STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **16**, December 2026, pp 6412–6434.
Published online in International Academic Press (www.IAPress.org)

CARO: Chaotic Artificial Rabbit Optimization Algorithm for Feature Selection

Mohammed Abdelrazek [1,*], A. H. El-Baz [2], Mohamed Abd Elaziz [3,4]

[1]*Department of Mathematics, Faculty of Science, Damietta University, New Damietta 34517, Egypt*
[2]*Department of Computer Science, Faculty of Computers and Artificial Intelligence, Damietta University, New Damietta 34517, Egypt*
[3] *College of Engineering and Information Technology, Ajman University, Ajman, UAE*
[4]*Artificial Intelligence Research Center (AIRC), College of Engineering and Information Technology, Ajman University, Ajman, UAE*

Abstract A modified version of Artificial Rabbit Optimization Algorithm (ARO) for feature selection is proposed in this paper. The ARO method is one of the swarm intelligence algorithms that mimic survival strategies of rabbits in nature. This research introduces a Chaotic Artificial Rabbit Optimization (CARO) method, which incorporates ten chaotic maps to improve feature selection performance. Utilizing a wrapper-based approach, CARO aims to identify optimal features that maximize classification accuracy. By integrating chaotic maps to modify key elements of the optimization process, the method seeks to accelerate convergence and enhance algorithmic effectiveness. Experiments on UCI classification datasets compare classification accuracy and F-measure against the original ARO, Ant Colony Optimization, Particle Swarm Optimization, and the Whale Optimization Algorithm. Selected-feature counts and reduction percentages are also examined for ARO and all CARO variants. Under the original holdout protocol, the Circle-map variant, CARO2, achieved the best overall accuracy ranking among the chaotic variants. Besides, the proposed method is evaluated using 12 functions from CEC'2022 benchmarks where it outperformed the existing methods.

Keywords Feature selection; Chaotic maps; Artificial Rabbit Optimization Algorithm (ARO); Meta-heuristic

DOI: 10.19139/soic-2310-5070-4040

1. Introduction

In machine learning and data mining, feature selection serves a critical purpose by identifying and extracting the most relevant attributes from a comprehensive feature set. Feature selection aims to enhance machine learning model performance by strategically reducing input dimensionality, systematically eliminating non-informative attributes, and extracting the most discriminative features that contribute significantly to predictive accuracy [1].

There are many different techniques for feature selection, including filter methods, wrapper methods, and embedded methods [2]. Here are some of the most common techniques:

1. Filter methods: Employ statistical techniques to evaluate and rank feature relevance. These approaches assess individual feature significance through methods like correlation analysis and mutual information quantification, systematically identifying the most informative attributes [3].
2. Wrapper methods: Iteratively evaluate model performance across different feature subsets, employing search strategies like recursive feature elimination and forward selection to identify optimal feature combinations that maximize predictive accuracy [4].
3. Embedded methods: Integrate feature selection directly into the model-building process, utilizing techniques like regularization in algorithms such as Lasso and Elastic Net to automatically diminish or eliminate less critical features by shrinking their coefficients to zero [5].

*Correspondence to: Mohammed Abdelrazek (Email: Mohammedrazek@gmail.com).

ISSN 2310-5070 (online) ISSN 2311-004X (print)
Copyright © 2026 International Academic Press

Contemporary research increasingly leverages meta-heuristic optimization techniques to pinpoint the most distinguishing features in complex datasets. These algorithms are a class of optimization techniques that can be used to solve complex optimization problems, including feature selection [6].

Meta-heuristic algorithms are increasingly being used for feature selection in machine learning. These algorithms are optimization techniques that are used to solve complex optimization challenges, including feature selection. By mimicking phenomena like evolutionary processes and collective animal behaviors, these algorithms efficiently navigate complex solution spaces [7]. One of the main advantages of using meta-heuristic algorithms for feature selection is that they can quickly explore a large search space and identify good solutions even when the problem is complex, and the search space is large. Conventional feature selection approaches often encounter computational limitations when confronting high-dimensional datasets, necessitating more sophisticated and efficient algorithmic solutions [8].

Meta-heuristic algorithms can be categorized into approaches targeting singular or multiple performance objectives. Single-objective algorithms are designed to optimize a single objective function, such as classification accuracy or prediction error [9]. Examples of single-objective algorithms that have been used for feature selection include Particle Swarm Optimization (PSO) [10], Genetic Algorithms (GA) [11], and Ant Colony Optimization (ACO) [12].

Satyajit et al. [13] proposed a hybrid algorithm that combines ACO and GA for feature selection in breast cancer diagnosis. The proposed algorithm was evaluated using three different datasets and achieved high classification accuracy and low feature selection error.

Feature selection by Cuckoo Search (CS) algorithm was proposed by Gunavathi and Premalatha [14]. They applied the CS algorithm for classifying microarray gene data related to cancer. They ranked the features using T and F-statistics, employing a K-nearest Neighbors (KNN) classifier as the fitness function. Their results indicated that the CS algorithm achieved 100% accuracy on most datasets.

In 2010, Min and Fangfang [15] created a two-step feature selection strategy that merged filtering and wrapper methods. The initial filtering stage employed multiple statistical approaches including correlative feature selection, relief, Mahalanobis distance, multivariate correlation, and mutual information to screen features. This was followed by a wrapper phase where they tested different evolutionary algorithms: adaptive genetic algorithms, chaotic binary particle swarm optimization, and a clonal selection algorithm. Among these wrapper methods, the clonal selection algorithm proved most effective at finding optimal feature combinations.

Hamraz et al. [16] developed a new feature selection method called (WSNR) weighted signal to noise ratio for high-dimensional microarray data analysis. The proposed algorithm was evaluated using eight gene expression datasets and achieved high classification accuracy while reducing the number of features.

Hybrid optimization methods called SSAPSO for the feature selection problem were developed by Rehab et al. [17], they combine the Slap Swarm Algorithm (SSA) and Particle Swarm Optimization (PSO). The algorithm improves the efficacy of the exploration and exploitation steps. The performance of SSAPSO is tested on benchmark functions and different UCI datasets, and it is shown that the algorithm outperforms other similar approaches in terms of accuracy without increasing computational effort. The experimental results provide evidence of the effectiveness of SSAPSO for feature selection.

Abdelrazek et al. [18] used different chaotic maps to develop dwarf mongoose optimization algorithm called CDMO which gives enhancement than original DMO and they compare it with most used meta heuristic algorithms on ten UCI datasets; their proposed method shows more effective feature selection performance than other well-known methods.

For multi-objective feature selection, Hamdani et al. [19] proposed algorithm based on Non-Dominated Sorting Genetic Algorithm (NSGA-II) for classification problems. The proposed algorithm was evaluated using five UCI datasets and achieved high classification accuracy while reducing the number of features and maintaining diversity among the selected features.

Recent studies have introduced different strategies to enhance the performance and applicability of metaheuristic optimization algorithms. Gharehchopogh et al. [20] developed a modified farmland fertility algorithm for constrained engineering optimization by improving its solution-update mechanisms. Anka et al. [21] proposed a discrete version of the Puma Optimizer for combinatorial optimization problems, demonstrating the importance of adapting continuous metaheuristics to discrete search spaces. Rishakan and Gharehchopogh [22] introduced

the Resultant Vector Strategy to improve the balance between exploration and exploitation and enhance solution quality. In addition, several improved variants of the Mountain Gazelle Optimizer were investigated to address convergence and search-efficiency limitations [23]. The Mountain Gazelle Optimizer has also been integrated with a hybrid deep-learning framework for diabetes prediction, illustrating the growing applicability of enhanced metaheuristic algorithms to machine-learning problems [24].

The comparison in Table 1 shows that previous studies have advanced feature selection through hybrid metaheuristics, filter-wrapper schemes, multi-objective formulations, and chaos-enhanced optimizers.

In this study, the proposed feature selection methodology employs Artificial Rabbit Optimization (ARO) [25] to systematically identify minimal feature subsets, utilizing K-Nearest Neighbor classification for comprehensive performance evaluation. Chaotic maps are combined with ARO to improve the results. These methods demonstrated superior efficiency and effectiveness compared to other contemporary techniques. The primary contributions of this paper can be summarized as follows:

- A novel hybrid approach named CARO is developed, integrating Artificial Rabbit Optimization with chaotic mapping techniques to enhance feature selection process, CARO replaced selected random variables in the detour-foraging and random-hiding updates with values generated by chaotic maps.
- Evaluate the proposed CARO method using ten datasets from UCI and employing the five-nearest-neighbor classifier to demonstrate its effectiveness.
- Ten chaotic maps are examined within the same feature-selection framework. Their classification results are compared using accuracy and F-measure to identify which maps provide useful improvements.
- The performance of the proposed feature selection method is compared with the original other well-known meta-heuristic-based feature selection methods which give superior results.
- Furthermore, the proposed method was additionally evaluated using the CEC 2022 benchmark functions, where it outperformed the existing methods.

The paper is organized as follows: Background information about the ARO algorithm and chaotic maps is presented in Section 2. Section 3 describes the proposed methodology in detail. The experimental findings and their analysis are discussed in Section 4, followed by conclusions in Section 5.

