Computer Systems Science & Engineering
DOI: 10.32604/csse.2022.023249
Article

Tech Science Press

Binary Representation of Polar Bear Algorithm for Feature Selection

Amer Mirkhan[1] **and Numan Çelebi**[2,*]

[1]Sakarya University, Computer Engineering Department, Sakarya, Turkey
[2]Sakarya University, Information Systems Engineering Department, Sakarya, Turkey
*Corresponding Author: Numan Çelebi. Email: ncelebi@sakarya.edu.tr
Received: 31 August 2021; Accepted: 27 October 2021

Abstract: In most of the scientific research feature selection is a challenge for researcher. Selecting all available features is not an option as it usually complicates the research and leads to performance drop when dealing with large datasets. On the other hand, ignoring some features can compromise the data accuracy. Here the rough set theory presents a good technique to identify the redundant features which can be dismissed without losing any valuable information, however, exploring all possible combinations of features will end with NP-hard problem. In this research we propose adopting a heuristic algorithm to solve this problem, Polar Bear Optimization PBO is a metaheuristic algorithm provides an effective technique for solving such kind of optimization problems. Among other heuristic algorithms it proposes a dynamic mechanism for birth and death which allows keep investing in promising solutions and keep dismissing hopeless ones. To evaluate its efficiency, we applied our proposed model on several datasets and measured the quality of the obtained minimal feature set to prove that redundant data was removed without data loss.

Keywords: Optimization; rough set; feature selection; heuristic algorithms

1 Introduction

Usually, researchers start their research by preparing data in order to start data analysis to discover hidden rules and insights. Before starting this process, mainly in huge datasets, only features-attributes-related to the research should be considered. However, deciding if a feature is necessary or not is not something intuitive, especially when the research is being done by someone out of the domain. For example, a computer or mathematical scientist is doing a research based on medical dataset, for researcher it will not be intuitive to decide which features are really needed to make decision and which are not. Features required to make that decision is called "Minimal Reduct" while other features will be called "Redundant Features". Redundant and unnecessary features will cause negative impact of the performance in terms of data processing execution time, in addition to inefficient utilization of memory and storage resources. Also, those redundant features will mislead the machine learning process and might result in generating invalid rules and decisions.

When dealing with huge datasets in terms of high number of attributes, the need for a technique to eliminate the unnecessary attributes becomes a must. In other words, when a dataset has too many attributes, there will be a need for a technique to calculate the importance of each attribute. The result should not be only to say if an attribute is necessary or not, however, there might be a gray area in which we need to have figures showing the effect of removing an attribute on the quality of the dataset. Here, and depends on the characteristics of each case, the researcher can decide how much tolerance he can offer. In some cases, in order to reduce the number of the selected attributes the decision might allow 10%-for example-tolerance to the quality of the dataset, while in other cases it might not be possible to accept any tolerance, in such cases, only attributes with zero impact on the quality of the dataset can be dismissed.

Pawlak in 1982 [1,2] has introduced the rough set theory, to provides a powerful mathematical based technique to handle inconsistent data and generate uncertain rules. In the machine learning, a various number of techniques can assess the relation between the attributes of a dataset, however, most of them cannot not provide a solution for inconsistent datasets where no definite rules can be obtained. The power of rough set lies in its ability to take a dataset as an input, and without any additional information or supervision it will be able to find the minimal reduct without affecting the quality of the original dataset. However, in spite of the capabilities of the rough set theory, when working with huge datasets with high number of features, the rough set's techniques will need to evaluate all the possible combinations of attributes in order to find the minimal reduct which eventually will be an NP-hard in order to generate all possible reducts and then selecting the minimal one. Heuristic algorithms can support this process allowing reaching the “optimal” solution by only evaluating a limited number of solutions without the need to explore the entire domain of solutions.

Polar Bear Optimization PBO, proposed by Połap et al. [3] is a meta-heuristic algorithm which in principle shares the behavior of the other population based heuristic algorithms in having a randomly generated population, and then inside a finite number of loops each member of the population will try to move toward the optimal solution based on a moving function. The implementation of function varies from one algorithm to another. The technique that PBO proposes is the dynamic control of population through a continues birth and dead mechanism that will allow investing more in good promising solutions and at same time will keep ignoring hopeless ones. Other swarm-based algorithms usually start with static population and in each iteration, an equal opportunity is given to all members in the population to improve solution they found.

2 Related Work

Due to the importance of reducing number of attributes within affordable processing time, many researches have implemented the rough set theory with the support of heuristic algorithms which allowed finding the minimal reduct without the need to explore all possible alternatives. Previously implemented heuristic algorithms, more or less, share the same concept of having static population generated at the beginning and then keep trying to enhance the founded solutions in each iteration using several techniques for moving and fitness functions. The uniqueness of PBO algorithm is the dynamic death/ reproduction technique. This technique allows giving additional chance to good solutions by generating new solutions out of good potentials while keep eliminating solutions that are not progressing well.

A considerable amount of literature focused on either using the rough set techniques to find optimal reduct in various areas, or just utilized the heuristic algorithms in solving NP-hard problems. However, in our summary here we have only listed studies implemented the rough set combined with heuristic techniques:

Chen et al. [4] developed a novel using the rough set and the fish swarm algorithm for feature selection. Su et al. [5] also proposed a novel search strategy to find the minimal features reduction using the rough set theory along with fish swarm algorithm to find the minimal reduct. Both could prove that using the Fish Swarm Algorithm can enhance the accuracy of finding the core features in addition to efficiency of convergence rate. Djaafar et al. [6] also presented a new cooperative swarm intelligence algorithm for feature selection based on a combination of Firefly Algorithm (FA) and Particle Swarm Optimization (PSO) using quantum computation and a high accuracy classification and better rate of feature reduction.

