentage of valid rules relative to diverse benchmarks. We noticed that our method proves its superiority against MSBARM and BSO-ARM in terms of PVR, which exceeds 70% for all the datasets. Furthermore, the CPU time consumption of the proposed method is very small in the face of exhaustive approaches, whereas the PVR is not less than 70%. These results prove the power and the necessity of optimization approaches instead of exact ones.

According to the obtained results, it can be noted the BGWO-ARM superiority against other approaches. This outperforming in terms of CPU time and number of generated rules can be explained by the

Table 6. Comparing our approach to exact approaches w.r.t CPU time (sec)

| **Datasets** | **BGWO-ARM** | **MSB-ARM** | **BSO-ARM** | **FP-Growth** | **Apriori** |
|---|---|---|---|---|---|
| Pumbs-star | 307 | 315 | **300** | 600 | 500 |
| BMS-WebView-1 | **272** | 348 | 400 | 850 | 1100 |
| Connect | **402** | 1094 | 950 | 2900 | 2600 |

Table 7. Comparing our approach to exact approaches w.r.t PVR (%)

| **Datasets** | **BGWO-ARM** | **MSB-ARM** | **BSO-ARM** | **FP-Growth** | **Apriori** |
|---|---|---|---|---|---|
| Pumbs-star | **70** | 54 | 60 | 100 | 100 |
| BMS-WebView-1 | **81** | 46 | 62 | 100 | 100 |
| Connect | **77** | 49 | 55 | 100 | 100 |

low complexity of our approach, which comes from the original GWO. Moreover, the new generation method based on the sigmoid function can lead our mining process to the best rules. Also, the mutation operator has an important role in the generated rule quality based on three fittest rules $\alpha$, $\beta$, and $\gamma$. On the other hand, we can observe that BGWO-ARM generates maximum of valid rules thanks to the good exploration of the search space, inherited from GWO, and the exploitation to extract the best local rules.

## 6 Conclusion and Future Works

In this paper, a new binary grey wolf optimizer with mutation for mining ARs in large database, called BGWOARM, has been presented. The proposed algorithm used a bitmap representation for the database, which reduces runtime and simplifies rule measures calculation. Moreover, a new rule generation method based on the sigmoid function is introduced, which produces a powerful rule generator. Afterward, the mutation algorithm is applied to generate the fittest candidate rule. The BGWOARM performances have been compared to five similar approaches recently published in the field of ARM in terms of quality, number of rules, and run time. Results proved the efficiency of the proposal, and that it outperformed these methods within most experiments. Moreover, our results are compared in the face of the classic methods in terms of rule validity, which shows the efficiency of the method in search space exploration. The technique must be upgraded and evaluated with a massive database. We also plan to further parallelize the technique and implement it on a GPU to improve both the quality of the solution and its execution time.

## References

[1] I. Bose and R. K. Mahapatra, "Business data mining—a machine learning perspective," *Information and management*, vol. 39, pp. 211–225, 2001.

[2] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques.* Elsevier, 2011.

[3] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," vol. 22, 1993, pp. 207–216.

[4] A. Telikani, A. H. Gandomi, and A. Shahbahrami, "A survey of evolutionary computation for association rule mining," *Information Sciences*, vol. 524, pp. 318–352, 2020. [Online]. Available: https://doi.org/10.1016/j.ins.2020.02.073

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

[6] Z. Kou and L. Xi, "Binary particle swarm optimization-based association rule mining for discovering relationships between machine capabilities and product features," 2018. [Online]. Available: https://doi.org/10.1155/2018/2456010

[7] K. E. Heraguemi, N. Kamel, and H. Drias, "Multi-swarm bat algorithm for association rule mining using multiple cooperative strategies," *Applied Intelligence*, vol. 45, pp. 1021–1033, Dec 2016.

[8] K. Heraguemi, H. Kadrii, and A. Zabi, "Whale optimization algorithm for solving association rule mining issue," *International*

*Journal of Computing and Digital Systems*, vol. 10, pp. 2210–142, 2021. [Online]. Available: http://journals.uob.edu.bh

[9] K. E. Heraguemi, N. Kamel, and H. Drias, "Multi-objective bat algorithm for mining numerical association rules," *International Journal of Bio-Inspired Computation*, vol. 11, pp. 239–248, 2018.