Table 1. Contribution and limitations of existing methods employed for feature selection

Authors	Technique	Contribution	Result	Limitation
Panigrahi et al. [13]	GACO	To resolve feature selection for breast-cancer diagnosis	High accuracy and low feature-selection error on three datasets	Single WDBC dataset; multi-objective accuracy-feature trade-offs were left for future work
Gunavathi and Premalatha [14]	CS	To select cancer microarray features using T/F statistics and KNN	100% accuracy on most evaluated datasets	Dataset-dependent results (92.68% on Prostate and 95% on Colon); limited to microarray data and KNN
Min and Fangfang [15]	Filter-wrapper hybrid	To combine statistical filtering with AGA, chaotic binary PSO, and clonal selection	The clonal-selection wrapper performed best among the tested alternatives	Multi-stage pipeline increases computational complexity; validation covered only three datasets
Hamraz et al. [16]	WSNR	To rank and select high-dimensional gene-expression features	High accuracy with fewer features on eight datasets	Supervised only; robust outlier handling, unsupervised use, and other SVM kernels remain untested

Authors	Technique	Contribution	Result	Limitation
Ibrahim et al. [17]	SSAPSO	To improve exploration and exploitation in feature selection	Improved accuracy without increased computational effort on UCI data	Selected feature subsets vary between runs, requiring their intersection to improve stability
Abdelrazek et al. [18]	CDMO	To enhance DMO with chaotic maps for wrapper feature selection	Improved the original DMO and produced competitive results on ten UCI datasets	Evaluation used benchmark/UCI data; real-world medical validation was left for future work
Hamdani et al. [19]	NSGA-II	To balance accuracy, feature count, and diversity using multi-objective feature selection	High accuracy with feature reduction on five UCI datasets	Produces a Pareto set requiring a final decision step; validation covered only five datasets
Gharehchopogh et al. [20]	MFFA	To improve solution-update rules for constrained engineering optimization	Improved search and solution-update performance	Tested only on constrained engineering designs; generality beyond this problem class is unverified
Anka et al. [21]	DPO	To adapt Puma Optimizer to discrete combinatorial spaces	Demonstrated effective discrete search	Requires problem-specific manual tuning and added computation; its non-binary design limits direct use on binary problems
Rishakan and Gharehchopogh [22]	RVS	To improve search direction and exploration-exploitation balance	Improved balance and solution quality	Validation focused on continuous single-objective benchmark and engineering problems; other settings remain unverified
Gharehchopogh and Rishakan [23]	IMGO/MGO	To improve MGO convergence and search efficiency	Enhanced convergence and search efficiency	Limited to continuous benchmark and engineering problems; discrete/binary and multi-objective cases were not evaluated
Valilou et al. [24]	MGO-HDL	To optimize hybrid deep learning for diabetes prediction	Demonstrated the applicability of enhanced metaheuristics to diabetes prediction	Validated on the Pima diabetes dataset only; no external or prospective clinical validation was reported
Wang et al. [25]	ARO	To introduce detour foraging and random hiding for global optimization	Effective performance on global and engineering optimization problems	Tested on continuous single-objective benchmarks and five engineering designs; other formulations were not evaluated

2. Background

2.1. Artificial Rabbit Optimization (ARO)

The Artificial Rabbit Optimization (ARO) algorithm is based on the survival strategies of rabbits in nature. Through evolutionary adaptation, rabbits have developed sophisticated survival mechanisms, including foraging strategies that involve exploring distant territories and utilizing random burrow concealment to evade potential threats. Rabbits have strong muscles and tendons that allow them to run fast, stop suddenly, turn sharply, and confuse their enemies. ARO takes inspiration from these strategies and applies them to optimization problems. The algorithm uses a detour foraging strategy as exploration and a random hiding strategy as exploitation. The algorithm also adapts by switching between these strategies according to energy levels [25].

The exploration is detouring foraging. During foraging, rabbits tend to look for food in faraway locations and ignore what is nearby, which is known as detour foraging. In their natural behavior, rabbits venture beyond their territories to graze in different areas. The ARO algorithm mimics this by assigning each artificial rabbit a designated foraging zone containing grass and burrows. These virtual rabbits then simulate natural foraging patterns by randomly visiting other zones and circling around food sources to maximize their food intake. The detour foraging behavior of ARO means that each search individual updates its position towards another randomly chosen individual in the swarm and adds a perturbation. The mathematical model of ARO [25] can be expressed as follows:

$$\overrightarrow{v}_i(t+1) = \overrightarrow{x}_j(t) + R\left(\overrightarrow{x}_i(t) - \overrightarrow{x}_j(t)\right) + \text{round}\left(0.5\left(0.05 + r_1\right)\right) n_1. \tag{1}$$

$$i,\ j = 1 \ldots\ldots\ldots n \ and \ j \neq i$$

$$R = L\,C. \tag{2}$$

$$L = \left(e - e^{\left(\frac{t-1}{T}\right)^2}\right) \sin(2\pi r_2). \tag{3}$$

$$c(k) = \begin{cases} 1 \ if \ k == g(l) \\ \\ 0 \ else \end{cases} \qquad k = 1, \ldots, d \text{ and } l = 1, \ldots\ldots, [r_3.d] \tag{4}$$

$$g = \text{randprem}(d). \tag{5}$$

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

This equation calculates where the *i*th rabbit will potentially move in the following iteration $(t+1)$, denoted as $\overrightarrow{v}_i(t+1)$, which is based on the rabbit's current position $\overrightarrow{x}_j(t)$ at time *t*, the size of the rabbit population is *n*, *d* is the dimension of the problem being solved. The maximum number of iterations is represented as *T*, round represents rounding up to the nearest integer, *randperm* generates a random permutation of integers, r_1, r_2 and r_3 which are random numbers between 0 and 1, *L* represents the rabbit's movement pace when performing detour foraging, and n_1 follows the standard normal distribution.

Rabbits typically create several burrows around their nest to avoid being caught by predators. In ARO, a rabbit generates *d* burrows along each dimension of the search space during each iteration. To reduce the likelihood of being preyed on, the rabbit randomly selects one of the burrows to hide in. The following equations show the mathematical model of random hiding strategy [25]:

$$\overrightarrow{v}_i(t+1) = \overrightarrow{x}_j(t) + R\left(r_4\ \overrightarrow{b}_{i,r}(t) - \overrightarrow{x}_i(t)\right), \quad i = 1, \ldots, n. \tag{7}$$

$$\overrightarrow{b}_{i,r}(t) = \overrightarrow{x}_i(t) + H\ g_r\ \overrightarrow{x}_i(t). \tag{8}$$

$$H = \frac{T - t + 1}{T} r_4. \tag{9}$$

$$g_r(k) = \begin{cases} 1 \; if \; k == [r_5.d] \\ \\ 0 \; else \end{cases} \qquad k = 1, \ldots, d. \tag{10}$$

Where $\overrightarrow{b}_{i,r}$ represents a randomly selected burrows for hiding from its d burrows, and r_4 and r_5 are two random numbers in (0, 1). The position updated of the ith rabbit after one of both detour foraging and random hiding achieved:

$$\overrightarrow{x}_i(t+1) = \begin{cases} \overrightarrow{x}_i(t), & f\left(\overrightarrow{x}_i(t)\right) \leq f\left(\overrightarrow{v}_i(t+1)\right), \\ \overrightarrow{v}_i(t+1), & f\left(\overrightarrow{x}_i(t)\right) > f\left(\overrightarrow{v}_i(t+1)\right). \end{cases} \tag{11}$$

If the fitness of the potential location for the ith rabbit is higher than its existing position, the rabbit will relinquish its existing spot and remain at the newly created candidate position, which is determined by either Eq. (1) or Eq. (7). Algorithm 1 describes the pseudocode of ARO [25].

Algorithm 1: Artificial rabbit Optimization Algorithm (ARO)
1: *Initialize the solutions* X_i
2*: Evaluate the fitness and best solution (*Fit_i *,* X_{best}*) for each solution*
3*: Repeat*
4: ***For*** *i=1 : max size of population*
5: *Calculate the energy factor A(t)*
6: $A(t) = 4\left(1 - \frac{t}{T}\right)\ln\frac{1}{r}$
7: *Compute the fitness function* Fit_i
8: *Update the best solution*
9: ***If A>1***
10: *Choose rabbit randomly*
11: *Use Eq.(1). For detour foraging*
12: *Update the position using Eq.(11).*
13: ***Else***
14: *Use Eq.(7). For random hiding*
15: *Update the position using Eq.(11).*
16: ***End if***
17: ***End for***
18: ***Until*** *criterion is not met*

2.2. Chaotic maps

Chaos represents a dynamic phenomenon characterized by nonlinear systems, wherein infinitesimal modifications to initial conditions can trigger substantial behavioral transformations [26]. This intricate phenomenon generates semi-random behaviors through deterministic nonlinear mechanisms. The Chaos Optimization Algorithm (COA) emerges as a sophisticated search strategy, strategically transporting variables and parameters between chaotic and solution domains [27]. The algorithm's computational prowess lies in its capacity to navigate global optimization challenges across stochastic, regular, and periodic chaotic motion configurations. Its remarkable simplicity and rapid convergence properties have catalyzed extensive utilization in computational research over the preceding decade. This investigation leverages ten established one-dimensional chaotic maps, frequently employed in computational research, to comprehensively explore complex optimization landscapes. The various chaotic maps depicted in Figure 1 exhibit distinct patterns, enabling an evaluation of how ARO performs across different mapping behaviors. The used chaotic maps are defined in Table 2. In this table, q denotes the index of the chaotic sequence,

and p_q is the *q-th* number in the chaotic sequence. The initial point p_0 is set to 0.6 for all chaos maps. Values ranging from 0.1 to 0.9 were evaluated, and 0.6 produced the best observed performance among the tested values. Therefore, this value was adopted for the CARO experiments and kept fixed when comparing the ten chaotic maps.

2.3. Chaotic maps for ARO

Chaotic maps generate deterministic sequences that can supply coefficients for the position-update equations of population-based optimization algorithms. Their use offers a way to vary search trajectories through the existing movement operators [28]. As shown in Fig.1, the maps have different behaviors and hence the behavior of ARO can be tested using different maps. The potential influence of a chaotic map depends on its dynamical properties and parameter settings. Under appropriate conditions, sensitivity to initial conditions produces different coefficient sequences from nearby starting values. For an ergodic map, trajectories visit regions of the coefficient domain according to its invariant measure. The resulting distribution and temporal ordering of values influence the inputs supplied to the movement equations [29].

The ARO algorithm models three aspects of rabbit behaviour, guided detour foraging for exploration, random hiding for exploitation, and energy shrinkage to regulate the transition between these phases. These search mechanisms are described in Equations (1), (8), and the energy factor (*A(t)*), defined in Algorithm 1, n_1 in the detour foraging strategy is a guided factor in the ARO algorithm. The algorithm produces a variety of solutions. They are subject to the normal distribution $N(0, 1)$ with zero mean and unit standard deviation.

In CARO, map-generated values replace selected random coefficients in these movement equations. During detour foraging, the substituted coefficient influences the activation of the Normal distribution. During random hiding, it scales the burrow-related term in the position update. Differences in the distributions and temporal patterns of the map-generated values can therefore affect both perturbation activation and the candidate positions produced during hiding. These mechanisms provide a rationale for using chaotic maps to diversify search trajectories and potentially reduce stagnation while retaining ARO's original search structure. Their practical effects are examined through comparisons between ARO and the ten CARO variants.