Lazo-Cortés et al. [7] also presented a new technique for obtaining the shortest reducts based on binary cumulative operations over a pair-wise comparison matrix, and a fast candidate evaluation process, the result of their experimental analysis showed that they were able to find the minimal reduct faster than other algorithms reported in the literature. Alweshah et al. [8] proposed a heuristic approach where they proposed a combination between wrapper approach and genetic programming algorithm, Wrapper Genetic Programming (WGP) to identify the most informative attributes. Their proposal increased the probability of finding high-quality reducts. Anaraki et al. [9] developed a new version of a binary shuffled frog leaping algorithm hybridizing with fuzzy-rough dependency degree (FRDD) for selecting the most informative features of a dataset. Also, a very recent research was conducted by Thuy et al. [10] proposed an attribute significance measures based on stripped quotient sets for calculating core and reduct, time for finding the minimal reduct was reduced compared with similar approaches. Zheng et al. [11] proposed an enhancement for heuristic attribute reduction (EHAR) in rough set is proposed, in some rounds of the process of adding attributes, those that have the same largest significance are not randomly selected, but build attribute combinations and compare their significances. Alijla et al. [12] assessed the potential of Master River Multiple Creeks Intelligent Water Drops (MRMC-IWD) using real-world optimization problems related to feature selection and classification. Hassanien et al. [13] proposed an improved moth-flame approach to automatically detect tomato diseases. The moth-flame fitness function depends on the rough sets dependency degree and it takes into a consideration the number of selected features. Moudani et al. [14] applied Dynamic programming technique in order to enumerate dynamically the optimal subsets of the reduced attributes of high interest by reducing the degree of complexity. Li et al. [15] proposed a new method based on the Wolf Search Algorithm (WSA) for optimizing the feature selection problem. Huang et al. [16] introduced an Incremental Weight Incorporated Rule Identification (IWIRI) algorithm to process in-coming data (objects) and generate updated decision rules without re-computation efforts in the database. Alia et al. [17] developed a new algorithm for Feature Selection based on hybrid Binary Cuckoo Search and rough set theory for classification on nominal datasets. Wang et al. [18] introduced distance measures into fuzzy rough sets and propose a novel method for attribute reduction. They first construct a fuzzy rough set model based on distance measure with a fixed parameter. Then, the fixed distance parameter is replaced by a variable one to better characterize attribute reduction with fuzzy rough sets. Sattar et al. [19] presented an improve of water cycle algorithm (IWCA) for rough set attribute reduction, by hybrid water cycle algorithm with hill climbing algorithm in order to improve the exploitation process of the WCA. Yang et al. [20] studied a pseudo-label strategy systematically in rough set theory. They first proposed a pseudo-label neighborhood relation then explored the attribute reductions based on the re-defined measures.

The rest of this paper is organized as: In Section 2, an introduction to rough set theory and its basic functionality, in addition to an explanation of the PBO algorithm in general, Section 3 will present how PBO algorithm was customized to be implemented along with rough set, Section 4 describes the experimental results, Section 5 concludes this research and list some open topics for future researches.

3 Background

We will start this section by explaining the basic concepts of the rough set theory, then we will go through the common characteristics of the heuristic algorithms focusing on the population-based ones, finally will describe the new technique introduced in PBO algorithm.

3.1 Rough Set Theory

Introduced by Pawlak [1,2], can be defined as a technique for knowledge discovery to handle fuzzy and unstable datasets where traditional classification algorithms cannot obtain exact and certain rules.

3.1.1 Information System

In rough set the information system (Information table) can be defined as a table of rows and columns. Columns will be called attributes or features, while rows will be called objects (instances), Formally, the information table can be defined as $I = (U, \mathrm{A})$, where U is the universe, a finite non-empty set of objects, A is a finite non-empty set of features.

3.1.2 Decision System

Information systems can also include one or more features for decision. For example, a doctor can give a decision for each patient-object-based on a list of input features. Information systems include several features, those features have a different influence on the decision. The definition of the decision system can be extended to be $I = (U,\ C \cup D,\ V,\ F)$, where V is a non-empty set of attribute values, and the function f is a Cartesian product of A and U into V.

3.1.3 Indiscernibility

Indiscernibility identifies the objects (instances) that, with regard to certain feature(s), cannot be distinguished from each other. It is simply an equivalence relation between objects. For a decision system $\propto = (U,\ C\ \cup\ D)$, there may exist an indiscernibility relation $IND_{\propto}(C)$:

$$IND_{\propto}(C) = \{\ (x_i,\ \ x_j)\ \in U^2 \mid \forall\, c\ \in Cc(xi) = c(xj)\} \tag{1}$$

3.1.4 Set Approximation

Assume B ⊂ C be the set of condition features, [x] B be the equivalence class of each object x ∈ U by the feature subset B. The approximation of the set of objects $X \subset$ U by using the equivalence class [x] B is given by the lower $\underline{\mathrm{B}}X$ and the upper approximation $\bar{B}X$. The lower approximation of X is defined by:

$$\underline{B}X = \{\ x \in U | [x]_B \subseteq X\} \tag{2}$$

The upper approximation of X is defined by:

$$\bar{B}X = \{x \in U | [x]_B \cap X \neq \varnothing\} \tag{3}$$

The lower approximation of X, or in some references is called the positive region, consists of all object that are certainly belongs to X. whereas the upper approximation, the negative region, consists of all object that might belong to X. The accuracy of a set can be calculated as the following:

$$\propto p(X) = \frac{|\underline{P}X|}{|\bar{P}X|} \tag{4}$$

The accuracy of a rough set is a ratio between 0 and 1. When this value is one, it means that upper and lower approximation match, in this case the set is not rough anymore and called "Crisp Set". On the other side, when the value is decreasing, this will increase the negative region which means a kind of inconsistency.

3.1.5 Dependency of Features

The dependency between a set of condition attributes B and a set of decision features R is given by the following formula:

$$\gamma_B(D) = \frac{|POS_B(D)|}{|U|} \leq 1 \tag{5}$$

$POS_B(D)$ is the positive region of B, so here we divide number of elements on the lower approximation by total number of objects.

When the value of $\gamma_B(D) = 0$, this means B is independent of D, and if D is fully dependent on B the value will be 1. This dependency of condition attributes on the decision attribute(s), is the core function of the rough set that we will calculate to evaluate the effect of removing some features on the dependency of decision. When a feature a is removed from B and dependency value is not changed, this means the attribute a is a redundant attribute and can be removed without affecting the quality of the dataset.

$$\gamma_B(D) = \gamma_{B-a}(D) \tag{6}$$

3.2 Polar Bear Optimization

Polar bear optimization can be classified under population based optimization algorithms proposed by Połap et al. [3], a simulation to the way that polar bear living and hunting in hard circumstances, the algorithm can be summarized as a population of bears, each of them tries to move toward the target a kind of local search, at the same time, each bear will keep checking if other bears have found a better solution, in this case the bear will move toward the bear who is more close to the best so far founded solution. Also, different from other swarm-based algorithms, this algorithm has a powerful control of the population by applying dynamic production and death mechanism, which allows finding the optimal solution in few numbers of bears and also in few numbers of iterations. The newly developed PBO algorithm showed an efficiency in solving optimization problems, this was verified by testing PBO against several optimization functions and comparing its performance to other metaheuristic algorithms. This algorithm can be used in several domains. For example, Nasr et al. [21] used PBO in their research "Neutronic and thermal-hydraulic aspects of loading pattern optimization during the first cycle of VVER-1000 reactor using Polar Bear Optimization method".

