Swarm Intelligent Optimization Algorithms to Solve Feature Selection Problem

Yu Liu, Yu-Cai Wang, Jie-Sheng Wang*, Ya-Ni Zhang

Abstract—**The objective of feature selection is to identify the optimal subset of features from all non-empty subsets of the attribute set that best approximates the distribution of the original dateset or is optimal according to a certain evaluation criterion, such as achieving the highest classification accuracy. In wrapper-based feature selection, swarm intelligence optimization algorithms are widely used. This paper addresses the feature selection problem using a wrapper-based approach, employing nine swarm intelligence optimization algorithms to solve it and compare their performance. The algorithms include Crow Search Algorithm (CSA), Aquila Optimizer (AO), Whale Optimization Algorithm (WOA), Harris Hawks Optimization (HHO), Arithmetic Optimization Algorithm (AOA), Butterfly Optimization Algorithm (BOA), Ant Lion Optimizer (ALO), Prairie Dog Optimization (PDO), and Sparrow Search Algorithm (SSA). Performance testing was conducted on 12 standard UCI datasets to validate the performance of these algorithms. Convergence curves and box plots of accuracy values for the nine swarm intelligence optimization algorithms across the 12 datasets are presented. The simulation results are compared based on the mean and standard deviation of fitness, the number of selected features, and accuracy.**

Index Terms—**Feature selection, Swarm intelligence optimization, KNN classifier, Performance evaluation**

I. Introduction

Feature selection is a widely used data preprocessing step in the field of artificial intelligence, aimed at reducing the complexity of datasets by removing irrelevant or redundant attributes. A dataset processed through feature selection becomes easier to understand, exhibits higher generalization capability, and achieves better computational efficiency. Feature selection methods, based on their integration with learning algorithms, can be classified into three categories: Filter, Wrapper, and Embedded. Filter methods operate independently of any learning algorithm; features are selected before the training phase. These methods primarily select features based on statistical relationships either among features or between features and the target variable. Essentially, Filter methods and learning algorithms are decoupled, with feature selection acting as a preprocessing step, while the learning algorithm functions as the validation mechanism. Filter methods usually identify a subset of features without requiring a classification algorithm, which allows for the rapid elimination of noise, redundancy, and irrelevant features. Despite their simplicity, speed, and high computational efficiency, especially when handling high-dimensional data, Filter methods have a significant drawback: they cannot detect interactions among features due to the lack of a subsequent learning phase. Consequently, their performance often falls short of that achieved by Wrapper methods.

Unlike Filter methods, Wrapper methods integrate feature selection with a learning algorithm. They leverage machine learning to identify feature interactions, optimizing model performance by selecting relevant features. This involves iteratively training and evaluating the model on different subsets, using evaluation metrics to find the optimal subset. Wrapper methods often achieve higher accuracy than Filter methods by assessing features based on the learner's performance. However, they incur significant computational costs due to the need for classifier training for each subset. Robust search strategies are essential for identifying the best feature subset. Exhaustive, heuristic, and random searches are primary strategies in Wrapper methods. Heuristic search algorithms, which reduce the search space, are widely used. These include swarm intelligence and physics-inspired meta-heuristic algorithms, commonly applied in practice. Swarm intelligence optimization algorithms [6] simulate the behavior of biological swarms to search for the optimal feature subset. Examples include the Honey Badger Algorithm (HBA) [7] and the Artificial Hummingbird Algorithm (AHA) [8]. Physics-inspired algorithms draw inspiration from natural phenomena or physical laws to design algorithms for solving optimization problems, such as the Arithmetic Optimization Algorithm (AOA) [9], the Flow Direction Algorithm (FDA) [10], and the Energy Valley Optimizer (EVO) [11]. Agrawal et al. proposed the Quantum Whale Optimization Algorithm (QWOA) for feature selection, which integrates quantum concepts with the Whale Optimization Algorithm to enhance the exploration and exploitation capabilities of the classical WOA [12]. Gokalp et al. introduced a novel emotion classification wrapper feature selection algorithm based on the IG meta-heuristic, where Multinomial Naive Bayes (MNB) is used as a classifier due to its high performance in emotion classification with selected features

Manuscript received August 7, 2024; revised December 2, 2024. This work was supported by the Basic Scientific Research Project of Institution of Higher Learning of Liaoning Province (Grant No. LJ222410146054), and Postgraduate Education Reform Project of Liaoning Province (Grant No. LNYJG2022137).

Yu Liu is a doctoral candidate of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan,114051, P. R. China (e-mail: lnasacl@126.com)

Yu-Cai Wang is a doctoral candidate of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, P. R. China (e-mail: 1275934857@qq.com).

Jie-Sheng Wang is a professor of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, P. R. China (Corresponding author, phone: 86-0412-2538246; fax: 86-0412-2538244; e-mail: wang_jiesheng@126.com).

Ya-Ni Zhang is an undergraduate student of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, P. R. China (e-mail: 1345502380@qq.com).

by IG [13]. Le et al. utilized Grey Wolf Optimization (GWO) and Adaptive Particle Swarm Optimization (APSO) to optimize a Multi-Layer Perceptron (MLP), reducing the number of input attributes required to predict early onset of diabetes [14]. Hu et al. proposed a Decentralized Foraging Strategy SMA (DFSMA), along with a Binary DFSMA (BDFSMA) using a transfer function, demonstrating superior performance over the original SMA and proving its practical engineering value in search space and feature selection [15]. Alzaqebah et al. combined a neighborhood search with the Moth-Flame Optimization (MFO) algorithm to address the feature selection problem, helping to avoid local optima and premature convergence [16]. Arora et al. introduced the Butterfly Optimization Algorithm (bBOA) for wrapper-based feature selection, which selects the optimal feature subset, reduces the length of the feature subset, and improves classification accuracy [17].

This paper is based on the Wrapper feature selection method, comparing the Crow Search Algorithm (CSA) with eight swarm intelligence optimization algorithms: AO, AOA, ALO, BOA, HHO, SSA, WOA, and PDO. The performance of these nine nature-inspired algorithms is simulated and compared across 12 datasets. The structure of the paper is as follows: Section 2 introduces the eight swarm intelligence optimization algorithms; Section 3 describes solving the feature selection problem using the Crow Search Algorithm; Section 4 presents the experimental simulations and results analysis; and finally, Section 5 provides the conclusions of the paper.