Table 2. Ten chaotic maps used in CARO

#map	Name	Definition	Range
1	Chebyshev	$P_{q+1} = cos(q\cos^{-1}(P_q))$	(-1,1)
2	Circle	$P_{q+1} = mod\left(P_q + r - \left(\frac{l}{2\pi}\right)\sin 2\pi P_q\right), 1), l = 0.5\ and\ r = 0.2$	(0,1)
3	Gauss/mouse	$P_{q+1} = f(x) = \begin{cases} 1 & , \ p_q = 0 \\ \frac{1}{mod(p_q, 1)}, & otherwise \end{cases}$	(0,1)
4	Iterative	$P_{q+1} = \sin\left(\frac{l\pi}{P_q}\right), l = 0.7$	(-1,1)
5	Logistic	$P_{q+1} = lP_q(1 - P_q), l = 4$	(0,1)
6	Piecewise	$P_{q+1} = f(x) = \begin{cases} \frac{P_q}{l}, & 0 \le P_q < l \\ \frac{P_q - l}{0.5 - l}, & l \le P_q < 0.5 \\ \frac{1-l-P_q}{0.5-l}, & 0.5 \le P_q < 1 - l \\ \frac{1-P_q}{l}, & 1 - l \le P_q < 1 \end{cases}, l = 0.4$	(0,1)
7	Sine	$P_{q+1} = \frac{l}{4}\sin(\pi P_q), l = 4$	(0,1)
8	Singer	$P_{q+1} = \mu\left(7.86P_q - 23.31P_q^2 + 28.75P_q^3 - 13.302875P_q^4\right), \mu = 1.07$	(0,1)
9	Sinusoidal	$P_{q+1} = lP_q^2\sin(\pi P_q), l = 2.3$	(0,1)

Table 2. Ten chaotic maps used in CARO

#map	Name	Definition	Range
10	Tent	$P_{q+1} = \begin{cases} \frac{P_q}{0.7}, & P_q < 0.7 \\ \frac{10}{3}(1 - P_q), & P_q \geq 0.7 \end{cases}$	(0,1)

3. The proposed algorithm

Artificial rabbit optimization (ARO) can be utilized for feature selection using a wrapper-based approach. In this approach, the classification method is used as a guide to optimize the selection of features to create an optimal feature subset. The KNN algorithm is used as the classification method to evaluate the quality of selected feature subset. KNN works by matching the unknown example to the majority category of its K nearest neighbors.

For feature selection, an adaptive version of the ARO algorithm is implemented to find the feature combination that yields the highest classification accuracy. The algorithm works by iteratively adjusting rabbit positions throughout the search space, beginning with a prime search agent. The rabbits then update their positions towards the best search agent using equations (1) and (7). The algorithm represents each potential solution as a real-valued vector that matches the dataset's dimensionality, with all values constrained between 0 and 1. When evaluating fitness, these solutions are converted to binary form. The fitness function combines two objectives: maximizing classification accuracy while minimizing the number of selected features, as formulated in Equation (12).

$$f\left(\overrightarrow{x}_i(t)\right) = \alpha E + (1 - \alpha)\left(\frac{|\overrightarrow{x}_i(t)|}{|c|}\right). \tag{12}$$

The term *E* denotes the classification error produced by the effective classifier, specifically the *K*-NN classifier. The second term accounts for the count of selected features. This formulation aims to achieve a balance between minimizing classification error and the number of features chosen, a parameter $\alpha \in$ [0, 1] is used. The factor α is used to control the importance of classification accuracy and number of selected features. As for any classifier, improving the accuracy is the first objective, the weight factor is usually set to value near one [30]. Thus, we set α for all the following experiments to 0.99. In this study, an enhanced feature selection method utilizing Chaotic Artificial Rabbit Optimization (CARO) has been implemented to enhance the accuracy of the ARO algorithm. The random numbers employed in equations (1) and (7) have been substituted with chaotic maps, which are reformulated as follows:

$$\overrightarrow{v}_i(t+1) = \overrightarrow{x}_j(t) + R\left(\overrightarrow{x}_i(t) - \overrightarrow{x}_j(t)\right) + \text{round}\left(0.5\left(0.05 + \rho_i\right)\right) n_1. \tag{13}$$

$$\overrightarrow{v}_i(t+1) = \overrightarrow{x}_j(t) + R\left(\rho_i \overrightarrow{b}_{i,r}(t) - \overrightarrow{x}_i(t)\right), \quad i = 1, \ldots, n. \tag{14}$$

Where ρ_i represents the value derived from well-established chaotic maps at the *i*-th iteration, as detailed in Table 2.

At the evaluation stages, a decision is made on whether to select a feature or not based on its associated variable, which is assessed using a static threshold of 0.5.

$$X_{i+1} = \begin{cases} x_{i,j} > 0.5, & \text{feature selected,} \\ x_{i,j} \leq 0.5, & \text{feature is not selected.} \end{cases} \tag{15}$$

Where $x_{i,j}$ is the continuous position of search agent *i* in dimension *j*. Consequently, the position of the rabbit signifies a selected feature set, with the position values increasing for feature dimensions that are candidates for

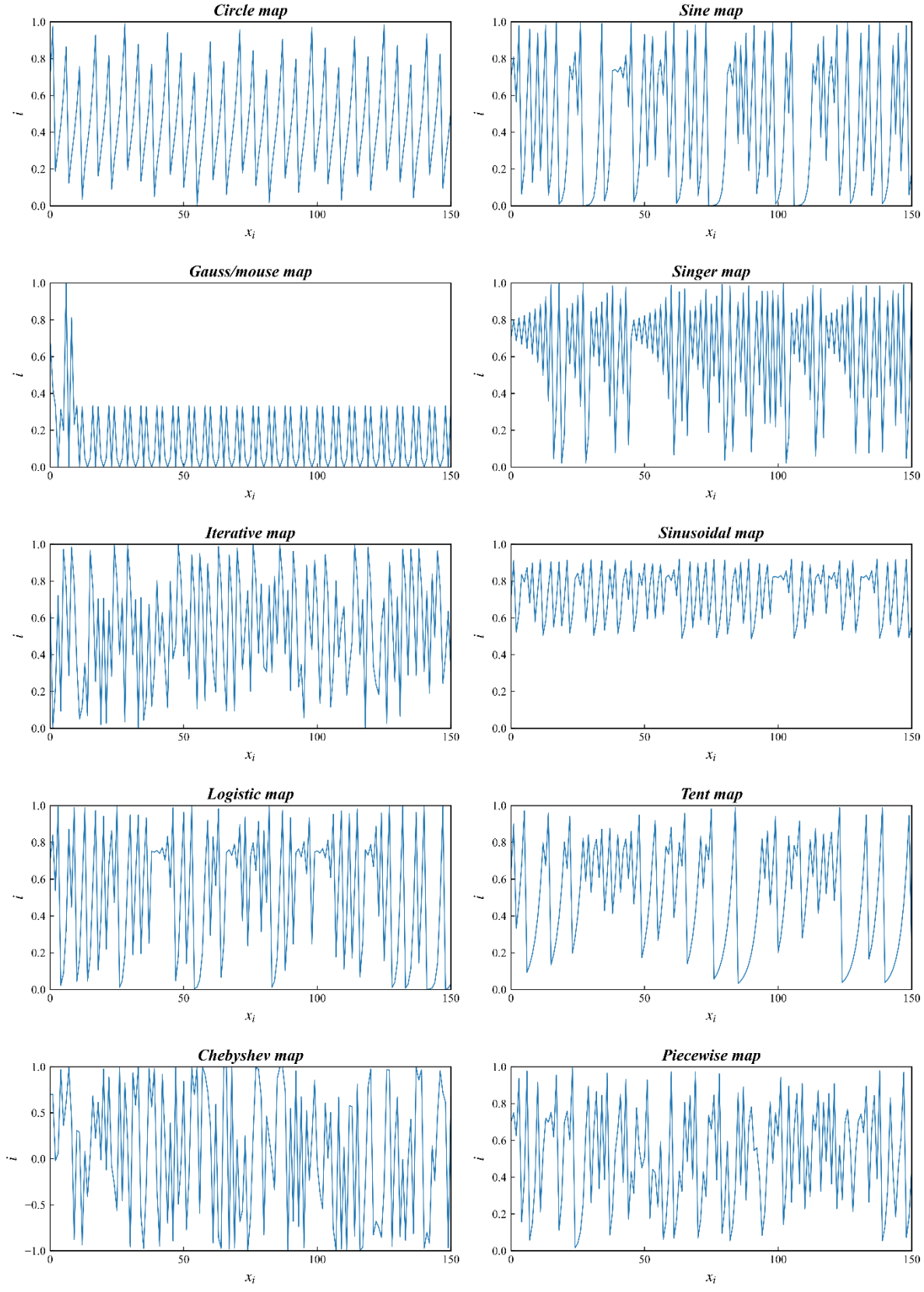


Figure 1. The ten chaotic maps

selection in the classification process. The details of this algorithm are presented in Algorithm 2 and illustrated in Figure 2.

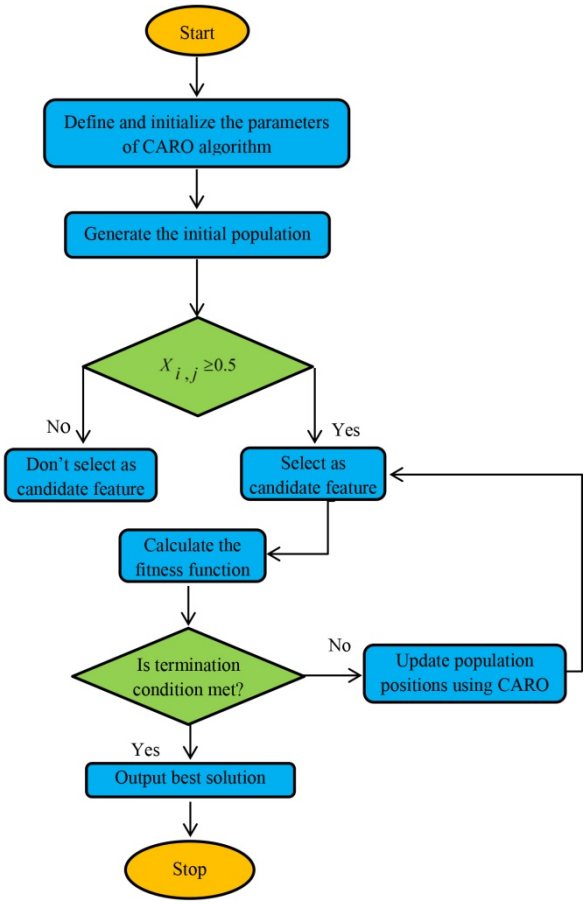


Figure 2. Flowchart of the proposed CARO algorithm

It is worth mentioning that the complexity of ARO as many optimization algorithms is $O(dM + MC)$, where d is the dimension of the problem, M is the number of solutions/particles/agents, and C represents the complexity of the objective function. The complexity of CARO is approximately identical to the ARO algorithm. This is because the complexity of the additional step that generates chaotic numbers in the CARO algorithm is $O(Max_{iter})$, where Max_{iter} is the maximum number of iterations. This phase is only used at the start of the CARO algorithm to generate all of the chaotic numbers needed for the iterations. However, the complexity of the step that generates random numbers in the ARO algorithm is also $O(Max_{iter})$ which is same in the CARO algorithm. As a consequence, both ARO and CARO algorithms approximately have the same time complexity.