3.2.1 Population Generation

Depending on an input variable (population size) the algorithm starts by creating the population elements and distributes them randomly within the scope of search domain, specifying number of bears which will be looking for the target solution, this input parameter should be selected very carefully, although increasing this number will increase the chance of finding the global optimal solution, but on the other hand, having too many bears will have negative impact on the performance.

3.2.2 Local Search

One step will be moved by all bears at each iteration, however, before moving the new potential location the new solution will be evaluated, if the new position is better than the current one then the bear will move, otherwise it will stay at current location. Formulas (7), (8) and Fig. 1 describe the local search behavior.

$$r = 4a\cos(\phi_0)\sin(\phi_0) \tag{7}$$

$$x_0^{new} = x_0^{actual} \mp r\cos(\phi_1) \tag{8}$$

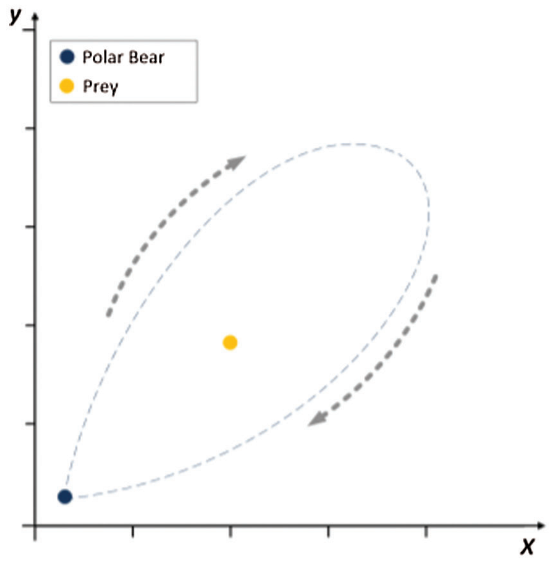

Figure 1: Local search

3.2.3 *Global Search*

Once in internal loop-local search-is complete and all the population elements had a chance to improve their locations, the global search starts by selecting one of the best solutions-bears-and all elements will try to make a step toward, however, a step will be first evaluated and only if the step will lead to a better solution, then the step will take place. The global search is performed according to the formula (9) and Fig. 2

$$(\bar{x}_j^t)^{(i)} = (\bar{x}_j^{t-1})^{(i)} + sign(\omega) \propto +\gamma \tag{9}$$

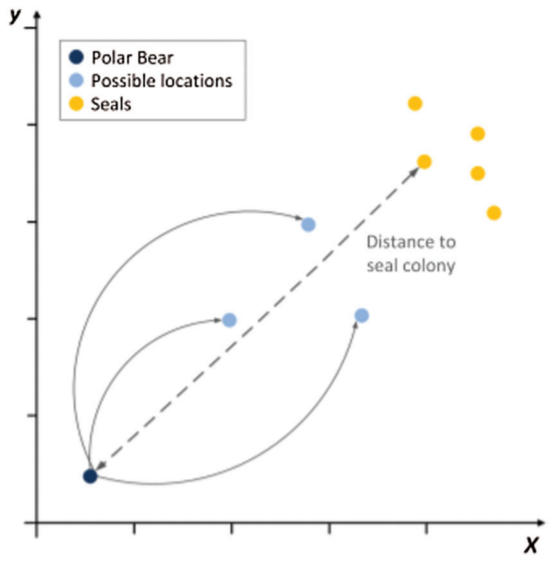

Figure 2: Global search

where ∝ is a random number in interval (0, 1), ω is the distance between two spatial coordinates and γ is a random value in the range of (0, ω).

3.2.4 Dynamic Control of Population

Unlike most of other swarm-based optimization algorithms, objects in PBO are not fixed. First the algorithm starts by generating only 75% of the population, and then after completing each iteration a decision is made whether to produce a new member or remove one based on a randomly generated variable. When reproduction is decided, then two of the best bears will generate a new solution by combining the two solutions, here we assume that combining two good solutions will produce another good solution. However, when the decision is to remove a bear, the bear having the worst value according to the fitness function will be removed from the population after checking that current number of bears will not be less than 50% of the given population size.

Those characteristics of BPO algorithm helped in finding the "optimal" solution using fewer numbers of objects-bears-and relatively less loops. Here, we put the word "optimal" inside quotations because there is no guarantee that the founded solution is the optimal one. This concept is common among all heuristic algorithms. Several researches were performed to find the optimal population size and number of required loops [22] is an example study, the summary of those studies can be summarized as: Increasing number of objects will help increase the search area and avoid stuck in local minimums, which leads in more chance to find global optimum-the optimal solution, while increasing the number of loops will result in having more accurate solutions, however, after a certain threshold, increasing the number of objects of loops might not be useful [23].

4 Proposed Model, Binary Polar Bear Optimization BPBO

In the previous section we explained how the rough set technique can be used to assess the quality of a subset of attributes for a given dataset. Then we have described the basic concepts of PBO algorithm. In this section we will explain our proposed approach of finding the minimal subset of features without the need to explore all possible combinations with the support of PBO algorithm.

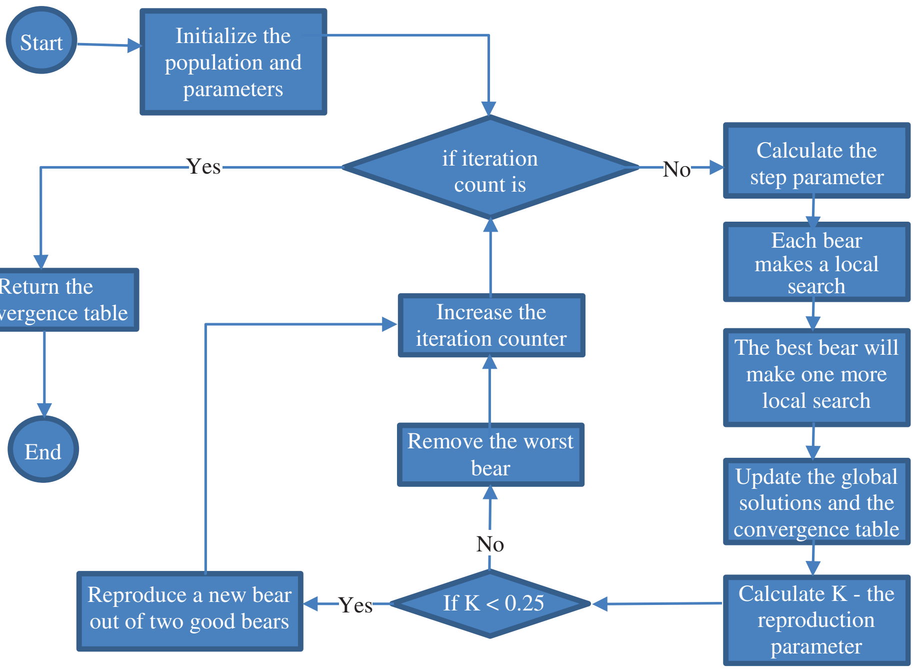
Start
Initialize the population and parameters
if iteration count is
Yes
No
Calculate the step parameter
Each bear makes a local search
The best bear will make one more local search
Update the global solutions and the convergence table
Calculate K - the reproduction parameter
If K < 0.25
Yes
No
Reproduce a new bear out of two good bears
Remove the worst bear
Increase the iteration counter
Return the convergence table
End

