Memory based ARO metaheuristic algorithm for feature selection problems

Ali Hamdipour[1], Abdolali Basiri[2*]

[1]Department of Mathematics and Computer Sciences, Damghan University, Damghan, Semnan, Iran (alihamdipour@std.du.ac.ir)

[2]Department of Mathematics and Computer Sciences, Damghan University, Damghan, Semnan, Iran (basiri@du.ac.ir)

Abstract:

The classification accuracy of datasets heavily depends on their features. The presence of irrelevant and redundant features in a dataset can lead to a reduction in classification accuracy. Identifying and removing such features is the main purpose of feature selection problem, which is an important step in the data science lifecycle. The aim of the Wrapper feature selection method is to reduce the number of selected features (SF) while improving the classification accuracy by optimizing a set of features. Feature selection is a challenging and computationally expensive problem that falls under the NP-complete category, so it requires computationally efficient algorithms to solve it. The Artificial Rabbits Optimization (ARO) is a biologically inspired optimization technique that mimics the unique and intelligent foraging tactics of rabbits in nature. This paper proposed a new feature selection method based on the ARO meta-heuristic algorithm, called the memory artificial rabbits optimization (MARO), to improve its performance for solving feature selection problems. The proposed MARO method is tested on a standard benchmark dataset and compared with four state-of-the-art feature selection algorithms. The results show the effectiveness of the proposed MARO algorithm in searching for an optimal subset of features.

* Abdolali Basiria, basiri@du.ac.ir

SCJ ACCEPTED

1. Introduction

The use of information technology in various fields such as social media, medicine, transportation, business, online education, and marketing has led to the generation of large and complex datasets. Analyzing and processing such data can be time-consuming and challenging. Therefore, it is necessary to preprocess the data before modeling. Data preprocessing involves several important steps, and one crucial step is feature selection, which is performed during the preprocessing phase [1]. The main aim of feature selection is to improve classification accuracy while reducing the number of selected features (SF) by eliminating irrelevant and redundant ones. By applying feature selection, the resulting dataset model becomes more powerful and helps prevent overfitting [2].

There are various approaches to obtaining the best subset of features. One such approach is a comprehensive search within the datasets. However, this approach is not efficient for large datasets because it requires evaluating 2^n.subsets of features for n features, which becomes highly complex [3]. Another approach is random selection, where subsets of features are chosen randomly. However, the complexity of this method can be reach to that of the comprehensive search method.

Feature selection is a NP-complete problem, which means that it falls into a class of problems for which no efficient solution has been discovered. It is highly challenging to solve this problem using deterministic methods. To overcome this, special feature selection methods have been developed, including filter methods, wrapper methods, embedded methods, and hybrid methods, as illustrated in Figure 1. These methods can find an appropriate solution for the feature selection problem within an acceptable timeframe.

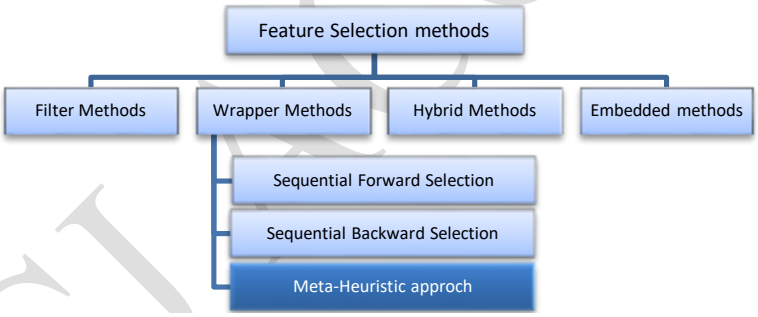


Figure 1: Feature Selection Methods

The filter method for feature selection involves assigning an importance score to each feature based on criteria such as dependency, distance, correlation, similarity, and consistency. Features with low scores are then removed, reducing the dimensionality of the datasets. This method can consider the mutual dependence between features, known as multivariate analysis, as discussed in [4]. Alternatively, if each feature is considered independent, it is referred to as univariate analysis, as discussed in [5].In the wrapper method for feature selection problem, a machine learning algorithm is used to evaluate the quality of different feature subsets. This category includes methods such as Sequential Forward Selection (SFS), Sequential Backward Selection (SBS), and meta-heuristic search. SFS methods start with an empty set of features and progressively add features until the evaluation function improves, as described in [6]. On the other hand, SBS assumes all features are initially included and then removes features one by one to achieve the desired result, as described in [7]. Additionally, meta-heuristic feature selection methods utilize meta-heuristic algorithms to search for the optimal subset of features, as studied in [8].

Embedded methods are another class of feature selection techniques where feature selection is integrated into the classifier algorithm. In these methods, the machine learning algorithm and feature selection problem interact to achieve the best classification accuracy, as used in [9]. Hybrid methods combine the capabilities of both filter and wrapper methods. In this approach, a filter method is initially applied to select a suitable subset of features, which is then passed to the wrapper method to determine the final subset, as proposed in [10].

The remainder of this paper is structured as follows: In section 2, a summary of previous works is given. Section 3 provides a brief overview of the functioning of the ARO algorithm. Section 4 elaborates the proposed approach in detail. In Section 5, we present the experimental results obtained from comparing the proposed algorithm with state-of-the-art algorithms. Lastly, Section 6 offers conclusions and outlines potential avenues for future research.

2. Lecture review

Solving real-world optimization problems poses various challenges. Some of these problems lack deterministic solutions, but meta-heuristic algorithms can effectively find suitable solutions within an acceptable time frame. Meta-heuristic algorithms possess advantages such as flexibility, simplicity, being guess free, and independence in optimization problems. These algorithms can be classified based on their characteristics: nature-inspired or non-nature-inspired, single-solution or population-based, and single-objective or multi-objective. They act as black boxes and find wide applicability in optimization problems, allowing us to

SCI ACCEPTED

obtain appropriate outputs without requiring in-depth knowledge of the problem's nature [11].