II. Swarm Intelligence Optimization Algorithms

A. Ant Lion Optimizer

The Ant Lion Optimizer (ALO) addresses optimization problems by numerically simulating the interactions between ants and antlions. It introduces a random walk mechanism for ants to achieve global search, combined with a roulette wheel selection and elitism strategy to maintain population diversity and enhance algorithm performance. In this algorithm, antlions represent solutions to the problem, and they update and preserve near-optimal solutions by preying on ants with higher fitness levels [18]. The random walk process is defined as:

$$X(t)=\begin{bmatrix} 0, cumsum\left(2r(t_1)-1\right), cumsum\left(2r(t_2)-1\right) \\ \ldots\ cumsum\left(2r(t_n)-1\right) \end{bmatrix} \quad (1)$$

where, $X(t)$ is the set of random walk steps for an ant, *cumsum* is calculates the cumulative sum, t is the number of steps (in this paper, the maximum number of iterations), and $r(t)$ is a random function.

$$r(t)=\begin{cases} 1, & rand > 0.5 \\ 0, & rand < 0.5 \end{cases} \quad (2)$$

where, $rand$ is a random number in the range $[0, 1]$. Since the feasible domain has boundaries, Eq. (1) cannot directly update the ant's position. To ensure randomness, Eq. (1) needs to be normalized as follows:

$$X_i^t=\frac{(X_i^t-a_i)*(d_i^t-c_i^t)}{(b_i-a_i)}+c_i t \quad (3)$$

where, a_i is the minimum value of the random walk, b_i the maximum value of the i-th dimension variable's random walk, c_i^t the minimum value of the i-th dimension variable at the t-th iteration, and d_i^t is the maximum value of the i-th dimension variable at the t-th iteration.

(1) Ants entering traps. The random walk of ants is influenced by the traps set by antlions:

$$\begin{cases} c_i^t = Antlion_i^t + c^t \\ d_i^t = Antlion_i^t + d^i \end{cases} \quad (4)$$

(2) Ants sliding towards antlions. Adaptive reduction of the trap region constrains the random walk of ants:

$$\begin{cases} c' = \dfrac{c'}{I} \\ d' = \dfrac{d'}{I} \end{cases} \quad (5)$$

(3) Catching prey. After hunting, the antlion eats the ant and relocates to its region to increase the probability of capturing new prey, updating its position as follows:

$$Antlion_j^t = Ant_i^t, iff(Ant_i^t) > f(Antlion_j^t) \quad (6)$$

(4) Elitism strategy. Elite antlions, which are the best antlions generated during each iteration, influence the random walk of ants as well:

$$Ant_i' = \frac{(R_A' + R_E')}{2} \quad (7)$$

B. Whale Optimization Algorithm

The Whale Optimization Algorithm (WOA) was proposed by Mirjalili et al. in 2016 and is inspired by the hunting behavior of whales [19]. As a relatively novel optimization algorithm, WOA has not been established for long. In this algorithm, the position of the whales represents feasible solutions. During the hunting process, whales either encircle their prey or use bubble-net feeding. Encircling refers to all whales moving toward other whales, while bubble-net feeding involves whales swimming in a circular motion in the water and releasing bubbles to drive away their prey. In each iteration of their movement, whales randomly select between these two actions. When they encircle their prey, they randomly decide whether to swim toward the best position found so far. The steps of the WOA are as follows:

(1) Initially, the population size of whales is set to X , and the positions of X whales are randomly generated. The parameters of the WOA, including $\alpha, A, C,\ l, p$ and Max_Iter are then initialized.

(2) The fitness of each whale is calculated and compared, ultimately determining the most suitable individual, which is defined as X^{ϕ};

(3) The algorithm then enters the main loop.

If $p < 0.5$ and $|A| < 1$, the position of each whale is updated according to Eq. (8); Otherwise, the position of the whale is updated according to Eq. (9).

$$\vec{X}(t+1)=\overrightarrow{X^{\phi}}(t)-\vec{A}\cdot\vec{D} \tag{8}$$

$$\vec{X}(t+1)=\overrightarrow{X_{rand}}-\vec{A}\cdot\vec{D} \tag{9}$$

$$\vec{A}=2\vec{a}\cdot\vec{r}-\vec{a} \tag{10}$$

$$\vec{D}=\left|\vec{C}\cdot \vec{X}_{rand}-\vec{X}\right| \tag{11}$$

where, $\vec{A}$ represents the convergence factor, and $\vec{D}$ denotes the distance between the individual whale and the best whale position, which are calculated using Eq. (10) and Eq. (11), respectively. Here, $\vec{r}$ is a random number in the range of [0,1]; $\vec{a}$ linearly decreases from 2 to 0 as the number of iterations increases; $\overrightarrow{X_{rand}}$ represents the position of any whale in the current population. If $p>0.5$, the position of each whale is calculated according to Eq. (12).

$$\vec{X}(t+1)=\vec{D}\cdot e^{bl}\cdot\cos(2\pi l)+\overrightarrow{X^{\phi}}(t) \tag{12}$$

$$\overrightarrow{D'}=\overrightarrow{X^{\phi}}(t)-\vec{X}(t) \tag{13}$$

where, $\overrightarrow{D'}$ represents the distance from the *i*-th whale to the food source; l is a random value in the range of [-1,1]; b is the spiral constant.

(4) The whale population is reassessed to identify the global optimal whale individual and its position.

(5) If the termination criteria of the WOA are met, the execution of the algorithm will stop. Otherwise, if the termination conditions are not satisfied, the algorithm will return to Step 2 and continue execution.

(6) The global optimal solution $\overrightarrow{X^{\phi}}$ is outputted.

C. *Sparrow Search Algorithm*

The Sparrow Search Algorithm (SSA) is a swarm intelligence optimization algorithm based on the behavior of sparrows, simulating the processes of foraging and avoiding predators. This algorithm employs a discoverer-follower model and incorporates a scouting alert mechanism. In this algorithm, the discoverer is the sparrow that locates the best food source, while the followers are other sparrows. Additionally, scouting sparrows are established to monitor safety. Structurally, the SSA is quite similar to the Artificial Bee Colony (ABC) algorithm, but there are some differences in the search operators, making it an improvement over the ABC algorithm [20].