Figure 3: Procedure of BPBO

The polar bear algorithm as proposed in [3] can only deal with solutions as spatial objects with n dimensions, solutions can be presented as a set of coordinates $(x_1,\ x_2,\ \ldots x_n)$ same for functions (local search, global search, moving functions, …), can only deal with this kind of solutions. However, each solution in related to feature selection is a list of selected and not selected attributes, A solution can be represented as a binary array, where "1" indicates that an attribute was selected while "0" means the opposite. For a single dataset all arrays will have the same length, example:

In Fig. 4 an example of dataset with eight features, in this subset of features, feature number 3 and 6 were not selected.

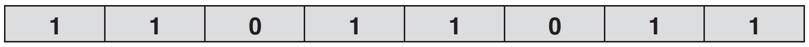

Figure 4: Solution presentation

According to the above, PBO cannot be applied as is to solve feature selection problems because both algorithms are speaking different languages. Following sub-sections will present our changes to the original PBO to be compatible with RST related problems:

4.1 Local Search

In order to make a step as a local search, we will be flipping over s attributes on and off, s is calculated according to Formula (10):

$$s = 4 - 4 * \frac{i}{maxc} \tag{10}$$

In Formula (10) i is the current outer loop counter, $maxc$ is the maximum number of iterations. The idea here is to start by making big steps and changing four features and then gradually will start decreasing the number of changed features to one when solution becomes closer to the optimal one. In the first quarter of the loop four random features are being changed to make a movement, while in the last quarter we assume that solutions became good enough and no need to make big steps. This technique was inspired from Simulated Annealing [24] and [25].

In each loop and after all bears make a step as a local search, we proposed give one more chance to one of the best bears. First, we randomly select one of the top 10% bears and apply the local again (Fig. 5).

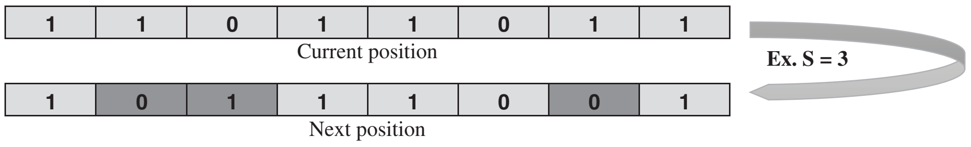

Figure 5: Local search

4.2 Global Search

Once local search is finished where all bears tried to enhance their locations, two of the best bears will be randomly selected in order to produce and new solution. Here we have applied two approaches (Fig. 6), first we produce a new solution by including attributes exist in in both solutions (AND operator), and then produce another solution by including attributes exist in at least one of the solutions (OR operator). Then we evaluate both solutions and select the solution with higher fitness.

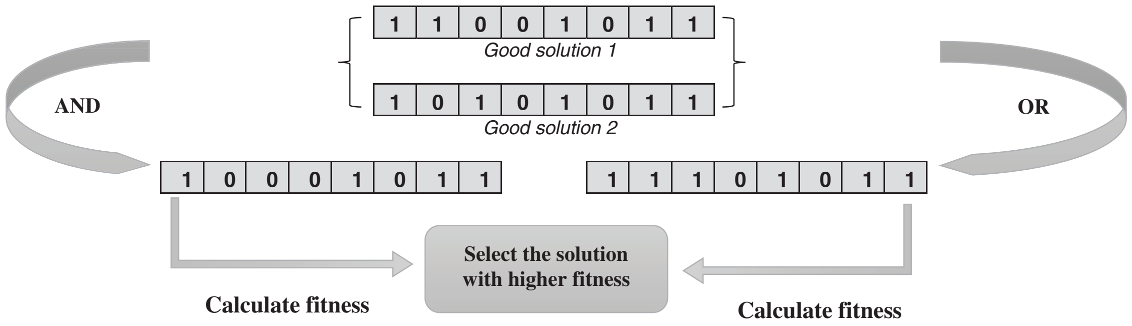

Figure 6: Reproduction

4.3 Fitness Function

The dependency will be calculated according to formula (5) in order to evaluate each generated solution, however, this formula does not give attention to the number of the selected features. In other words, if two solutions have the same quality, formula (5) considers them as equal solutions. We needed a mechanism that gives weighting to the quality of the proposed solutions in addition to the number of the selected features. Here our fitness function will be the formula (11) as proposed by Wang et al. [26]:

$$Fitness = \alpha * \ \gamma_R(D) + \beta * \frac{|C| - |R|}{|C|} \tag{11}$$

The first part of the equation $\alpha * \ \gamma_R(D)$ is the classification quality as defined in formula (5), while |C| is the total number of attributes in the dataset and |R| is the number of only selected attributes. This formula allows a kind of tolerance to quality calculation, where α and β can be considered as tolerance variable that will be an input for the algorithm. Increasing parameter β can increase the tolerance and might compromise the quality by accepting more solutions with fewer attributes. Those parameters can be adjusted according to the nature of problem, when dealing with critical data where accuracy is very important, then α and β should be set to one and zero respectively. According to Kirkpatrick et al. [24], proposed values for α, β can be 0.9 and 0.1 respectively.