The class of meta-heuristic methods for feature selection utilizes these meta-heuristic algorithms to solve the feature selection problem by formulating it as an optimization problem with a suitable fitness function.

Genetic Algorithm (GA) is a prominent meta-heuristic algorithm that employs three operations mutation, selection, and crossover to update the positions of individuals within a population. Each individual, represented as a chromosome, represents a potential solution to the optimization problem [12]. Several meta-heuristic feature selection methods are based on genetic algorithms, and some combine GA with other algorithms to achieve improved results. For example, the algorithm proposed in [13] utilizes genetic algorithms and neural networks to identify the appropriate feature subset. Other studies such as [14] also employ genetic algorithm to solve feature selection problems.

Particle Swarm Optimization (PSO) is another popular meta-heuristic algorithm inspired by the behavior of birds. PSO updates the positions of particles based on the best solution found so far. This algorithm exhibits a favorable convergence rate, although it may become trapped in local optima. Several PSO-based meta-heuristic methods for feature selection have been proposed in studies such as [15].

The Ant Colony Optimization (ACO) algorithm is another widely used meta-heuristic algorithm. It is inspired by the foraging behavior of ants in their search for the shortest path between their nest and food sources [16]. Several meta-heuristic feature selection methods based on ACO have been introduced, such as the algorithm presented in [17].

Researchers have proposed many solutions to the problem of feature selection problem. In Table 1 a summary of the works based on algorithms Salp Swarm Algorithm (SSA) [28], Gray Wolf Optimization (GWO) [11], Cuckoo Optimization Algorithm (COA) [29],and Differential Evolution (DE) [30] is given.

In recent years, numerous meta-heuristic algorithms have been developed, each with its own advantages and disadvantages. The Artificial rabbits optimization (ARO) is a meta-heuristic optimization algorithm that distinguishes itself from the other algorithms due to its unique biological inspiration [31]. In the next section, a brief description of this algorithm is provided.

This method is primarily proposed to prevent the re-evaluation of candidate solutions. Given that the most time-consuming aspect of feature selection is evaluating candidate solutions, the proposed approach can significantly enhance the performance of metaheuristic algorithms in feature selection.

3. ARO (Artificial Rabbits Optimization)

Rabbits in nature have special behaviors to feed, escape or hide from predators. ARO is an optimization algorithm designed based on these behaviors of rabbits. Rabbits do not forage on the grass near their nest to avoid detection of their nest site. This behavior, called detour foraging, is used for the exploration phase of the ARO algorithm. Rabbits also dig holes in the side of their nests so that when predators attack them, they can accidentally hide in one of them. This strategy, called random hiding, is used in the exploitation phase for the ARO algorithm. Given that the energy required to escape from predators is limited in rabbits, they must strike a balance between detour foraging and random hiding because the farther they are from their nest, the greater the distance they must run to escape than when predators attack. ARO uses this pattern to balance between exploration and exploitation phases.[32]

3.1. Mathematical modeling of rabbit behavior

The ARO algorithm considers search agents (rabbits) in d-dimensional space and then updates the position of each agent according to the mentioned behaviors. One of the behaviors of rabbits is detour foraging. The following equations are used for mathematical simulation of this behavior.

$$\vec{v_i}(t+1) = \vec{x_j}(t) + R.(\vec{x_i}(t) - \vec{x_j}(t)) + round(0.05 + (0.05 + r_1)).n_1 \quad (1)$$

$$i, j = 1, ..., n \text{ and } \quad j \neq i$$

$$R = L \cdot c \quad (2)$$

$$L = (e - e^{(\frac{t+1}{T})^2}) \cdot \sin(2\pi r_2) \quad (3)$$

Table 1: Some of recent works on the feature selection problem

Base on	Authors	Type
SSA	[18], [19]	Wrapper
GWO	[20] , [21]	Wrapper
GWO	[22]	Hybrid
COA	[23]	Wrapper
ABC	[24] , [25]	Wrapper
DE	[26]	Wrapper
DE	[27]	Filter

SCI ACCEPTED

$$(k)=\begin{cases}1\ if\ k==g(l)\\ 0\ else\end{cases} \qquad k=1,\dots,d\ and\ l=1,\dots,\lceil r_3,d\rceil \tag{4}$$

$$g=randperm(d) \tag{5}$$

$$n_1\sim N(0,1) \tag{6}$$

where $\vec{v}_i\left(t+1\right)$ is a temporary variable to update the position of each search agent (rabbit) in the search space. $x_i\left(t\right)$, n, and T are the position of the ith search agent in time t, the size of the population of rabbits and the maximum number of iterations, respectively. Also, r_1, r_2 and r_3 are random numbers in the range $(0,1)$. n_1 is the standard normal distribution. The $randperm(k)$ function returns a random permutation of integers between 0 and k. Equation ۰ provides the detour foraging.

In equation 2, R has a direct relation with L. Also, L, which is the length of the rabbit's steps, has a large value at the beginning and gradually decreases. The detour foraging behavior makes the ARO algorithm not get stuck in the local optimal points and converges to the global solution.