Based on the aforementioned description of sparrows, a mathematical model can be established to construct the SSA. This model employs virtual sparrows to simulate the tracking of food sources, where each sparrow's position is represented by a position vector, defined as follows:

$$X=\begin{pmatrix} x_{1,1} & \cdots & x_{1,D} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,D} \end{pmatrix} \tag{14}$$

where the number of sparrows is denoted by N, and the dimension of the optimization is represented by D. The following vector can be used to represent the fitness values of all sparrows:

$$f(x)=\begin{pmatrix} f([\,x_{1,1} \;\; x_{1,2} \;\; \cdots \;\; x_{1,D}]) \\ \vdots \\ f([x_{N,1} \;\; x_{N,1} \;\; \cdots \;\; x_{N,D}]) \end{pmatrix} \tag{15}$$

Where, each row corresponds to the value $f(x)$.

$$x_{(i.j)}^{t+1}=\begin{cases} x_{(i.j)}^{t}\cdot exp\left(\dfrac{-i}{\alpha\cdot T}\right), & if\ R_2<ST \\ x_{(i.j)}^{t}+Q\cdot L_{i,j}, & if\ R_2\geq ST \end{cases} \tag{16}$$

The scavengers will continue to monitor the producers until the latter locate a suitable food source, at which point the producers will leave their current position to pursue the target food source. If successful, the scavengers will obtain food from the producers; otherwise, they will maintain their vigilance tasks. The foragers update their positions according to the following equation.

$$x_{i,j}^{t+1}=\begin{cases} Q\cdot exp\left(\dfrac{x_{worst,j}^{t}-x_{i,j}^{t}}{i^2}\right), & if\ i>\dfrac{N}{2} \\ x_{p,j}^{t+1}+\left|x_{i,j}^{t}-x_{p,j}^{t+1}\right|\cdot A_{0,j}^{+}\cdot L, & if\ i\leq\dfrac{N}{2} \end{cases} \tag{17}$$

$$x_{i,j}^{t+1}=\begin{cases} x_{worst,j}^{t}+\beta\cdot\left|x_{i,j}^{t}-x_{best,j}^{t+1}\right|, & if\ f_i>f_g \\ x_{i,j}^{t}+K\cdot\left(\dfrac{\left|x_{i,j}^{t}+x_{worst,j}^{t}\right|}{(f_i-f_w)+\varepsilon}\right), & if\ i=\dfrac{N}{2} \\ x_{i,j}^{t}, & if\ i<\dfrac{N}{2} \end{cases} \tag{18}$$

D. *Butterfly Optimization Algorithm*

The Butterfly Optimization Algorithm (BOA) was introduced by Arora et al. in 2019. This algorithm is inspired by the survival and reproductive behaviors of butterflies in nature. Butterflies analyze airborne scents to locate food sources or other butterflies, relying on their sensory perception to identify these resources. In the BOA, each butterfly emits a scent that diffuses through the environment, with the scent intensity correlating to the butterfly's fitness. As a butterfly's position changes, so does its fitness. When a butterfly detects the scent of another, it moves closer to that butterfly, a process referred to as "global searching." Conversely, if a butterfly does not detect a more fragrant counterpart, it will engage in random movement, known as "local searching." The scent intensity is represented by the stimulus intensity, as shown in Eq. (19):

$$f=cI^{\alpha} \tag{19}$$

where, c is the sensory factor; I represents the intensity of the stimulus; α is the power exponent. The intensity I is related to the fitness of the butterfly. During the global search, butterflies move towards the optimal solution g^*, as expressed in Eq. (20).

$$x_i^{t+1}=x_i^t+(r^2*g^*-x_i^t)*f_i \tag{20}$$

where, x_i^t represents the solution vector of the *i*-th butterfly in the *t*-th iteration; g^* denotes the best solution found so far; f_i is the scent of the *i*-th butterfly; r is a random number in the range of [0,1]. The local search phase can be expressed as follows in Eq. (21).

$$x_i^{t+1} = x_i^t + (r^2 * x_j^t - x_k^t) * f_i \tag{21}$$

where, x_i^t and x_j^t represent the k-th and j-th butterflies randomly selected from the solution space. During the foraging process of the butterflies, both global and local searches are performed, and the parameter p can be employed to facilitate the transition between these two search modes. Each iteration utilizes the random values of r and p generated by Eq. (21) to determine whether to engage in global or local search strategies.

E. Harris Hawks Optimization

The Harris Hawks Optimization (HHO) algorithm is inspired by the hunting strategies of Harris hawks, where the hawks symbolize candidate solutions and the optimal solution is referred to as the prey. In the HHO algorithm, the Harris hawks utilize their keen eyesight to track their prey and subsequently execute a surprise attack to capture it. The HHO algorithm comprises three main components: the exploration phase, the transition phase between exploration and exploitation, and the exploitation phase [22].

(1) Exploration Phase

Harris hawks randomly inhabit a location and employ two strategic approaches to locate their prey, as described in Eq. (22).

$$X(\tau+1) = \begin{cases} X_{rand}(\tau) - r_1 \,|\, X_{rand}(\tau) - 2r_2 X(\tau) \,|, & q \ge 0.5 \\ [X_{rabit}(\tau) - X_m(\tau)] - r_3[l^b + r_4(u^b - l^b)], & q < 0.5 \end{cases} \tag{22}$$

where, $X(\tau)$ and $X(\tau+1)$ represent the positions of individuals in the current and next iterations, respectively; τ denotes the iteration count. $X_{rand}(\tau)$ refers to the position of a randomly selected individual, while $X_{rabbit}(\tau)$ indicates the position of the prey, which corresponds to the individual with the best fitness. The variables $r_1 - r_4$ and q are random numbers uniformly distributed between [0,1]; q is used to determine the strategy to be adopted in the random selection process. $X_m(\tau)$ represents the average position of individuals, as shown in Eq. (23).

$$X_m(\tau) = \frac{1}{M} \sum_{k-1}^{M} X_k(\tau) \tag{23}$$

where, $X_k(\tau)$ represents the position of the k individual in the population, and M denotes the size of the population.

(2) Transition Phase

The HHO algorithm operates based on the energy of the prey's escape, which is defined by Eq. (24).

$$E = 2E_0(1 - \frac{\tau}{T}) \tag{24}$$

where, E_0 represents the initial energy of the prey, τ is the iteration count, and T denotes the maximum number of iterations. When $|E| \ge 1$, the algorithm enters the exploration phase; conversely, when $|E| < 1$, it transitions to the exploitation phase.

(3) Exploitation Phase

Definition: r is a random number uniformly distributed in the interval [0, 1], used to select different exploitation strategies.

(a) When $0.5 \le |E| < 1$ and $r \ge 0.5$, the soft encirclement strategy is employed for position updates, as expressed in Eq. (25).

$$X(\tau+1) = \Delta X(\tau) - E\left|JX_{rabbit}(\tau) - X(\tau)\right| \tag{25}$$

$$\Delta X(\tau) = X_{rabbit}(\tau) - X(\tau) \tag{26}$$

where, $\Delta X(\tau)$ denotes the displacement of the prey's current position, while J is a random variable uniformly distributed within the interval [0, 2].