4.4 The Proposed Algorithm

Our main proposed algorithm along with sub-algorithms is represented in Fig. 3 and by the following pseudo-code:

BPBO Algorithm

Input: Information table IS (table), Polulation Size (bears), number of loops (maxc) and tolerance variables α and β

Output: convergence table

```
(1:)    Initialize variables: convergence, g_min, g_fitness, i = 0
(2:)    Create only 75% of the population and calculate the initial fitness of each bear according to (11)
(3:)    while (i <= maxc)
(4:)    {
(5:)        Calculate number of attributes to change "step" according to (10)
(6:)        for each element (curr) ∈ population
(7:)            {
(8:)                    next = move (curr, step) according to moving algorithm
(9:)                    If fitness of new position is better than current one
(10:)                           Move the current element to the new position
(11:)           if fitness of the new position is higher than the g_optimal or same fitness with a smaller
number of attributes
(12:)                                   g_fitness = current_fitness
(13:)                                   update g_min = len(curr)
(14:)                                   add current element to convergence table (output)
(15:)           }
(16:)           Randomly select one of the best 10% and make one additional move according to lines 8–15
(17:)           Generate random number κ between 0 and 1
(18:)           If k < 0.25
(19:)                   Randomly choose two of the top 10% bears to reproduce a new one according to
formula
(20:)           else
(21:)                   if current population size > bears * 0.5
(22:)                           remove the worst element from population
(23:)           i = i + 1
(24:)       }
(25:)       Return convergence table
```

Reproduction Algorithm

```
Input bear1, bear2
Output new bear
(1:)    Solution_AND = selecting common attributes in bear1 and bear2
(2:)    Fitness_AND = Fitness of Solution_AND
(3:)    Solution_OR = selecting attributes exists in both bear1 or bear2
(4:)    Fitness _OR = Fitness of Solution_OR
(5:)    If Fitness _AND > Fitness_OR
(6:)            New_Solution = Solution_AND
(7:)    else
(8:)            New_Solution = Solution_OR
Output New_Solution
```

Moving Algorithm

```
Input bear, steps
Output new proposed solution
(1:)    next = bear
(2:)    for i = 1 to steps
(3:)            select random pos between 1 and length of bear
(4:)            if attribute[pos] of bear = 0
(5:)                    attribute[pos] of next = 1
(6:)            else
(7:)                    attribute[pos] of next = 0
Output next
```

5 Experimental Analysis

We have selected eight benchmark datasets from Machine Learning Repository UCI [27] to evaluate the performance of our proposal. We have used same datasets of [4] in order to compare our results with other similar algorithms. The analysis was done on a computer with 1.8 GHz CPU and 16 GB RAM running Windows 10. The analysis will prove the effectiveness of our algorithm in finding the minimal reduct using small population and few numbers of iterations, however, the execution time might not an accurate indicator to be compared with figures reported in [4] because both researches were performed on different environments. Because algorithms are stochastic based process, each execution might give different results, so we have evaluated each dataset ten times and the optimal results were reported in our analysis.

The algorithm will need four parameters as an input, population size, number of iterations in addition to two tolerance parameters. Population size and number of iterations should depend on the size of the dataset, especially number of features. The total number of possible solutions increases when number of features increases. According to that and based on our experimental analysis we noticed that the optimal size for population is same number of attributes. Same logic also applies for number of iterations, increasing the number of features will require more trials to find the optimal solution. We noticed that two times number of attributes will be needed as iteration count. We did not see any need to modify those parameters according to number of instances in the dataset.

The Tab. 1 contains a list of the used datasets with information related to number of records and number of features excluding the class-decision-feature, in addition to number of available unique values for the class attribute. Tab. 2 includes dataset name, number of instances and features, then the minimum reduct obtained using brute force by examining all possible solutions. Then we have added the results for the optimal reduct obtained by three heuristic algorithms, RSAR [23], EBR [28] and FSARSR [4] along with execution time needed to calculate this reduct. Last two columns of the table are the results of our proposed algorithm.

Table 1: List of used datasets

No	Dataset	Samples	Features	Classes
1	Audiology	200	69	24
2	Balance	625	4	3

(Continued)

Table 1 (continued)

No	Dataset	Samples	Features	Classes
3	Chess	3196	36	2
4	Lung	32	56	3
5	Mushroom	8124	22	2
6	Soylarge	307	35	19
7	Soysmall	47	35	19
8	Vote	435	16	2

Table 2: Optimal solution by several heuristic algorithm

Dataset	Instances	Features	Min	RSAR	Time	EBR	Time	FSARSR	Time	BPBO	Time
Audiology	200	69	-	-	-	-	-	13	4765.51	14	14
Balance	625	4	4	4	0.492	4	0.489	4	1.98	4	<1
Chess	3196	36	-	31	766.52	33	547.13	29	3343	21	275
Lung	32	56	4	5	1.22	4	0.64	4	128.56	4	1
Mushroom	8124	22	4	5	286.27	5	225.839	4	2767.12	4	47
Soylarge	307	35	10	13	23.904	10	12.21	10	573.31	9	6
Soysmall	47	35	2	4	0.24	2	0.40	2	6.89	2	<1
Vote	435	16	9	10	1.73	13	1.28	9	45.32	8	<1

From Tab. 2 and Fig. 7 we can compare our results with three other algorithms in terms of number of features of the minimal reduct found by each algorithm in addition to the execution time. We can see that our algorithm was able to find the optimal reduct similar or even better than other algorithms. Especially for the Chess dataset, which is the largest dataset in our analysis in terms of number of attributes and number of samples, brute force was not able to find the minimal reduct for this database due to the high number of possible solutions and the needed time to calculate one reduct due to high number of instances. In this dataset we were able to find a minimal reduct with 21 attributes where similar researches could find solutions with minimum 29 features. Also, when we compare the execution time, we can see a big difference with other algorithms, sometimes it is almost ten times which cannot be only due to difference in hardware and the execution platform.

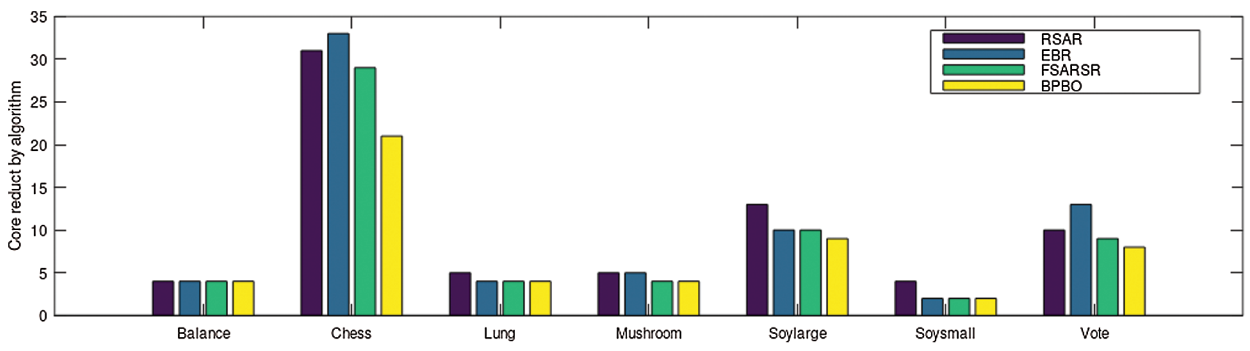

Figure 7: Convergence chart per dataset/heuristic algorithm

From Fig. 8, we can see the convergence for all datasets we have used, based on the complexity (number of instances and number of attributes) of the dataset, the figure can clearly show that our algorithm is trying to improve the fitness of all dataset. We noticed that-except for vote dataset-all datasets start at fitness 0.9, the fitness is calculated according to formula (11), the algorithm starts by calculating the fitness by including all features, in that case the right operand of the + sign in formula (11) will be zero, while the left side will be 0.9 when the original dataset is crisp [2]. The fitness of vote dataset was below 0.9 when selecting all features, this means the dataset is rough even when all features were selected. Another interesting result was the balance dataset as it was started with fitness 0.9 and no enhancement was possible because the dataset contains 4 features and the minimum reduct was also 4, so no features can be dismissed.