As mentioned at the beginning of this section, another strategy of rabbits is random hiding. Rabbits hide when predators attack in holes they dig around their nests. To simulate this behavior, the ARO algorithm assumes d number of holes in each dimension of the search space. The following equations is used to generate these holes.

$$\overrightarrow{b_{i,j}}(t)=\overrightarrow{x_i}(t)+H.g.\overrightarrow{x_i}(t) \qquad i=1,\dots,n\ and\ j=1,\dots,d \tag{7}$$

$$H=\frac{T-t+1}{T}\cdot r_4 \tag{8}$$

$$n_2\sim N(0,1) \tag{9}$$

$$g(k)=\begin{cases}1 & if\ k==j\\ 0 & else\end{cases} \qquad k=1,\dots,k \tag{10}$$

where $b_{i,j}(t)$ is the jth hole for the ith rabbit. r_4 and n_2 are random numbers in the range $(0,1)$ and standard normal distribution, respectively. The following equation is used to simulate the random hiding of rabbits in the produced holes.

$$\overrightarrow{v_i}(t+1)=\overrightarrow{x_i}(t)+R\cdot\left(r_4\cdot\overrightarrow{b_{i,j}}(t)-\overrightarrow{x_i}(t)\right) \qquad i=1,\dots,n \tag{11}$$

$$g_r(k)=\begin{cases}1 & if\ k==\lceil r_5,d\rceil\\ 0 & else\end{cases} \qquad k=1,\dots,d \tag{12}$$

$$\overrightarrow{b_{i,r}}(t)=\overrightarrow{x_i}(i)+H\cdot g_r\cdot\overrightarrow{x_i}(i) \tag{13}$$

where r_5 is a random number in the range $(0,1)$. After choosing one of the random foraging or hiding strategies and calculating the corresponding value v_i, the following equation is used to calculate the next position of each rabbit.

$$\vec{x}_i(t+1)=\begin{cases}\overrightarrow{x_i}(t) & f(\overrightarrow{x_i}(t))\le f(\overrightarrow{v_i}(t+1))\\ \overrightarrow{v_i}(t+1) & f(\overrightarrow{x_i}(t))>f(\overrightarrow{v_i}(t+1))\end{cases} \tag{14}$$

where $x_i(t+1)$ indicates the position of the ith agent at time $t+1$. The ARO algorithm models the energy depletion of rabbits during running to switch between exploration and exploitation phases. To do it in the initial iterations, more detour foraging should be done and gradually detour foraging is reduced and random hiding is increased in subsequent iterations. In each iteration of ARO, the following equation is used to select the desired strategy.

$$A(t)=4(1-\frac{t}{T})\ln\frac{1}{r} \tag{15}$$

where r is a random number in the range $(0,1)$. In the next section, this algorithm is improved and used to solve feature selection problems.

4. Proposed Method

In this section, the main idea of the paper is described. The aim is to improve the efficiency of ARO metaheuristic algorithms in the feature selection problem by removing repeated fitting values. The implementation of meta-heuristic algorithms consists of a number of populations whose positions are updated in each iteration. Often, the most time-consuming part in each iteration of the meta-heuristic algorithm is evaluating the positions with the fitness function. One of the disadvantages of these algorithms is the evaluation of the repeated position with fitness function. In order to solve this problem, a memory table (MT) is introduced in this paper, where previously positions are stored to avoid re-evaluation them. Therefore, if repeated values are reproduced, they will be ignored and new values that have not been generated before will be generated.

In this table, the positions that has already been evaluated is stored and it is prevented from being re-evaluated in new calculations. In the same way, it creates a new position and if that position is not already in the table, the algorithm continues and otherwise it creates a new position.

By preventing the re-evaluation of the items in the MT by generating new items, the probability of reaching new area in the search space increases and the efficiency of the algorithm enhances. Figure 3 shows

SCI ACCEPTED

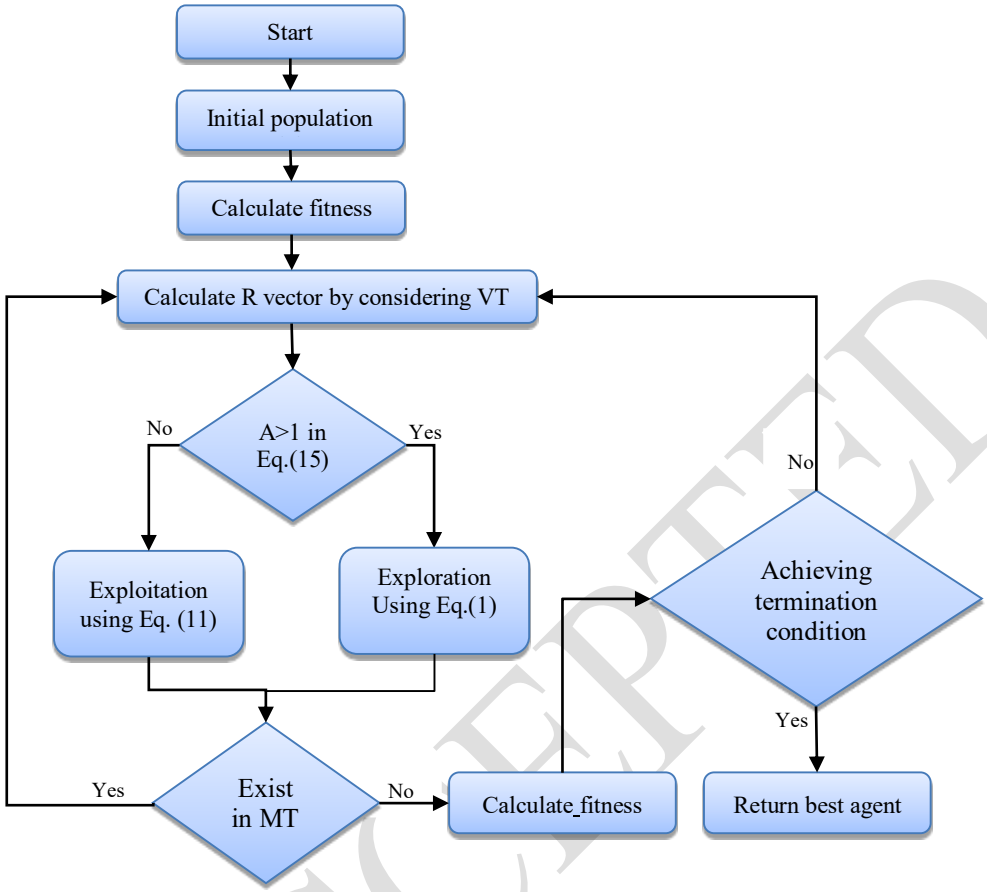


Figure 2: MARO flowchart

number of repeated positions generated for the CongressEW benchmark [33] datasets in different iterations. There are many repeated cases in the last iterations because in the last iterations the algorithm enters the exploitation phase and the evaluated positions get closer together. Therefore, the possibility of creating repeated positions increases.