Boundary handling is performed by uniformly reinitializing each component that leaves [0,1], rather than by clipping it to the nearest bound. An empty binary mask is assigned fitness 1. The original ARO phase switch is retained: A $>$ 1 activates detour foraging, and $A \leq 1$ activates random hiding. These details are now included because they affect reproducibility and the interpretation of CARO.

The maps have different native ranges. The reported implementation used the generator-specific outputs, and a common range transformation was not applied identically to every map. Accordingly, the comparison should be interpreted as a comparison of the implemented map variants. A future controlled ablation should normalize every sequence to [0,1] before substituting coefficients that are defined on that interval.

Algorithm 2: Steps of the proposed CARO for feature selection
Input
X: the dataset, where each row represents a sample and each column represents a feature
Y: the corresponding labels
Begin
Initialize the algorithm parameters:
Initialize number of rabbits (search agents): n.
Set maximum number of iterations MaxIter
For *t =1:max_iter and for each rabit*

Compute the ARO running operator and energy factor
if A>1, generate a candidate by detour foraging using Eq. (13).

Otherwise, generate ir by random hiding using Eq. (14).

Apply a random disturbance to each rabbit's position to promote exploration:
$X_i(t+1) = X_i(t+1) + rand * (U - L)$

Where U and L is the upper and lower bound for search space respectively

Evaluate the fitness of each rabbit's new position using the fitness function.

Update the current best solution (S) if a better solution is found.

Calculate the classification metrics (accuracy, F1-measure) for the best solution so far.
End for
Return *the optimal solution obtained across all iterations.*
Compute fitness measures (max fitness, min fitness, mean fitness, standard deviation (corresponding to the optimal solution from each iteration
End

4. Experimental results

4.1. Dataset and parameters setting

The study employed ten datasets sourced from the renowned UCI repository [31], which are detailed in Table 3. These datasets were strategically selected to encompass diverse complexity patterns and varying instance counts, enabling comprehensive evaluation of the proposed method's performance across multiple computational scenarios. The datasets range from 13 to 60 features, representing small to medium-dimensional feature spaces. The experimental protocol follows a standard hold-out validation approach. Each dataset was partitioned into a training subset (80%) and a holdout validation subset (20%). The training subset was used to train the KNN classifier, while classification error on the holdout validation subset guided the feature-selection process.

Table 3. Description of the experimental datasets

ID	Dataset	# dims	# instances
D1	base_BreastEW	30	569
D2	base_Exactly	13	1000
D3	base_M-of-n3	13	1000
D4	breastEW	30	569
D5	CongressEW	16	435
D6	Ionosphere	34	351
D7	KrvskpEW	36	3196

Table 3. Description of the experimental datasets

ID	Dataset	# dims	# instances
D8	SonarEW	60	208
D9	SpectEW	22	267
D10	WaveformEW	40	5000

The K-nearest neighbor (KNN) classifier was selected for its widespread utility and computational simplicity in machine learning applications. To streamline the analytical process, a fixed neighborhood size of K=5 was predetermined for the classification task [32]. Classification error on the holdout partition guided feature selection, and the reported classification results therefore represent validation performance under this experimental protocol. The experimental design targeted 25 runs per algorithm–dataset configuration.

4.2. Performance metrics

The evaluation covered classification performance, selected-feature counts, and feature-reduction percentages. Summary statistics were calculated from the available original runs for each algorithm–dataset configuration. The performance evaluation of the proposed method incorporates two established metric categories: fitness metrics and classification metrics, comprehensively detailed in Table 4, to provide a multifaceted assessment of the algorithm's effectiveness.

Table 4. Fitness and classification metrics

Type	**Formula**
Max fitness	$\max_{i=1,\ldots,\mathrm{Run}} BS_i$
Min fitness	$\min_{i=1,\ldots,\mathrm{Run}} BS_i$
Mean fitness	$\frac{1}{\mathrm{Run}} \sum_{i=1}^{\mathrm{Run}} BS_i$
Standard deviation (SD)	$\sqrt{\frac{\sum_{i=1}^{\mathrm{Run}} (BS_i - \mu)^2}{\mathrm{Run}}}$
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
F-score	$\frac{2\,(\mathrm{Precision} \cdot \mathrm{Recall})}{\mathrm{Precision} + \mathrm{Recall}}$

Where *BS* is the best score gained in each iteration, *L* is the number of features. *TP* is the true positive, *FP* is the false positive, *FN* is the false negative and *TN* represents the true negative.

4.3. Analysis and discussion

To assess the performance of the proposed ARO and CARO algorithms, 10 distinct datasets from the UCI repository are utilized. The results produced by ARO are benchmarked against established meta-heuristic algorithms namely, PSO, ACO, WOA, and then ARO itself compared with the ten chaotic ARO maps "CARO". To evaluate the convergence capability, each algorithm was executed 25 times. The average results from these 25 runs were computed and compared across algorithms. The parameter settings for each algorithm are detailed in Table 5. The experimental evaluation is structured into two parts. The first part involves benchmarking the 10 CARO variants to identify the most effective chaotic map, as presented in Tables 6–7. The second part focuses on comparing the best-performing chaotic map against the original ARO and several established meta-heuristic algorithms ACO, PSO and WOA as shown in Tables 8-9.

Table 5. Parameter settings

Parameter	Value
k-value of KNN	5
number of populations	20
number of iterations	100
problem dimensions	Number of features in each data
Data search domain	[0 1]
Repetition of runs	25
Pheromone in PSO	0.2
B constant in WOA	1
Initial value of Chaotic	0.6
Iteration number in chaotic	500

Table 6 shows the accuracy and F-measure of the average runs for the ten CARO where the number after CARO refers to the map number in Table 1, for example CARO2 is Circle map. Results in Table 6 show that the Circle map which is CARO2 has higher results in five datasets, CARO10 has best results in two datasets, CARO4, CARO5 and CARO7 have best result in one dataset for each regarding accuracy. So, CARO2 has been chosen to be compared with ARO, ACO, PSO and WOA algorithms.

Table 7 shows the accuracy and F-measure of the average runs for CARO2, ARO, PSO, ACO and WOA, which clearly show the superiority of CARO2 in 5 datasets for accuracy and 4 datasets in F-measure, ARO is better in 3 datasets for accuracy and 4 in F-measure which means the developing in results after using chaotic maps. Table 8 evaluates the Min, Max and Mean fitness value, also it's indicated the standard deviation. CARO2 gives best results in Max and Mean than other algorithms. A clarification on the fitness-metric convention used in this paper is essential for correctly interpreting Table 8. The fitness function f(x) in Equation (12) is a Minimization objective; lower values indicate better solutions (lower classification error and fewer selected features). Therefore, in Table 8: "Max fitness" refers to the worst fitness value observed across all runs (the highest, i.e., least good, value of f seen), "Min fitness" refers to the best fitness value (the lowest, i.e., most optimized, value of f found), and "Mean fitness" is the average of the best fitness values across the 25 runs. The term "Max" in Table 8 column header refers to the maximum of the best solution obtained per run (worst-case best), while "Min" refers to the minimum of the best solution per run (best-case best).

Figure 3 shows the convergence curve for ARO and the ten CARO. In this figure, 100 iterations were conducted. The convergence curves show three behavioral stages: an aggressive early descent (iterations 1–20) driven by the ergodic diversity of the chaotic maps, a sustained improvement phase (iterations 20–60) where CARO avoids the stagnation plateaus visible in ARO "colored with pink", and a final convergence phase (iterations 60–100) where CARO variants settle at lower fitness values. The exception on D9 (SpectEW, 22 features) is expected, as the small search space leaves no room for chaotic diversity to provide additional benefit.

Table 9 reports the selected-feature counts of ARO and all ten CARO variants as mean ± sample Standard Deviation (SD), The results highlight the potential of incorporating chaotic maps into ARO to obtain more compact feature subsets. On Ionosphere, CARO4 selects an average of 7.16 ± 2.90 features, compared with 9.76 ± 4.82 for ARO. Similarly, on SonarEW, CARO4 selects 26.04 ± 5.26 features, compared with 27.36 ± 3.89 for ARO. These examples provide descriptive evidence that incorporating a chaotic map can improve feature reduction relative to the original algorithm. The observed gains support the potential usefulness of chaotic maps for feature selection, with their effectiveness depending on the chosen map and dataset.

Table 10 presents the corresponding mean feature-reduction percentages. The ten displayed entries represent ten unique datasets. For the cross-dataset summary, we use the ten datasets with 25 available runs for every method. The unweighted mean reduction is highest for CARO4 (Iterative, 54.26%), followed by CARO7 (Sine, 53.90%) and ARO (53.56%). CARO2 (Circle) obtains 53.48%. These results highlight the potential benefit of incorporating chaotic maps into ARO for obtaining more compact feature subsets. Across the ten datasets with complete results for all methods, CARO4 and CARO7 achieved mean feature reductions of 54.26% and 53.90%, respectively,

compared with 53.56% for the original ARO. These observed improvements suggest that specific chaotic maps can enhance feature reduction, with the extent of the benefit depending on the map and dataset.