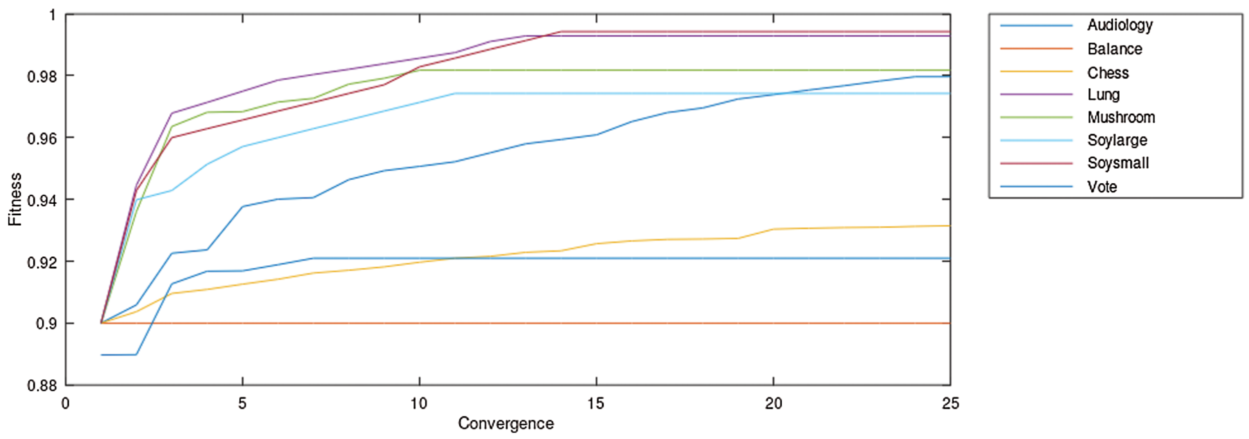

Figure 8: BPBO convergence chart per dataset

As an input parameter, we should decide the needed population size for each dataset, randomly several values of bears were used across all datasets, we have noticed that the in most of the times that the optimal value for the population size is almost same number of attributes. Taking into consideration that selecting the population size correctly has a major impact on the performance. Selecting a very small number might not allow the algorithm finding good results specially when number of attributes is high. On the other hand, selecting high number might result in finding good solutions but will have negative impact on the execution time. In Figs. 9–11, we have presented the result of using several numbers of bears to find the minimal reduct of the most complicated datasets, Chess, Audiology and Soylarge, we can notice at certain threshold having more bears will not produce better solutions anymore as the optimal solution is already reached. According to our analysis, this threshold is usually the total number of attributes in the dataset.

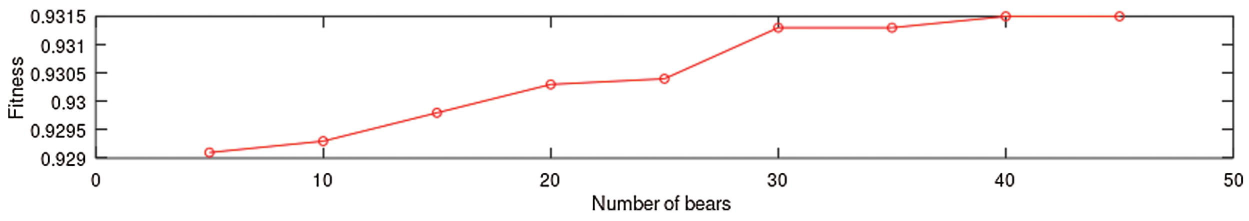

Figure 9: Chess fitness/population size

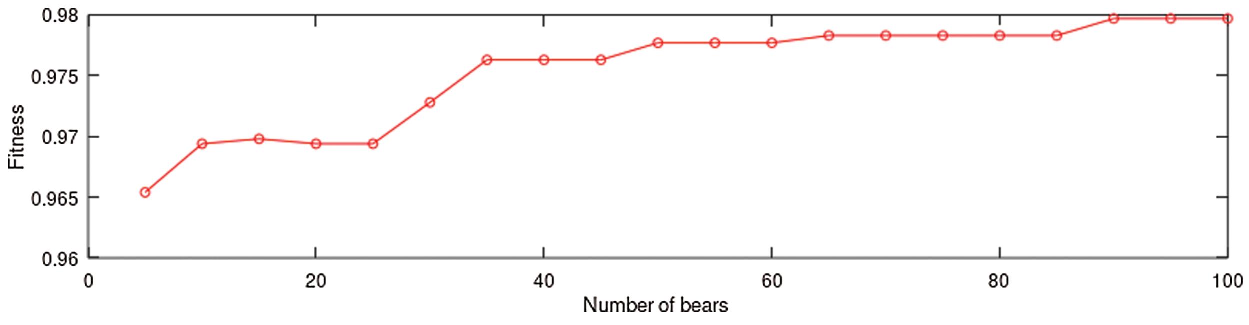

Figure 10: Audiology fitness/population size

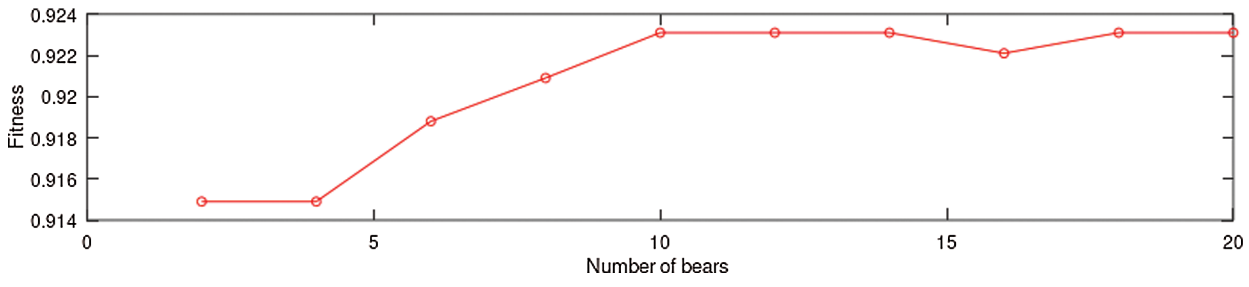

Figure 11: Soylarge fitness/population size

One more important factor is the iteration count, this parameter also has impact on the possibility of finding the optimal solution and also has affects the total runtime. Similar to population size, this parameter should be selected carefully, and also according to our analysis and as we can see in Figs. 12–14, the optimal solution is found at a certain threshold, which is in most of the cases is almost two times the feature's count, for example in the Chess database where we have 36 attribute, the optimal number of iterations is 72, after this threshold, the algorithm will be looping but no more solution are found, assuming that population size is also selected according to our above recommendation.