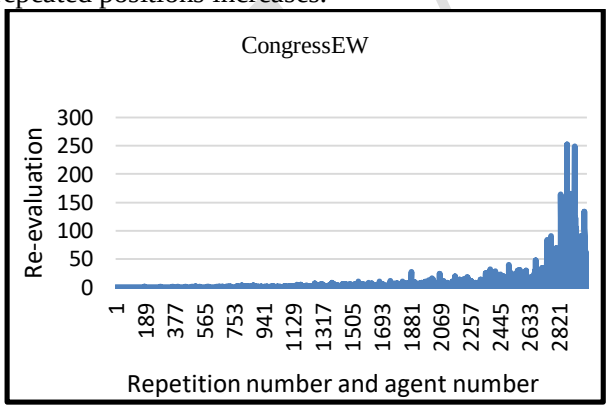


Figure 3: The number of repeated positions in iteration 1 to 100 with 30 search agents for CongressEW dataset

In equation 1, *R* specifies the direction of population movement. To increase the exploration of ARO algorithm, in this paper, also a different method to recognize *R* vector is introduced. This method used a table that stores the used R vectors that call vector table (VT). In each iteration of the algorithm, the *R* vector is generated 10 times, and among them, the case that has the greatest difference with the previous cases is selected. The selected item is added to the VT and the position of search agents is also updated with this vector. The number of vectors produced in each iteration (10) was experimentally obtained. Using the vector with the most deviations allows access to unvisited area in the search space which improves the performance of the ARO algorithm.

In the second section, MT and VT memory tables were introduced to improve the performance of ARO algorithm in feature selection problems. Figure 3 shows the flowchart of proposed algorithm. In the next section, the introduced method is tested on benchmark detasets.

ACCEPTED

Table 2: UCI benchmark datasets

	Datasets	No.of Objects	No.of Features
1	PenglungEW	73	325
2	Sonar	208	60
3	WaveformEW	5000	40
4	KrVsKpEW	3196	36
5	Ionosphere	351	34
6	BreastEW	569	30
7	SpectEW	267	22
8	Lymphography	148	18
9	CongressEW	435	16
10	Vote	300	16
11	Zoo	62	16
12	Exactly	1000	13
13	Exactly2	1000	13
14	M-of-n	1000	13
15	HeartEW	270	13
16	Wine	178	13
17	Tic-tac-toe	958	9
18	BreastCancer	699	9

5. Experiments and results

5.1. Definition of experiments

The proposed methods in this article were implemented using the Python programming language. The UCI standard benchmark datasets in Table 2, which were sourced from [33] work, were used to evaluate the efficiency of the proposed method. The datasets were partitioned such that 80% of the data was reserved for training, while the remaining portion was utilized for testing.

To assess the classification accuracy, the K-nearest neighbor (KNN) classifier was applied to the split datasets. The obtained classification accuracy was then incorporated into the following fitness function

$$fitness = \omega * (1 - ACC) + (1 - \omega)\frac{SF}{TF} \qquad (16)$$

where SF and TF are number of selected and total features respectively. ω is a coefficient to control the importance of the classification accuracy and the number of SF, which in our experiments $\omega = 0.99$ is considered. Therefore, the feature selection problem with a suitable fitness function became a minimization problem, which can be solved by the proposed method.

The MARO algorithm employed to solve feature selction problem to find the appropriate subset of features. This algorithm was compared with five state-of-art algorithms: ARO, AIEOU, SMO, WOA_CM, ASGW. All five algorithms were applied to the same split datasets. However, it is worth noting that the test and training datasets were randomly divided and resulting in average ten different executions with different test and training sets. Also, the number of algorithms population are 30 and the number of repetitions are 100 time. All experiments were repeated 5 times and their results were averaged to provide reliable results.

The implementation of the experiment was carried out on a system with the following specifications:

CPU: Intel Core i3, 2.4 GHz

RAM: 4 GB.

The algorithms are compared based on three criteria:

- Classification accuracy (Acc): The primary objective is to enhance the classification accuracy, which is measured using the K-nearest neighbor (KNN) algorithm. This is achieved by eliminating irrelevant and redundant features from the datasets.
- Selected features (SF): The second priority is to reduce the number of selected features.
- Time: One of the important parameters in the implementation of algorithms is its run time. Therefore, the execution time has also been examined in these experiments.

5.2. Experiments Results

This section presents a comparison between MARO and five state-of-the-art algorithms, demonstrating the superior performance of MARO. Compared algorithms include ARO, AIEOU, SMO, WOA_CM, ASGW. Throughout the experiments, a population size of 30 and a maximum iteration count of 100 were chosen, with the experiment being repeated 5 times.

All the investigated algorithms and the proposed algorithm have been implemented in Python and are accessible in an online repository[1]. Additionally, the datasets used were obtained from the UCI [2]website.

Table presents Table 3 a comparison of MARO with five state-of-the-art algorithms based on their average classification accuracy. MARO beats ARO in 10 datasets on the average classification accuracy In the other 7 datasets, they give equal results and ARO performs better only in one datasets. As a result, MARO performs better than ARO in average classification accuracy over benchmark datasets.