(b) When the condition $|E| < 0.5$ and $r \ge 0.5$ is met, the position is updated using the hard encirclement strategy, as formulated in Eq. (27).

$$X(\tau+1) = X_{rabbit}(\tau) - E\left|\Delta X(\tau)\right| \tag{27}$$

(c) When the condition 0.5≤|E|<1 and r<0.5 is satisfied, the position is updated using the progressive soft encirclement approach, as delineated in Eq. (28).

$$X(\tau+1) = \begin{cases} Y, & f(\mathrm{Y}) < f(\mathrm{X}(\tau)) \\ Z, & f(\mathrm{Z}) < f(\mathrm{X}(\tau)) \end{cases} \tag{28}$$

$$Y = X_{rabbit}(\tau) - E\left|JX_{rabbit}(\tau) - X(\tau)\right| \tag{29}$$

$$Z = Y + S \times LF(2) \tag{30}$$

where, $f(\cdot)$ denotes the fitness function; s is a two-dimensional random vector; and $LF(\cdot)$ represents the mathematical formulation of the Lévy flight.

(d) When the condition $|E| < 0.5$ and $r < 0.5$ is met, the position is updated using the asymptotic hard encirclement method as described by Eq. (28), with Y defined in Eq. (31).

$$Y = X_{rabbit}(\tau) - E\left|JX_{rabbit}(\tau) - X_m(\tau)\right| \tag{31}$$

F. Arithmetic Optimization Algorithm

The Arithmetic Optimization Algorithm (AOA) fundamentally relies on exploration and exploitation mechanisms. During the exploration phase, a comprehensive traversal of the search space is required since the optimal solution could be situated anywhere within this space [23]. Conversely, the exploitation phase capitalizes on the effective information, utilizing correlations among high-quality solutions to incrementally adjust and refine the search trajectory from the initial guess towards the optimal solution. In AOA execution, the switching between exploration and exploitation phases is governed by the function value of the Mathematical Optimizer Accelerator (MOA). When $r1 > MOA$, global exploration is activated; whereas, when $r1 < MOA$, the algorithm transitions to local exploitation.

$$MOA(t) = Min + t * (\tfrac{Max - Min}{T}) \tag{32}$$

(1) Global exploration. Division and multiplication operations generate high distribution values or decision metrics conducive to the global exploration mechanism.

$$x_{i,j}(t+1) = left(bestx_j \div (MOP+\varepsilon) * (UB_j - LB_j) * \mu + LB_J), r_2 < 0.5 \quad (33)$$

$$x_{i,j}(t+1) = left(bestx_j * MOP * (UB_j - LB_j) * \mu + LB_J), otherwise \quad (34)$$

The Mathematical Optimizer Probability (MOP) is formulated as:

$$MOA(t) = 1 - \frac{t^{\frac{1}{a}}}{T^{\frac{1}{a}}} \quad (35)$$

(2) Local exploration. Subtraction and addition operations result in high-density values, which enhance the algorithm's capacity for local exploitation.

$$x_{i,j}(t+1) = left(bestx_j - MOP * (UB_j - LB_j) * \mu + LB_J), r_3 < 0.5 \quad (36)$$

$$x_{i,j}(t+1) = left(bestx_j + MOP * (UB_j - LB_j) * \mu + LB_J), otherwise \quad (37)$$

G. Aquila Optimizer

As an innovative intelligent optimization algorithm, the core concept of the Arithmetic Optimization (AO) algorithm is to emulate the natural behavior of eagles in capturing their prey, with the goal of achieving optimization objectives. This algorithm exhibits exceptional optimization capabilities and efficient convergence rates, offering an effective solution for addressing optimization challenges [24]. The population is randomly initialized within the defined search space.

$$X_{i,j} = rand \times (UB_j - LB_j) + LB_j,\ i = 1,2,\ldots,N_j = 1,2,\ldots \quad (38)$$

In this context, *rand* is a random vector, *LB* denotes the lower bound of the *j*-th dimension of the given problem, and *UB* denotes the upper bound of the *j*-th dimension of the given problem. Expansion of search. Eagles identify hunting areas by soaring high and diving vertically, a behavior that can be modeled mathematically as follows:

$$x_1(t+1) = X_{best}(t) \times \left(1 - \frac{t}{T}\right) + \left(X_M(t) - X_{best}(t) * rand\right) \quad (39)$$

$$x_M(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t), \quad \forall j = 1,2,\cdots,Dim \quad (40)$$

where, $x_1(t+1)$ represents the solution generated at iteration $t+1$; $X_{best}(t)$ is the best solution obtained up to iteration t, approximating the prey's location; $(1-t/T)$ controls the search expansion over iterations; $X_M(t)$ denotes the mean value of the current solutions at iteration t, calculated using Eq. (39); *rand* is a random value between 0 and 1; r and T represent the current and maximum iteration numbers, respectively; *Dim* indicates the problem's dimensionality, and *N* is the population size (number of candidate solutions).

H. Prairie Dog Optimization

The fundamental concept of the Prairie Dog Optimization (PDO) algorithm is to model prairie dog behavior in two distinct stages [25]. The first stage, global exploration, encompasses two behavioral modes: searching for food and constructing burrows. The second stage, local exploitation, includes two behavioral modes: responding to food source signals and reacting to predator signals.

(1) Exploration Stage

The exploration stage employs two strategies:

Strategy One. Individuals search for new food sources within their burrows.

$$PD_{i+1,j+1} = GBest_{i,j} - eCBest_{i,j} \times \rho - CPD_{i,j} \times Levy(n) \forall \frac{Maxiter}{4} \quad (41)$$

Strategy two. Individuals continuously excavate new burrows.

$$PD_{i+1,j+1} = GBest_{i,j} \times rPD \times DS \times Levy(n) \forall \frac{Maxiter}{4} \le iter < \frac{Maxiter}{2} \quad (42)$$

where, $PD_{i,j}$ represents the position of an individual, while $GBest_{i,j}$ signifies the current global best solution. $Levy(n)$ refers to the standard Lévy flight, and ρ is a specialized food source alarm fixed at 0.1 Hz for this experiment. The mathematical models for $eCBest_{i,j}$, $CPD_{i,j}$ and DS are formulated as follows:

$$eCBest_{i,j} = GBest_{i,j} \times \Delta + \frac{CPD_{ij} \times mean(PD_{n,m})}{GBest_{i,j} \times (UB_j - LB_j) + \Delta} \quad (43)$$

where, $eCBest_{i,j}$ evaluates the effectiveness of the current best solution, with Δ representing minor differences between individual prairie dogs. CPD_{ij} is the cumulative random effect of all prairie dogs in the population, and *rPD* is the position of the random solution.