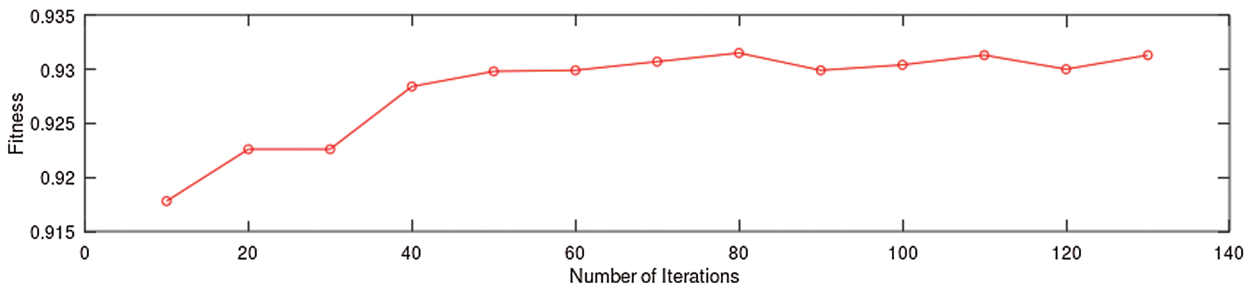

Figure 12: Chess fitness/iteration count

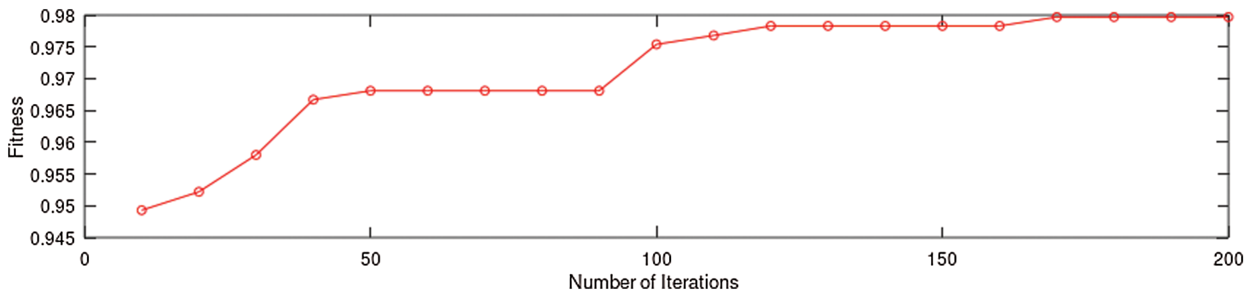

Figure 13: Audiology fitness iteration count

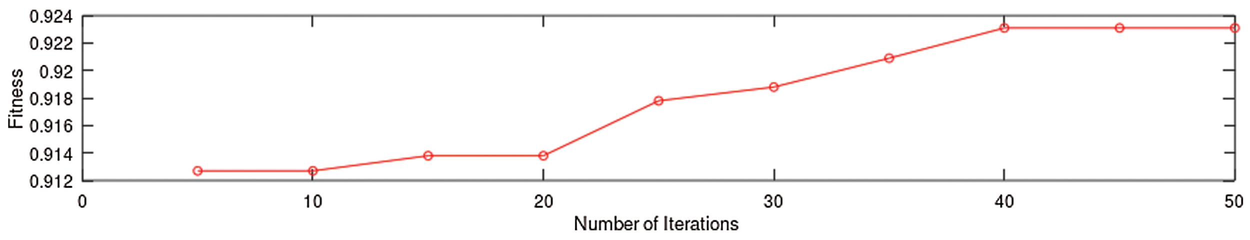

Figure 14: Soylarge fitness/iteration count

The main contribution in this research is the dynamic population and re-production/death strategy, however, to prove that we have also included additional information to each generated solution including the iteration number which produced that solution and the number of steps executed as a local search until reaching the optimal solution, from Tab. 3 we can see that the optimal solution was mostly produced as a result of the re-production mechanism and not by the initial population, and we can also notice that solutions generated during the loop are close to the optimal solution and only few steps in local search were needed to move toward the optimal solution. From this table we can understand that objects in the initial population were not able to find the optimal reduct using such few bears/iterations, and only the re-produced bears allowed finding the target solution efficiently.

Table 3: Analysis for the obtained solutions

Dataset	Samples	Features	Iterations	Best solution	Changed	Duration(s)
Audiology	200	69	140	107	111	12
Balance	625	4	10	1	1	<1
Chess	3196	36	74	56	68	220
Lung	32	56	114	68	78	1
Mushroom	8124	22	46	28	29	46
Soylarge	307	35	72	42	70	12
Soysmall	47	35	72	21	29	1
Vote	435	16	34	11	33	1

6 Conclusion and Future Work

In our research we have discussed the importance of reducing the size of the dataset before starting any research and how the rough set theory provides a powerful technique to find the minimal dataset's reduct. We also explained how the rough set by itself might not be able to find the minimal reduct as this might require calculating all combinations of attributes which is not possible in large datasets. The heuristic algorithms, especially population-based ones, can play a vital role in solving such NP-Hard problems. In the literature, several heuristic algorithms were utilized along with rough set techniques to find the minimal product. We proposed a binary representation of the Polar Bear Optimization algorithm to find the optimal reduct of a dataset. The polar bear algorithm can only deal with solutions represented as spatial coordinates while the solutions in the rough set are binary array of of selected and unselected features of the dataset, we had to make some amendments to the original functions of PBO to be make it compatible with rough set terminologies. We first represented the objects-bears-in binary format, then we modified the local and global search functions by using binary operators. To evaluate our proposed algorithm, we have selected several datasets from UCI and compared our results with other similar algorithms. Our experimental analysis showed that the dynamic population behavior of our proposed algorithm allowed finding the minimal reduct in a very efficient way in comparing with similar algorithms.