Figure 4: The number of datasets compared to each other in terms of SF

[1] https://github.com/alihamdipour/MARO

[2] https://archive.ics.uci.edu/datasets

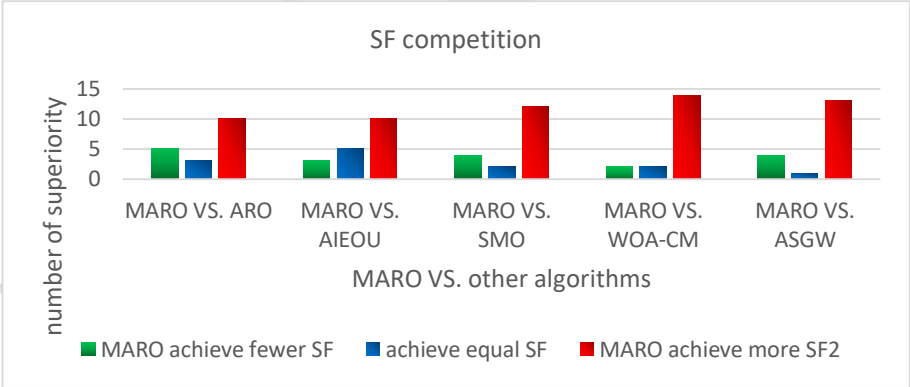
SUBMITTED

Table 3: Comparison of algorithms with the proposed method in terms of classification accuracy

Dataset	MARO	ARO	AIEOU	SMO	WOA-CM	ASGW	Original
Exactly	**99.8**	99.2	97.7	98.1	97.9	81.2	70
Wine	**100**	**100**	**100**	**100**	**100**	98.89	69.44
Vote	**98.33**	**98.33**	**98.33**	**98.33**	**98.33**	97.33	91.67
SpectEW	**91.48**	90.74	91.11	90.74	91.11	90.74	81.48
CongressEW	**97.47**	96.78	**97.47**	97.24	96.78	96.55	93.1
Ionosphere	**98**	96.86	94.57	94	94.29	93.43	84.29
Zoo	**100**	**100**	**100**	**100**	**100**	**100**	80
Breastcancer	**98.57**	**98.57**	**98.57**	**98.57**	**98.57**	98	58.57
Lymphography	96.67	95.33	**98**	95.33	94.67	92.67	66.67
Exactly2	78.4	78.3	78.4	**78.5**	78.4	77.1	77.5
HeartEW	**85.56**	85.19	84.07	84.07	82.22	74.44	61.11
M-of-n	**99.9**	99.8	99.5	98.7	99.3	93.3	88.5
Sonar	**97.62**	97.14	**97.62**	**97.62**	**97.62**	96.67	90.48
BreastEW	**96.49**	**96.49**	**96.49**	**96.49**	**96.49**	**96.49**	90.35
PenglungEW	**100**	**100**	**100**	**100**	**100**	97.33	86.67
WaveformEW	**84.72**	84.64	83.56	83.84	83.64	83.54	81.5
KrVsKpEW	98.2	**98.28**	97.57	97.89	97.26	97.42	94.84
Tic-tac-toe	**83.85**	**83.85**	**83.85**	**83.85**	**83.85**	**81.77**	83.85
Friedman Test Rank	**5.86**	4.81	4.78	4.64	4.42	2.25	1.25
Assigned Rank	1th	2th	3th	4th	5th	6th	7th

MARO beats AIEOU in 7 datasets on the average classification accuracy criteria. In the other 10 datasets, they give equal results and AIEOU performs better only in one datasets. As a result, MARO performs better than AIEOU in average classification accuracy over benchmark datasets. MARO demonstrates superior average classification accuracy results compared to the five state-of-the-art algorithms that were evaluated.

In Figure 5, the superiority of MARO is shown statistically. Green color indicates the number of superiority, blue color indicates the number of equality, and red color indicates the scarcity in the benchmark dataset. As can be seen, MARO has performed better in terms of classification accuracy compared to all 5 compared algorithms.



SMO had better result only one dataset. MARO has better results in 9 datasets and equal results have been obtained in the rest datasets. As a result, MARO performs better than SMO in average classification accuracy over datasets. MARO beats WOA-CM in 9 datasets and has been obtained the same result in the rest 9 dataset over the average classification accuracy results. As a result, MARO performs better than WOA-CM in average classification accuracy over datasets. ASGW has not had better results MARO in any dataset. MARO has better results in 16 datasets and equal results have been obtained in the rest datasets. As a result, MARO performs better than ASGW in average classification accuracy over datasets. Overall,

The average SF achieved of MARO and five state-of-the-art algorithms are presented in Table 3*Figure 5*. MARO beats AIEOU in 10 datasets and AIEOU beats MARO in 3 datasets. In the other one dataset, they give equal results. As a result, MARO performs the better than AIEOU in average count of selected features across datasets. SMO had better results in 4 dataset based on average SF. MARO has better results in 12 datasets and equal results have been obtained in the 2 datasets. As a result, MARO performs better

than SMO in average count of selected features across datasets. MARO beats WOA-CM in 14 datasets based on average SF and in the other 2 dataset, they give equal results. WOA-CM has better results MARO in 2 dataset. As a result, MARO performs better than WOA-CM in average count of selected features across datasets. ASGW beats MARO in just 4 datasets and MARO beats ASGW in other 13 datasets based on the average SF. As a result, MARO performs better than ASGW in average count of selected features across datasets. So generally, MARO has obtained better results than 4 state-of-art feature selection algorithms based on the average SF.

Table 3: The results of comparing the selected features of MARO and state-of-art algorithms

Dataset	MARO	ARO	AIEOU	SMO	WOA_CM	ASGW	Original
Exactly	**6.4**	**6.4**	7	6.8	7	9	13
Wine	**3.8**	4	4.4	4.4	4.8	6.4	13
Vote	**3.4**	3.6	4.6	5.8	6.6	7.2	16
SpectEW	10.4	**6**	9	8	12.2	11.6	22
CongressEW	7.40	**6.20**	7.60	6.80	6.80	7.40	16
Ionosphere	**5**	7.6	14.2	13.4	16	14	34
Zoo	**9**	9.2	**9**	9.2	9.2	10.2	16
Breastcancer	**3**	**3**	**3**	3.2	3.4	5.6	9
Lymphography	9	**6**	7.8	8	9.6	8.4	18
Exactly2	8.6	8.2	8.6	9	8.6	**7.6**	13
HeartEW	**4.2**	4.6	**4.2**	**4.2**	4.6	5.8	13
M-of-n	**6.2**	6.4	6.6	7	7	9.4	13
Sonar	**25.4**	26.8	30.6	31	36.2	37.4	60
BreastEW	**13.6**	15.4	14.8	15.2	19	18.6	30
PenglungEW	**81**	91.8	168.4	165.4	197	185	325
WaveformEW	28.4	27.4	24.6	**23**	25.8	28	40
KrVsKpEW	**18.5**	24	21.5	20	24.5	23	36
Tic-tac-toe	9	9	9	9	9	**7**	9
Friedman Test Rank	**2.39**	2.58	3.22	3.28	4.86	4.81	6.86
Assigned Rank	**1th**	2th	3th	4th	6th	5th	7th