$$CPD_{ij} = \frac{GBest_{i,j} - rPD_{i,j}}{GBest_{i,j} + \Delta} \quad (44)$$

DS denotes the digging strength of small groups, which depends on the quality of the food source. r introduces randomness to ensure effective exploration, taking the value -1 for odd iterations and 1 for even iterations. Max_{iter} is the maximum number of iterations, and *iter* represents the current iteration number.

$$DS = 1.5 \times r \times \left(1 - \frac{iter}{Max_{iter}}\right)^{2\frac{iter}{Max_{iter}}} \quad (45)$$

(2) Exploitation Stage

In the exploitation stage, the first type of sound indicates the location and quality of a food source. When an individual discovers a high-quality food source, other individuals converge at the sound source to satisfy their food needs. The position update for this scenario is mathematically represented as follows:

$$PD_{i+1,j+1} = GBest_{i,j} - eCBest_{i,j} \times \varepsilon - CPD_{i,j} \times rand \forall \frac{Max_{iter}}{2} < \frac{Max_{iter}}{4} \quad (46)$$

The second type of sound warns of a predator's presence. Prairie dogs on the predator's path will hide. The position update for this scenario is mathematically represented as follows:

$$PD_{i+1,j+1} = GBest_{i,j} \times PE \times rand \forall \frac{3Max_{iter}}{4} < Max_{iter} \quad (47)$$

where, PE denotes the predator effect, and $rand$ is a random number between 0 and 1.

$$PE = 1.5 \times r \times (1 - \frac{iter}{Max_{iter}})^{(2\frac{iter}{Max_{iter}})} \quad (48)$$

III. Solving the Feature Selection Problem Using the Crow Search Algorithm

A. Crow Search Algorithm

In 2016, Askarzadeh et al. conducted an in-depth study on the foraging behavior of crows and successfully developed an innovative algorithm known as the Crow Search Algorithm (CSA). The CSA simulates two behaviors of crows: foraging and hiding food, and it is characterized by its parameter efficiency, simplicity, ease of understanding, and strong global search capability [26].

The CSA is inspired by the food-storing behavior of crows, which involves hiding leftover food for retrieval when needed. Crows are known for their greed, often following each other to discover food resources during foraging. Locating food hidden by a crow is not an easy task; once a crow detects that other crows are trailing it, it will quickly adopt a strategy to mislead its pursuers by altering its direction of movement, thereby protecting its food source. The CSA utilizes a population of seekers to explore the search space. By employing a population-based approach, the probability of finding a good solution and escaping local optima is enhanced. The CSA algorithm increases solution diversity and employs a memory mechanism to retain the best solutions. During the iterations of the Crow Search Algorithm, each crow randomly selects either another crow or itself and migrates towards its hidden position(i.e., the best solution found by the crow). This signifies that in each iteration, the best-known position will be directly used in subsequent searches to find even better solutions.

(1) Crows live in a social structure, exhibiting group behavior.

(2) Crows possess the ability to remember the locations where they have stored food.

(3) There are instances of mutual chasing and theft among crows.

(4) When being followed, crows will protect their stored food with a certain probability.

In each iteration, the position update of crow i is closely related to a randomly selected crow j. Crow i will follow crow j to its food storage location m_j_iter (represented by the position of crow j in the code). A perception probability parameter $AP = 0.1$ is set to determine whether crow j detects the tracking behavior of crow i. Consequently, there are two potential position update scenarios for crow i.

1) Successful Tracking

In this case, crow j does not notice the tracking behavior of crow i:

$$X^{i,iter+1} = X^{i,iter} + r_i \times fl^{i,iter} \times (m^{j,iter} - X^{i,iter}) \quad (49)$$

where, r_i represents a uniformly distributed random number in the interval [0, 1], and $fl^{i,iter}$ denotes the flight distance of crow i at the current iteration $iter$. The flight distance determines the step size of the movement towards the selected hidden position. When the flight distance is relatively short, the crows typically engage in local search; however, as the flight distance increases, the crows are more likely to undertake global search behaviors.

2) Tracking Failure

At this point, the tracking behavior of crow i has been detected, prompting crow j to guide it to a random position. Therefore, in conjunction with step 1, the position update method for crow i is given by:

$$X^{i,iter+1} = \begin{cases} X^{i,iter+1} = X^{i,iter} + r_i \times fl^{i,iter} \times (m^{j,iter} \\ -X^{i,iter}), r_j \geq AP^{j,iter} \\ a\ random\ position, \quad otherwise \end{cases} \quad (50)$$

where, r_j represents a uniformly distributed random number in the interval [0, 1]. $AP^{j,iter}$ indicates the adaptability level of crow j after iterations. A smaller AP value leads to a more intense $iter$ search behavior, while a larger AP value enhances the diversity of the search behavior. The algorithm proceeds as follows:

(1) Parameter initialization. Define the decision variables and determine the perception probability (AP), flight distance (fl), maximum number of iterations ($iter\ max$), and the number of crows (N).

(1) Position and memory initialization. Crows are randomly distributed within the search space. In the initial iteration, each crow hides food at its starting position.

(3) Evaluation of the fitness (objective) function. For each crow, compute its objective function value.

(4) Update crow positions. Determine new positions based on Eq. (49).

(5) Verify the feasibility of new positions. Check the validity of the new position for each crow. If the new position is feasible, the crow will update its position; otherwise, it retains its current position.

(6) Compute the objective function for the new position. Calculate the objective function value for each crow's new position.

(7) Update crow memory. If the objective function value at the new position is better, update the memory to reflect the information of the new position.

(8) Termination condition. Repeat Steps 4 to 7 until the termination condition is satisfied, at which point output the position of the best objective function value.

B. K-Nearest Neighbors

The K-Nearest Neighbors (KNN) classifier is a supervised learning algorithm that determines the category of an unknown sample by calculating the distances between it and the samples of known categories. It selects the K nearest samples as neighbors and subsequently classifies the unknown sample based on the obtained category information through majority voting or weighted majority voting. The algorithm can be described as follows:

(1) Calculate the distances between the test data and the training data.

(2) Sort the distances in ascending order.

(3) Select the K nearest points.

(4) Count the frequency of each category among the K points.

(5) Assign the category with the highest frequency among the K points as the predicted classification for the test data.