In this research we have implemented AND/OR as binary operators to reproduce new solutions based on two good solutions after each iteration, the results were very good and showed that this was really a promising technique, however, we believe that implementing more advanced binary operators worth evaluation and might even give much better results, this could be a subject for future researches to be evaluated.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Z. Pawlak, “Rough sets,” *International Journal of Computer & Information Sciences*, vol. 11, no. 5, pp. 341–356, 1982.

[2] Z. Pawlak, “Some issues on rough sets,” *Transactions on Rough Set, I, Journal Subline, Lecture Notes in Computer Science*, vol. 3100, pp. 1–58, 2004.

[3] D. Połap and M. Woźniak, “Polar bear optimization algorithm: Meta-heuristic with fast population movement and dynamic birth and death mechanism,” *Symmetry*, vol. 9, no. 10, article 203, 2017.

[4] Y. Chen, Q. Zhu and H. Xu, “Finding rough set reducts with fish swarm algorithm,” *Knowledge-Based Systems*, vol. 81, pp. 22–29, 2015.

[5] Y. Su and J. Guo, “A novel strategy for minimum attribute reduction based on rough set theory and fish swarm algorithm,” *Computational Intelligence and Neuroscience*, vol. 2017, pp. 1–7, 2017.

[6] Z. Djaafar and F. B. Abdelaziz, “A cooperative swarm intelligence algorithm based on quantum-inspired and rough sets for feature selection,” *Computers & Industrial Engineering*, vol. 115, pp. 26–36, 2018.

[7] M. Lazo-Cortés, J. Martínez, M. Francisco, A. Carrasco-Ochoa and G. Diaz, “A new algorithm for computing reducts based on the binary discernibility matrix,” *Intelligent Data Analysis*, vol. 20, no. 2, pp. 317–337, 2016.

[8] M. Alweshah, O. A. Alzubi, J. A. Alzubi and S. Alaqeel, “Solving attribute reduction problem using wrapper genetic programming,” *International Journal of Computer Science and Network Security*, vol. 16, no. 5, pp. 78–84, 2016.

[9] J. Anaraki, S. Samet, M. Eftekhari and C. Ahn, “A fuzzy-rough based binary shuffled frog leaping algorithm for feature selection,” *International Journal of Computer and Information Engineering*, vol. 12, no. 9, pp. 722–729, 2018.

[10] N. N. Thuy and S. Wongthanavasu, “A new approach for reduction of attributes based on stripped quotient sets,” *Pattern Recognition*, vol. 97, pp. 1–13, 2017.

[11] K. Zheng, J. Hu, Z. Zhan, J. Ma and J. Qi, “An enhancement for heuristic attribute reduction algorithm in rough set,” *Expert Systems with Applications*, vol. 41, no. 15, pp. 6748–6754, 2014.

[12] B. Alijla, C. Lim, L. Wong, A. Khader and M. Al-Betar, “An ensemble of intelligent water drop algorithm for feature selection optimization problem,” *Applied Soft Computing*, vol. 65, pp. 531–541, 2018.

[13] A. E. Hassanien, T. Gaber, U. Mokhtar and H. Hefny, “An improved moth flame optimization algorithm based on rough sets for tomato diseases detection,” *Computers and Electronics in Agriculture*, vol. 136, no. C, pp. 86–96, 2017.

[14] W. Moudani, A. Chahine and F. Chakik, “Dynamic rough sets features reduction,” *Journal of Computer Science and Information Security*, vol. 9, no. 4, pp. 1–10, 2011.

[15] J. Li, S. Fong, R. Wong, R. Millham and K. Wong, “Elitist binary wolf search algorithm for heuristic feature selection in high-dimensional bioinformatics datasets,” *Scientific Reports*, vol. 7, no. 1, pp. 1–14, 2017.

[16] C. Huang, T. Tseng and C. Y. Tang, “Feature extraction using rough set theory in service sector application from incremental perspective,” *Computers & Industrial Engineering*, vol. 91, pp. 30–41, 2015.

[17] A. Alia and A. Taweel, “Feature selection based on hybrid binary cuckoo search and rough set theory in classification for nominal datasets,” *International Journal of Information Technology and Computer Science*, vol. 9, pp. 63–72, 2017.

[18] C. Wang, Y. Huang, M. Shao and X. Fan, “Fuzzy rough set-based attribute reduction using distance measures,” *Knowledge-Based Systems*, vol. 164, pp. 205–212, 2019.

[19] A. Sattar and J. Al-saedi, “Hybrid water cycle algorithm for attribute reduction problems,” in *Proc. WCECS*, San Francisco, USA, 2015.

[20] X. Yang, S. Liang, H. Yu, S. Gao and Y. Qian, “Pseudo-label neighborhood rough set: Measures and attribute reductions,” *International Journal of Approximate Reasoning*, vol. 105, pp. 112–129, 2019.

[21] M. A. Nasr, M. Zangian, M. Abbasi and A. Zolfaghari, “Neutronic and thermal-hydraulic aspects of loading pattern optimization during the first cycle of VVER-1000 reactor using polar bear optimization method,” *Annals of Nuclear Energy*, vol. 133, pp. 538–548, 2019.

[22] T. Chen, K. Tang, G. Chen and X. Yao, “A large population size can be unhelpful in evolutionary algorithms,” *Theoretical Computer Science*, vol. 436, pp. 54–70, 2012.

[23] R. Jensen and Q. Shen, “Finding rough set reducts with ant colony optimization,” *Journal of Fussy Sets and Systems*, vol. 49, pp. 15–22, 2003.

[24] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, “Optimization by simulated annealing,” *Science, New Series*, vol. 220, no. 4598, pp. 671–680, 1983.

[25] P. J. van Laarhoven and E. H. Aarts, “Simulated annealing,” in *Simulated Annealing: Theory and Applications*, 1st ed., vol. 37, Netherlands: Springer, pp. 7–15, 1987.

[26] X. Wang, J. Yang, X. Teng, W. Xia and R. Jensen, “Feature selection based on rough sets and particle swarm optimization,” *Pattern Recognition Letters*, vol. 28, no. 4, pp. 459–471, 2007.

[27] D. Dua, and C. Graff, “UCI machine learning repository,” Retrieved from http://archive.ics.uci.edu/ml*, 2017.*

[28] D. Q. Miao and G. R. Hu, “A heuristic algorithm for reduction of knowledge,” *Journal of Computer Research and Development*, vol. 36, no. 6, pp. 681–684, 1999.