Table 6. Comparison of ten CARO variants (accuracy and F-measure)

	Measure	CARO1	CARO2	CARO3	CARO4	CARO5	CARO6	CARO7	CARO8	CARO9	CARO10
D1	Accuracy	98.7257	**99.0088**	51.3274	98.7965	98.9027	96.2832	98.4425	98.3009	98.2301	98.4779
	F-Measure	98.9921	**99.2187**	38.2022	99.0489	99.1320	97.0518	98.7703	98.6627	98.6017	98.8022
D2	Accuracy	100	100	72.74	100	100	73.5	100	100	100	100
	F-Measure	100	100	48.588	100	100	51.376	100	100	100	100
D3	Accuracy	100	100	86.70	100	100	90.5	100	100	100	100
	F-Measure	100	100	89.57	100	100	92.72	100	100	100	100
D4	Accuracy	98.7257	**99.0088**	51.3274	98.7965	98.9027	51.3274	**99.0088**	98.7611	98.7611	98.4425
	F-Measure	98.9921	99.2187	38.2022	99.0489	99.1320	38.2022	**99.2206**	99.0240	99.0205	98.7722
D5	Accuracy	98.6667	98.6207	96.3218	98.6207	98.5747	96.3218	98.5747	98.6207	98.5747	**98.7126**
	F-Measure	98.9058	98.8755	96.9961	98.8705	98.8353	96.9961	98.8311	98.8768	98.8360	**98.9510**
D6	Accuracy	94.6286	**95.7143**	91.4285	**95.7143**	95.6571	91.4285	95.7143	94.6286	94.4000	94.1143
	F-Measure	95.9313	**96.7547**	93.6170	96.7472	96.7132	93.4782	96.7489	95.9417	95.7723	95.5757
D7	Accuracy	98.5164	**98.6040**	93.7277	98.5790	98.3849	95.7746	98.4350	98.4663	98.4726	98.4100
	F-Measure	98.5758	**98.6590**	94.0118	98.6358	98.4474	95.9152	98.5027	98.5297	98.5327	98.4736
D8	Accuracy	97.4634	97.2682	87.4146	97.2682	**97.6585**	90.2439	97.0731	96.8780	97.3658	97.3658
	F-Measure	97.2212	96.9975	86.0019	96.9874	**97.4471**	89.4736	96.7929	96.6220	97.1295	97.1019
D9	Accuracy	92.1509	92	87.0188	91.7735	92.0754	86.7924	91.9245	92	91.9245	**92.2264**
	F-Measure	79.9046	79.5310	67.6078	79.4853	80.1942	72	78.0422	**80.3032**	79.5944	79.9870
D10	Accuracy	82.24	**82.744**	76.548	82.536	82.02	77.1	82.24	82.368	82.132	82.284
	F-Measure	88.6782	**89.1509**	84.2646	88.9525	88.5980	84.3585	88.7239	88.8257	88.8852	88.8916

Table 7. Accuracy and F-measure for CARO2 compared with meta-heuristic algorithms

Dataset	**Measure**	**CARO2**	**ACO**	**PSO**	**WOA**	**ARO**
D1	Accuracy	**99.008**	98.23	98.69	98.23	98.831
	F-Measure	99.218	97.605	98.223	98.23	**99.937**
D2	Accuracy	**100**	97.96	91.26	83.729	99.8
	F-Measure	**100**	98.549	93.904	88.903	98.454
D3	Accuracy	**100**	99.52	97.48	96.916	**100**
	F-Measure	**100**	99.345	96.526	95.761	**100**
D4	Accuracy	**99.008**	98.557	98.371	97.676	98.831
	F-Measure	**99.218**	98.046	97.788	96.841	98.654

Table 7. Accuracy and F-measure for CARO2 compared with meta-heuristic algorithms

Dataset	Measure	CARO2	ACO	PSO	WOA	ARO
D5	Accuracy	**98.620**	97.112	97.662	97.318	98.39
	F-Measure	**98.875**	96.808	97.008	96.556	98.30
D6	Accuracy	95.714	95.542	93.085	93.542	**96**
	F-Measure	96.754	96.602	94.853	95.178	**96.985**
D7	Accuracy	98.604	97.489	97.446	97.218	**98.741**
	F-Measure	**98.659**	97.363	97.341	97.077	98.383
D8	Accuracy	97.268	93.756	96.195	92.479	**98.341**
	F-Measure	96.997	94.196	96.451	93.032	**97.824**
D9	Accuracy	**92**	90.037	91.169	88.83	91.32
	F-Measure	79.531	93.805	94.520	93.042	**94.553**
D10	Accuracy	82.744	**85.623**	86.376	85.561	82.276
	F-Measure	89.150	89.272	89.821	89.225	**89.526**

Table 8. Comparison between the best CARO and other algorithms in fitness metrics

Dataset	Algorithm	SD	Min	Max	Mean
D1	CARO2	**0.005147**	0.031858	**0.009912**	**0.012995**
	ACO	0.0708	0.0204	0.2478	0.0811
	PSO	0.0289	0.0874	0.219	0.0994
	WOA	0.0214	0.1627	0.2435	0.1737
	ARO	0.0553	**0.002**	0.2528	0.0214
D2	CARO2	0.020775	0.1136	**0**	**0.00552**
	ACO	0.0034	0.0176	0.0331	0.0202
	PSO	0.0038	**0.0116**	0.0290	0.0141
	WOA	**0.0014**	0.0176	0.0250	0.0180
	ARO	0.0054	**0.0116**	0.0336	0.0149
D3	CARO2	0.020775	0.1136	**0**	0.00552
	ACO	0.0218	0.0048	0.0952	0.0200
	PSO	**0.0105**	0.0252	0.0936	0.0279
	WOA	0.0116	0.0308	0.0775	0.0365
	ARO	0.0117	**0**	0.0788	**0.0025**
D4	CARO2	0.005147	0.031858	**0.009912**	**0.012995**
	ACO	0.0034	0.0144	0.0301	0.0167
	PSO	0.0028	0.0162	0.0297	0.0177
	WOA	**0.0024**	0.0232	0.0328	0.0243
	ARO	0.0048	**0.0116**	0.0336	0.0151
D5	CARO2	0.003936	0.033103	**0.013793**	**0.015724**
	ACO	0.0033	0.0251	0.0394	0.0273
	PSO	0.0031	0.0255	0.0446	0.0268
	WOA	**0.0017**	0.0278	0.0368	0.0286
	ARO	0.0095	**0.0125**	0.0539	0.0187
D6	CARO2	0.0174	0.1091	**0.0428**	0.0547
	ACO	0.0154	0.0624	0.1317	0.0735
	PSO	0.0146	0.0351	0.1034	0.0456

Table 8. Comparison between the best CARO and other algorithms in fitness metrics

Dataset	Algorithm	SD	Min	Max	Mean
D7	WOA	**0.0054**	0.0752	0.0975	0.0778
	ARO	0.0206	**0.0165**	0.0985	**0.0309**
	CARO2	0.0090	**0.0590**	**0.0139**	**0.0195**
	ACO	0.0089	0.0988	0.1381	0.1076
	PSO	0.0056	0.0852	0.1101	0.0890
	WOA	**0.0032**	0.1116	0.1245	0.1132
	ARO	0.0103	0.0867	0.1283	0.0938
	CARO2	0.0223	**0.1160**	**0.0273**	**0.0442**
D8	ACO	**0.0028**	0.1437	0.1599	0.1452
	PSO	0.0045	0.1361	0.1585	0.1392
	WOA	0.0031	0.1443	0.1594	0.1457
	ARO	0.0130	0.1772	0.2337	0.1870
	CARO2	0.0104	0.1237	0.08	0.0874
D9	ACO	0.0042	0.0226	0.0421	0.0997
	PSO	0.0042	0.0226	0.0421	0.0997
	WOA	**0.0024**	0.0268	0.0373	0.0278
	ARO	0.0042	**0.0160**	**0.0372**	**0.0179**
	CARO2	0.0133	0.2271	0.1725	0.1823
D10	ACO	0.0134	0.0445	0.0971	0.0550
	PSO	0.0071	0.0657	0.0977	0.0697
	WOA	**0.0059**	0.0645	**0.0914**	0.0670
	ARO	0.0175	**0.04**	0.1034	**0.0532**