In this experiment, KNN is employed for the classification task, calculating the Euclidean distance D_E between the training set and the test data to determine the closest samples, as expressed in Eq. (51).

$$D_E = \sqrt{\sum_{i-1}^{K} (Train_{Fi} - Test_{Fi})^2} \tag{51}$$

C. Fitness Function

genetic and evolutionary algorithms. It evaluates the quality of individual solutions to a problem, with higher fitness values indicating superior solutions that are favored for survival and reproduction, while lower values suggest inferiority and lead to elimination. Typically derived from the problem's objective function, the fitness function can be represented by the numerical value of the objective function, related metrics, or an evaluation function based on the problem's characteristics and individual solutions. Designing the fitness function requires consideration of the problem's features and objectives, selecting appropriate metrics to ensure a meaningful and discriminative evaluation. Feature subsets are represented as binary vectors, with '1' for selected features and '0' for non-selected features. These conflicting objectives are encapsulated in the fitness function, as shown in Eq. (52).

$$fitness = h_1 \gamma_R(D) + h_2 \frac{|M|}{|N|} \tag{52}$$

where, $\gamma_R(D)$ represents the classification error rate corresponding to the feature subset selected by the classifier. $|M|$ denotes the number of selected features, $|N|$ is the total number of features, h_1 and h_2 are two weight coefficients reflecting the subset's classification accuracy and length, respectively, with the constraint $h_1 + h_2 = 1$. Given the necessity for an accurate classification model, the classification accuracy is assigned a higher inertia weight. In this study, h_1 and h_2 are set to 0.99 and 0.01, respectively.

D. Performance Evaluation of Feature Selection

Metrics are often utilized when evaluating and interpreting the results of feature selection problems. These evaluation metrics include the fitness value, classification accuracy, and the average number of selected features. Eqs. (53) to (58) sequentially represent the calculation methods for average classification accuracy, the average number of selected features, and the mean and standard deviation of fitness values.

$$Mean\_accuracy = \frac{1}{20} \sum_{i=1}^{20} Accuracy_i \tag{53}$$

where, $Mean\_accuracy$ represents the average classification accuracy obtained from 20 independent runs of the algorithm, while $Accuracy_i$ denotes the classification accuracy achieved in each individual run. The classification accuracy is calculated as follows:

$$Accuracy = \frac{1}{N} \sum_{i=1}^{N} match(Pl_i, Al_i) \tag{54}$$

where, N represents the number of test instances, which corresponds to the number of instances in the dataset. Pl_i denotes the predicted class label for data point i, while Al_i refers to the actual class label from the labeled data, serving as the reference class label. The function $match(Pl_i, Al_i)$ acts as a comparison function. When $Pl_i = Al_i$, $match(Pl_i, Al_i) = 1$, otherwise, $match(Pl_i, Al_i) = 0$.

$$Mean\_feature = \frac{1}{20} \sum_{i=1}^{20} feature_i \tag{55}$$

where, $Mean\_feature$ represents the average number of selected features obtained from M independent algorithm runs, while $feature_i$ denotes the number of selected features obtained in each individual run.

$$Mean\_fitness = \frac{1}{20} \sum_{i=1}^{20} fitness_i \tag{56}$$

where, $Mean\_fitness$ represents the average fitness obtained from M independent runs of the algorithm, while f_i denotes the best fitness achieved in each individual run. The method for calculating fitness is as follows.

$$\begin{aligned} fitness &= 099 * (1 - Accuracy) + \\ &0.01 * \frac{|Selected\ features\ Count|}{|Total\ features\ Count|} \end{aligned} \tag{57}$$

where, $Accuracy$ refers to the classification accuracy.

$$Std\_fitness = \sqrt{\frac{1}{20} \sum (fitness_i - Mean\_fitness)^2} \tag{58}$$

where, $Std\_fitness$ represents the standard deviation of the fitness values, $fitness_i$ denotes the fitness value obtained in the i iteration, and $Mean\_fitness$ is calculated using Eq. (56).

IV. Experimental Results and Discussions

A. Selection of Experimental Data

The experiments involved selecting 12 datasets from the UCI repository for classification studies. These datasets, with varying instances, feature counts, and classes, provide diverse perspectives to evaluate the advantages and disadvantages of different nature-inspired algorithms across various datasets. Table I details the information of these datasets.

In this study, the K-nearest neighbor (KNN) algorithm with K=5 was employed to calculate the classification accuracy in the fitness function, as KNN is proven to be faster and simpler. The experiments were repeated 20 times with different random seeds. Additionally, to prevent over-fitting, five-fold cross-validation was used. The dataset was divided into training and test sets. In the first iteration, 80% of the feature vectors were used for training, and the remaining 20% were used for testing. Subsequently, another 20% of the feature vectors were used for testing, while the remaining 80% were used for training. This process was repeated until all feature vectors were used for testing. Finally, the average statistical measurements over 20 independent runs were collected and presented as the final results.

All experiments were conducted using MATLAB R2020a on an Intel Core i3-1005G1 machine with a CPU of 1.20 GHz, 4GB of RAM, and a Windows 10 operating system. In this study, the population size for each algorithm was set to 10, and the maximum number of iterations was set to 100. The common parameters for the nine algorithms were kept consistent. The dimensionality of the search space equaled the total number of features. According to previous research, setting the parameter h_1 to 0.99 resulted in optimal classification performance.

B. Feature Selection Results and Analysis

To evaluate the effectiveness and superiority of the Crow Search Algorithm (CSA) in feature selection, CSA was compared with eight commonly used swarm intelligence optimization algorithms across 12 different UCI datasets. These algorithms include Ant Lion Optimizer (ALO), Arithmetic Optimization Algorithm (AOA), Butterfly Optimization Algorithm (BOA), Harris Hawks Optimization (HHO), Population-Based Optimization Algorithm (PDO), Sparrow Search Algorithm (SSA), Whale Optimization Algorithm (WOA), and Eagle Strategy (AO). The results were evaluated based on the mean and standard deviation of fitness, the number of selected features, and accuracy, with the optimal values highlighted in bold. Data tables provided the results for the mean fitness, accuracy standard deviation, mean number of selected features, and accuracy. Additionally, convergence curves and box plots of precision values for the nine swarm intelligence optimization algorithms on 12 datasets were presented.

Simulation results are shown in Tables II- IV, with the best results highlighted in bold. Table II presents the mean and standard deviation of fitness for the nine swarm intelligence optimization algorithms. Tables III- IV show the comparison of accuracy values and the mean number of selected features for the different algorithms, respectively. In these tables, the best results are highlighted in bold. In Table II, the CSA algorithm achieved the highest average fitness values on most datasets (12 datasets). Table III demonstrates that the CSA algorithm holds a significant advantage in accuracy. Table IV shows the number of selected features for all swarm intelligence optimization algorithms, with CSA winning by an absolute margin. These results underscore the superior performance of the CSA algorithm in terms of fitness, accuracy, and feature selection, illustrating its robustness and effectiveness in various classification scenarios.