[10] E. V. Altay and B. Alatas, "Performance analysis of multi-objective artificial intelligence optimization algorithms in numerical association rule mining," *Journal of Ambient Intelligence and Humanized Computing*, 2019. [Online]. Available: https://doi.org/10.1007/s12652-019-01540-7

[11] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, Mar 2014.

[12] Z. Abderrahim, K. E. Herraguemi, and M. Sabir, "A new improved variable step size mppt method for photovoltaic systems using grey wolf and whale optimization technique based pid controller," *Journal Europeen des Systemes Automatises*, vol. 54, pp. 175–185, Feb 2021.

[13] Q. Al-Tashi, H. M. Rais, S. J. Abdulkadir, H. Alhussian, and S. Mirjalili, "A review of grey wolf optimizer-based feature selection methods for classification," pp. 273–286, 2020. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-32-9990-0_13

[14] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *ACM sigmod record*, vol. 29, no. 2, pp. 1–12, 2000.

[15] S. M. Ghafari and C. Tjortjis, "A survey on association rules mining using heuristics," *WIREs Data Mining and Knowledge Discovery*, vol. 9, Jul 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1307

[16] H. Drias, "Genetic algorithm versus memetic algorithm for association rules mining," *2014 6th World Congress on Nature and Biologically Inspired Computing, NaBIC 2014*, pp. 208–213, Oct 2014.

[17] A. Derouiche, A. Layeb, and Z. Habbas, "Mining interesting

association rules with a modified genetic algorithm," *Pattern Recognition and Artificial Intelligence*, vol. 1322, p. 274, 2021. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7972016/

[18] R. J. Kuo, C. M. Chao, and Y. T. Chiu, "Application of particle swarm optimization to association rule mining," *Applied Soft Computing*, vol. 11, pp. 326–336, 2011.

[19] K. Sarath and V. Ravi, "Association rule mining using binary particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 26, pp. 1832–1840, 2013.

[20] A. Derouiche, A. Layeb, and Z. Habbas, "Chemical reaction optimization metaheuristic for solving association rule mining problem," vol. 2017-Octob. IEEE Computer Society, 3 2018, pp. 1011–1018.

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

[22] Y. Gheraibia, A. Moussaoui, Y. Djenouri, S. Kabir, and P. Y. Yin, "Penguins search optimisation algorithm for association rules mining," *CIT. Journal of Computing and Information Technology*, vol. 24, pp. 165–179, 2016.

[23] K. E. Heraguemi, N. Kamel, and H. Drias, "Association rule mining based on bat algorithm," *Journal of Computational and Theoretical Nanoscience*, vol. 12, no. 7, pp. 1195–1200, 2015.

[24] 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.

[25] U. Mlakar, M. Zorman, and I. Fister, "Modified binary cuckoo search for association rule mining," vol. 32, 2017, pp. 4319–4330.

[26] P. N. Tan, V. Kumar, and J. Srivastava, "Selecting the right objective measure for association analysis," vol. 29, 2004, pp. 293–313.

[27] A. A. Freitas, *Data mining and knowledge discovery with evolutionary algorithms.* Springer Science & Business Media, 2002.

[28] B. Goethls and M. J. Zaki, "Frequent itemset mining dataset repository," 2003. [Online]. Available: http://fimi.ua.ac.be/data/

[29] H. A. Guvenir and I. Uysal, "Bilkent university function approximation repository," 2000. [Online]. Available: http://funapp.cs.bilkent.edu.tr/DataSets/

[30] Y. Djenouri, H. Drias, Z. Habbas, and H. Mosteghanemi, "Bees swarm optimization for web association rule mining," vol. 3, 2012, pp. 142–146.

KamelEddine Heraguemi, Nadjet Kamel,
Majdi M. Mafarja

Received December 10, 2023
Accepted January 26, 2024

KamelEddine Heraguemi
ORCID: https://orcid.org/0000-0001-6992-5536
The Networks & Distributed Systems Laboratory.
National School of Artificial Intelligence
Algiers, Algeria
E–mail: kameleddine.heraguemi@ensia.edu.dz

Nadjet Kamel
ORCID: https://orcid.org/0000-0003-3608-8895
The Networks & Distributed Systems Laboratory.
University Setif1 Ferhat Abbas.
Sétif, Algeria
E–mail: nkamel@univ-setif.dz

Majdi M. Mafarja
ORCID: https://orcid.org/0000-0002-0387-8252
Department of Computer Science, Faculty of Engineering and Technology, Birzeit University
Birzeit, Palestine
E–mail: mmafarja@birzeit.edu