Table 9. Selected-feature counts (mean ± sample SD) for ARO and CARO

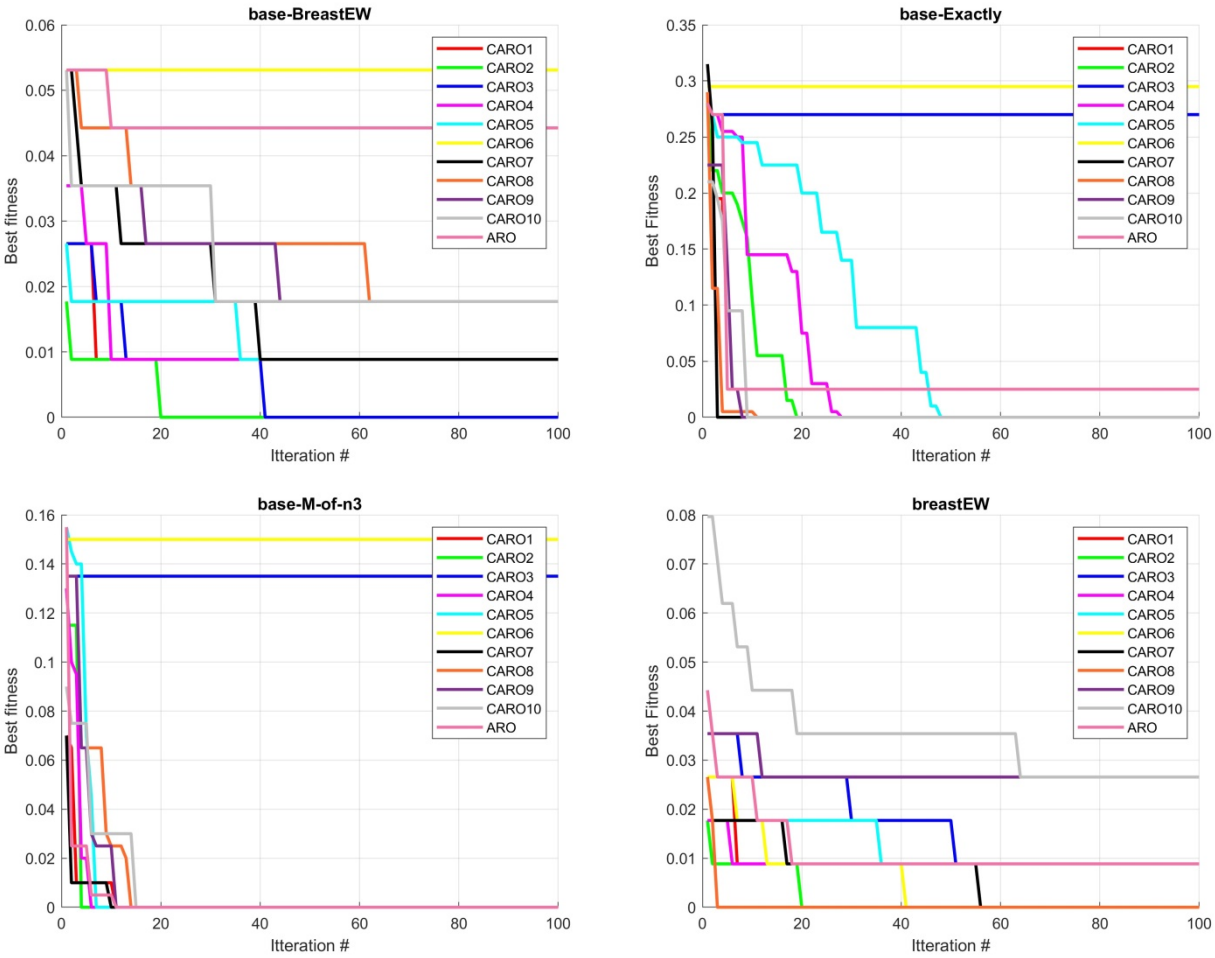
Method	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
ARO	14.80 ± 2.84	6.28 ± 0.46	6.78 ± 0.43	15.00 ± 2.96	6.76 ± 2.03	9.76 ± 4.82	20.48 ± 2.54	27.36 ± 3.89	10.24 ± 2.26	22.72 ± 3.09
CARO1	16.40 ± 2.58	6.36 ± 0.49	6.76 ± 0.52	16.40 ± 2.58	7.44 ± 1.85	12.48 ± 3.29	20.84 ± 3.36	28.28 ± 3.16	11.04 ± 2.05	22.84 ± 2.73
CARO2	14.96 ± 3.13	6.52 ± 0.51	6.84 ± 0.47	14.96 ± 3.13	7.16 ± 2.21	7.52 ± 4.32	20.76 ± 3.48	28.20 ± 5.55	10.64 ± 2.53	22.24 ± 2.82
CARO3	16.00 ± 2.98	8.12 ± 2.39	8.48 ± 1.98	16.00 ± 2.98	8.04 ± 2.05	15.28 ± 3.01	20.00 ± 3.21	30.56 ± 4.00	11.12 ± 2.39	21.64 ± 2.50
CARO4	16.48 ± 2.02	6.40 ± 0.50	6.92 ± 0.40	16.48 ± 2.02	7.36 ± 1.82	7.16 ± 2.90	20.32 ± 3.28	26.04 ± 5.26	10.88 ± 2.80	21.84 ± 3.46
CARO5	16.00 ± 3.08	6.40 ± 0.50	6.84 ± 0.47	16.00 ± 3.08	7.12 ± 2.11	8.12 ± 4.20	20.52 ± 3.68	27.68 ± 6.07	11.00 ± 3.39	22.84 ± 3.95
CARO6	15.28 ± 2.62	8.12 ± 2.39	8.48 ± 1.98	15.28 ± 2.62	8.04 ± 2.05	15.28 ± 3.01	20.00 ± 3.21	30.56 ± 4.00	11.12 ± 2.39	21.64 ± 2.50
CARO7	16.24 ± 2.54	6.24 ± 0.44	6.80 ± 0.58	14.80 ± 2.24	7.32 ± 2.30	8.08 ± 3.84	20.60 ± 2.29	27.20 ± 4.68	10.28 ± 2.82	22.36 ± 3.08
CARO8	15.04 ± 3.02	6.24 ± 0.44	6.88 ± 0.53	17.72 ± 4.18	7.44 ± 2.24	10.40 ± 4.27	21.64 ± 2.91	28.32 ± 4.45	10.48 ± 2.68	22.92 ± 2.87
CARO9	15.88 ± 2.09	6.52 ± 0.51	6.84 ± 0.47	20.24 ± 3.73	7.92 ± 1.82	12.60 ± 3.34	21.12 ± 3.49	27.68 ± 3.80	9.88 ± 2.09	22.92 ± 2.06
CARO10	15.24 ± 2.63	6.40 ± 0.50	6.76 ± 0.44	16.52 ± 2.47	7.84 ± 2.46	12.88 ± 2.88	20.08 ± 2.63	29.40 ± 3.89	10.04 ± 1.99	23.04 ± 2.47

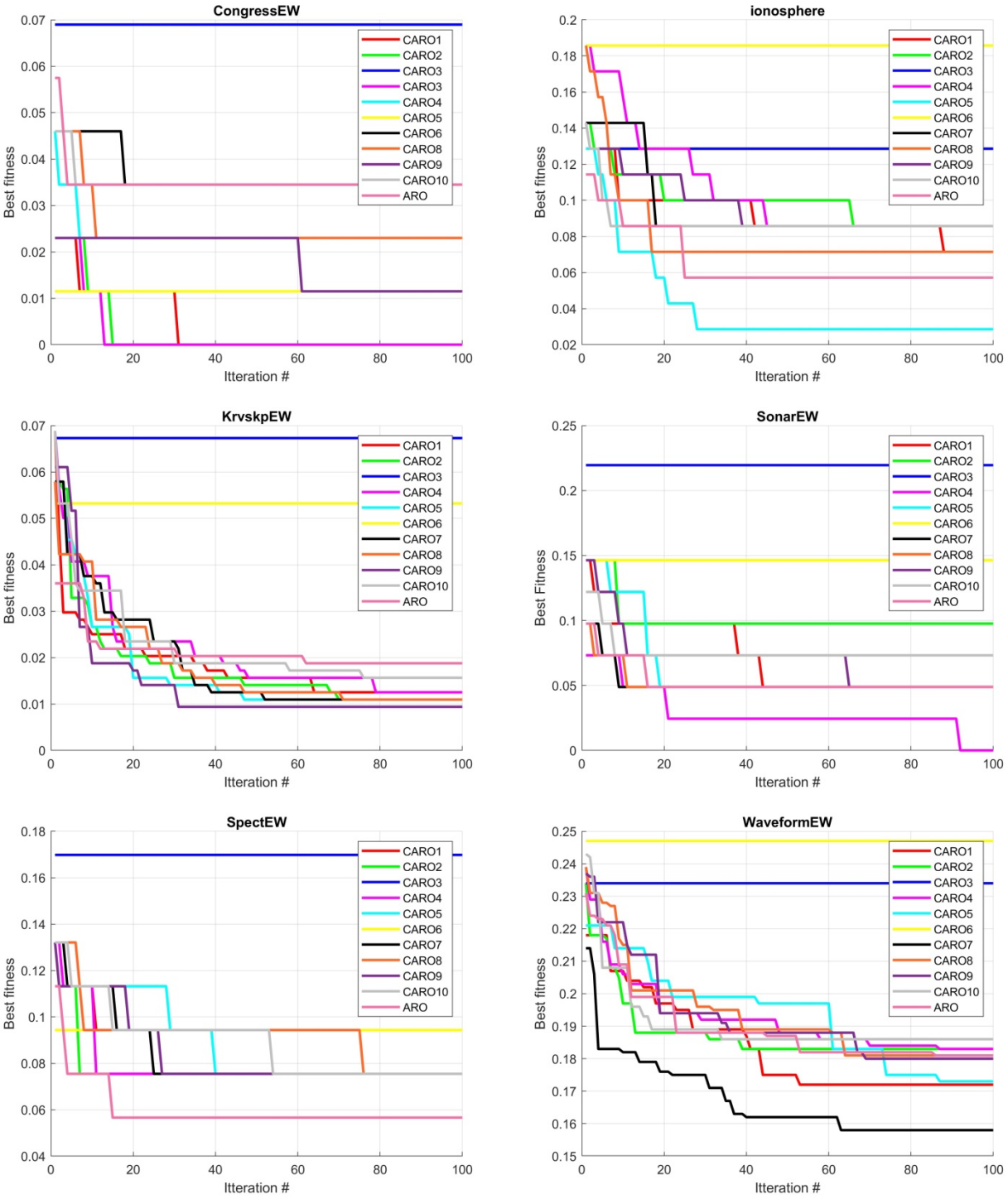
Table 10. Mean feature reduction (%) for ARO and ten CARO variants

Method	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
ARO	50.67	51.69	47.86[a]	50.00	57.75	71.29	43.11	54.40	53.45	43.20
CARO1	45.33	51.08	48.00	45.33	53.50	63.29	42.11	52.87	49.82	42.90
CARO2	50.13	49.85	47.38	50.13	55.25	77.88	42.33	53.00	51.64	44.40
CARO3	46.67	37.54	34.77	46.67	49.75	55.06	44.44	49.07	49.45	45.90
CARO4	45.07	50.77	46.77	45.07	54.00	78.94	43.56	56.60	50.55	45.40

Method	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
CARO5	46.67	50.77	47.38	46.67	55.50	76.12	43.00	53.87	50.00	42.90
CARO6	49.07	37.54	34.77	49.07	49.75	55.06	44.44	49.07	49.45	45.90
CARO7	45.87	52.00	47.69	50.67	54.25	76.24	42.78	54.67	53.27	44.10
CARO8	49.87	52.00	47.08	40.93	53.50	69.41	39.89	52.80	52.36	42.70
CARO9	47.07	49.85	47.38	32.53	50.50	62.94	41.33	53.87	55.09	42.70
CARO10	49.20	50.77	48.00	44.93	51.00	62.12	44.22	51.00	54.36	42.40

Figure 3. Comparison between ten CARO variants and ARO




CongressEW
ionosphere
KrvskpEW
SonarEW
SpectEW
WaveformEW
Best fitness
Best Fitness
Itteration #
CARO1
CARO2
CARO3
CARO3
CARO4
CARO5
CARO6
CARO7
CARO8
CARO9
CARO10
ARO

The Friedman test [33] has been used to test the ranks of our method and other algorithms. It is a non-parametric test and multiple comparisons test that can find significant differences among the behaviors of multiple optimizers. The ranks of the Friedman test are conducted on all 10 UCI datasets for all the used algorithms. First the ranks among 10 CARO algorithms are compared in terms of accuracy, F-measure, standard deviation, best fitness, worst fitness and the mean fitness as shown in Table 11, which also shows that CARO2 performing the best. Then in the second stage, CARO2 has been compared with ARO, PSO, ACO and WOA algorithms, Table 12 shows that CARO2 is the highest performing optimizer of the comparison for the Friedman, which gives more evidence that our proposed method gives more efficient results than other algorithms.