To provide a more intuitive and vivid demonstration of the differences among the nine swarm intelligence optimization algorithms, convergence curves and accuracy box plots were generated based on the best classification accuracy calculated by the KNN classifier during the execution of the algorithms. The convergence curves for the nine swarm intelligence optimization algorithms across the 12 datasets are illustrated in Fig. 1. Seen from Fig.1, the horizontal axis represents the number of iterations, while the vertical axis depicts the average accuracy values obtained after each algorithm is independently executed 20 times. These curves highlight the convergence behavior of each algorithm, with the CSA algorithm consistently exhibiting superior convergence performance. Additionally, the accuracy box plots are presented in Fig. 2. Offering insights into the variability and robustness of their performance. The combination of convergence curves and accuracy box plots offers a comprehensive visual comparison, emphasizing the advantages of the CSA algorithm in terms of both convergence speed and accuracy stability.

TABLE I. 12 DATASETS USED IN SIMULATION EXPERIMENTS

Number	Datasets	Features	Instances	Classes
1	Algerian Forest Fires	12	244	2
2	Brain Tumor2	10366	50	3
3	Bupa	5	345	2
4	Clean1	166	476	2
5	Climate Model Simulation Crashes	19	540	2
6	Connectionist Bench	59	208	2
7	Forest type mapping	26	523	4
8	Handwritten	255	1593	10
9	HAPTDataSet	560	360	9
10	Heart	12	270	2
11	Wine	12	178	3
12	Zoo	15	101	7

TABLE II. COMPARISON OF MEAN FITNESS AND ACCURACY STANDARD DEVIATION

Dataset	Measure	ALO	CSA	WOA	SSA	BOA	HHO	AOA	AO	PDO
Algerian	AVG	0.0092	0.0065	0.0125	0.0084	0.0216	0.0182	0.0143	0.0082	0.0114
	STD	0.0093	0.0072	0.0122	0.0087	0.0067	0.0096	0.0100	0.0096	0.0101
Brain	AVG	0.0013	0.0049	0.0015	0.0049	0.0076	0.0116	0.0109	0.0021	0.0004
	STD	0.0012	0.0000	0.0015	0.0012	0.0219	0.0306	0.0302	0.0014	0.0011
Bupa	AVG	0.2826	0.2839	0.2852	0.2892	0.3058	0.2982	0.2826	0.2826	0.2867
	STD	0.0000	0.0039	0.0092	0.0111	0.0163	0.0179	0.0000	0.0000	0.0108
Zoo	AVG	0.1001	0.0997	0.1069	0.1033	0.1175	0.1033	0.1083	0.1037	0.0903
	STD	0.0253	0.0151	0.0226	0.0247	0.0202	0.0218	0.0240	0.0199	0.0292

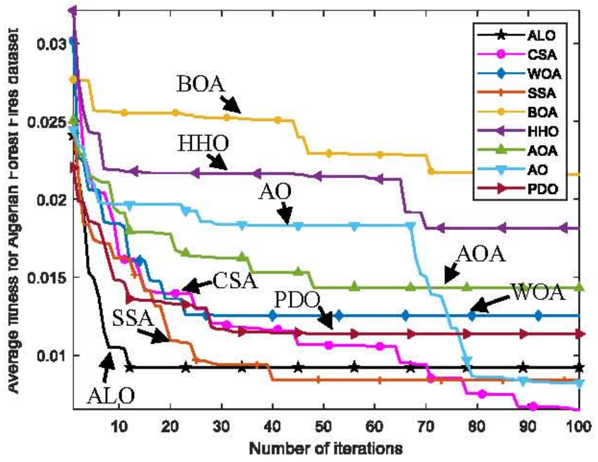
Clean1	AVG	0.0292	0.0090	0.0385	0.0238	0.0438	0.0486	0.0209	0.0136	0.0310
	STD	0.0295	0.0178	0.0205	0.0156	0.0174	0.0246	0.0189	0.0141	0.0221
Climate	AVG	0.0536	0.0503	0.0735	0.0619	0.0824	0.0587	0.0779	0.0638	0.0541
	STD	0.0170	0.0092	0.0150	0.0134	0.0093	0.0117	0.0155	0.0101	0.0163
Connectionist	AVG	0.0615	0.0503	0.0704	0.0616	0.0736	0.0799	0.0535	0.0437	0.0505
	STD	0.0202	0.0264	0.0354	0.0299	0.0212	0.0367	0.0295	0.0321	0.0353
Wine	AVG	0.0394	0.0113	0.0434	0.0296	0.0650	0.0469	0.0175	0.0135	0.0206
	STD	0.0246	0.0135	0.0216	0.0252	0.0385	0.0383	0.0250	0.0144	0.0192
Forest	AVG	0.0866	0.0969	0.0971	0.1199	0.1060	0.1058	0.0879	0.0719	0.0812
	STD	0.0084	0.0069	0.0083	0.0067	0.0102	0.0107	0.0097	0.0055	0.0050
HAPTD	AVG	0.0263	0.0220	0.0300	0.0246	0.0361	0.0317	0.0262	0.0199	0.0208
	STD	0.0157	0.0164	0.0146	0.0120	0.0124	0.0179	0.0153	0.0106	0.0136
Heart	AVG	0.0894	0.0811	0.0934	0.0853	0.1093	0.0981	0.1033	0.0894	0.0774
	STD	0.0348	0.0206	0.0428	0.0266	0.0306	0.0472	0.0299	0.0236	0.0455
Handwritten	AVG	0.0269	0.0269	0.0522	0.0286	0.0608	0.0578	0.0288	0.0208	0.0265
	STD	0.0077	0.0044	0.0052	0.0094	0.0062	0.0096	0.0071	0.0069	0.0088