In Figure 4, the superiority of MARO is shown statistically. Green color indicates the number of superiority, blue color indicates the number of equality, and red color indicates the scarcity in the benchmark dataset. As can be seen, MARO has performed better in terms of selected feature compared to all 5 compared algorithms.

Figure 6 shows the value of the fitness function for all 18 benchmark datasets. In each of the graphs of this figure, it is possible to compare the proposed algorithm with 5 other state-of-art algorithms in the corresponding dataset. These graphs are the value of the fitting function resulting from equation (16) in different iterations. These graphs are the result of the average execution of 5 times. Each time, the dataset is separated from a different part for training and testing to obtain reliable results. By using these graphs, we can check and analyze the performance of all compared algorithms compared to each other in different iterations. As

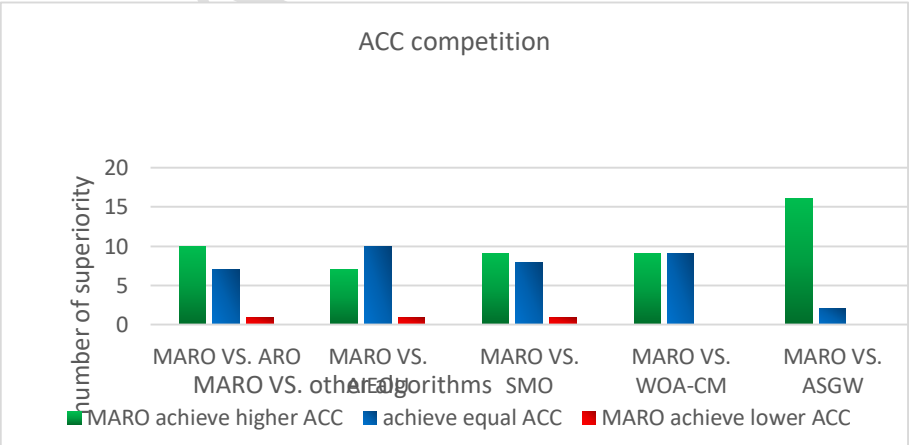


Figure 5: The number of datasets compared to each other in terms of classification accuracy