Table 11. Comparison between CARO variants using the Friedman test

	Accuracy	**F-Measure**	**SD**	**Max**	**Min**	**Mean**
CARO1	7	6.6000	2.9500	3.0500	4.6500	3.8500
CARO2	8.0500	8	9.9500	9.7000	2.9000	6.4000
CARO3	1.2500	1.2000	7.6000	5.9000	2.3000	1.9000
CARO4	7.0500	6.9000	4	3.4500	4.7500	4.1500
CARO5	6.5000	6.6000	1	1.7000	3.7000	2.7000
CARO6	1.7500	1.8000	6.5500	5	7.1000	7.1000
CARO7	5.9500	5.7000	3.6000	3.6000	5.1000	4.4000
CARO8	5.8500	6.2000	8.1500	6.4000	6.6000	6.6000
CARO9	5.2500	5.8000	5.2500	7.5000	8.4000	8.4000
CARO10	6.3500	6.2000	5.9500	8.7000	9.5000	9.5000
P-value	7.8541e-09	2.44E-08	7.7640e-12	5.5325e-11	1.3122e-08	9.4874e-10

Table 12. Comparison between CARO2 and other algorithms using the Friedman test

	Accuracy	F-Measure	SD	Min	Max	Mean
CARO2	4.3500	3.8500	4	4	1.8000	1.9000
ACO	2.5500	2.7000	3.2000	2.6000	3.8500	3.5500
PSO	2.6000	2.6000	2.3000	2.7000	2.9500	3.1500
WOA	1.45000	1.5000	1.3000	4.0500	2.4000	4.3000
ARO	4.0500	4.3500	4.2000	1.6500	4	2.1000
P-value	1.2722e-04	4.3940e-04	8.2941e-05	0.2000	0.0068	0.0027

The mean of ranks of the Friedman test is shown in Figure 4 for 10 UCI datasets using 10 CARO algorithms based on accuracy. This figure summarizes the average accuracy ranks of the ten CARO variants across the evaluated datasets. Under the ranking convention used in this figure, larger values indicate better performance. CARO2 obtains the highest average rank, followed by CARO4 and CARO1, while CARO3 and CARO6 occupy the lowest positions. Besides, Figure 5 compares the average accuracy ranks of CARO2, ARO, ACO, PSO, and WOA, with larger ranks indicating better performance. CARO2 achieves the highest average rank, followed closely by ARO, while PSO and ACO obtain similar intermediate ranks and WOA receives the lowest rank.

4.4. Performance evaluation on CEC'22 benchmark functions

This section presents an evaluation of CARO2's effectiveness in addressing optimization tasks. Specifically, the algorithm's numerical performance is assessed through its application to a suite of twelve benchmark functions provided by CEC'22.

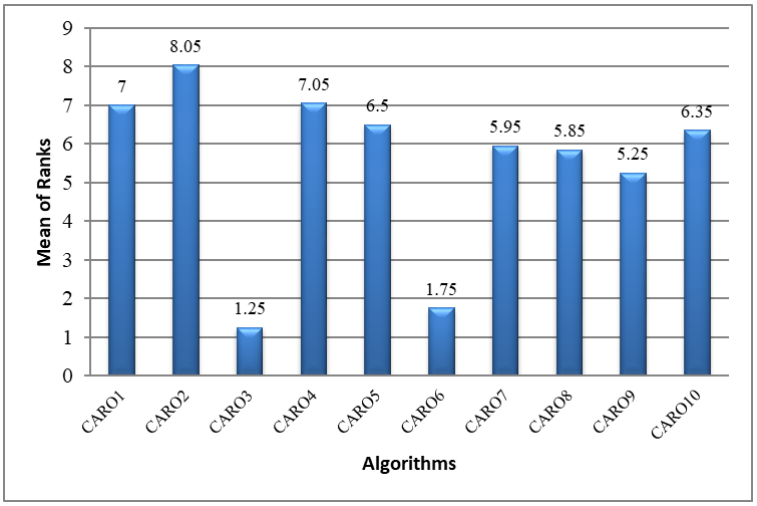


Figure 4. Mean ranks obtained by Friedman test for 10 CARO algorithms based on accuracy

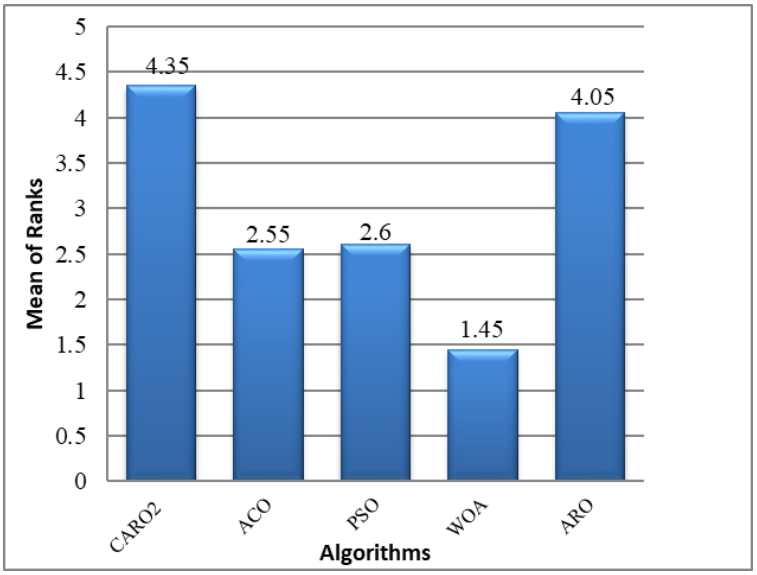


Figure 5. Mean ranks obtained by Friedman test for best CARO and other algorithms based on accuracy

To validate the robustness of CARO2, its performance was rigorously compared against six contemporary optimization algorithms: Artificial Hummingbird Algorithm (AHA) [34], African Vultures Optimization Algorithm (AVOA) [35], Crow Search Algorithm (CSA) [36], Harris Hawks Optimization (HHO) [37], Northern Goshawk Optimization (NGO) [38], and Satin Bowerbird Optimizer (SBO) [39], as documented in Table 13.

Furthermore, to substantiate CARO2's capacity for solving complex optimization challenges, additional comparative analyses were conducted with two recently refined algorithmic approaches: an adaptive quadratic interpolation and rounding mechanism Sine Cosine Algorithm (ARSCA) [40] and a trigonometrically enhanced Archimedes Optimization Algorithm (SCAOA) [41]. Experimental outcomes demonstrated that the proposed method consistently outperformed these comparative algorithms, underscoring its computational efficacy and algorithmic sophistication.

Table 13. Algorithm performance on CEC'2022 benchmark (30 runs; best results in bold)

Fun		AHA	AVOA	CSA	HHO	NGO	SBO	ARSCA	SCAOA	CARO2
F1	Mean	0.300E+03	0.300E+03	0.604E+04	0.300E+03	0.300E+03	0.300E+03	0.300E+03	0.323E+03	**-0.217E-22**
	STD	0.134E-10	0.724E-13	0.268E+04	0.136E-00	0.559E-13	0.213E-00	0.196E-01	0.179E+02	**0.294E-21**
F2	Mean	0.407E+03	0.416E+03	0.610E+03	0.422E+03	0.404E+03	0.410E+03	0.405E+03	0.404E+03	**-0.994E-23**
	STD	0.176E+02	0.270E+02	0.944E+02	0.291E+02	0.130E+02	0.209E+02	0.131E+02	0.295E+01	**0.191E-22**
F3	Mean	0.600E+03	0.604E+03	0.628E+03	0.618E+03	0.600E+03	0.604E+03	0.604E+03	0.601E+03	**0.216E-17**
	STD	0.971E-02	0.385E+01	0.614E+01	0.119E+02	0.194E-00	0.705E+01	0.331E+01	0.189E-00	**0.320E-16**
F4	Mean	0.823E+03	0.826E+03	0.835E+03	0.829E+03	0.809E+03	0.827E+03	0.820E+03	0.811E+03	**0.194E-20**
	STD	0.764E+01	0.933E+01	0.963E+01	0.739E+01	0.286E+01	0.969E+01	0.629E+01	0.290E+01	**0.126E-18**
F5	Mean	0.922E+03	0.103E+04	0.111E+04	0.138E+04	0.900E+03	0.133E+04	0.910E+03	0.900E+03	**0.184E-00**
	STD	0.429E+02	0.114E+03	0.843E+02	0.207E+03	0.155E+01	0.272E+03	0.205E+02	**0.201E-00**	0.286E-00
F6	Mean	0.205E+04	0.342E+04	0.341E+06	0.299E+04	0.197E+04	0.251E+04	0.353E+04	0.296E+04	**0.120E-01**
	STD	0.479E+03	0.138E+04	0.107E+07	0.144E+04	0.224E+03	0.957E+03	0.194E+04	0.100E+04	**0.219E-00**
F7	Mean	0.201E+04	0.203E+04	0.205E+04	0.203E+04	0.201E+04	0.205E+04	0.202E+04	0.202E+04	**-0.863E-02**
	STD	0.942E+01	0.103E+02	0.148E+02	0.111E+02	0.667E+01	0.454E+02	0.810E+01	0.626E+01	**0.315E-03**
F8	Mean	0.222E+04	0.222E+04	0.223E+04	0.223E+04	0.222E+04	0.227E+04	0.222E+04	0.222E+04	**0.655E+02**
	STD	0.678E+01	0.653E+01	0.453E+01	0.955E+01	0.882E+01	0.877E+02	0.691E+01	0.802E+01	**0.419E-02**
F9	Mean	0.253E+04	0.253E+04	0.265E+04	0.255E+04	0.253E+04	0.253E+04	0.253E+04	0.253E+04	**-0.484E-09**
	STD	**0.164E-09**	0.911E+01	0.306E+02	0.508E+02	0.463E-12	0.268E+02	0.268E+02	0.708E+01	0.133E-08
F10	Mean	0.250E+04	0.250E+04	0.251E+04	0.261E+04	0.253E+04	0.269E+04	0.254E+04	0.251E+04	**-0.471E-16**
	STD	0.117E-00	0.134E-00	0.811E+01	0.746E+02	0.467E+02	0.197E+03	0.578E+02	0.367E+02	**0.161E-15**
F11	Mean	0.262E+04	0.264E+04	0.292E+04	0.280E+04	0.264E+04	0.274E+04	0.266E+04	0.261E+04	**0.221E-08**
	STD	0.807E+02	0.648E+02	0.871E+02	0.133E+03	0.775E+02	0.162E+03	0.131E+03	0.160E+02	**0.959E-08**
F12	Mean	0.287E+04	0.288E+04	0.289E+04	0.289E+04	0.286E+04	0.295E+04	0.287E+04	0.286E+04	**-1.000E-00**
	STD	0.497E+01	0.890E+01	0.148E+02	0.261E+02	0.165E+01	0.533E+02	0.598E+01	0.134E+01	**0.443E-03**

5. Conclusion and future work

Artificial Rabbit Optimization Algorithm (ARO) and its hybridization with chaos (CARO) were proposed for feature selection. The main contributions are the chaotic modification of ARO, the comparison of ten chaotic maps within the feature selection framework, and the assessment of their effects on classification performance and the number of selected features. To augment the proposed technique's performance, ten distinct chaotic maps were strategically integrated, implementing CARO as a wrapper-based feature selection methodology. The experimental results demonstrated CARO's superior performance compared to established meta-heuristic algorithms, including Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Whale Optimization Algorithm (WOA), and the original Artificial Rabbit Optimization (ARO). Comprehensive analyses and experimental findings conclusively validate CARO's unique capability to identify and select optimal feature subsets, consistently leading to superior classification accuracy across diverse computational scenarios.

Moreover, the integration of chaotic maps significantly enhanced the original Artificial Rabbit Optimization (ARO) algorithm, improving both classification and fitness metrics. To further validate the algorithm's robustness, the proposed method underwent rigorous testing against contemporary optimization benchmarks from the CEC'22, providing additional empirical evidence of its computational effectiveness.

In future work, we can extend this study to other applications related to various fields such as engineering problems, medical image classification, signal and image processing. In addition, it would be interesting to investigate the hybridization of ARO algorithm with other swarm meta-heuristic algorithms, Future work also should evaluate the complete selection procedure using nested cross-validation, extend the parameter analysis to population size and iteration budget, and investigate objectives that explicitly balance classification error and subset size.

Competing interest The authors declare no competing interests.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Funding Declaration Authors state no funding involved.

REFERENCES

1. Pudjihartono, N., Fadason, T., Kempa-Liehr, A.W. and O'Sullivan, J.M., 2022. A review of feature selection methods for machine learning-based disease risk prediction. *Frontiers in Bioinformatics*, *2*, p.927312.
2. Manikandan, G. and Abirami, S., 2021. Feature selection is important: state-of-the-art methods and application domains of feature selection on high-dimensional data. *Applications in Ubiquitous Computing*, pp.177–196.
3. Bommert, A., Sun, X., Bischl, B., Rahnenführer, J. and Lang, M., 2020. Benchmark for filter methods for feature selection in high-dimensional classification data. *Computational Statistics & Data Analysis*, *143*, p.106839.
4. Wah, Y.B., Ibrahim, N., Hamid, H.A., Abdul-Rahman, S. and Fong, S., 2018. Feature selection methods: Case of filter and wrapper approaches for maximising classification accuracy. *Pertanika Journal of Science & Technology*, *26*(1).
5. Maldonado, S. and Weber, R., 2009. A wrapper method for feature selection using support vector machines. *Information Sciences*, *179*(13), pp.2208–2217.
6. Abdollahzadeh, B. and Gharehchopogh, F.S., 2022. A multi-objective optimization algorithm for feature selection problems. *Engineering with Computers*, *38*(Suppl 3), pp.1845–1863.
7. El-Kenawy, E.S.M., Mirjalili, S., Alassery, F., Zhang, Y.D., Eid, M.M., El-Mashad, S.Y., Aloyaydi, B.A., Ibrahim, A. and Abdelhamid, A.A., 2022. Novel meta-heuristic algorithm for feature selection, unconstrained functions and engineering problems. *IEEE Access*, *10*, pp.40536–40555.
8. Sharma, M. and Kaur, P., 2021. A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem. *Archives of Computational Methods in Engineering*, *28*, pp.1103–1127.
9. Houssein, E.H., Mahdy, M.A., Shebl, D. and Mohamed, W.M., 2021. A survey of metaheuristic algorithms for solving optimization problems. In *Metaheuristics in machine learning: theory and applications*, Springer, 14, pp.515–543.
10. Wang, D., Tan, D. and Liu, L., 2018. Particle swarm optimization algorithm: an overview. *Soft Computing*, *22*, pp.387–408.
11. Reeves, C.R., 2010. Genetic algorithms. *Handbook of Metaheuristics*, pp.109–139.
12. Dorigo, M., Birattari, M. and Stutzle, T., 2006. Ant colony optimization. *IEEE Computational Intelligence Magazine*, *1*(4), pp.28–39.
13. Panigrahi, S., Swapnarekha, H. and Subudhi, S., 2022. GACO: A Genetic Algorithm with Ant Colony Optimization—Based Feature Selection for Breast Cancer Diagnosis. In *Nature-Inspired Optimization Methodologies in Biomedical and Healthcare*, Springer, 233, pp.269–293.
14. Gunavathi, C. and Premalatha, K., 2015. Cuckoo search optimisation for feature selection in cancer classification: a new approach. *International Journal of Data Mining and Bioinformatics*, *13*(3), pp.248–265.

15. Min, H. and Fangfang, W., 2010, December. Filter-wrapper hybrid method on feature selection. In *2010 Second WRI Global Congress on Intelligent Systems*, *IEEE*, 3, pp.98–101.
16. Hamraz, M., Ali, A., Mashwani, W.K., Aldahmani, S. and Khan, Z., 2023. Feature selection for high dimensional microarray gene expression data via weighted signal to noise ratio. *PLOS ONE*, *18*(4), e0284619.
17. Ibrahim, R.A., Ewees, A.A., Oliva, D., Abd Elaziz, M. and Lu, S., 2019. Improved salp swarm algorithm based on particle swarm optimization for feature selection. *Journal of Ambient Intelligence and Humanized Computing*, *10*, pp.3155–3169.
18. Abdelrazek, M., Abd Elaziz, M. and El-Baz, A.H., 2024. CDMO: Chaotic Dwarf Mongoose Optimization Algorithm for feature selection. *Scientific Reports*, *14*(1), p.701.
19. Hamdani, T.M., Won, J.M., Alimi, A.M. and Karray, F., 2007. Multi-objective feature selection with NSGA II. In *Adaptive and Natural Computing Algorithms: 8th International Conference, ICANNGA 2007, Warsaw, Poland, April 11–14, 2007, Proceedings, Part I 8*, pp.240–247.
20. Gharehchopogh, F.S., Farnad, B. and Alizadeh, A., 2021. A modified farmland fertility algorithm for solving constrained engineering problems. *Concurrency and Computation: Practice and Experience*, *33*(17), p.e6310.
21. Anka, F., Soleimanian Gharehchopogh, F., Mousavirad, S.J. and Tejani, G.G., 2026. Discrete puma optimizer to solve combinatorial optimization problems. *International Journal of Computational Intelligence Systems*, *19*(1), p.73.
22. Rishakan, K.F. and Gharehchopogh, F.S., 2026. Resultant vector strategy: A new strategy for improved performance of metaheuristic algorithm. *Knowledge-Based Systems*, p.115833.
23. Gharehchopogh, F.S. and Rishakan, K.F., 2026. Several Improved Models of the Mountain Gazelle Optimizer for Solving Optimization Problems. *Computer Modeling in Engineering & Sciences*, *146*(1).
24. Valilou, M., Valilou, S. and Gharehchopogh, F.S., 2026. An enhanced diabetes prediction using an improved hybrid deep learning algorithm with mountain gazelle optimizer. *Journal of Diabetes & Metabolic Disorders*, *25*(1), p.46.
25. Wang, L., Cao, Q., Zhang, Z., Mirjalili, S. and Zhao, W., 2022. Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. *Engineering Applications of Artificial Intelligence*, *114*, p.105082.
26. Chuang, L.Y., Yang, C.H. and Li, J.C., 2011. Chaotic maps based on binary particle swarm optimization for feature selection. *Applied Soft Computing*, *11*(1), pp.239–248.
27. Yang, D., Liu, Z. and Zhou, J., 2014. Chaos optimization algorithms based on chaotic maps with different probability distribution and search speed for global optimization.
28. Allam, A.A., Tawhid, M.A. and Owais, M., 2026. Chaotic Artificial Rabbits Optimization for Minimax Problems. *Mathematical and Computational Applications*, *31(3)*, p.83.
29. Richter, H. and Nörenberg, P.M., 2024, June. Random and Chaotic Sequences, and the Effect of their Distributions on PSO Performance. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pp.1–8. IEEE.
30. Sarafrazi, S. and Nezamabadi-Pour, H., 2013. Facing the classification of binary problems with a GSA-SVM hybrid system. *Mathematical and Computer Modelling*, *57*(1–2), pp.270–278.
31. Frank, A. and Asuncion, A., 2010. *UCI machine learning repository*.
32. Peterson, L.E., 2009. K-nearest neighbor. *Scholarpedia*, *4*(2), p.1883. DOI: 10.4249/scholarpedia.1883.
33. Zimmerman, D.W. and Zumbo, B.D., 1993. Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks. *The Journal of Experimental Education*, *62*(1), pp.75–86.
34. Zhao, W., Wang, L. and Mirjalili, S., 2022. Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications. *Computer Methods in Applied Mechanics and Engineering*, *388*, p.114194.
35. Abdollahzadeh, B., Gharehchopogh, F.S. and Mirjalili, S., 2021. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Computers & Industrial Engineering*, *158*, p.107408.
36. Askarzadeh, A., 2016. A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. *Computers & Structures*, *169*, pp.1–12.
37. Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M. and Chen, H., 2019. Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems*, *97*, pp.849–872.
38. Dehghani, M., Hubálovský, Š. and Trojovský, P., 2021. Northern goshawk optimization: a new swarm-based algorithm for solving optimization problems. *IEEE Access*, *9*, pp.162059–162080.
39. Moosavi, S.H.S. and Bardsiri, V.K., 2017. Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation. *Engineering Applications of Artificial Intelligence*, *60*, pp.1–15.
40. Yang, X., Wang, R., Zhao, D., Yu, F., Huang, C., Heidari, A.A., Cai, Z., Bourouis, S., Algarni, A.D. and Chen, H., 2023. An adaptive quadratic interpolation and rounding mechanism sine cosine algorithm with application to constrained engineering optimization problems. *Expert Systems with Applications*, *213*, p.119041.
41. Neggaz, I., Neggaz, N. and Fizazi, H., 2023. Boosting Archimedes optimization algorithm using trigonometric operators based on feature selection for facial analysis. *Neural Computing and Applications*, *35*(5), pp.3903–3923.