SCCEPTED

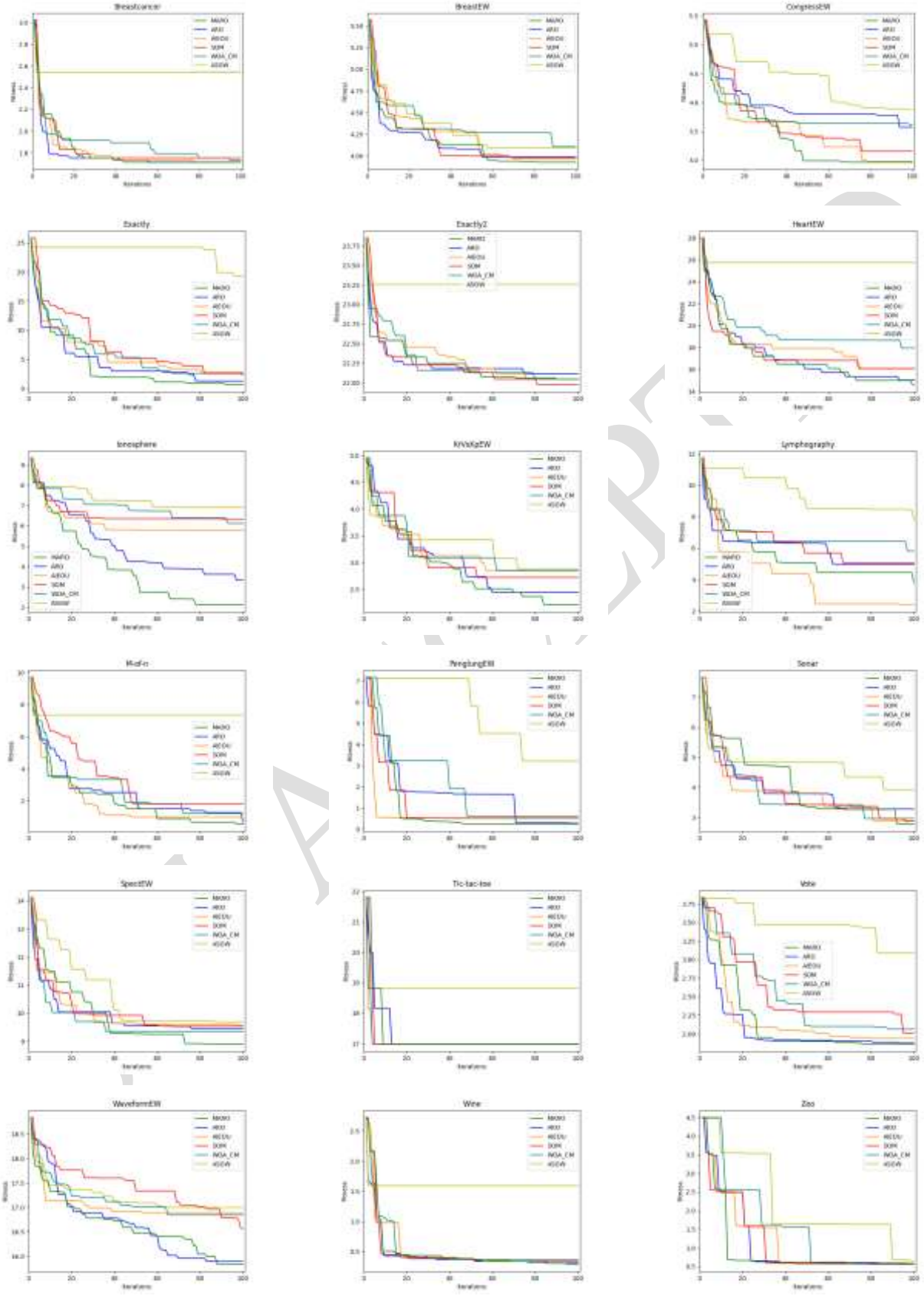


Figure 6: Fitness comparison graphs for 18 benchmark datasets with all 6 compared algorithms

Table 4: The results of comparing the selected execution time of MARO algorithm and state-of-art algorithms

datasets	ARO	AIEOU	ASGW	WOA_CM	SMO	MARO
Exactly	62.13	219.25	63.38	62.48	61.95	100.87
Wine	16.16	62.34	15.84	15.12	15.7	41.69
Vote	24.66	77.65	24.91	24.76	24.82	42.16
SpectEW	26.79	75.38	24.85	24.67	26.45	41.65
CongressEW	27.74	103.39	28.27	27.69	28.31	38.51
Ionosphere	28.72	113.23	30.58	29.97	31.83	36.6
Zoo	13.12	50.14	12.99	12.76	13.28	22.23
Breastcancer	44.16	158.64	44.38	43.38	43.55	172.95
Lymphography	13.78	51.97	13.22	13.15	14.31	20.02
Exactly2	64.42	223.68	63.25	62.55	62.4	83.88
HeartEW	23.87	77.6	24.95	24.78	24.71	34.29
M-of-n	64.48	235.23	66.37	64.6	63.9	102.33
Sonar	25.28	88.15	25.46	25.45	25.18	25.24
BreastEW	38.99	150.64	40.08	39.57	39.87	46.24
PenglungEW	14.64	93.93	17.9	16.02	27.13	19.39
WaveformEW	271.85	994.68	270.37	267.44	271.84	280.35
KrVsKpEW	342.34	1249.24	381.43	360.82	336.43	340.08
Tic-tac-toe	62.92	227.96	60.09	58.42	57.06	192.09
Friedman Test Rank	2.56	5.94	3.33	2.00	2.50	4.67

can be seen, the proposed algorithm has obtained a lower value of the fitness function in most cases.

The average execution time of each algorithm while they have been less classification time than MARO, they have been less classification accurate than MARO. Therefore, MARO can find the best answer among the compared algorithms based on the average classification accuracy and average SF in an acceptable time.

6. Conclusion

Two approaches for feature selection problem are presented in this work. The first approach employs the ARO algorithm for feature selection, which constitutes the first systematic effort to apply this algorithm to feature selection problem. Second, the improved ARO algorithm was introduced. The comparison of these two approaches for feature selection problems reveals that the proposed improved algorithm is more efficient. Following that, the proposed algorithm is tested against five of the state-of-art algorithms. According to the comparison results, the proposed algorithm is capable of improving classification accuracy while reducing the SF more effectively than the other compered algorithms. Moreover, given the p-value < 0.001, a statistically significant difference is observed among the algorithms' performance.

For the future, a binary version of MARO algorithm can be introduced. Since the feature selection problem is inherently a binary problem. By binarizing we can avoid re-evaluating previous values more and more precisely. With this, the speed of the algorithm improved and the classification accuracy increases. Moreover, by enhancing the memory table structure for faster access and retrieval of positions, the performance of this method can be improved.

7. References

[1] S. Arora and P. Anand, "Binary butterfly optimization approaches for feature selection," *Expert Syst Appl*, vol. 116, no. 12, pp. 147–160, 2019, doi: 10.1016/j.eswa.2018.08.051.

[2] Y. Saeys, I. Inza, and P. Larranaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007, doi: 10.1093/bioinformatics/btm344.

[3] E.-G. Talbi, *Metaheuristics : from design to implementation*. John Wiley & Sons, 2009.

[4] Y. Gao, Y. Zhou, and Q. Luo, "An Efficient Binary Equilibrium Optimizer Algorithm for Feature Selection," *IEEE Access*, vol. 8, no. 12, pp. 140936–140963, 2020, doi: 10.1109/access.2020.3013617.

[5] M. Gan and L. Zhang, "Iteratively local fisher score for feature selection," *Applied Intelligence*, vol. 51, no. 8, pp. 6167–6181, 2021, doi: 10.1007/s10489-020-02141-0.

[6] P. Somol, P. Pudil, J. Novovičová, and P. Paclík, "Adaptive floating search methods in feature selection," *Pattern Recognit Lett*, vol. 20, no. 11–13, pp. 1157–1163, 1999, doi: 10.1016/s0167-8655(99)00083-5.

[7] K. Yan, L. Ma, Y. Dai, W. Shen, J. Z, and D. Xie, "Cost-sensitive and sequential feature selection for chiller fault detection and diagnosis," *International Journal of Refrigeration-revue Internationale Du Froid*, vol. 86, no. 12, pp. 401–409, 2018, doi: 10.1016/j.ijrefrig.2017.11.003.

SOL ACCEPTED

[8] H. M. Abdulwahab, S. Ajitha, and M. A. N. Saif, "Feature selection techniques in the context of big data: taxonomy and analysis," *Applied Intelligence*, no. 12, 2022, doi: 10.1007/s10489-021-03118-3.

[9] R. Guha, M. Ghosh, S. Mutsuddi, R. Sarkar, and S. Mirjalili, "Embedded chaotic whale survival algorithm for filter–wrapper feature selection," *Soft comput*, vol. 24, no. 17, pp. 12821–12843, 2020, doi: 10.1007/s00500-020-05183-1.

[10] J. Zhang *et al.*, "Fast Multilabel Feature Selection via Global Relevance and Redundancy Optimization," *IEEE Trans Neural Netw Learn Syst*, vol. 35, no. 4, pp. 5721–5734, 2024, doi: 10.1109/tnnls.2022.3208956.

[11] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[12] sajad esfandyari and vahid rafe, "Using the Particle Swarm Optimization Algorithm to Generate the Minimum Test Suite in Covering Array with Uniform Strength," *Soft Computing Journal*, vol. 8, no. 2, pp. 66–79, 2021, doi: 10.22052/8.2.66.

[13] S. Oreski and G. Oreski, "Genetic algorithm-based heuristic for feature selection in credit risk assessment," *Expert Syst Appl*, vol. 41, no. 4, pp. 2052–2064, 2014, doi: 10.1016/j.eswa.2013.09.004.

[14] I. Jain, V. K. Jain, and R. Jain, "Correlation feature selection based improved-Binary Particle Swarm Optimization for gene selection and cancer classification," *Appl Soft Comput*, vol. 62, pp. 203–215, 2018, doi: 10.1016/j.asoc.2017.09.038.

[15] K. Chen, F. Zhou, and X. Yuan, "Hybrid particle swarm optimization with spiral-shaped mechanism for feature selection," *Expert Syst Appl*, vol. 128, pp. 140–156, 2019, doi: 10.1016/j.eswa.2019.03.039.

[16] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996, doi: 10.1109/3477.484436.

[17] H. Shafiei, V. Rafe, and M. Amiri, "Optimization of vehicle routing based on the combination of ant colony and particle swarm algorithms with the heuristic function of the cosine of angles," *Soft Computing Journal*, vol. 12, no. 2, pp. 146–164, 2024, doi: 10.22052/scj.2023.248702.1118.

[18] R. A. Ibrahim, A. A. Ewees, D. Oliva, M. Abd Elaziz, and S. Lu, "Improved salp swarm algorithm based on particle swarm optimization for feature selection," *J Ambient Intell Humaniz Comput*, vol. 10, no. 8, pp. 3155–3169, 2018, doi: 10.1007/s12652-018-1031-9.

[19] N. Neggaz, A. A. Ewees, M. A. Elaziz, and M. Mafarja, "Boosting salp swarm algorithm by sine cosine algorithm and disrupt operator for feature selection," *Expert Syst Appl*, vol. 145, p. 113103, 2020, doi: 10.1016/j.eswa.2019.113103.

[20] M. Abdel-Basset, D. El-Shahat, I. El-henawy, V. H. C. de Albuquerque, and S. Mirjalili, "A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection," *Expert Syst Appl*, vol. 139, p. 112824, 2020, doi: 10.1016/j.eswa.2019.112824.

[21] Q. Tu, X. Chen, and X. Liu, "Multi-strategy ensemble grey wolf optimizer and its application to feature selection," *Appl Soft Comput*, vol. 76, pp. 16–30, 2019, doi: 10.1016/j.asoc.2018.11.047.

[22] E. Emary, W. Yamany, A. E. Hassanien, and V. Snasel, "Multi-Objective Gray-Wolf Optimization for Attribute Reduction," *Procedia Comput Sci*, vol. 65, pp. 623–632, 2015, doi: 10.1016/j.procs.2015.09.006.

[23] M. Prabukumar, L. Agilandeeswari, and K. Ganesan, "An intelligent lung cancer diagnosis system using cuckoo search optimization and support vector machine classifier," *J Ambient Intell Humaniz Comput*, vol. 10, no. 1, pp. 267–293, 2017, doi: 10.1007/s12652-017-0655-5.

[24] X. Wang, Y. Zhang, X. Sun, Y. Wang, and C. Du, "Multi-objective feature selection based on artificial bee colony: An acceleration approach with variable sample size," *Appl Soft Comput*, vol. 88, p. 106041, 2020, doi: 10.1016/j.asoc.2019.106041.

[25] Y. Zhang, S. Cheng, Y. Shi, D. Gong, and X. Zhao, "Cost-sensitive feature selection using two-archive multi-objective artificial bee colony algorithm," *Expert Syst Appl*, vol. 137, pp. 46–58, 2019, doi: 10.1016/j.eswa.2019.06.044.

[26] E. Hancer, "A new multi-objective differential evolution approach for simultaneous clustering and feature selection," *Eng Appl Artif Intell*, vol. 87, p. 103307, 2020, doi: 10.1016/j.engappai.2019.103307.

[27] E. Hancer, B. Xue, and M. Zhang, "Differential evolution for filter feature selection based on information theory and feature ranking," *Knowl Based Syst*, vol. 140, pp. 103–119, 2018, doi: 10.1016/j.knosys.2017.10.028.

[28] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems," *Advances in Engineering Software*, vol. 114, pp. 163–191, 2017, doi: 10.1016/j.advengsoft.2017.07.002.

[29] R. Rajabioun, "Cuckoo Optimization Algorithm," *Appl Soft Comput*, vol. 11, no. 8, pp. 5508–5518, 2011, doi: 10.1016/j.asoc.2011.05.008.

SOFT ACCEPTED

[30] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, 1997, doi: 10.1023/A:1008202821328.

[31] L. Wang, Q. Cao, Z. Zhang, S. Mirjalili, and W. Zhao, "Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems," *Eng Appl Artif Intell*, vol. 114, p. 105082, 2022, doi: 10.1016/j.engappai.2022.105082.

[32] A. Hamdipour, A. Basiri, and M. Zaare, "Numerical solution of nonlinear equations systems with improved meta-heuristic ARO algorithm," *Soft Computing Journal*, p., 2024, doi: 10.22052/scj.2025.254888.1244.

[33] Blake and Catherine L, *UCI repository of machine learning databases*. Department of Information and Computer Science, University of California, 1998.

SCJ ACCEPTED