TABLE III. Comparison of Accuracy Values for Different Algorithms

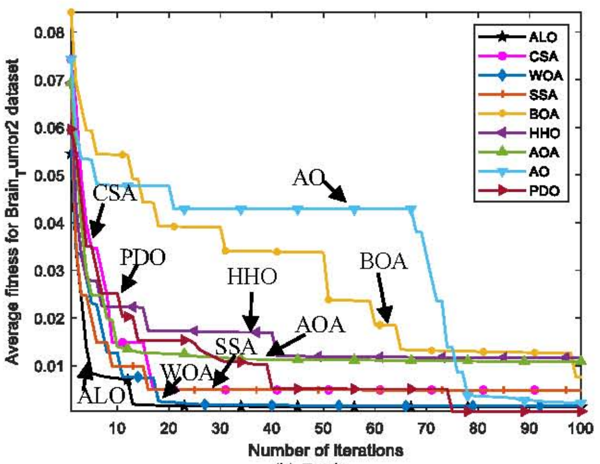
Dataset	ALO	CSA	WOA	SSA	BOA	HHO	AOA	AO	PDO
Algerian	0.9938	0.9969	0.9896	0.9948	0.9813	0.9844	0.9875	0.9938	0.9906
Brain	1.0000	1.0000	1.0000	0.9950	0.9900	0.9900	1.0000	1.0000	1.0000
Bupa	0.7246	0.7232	0.7217	0.7167	0.6964	0.7058	0.7246	0.7246	0.7196
Zoo	0.9750	0.9950	0.9650	0.9800	0.9600	0.9550	0.9825	0.9900	0.9725
Clean1	0.9500	0.9537	0.9295	0.9421	0.9200	0.9442	0.9232	0.9389	0.9505
CliMate	0.9417	0.9532	0.9319	0.9421	0.9292	0.9222	0.9486	0.9597	0.9537
Connectionist	0.9146	0.9061	0.9037	0.8829	0.8951	0.8951	0.9122	0.9293	0.9207
Wine	0.9629	0.9914	0.9586	0.9729	0.9371	0.9557	0.9843	0.9886	0.9814
Forest	0.9798	0.9827	0.9755	0.9798	0.9678	0.9731	0.9774	0.9851	0.9846
HAPTD	0.9153	0.9229	0.9111	0.9188	0.8931	0.9056	0.8986	0.9139	0.9285
Heart	0.8222	0.8593	0.8130	0.8537	0.8093	0.7954	0.8380	0.8556	0.8444
Handwritten	0.9061	0.9042	0.8989	0.9008	0.8860	0.9017	0.8953	0.9011	0.9159

TABLE IV. Comparison of Average Selected Feature Count

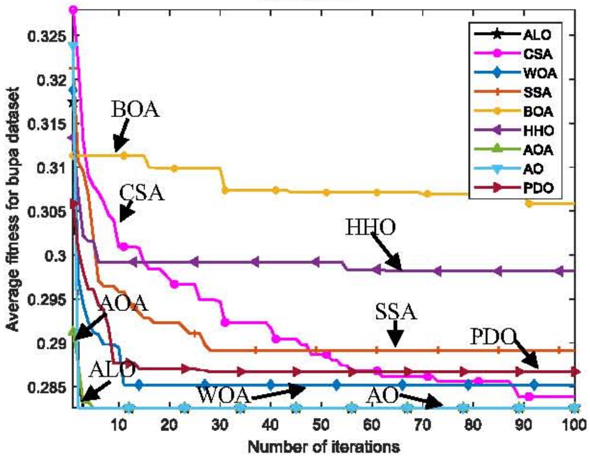
Dataset	ALO	CSA	WOA	SSA	BOA	HHO	AOA	AO	PDO
Algerian	3.95	4.45	2.90	4.20	3.90	3.50	2.50	2.65	2.70
Brain	1362.4	5046.2	1571.7	5050.2	2724.4	1807.3	992.7	2156.8	411.5
Bupa	6.00	2.90	5.85	5.20	3.15	4.15	6.00	6.00	5.45
Zoo	7.15	5.55	6.15	6.40	6.75	6.55	5.65	5.95	6.10
Clean1	68.10	74.60	61.80	76.85	52.90	58.75	30.70	56.45	84.70
Climate	7.40	5.15	6.10	8.70	7.00	5.75	5.30	7.60	9.25
Connectionist	12.80	23.45	10.40	23.90	13.10	11.60	5.60	11.50	16.20
Wine	3.40	2.25	3.15	3.50	3.65	4.00	2.50	2.85	2.85
Forest	17.10	9.85	15.40	12.40	11.45	13.65	10.35	14.00	14.95
HAPTD	310.45	269.65	303.40	270.75	189.75	256.45	162.35	232.00	367.80
Heart	5.25	6.25	4.80	5.65	4.55	5.90	5.05	5.50	5.50
Handwritten	183.05	124.45	174.40	130.05	119.20	154.65	118.30	149.45	179.40



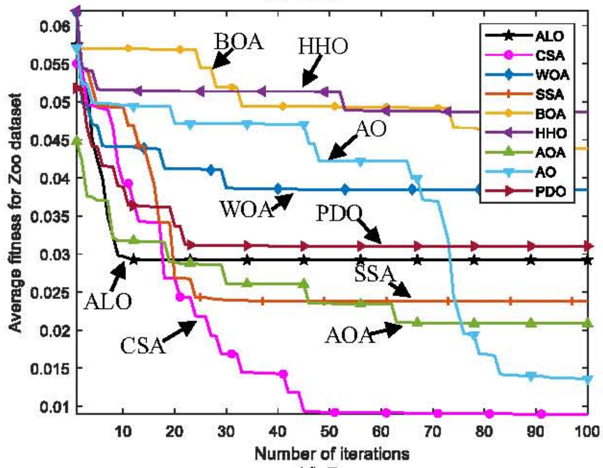
(a) Agerian



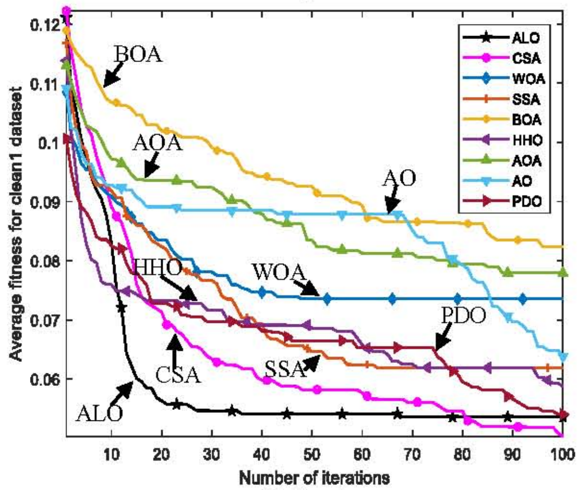
(b) Brain



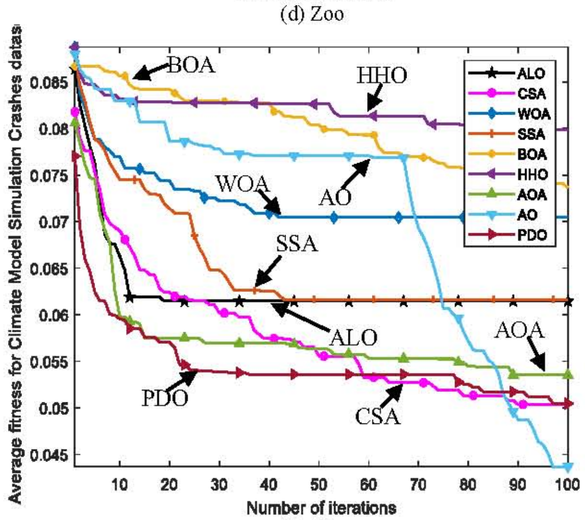
(c) Bupa



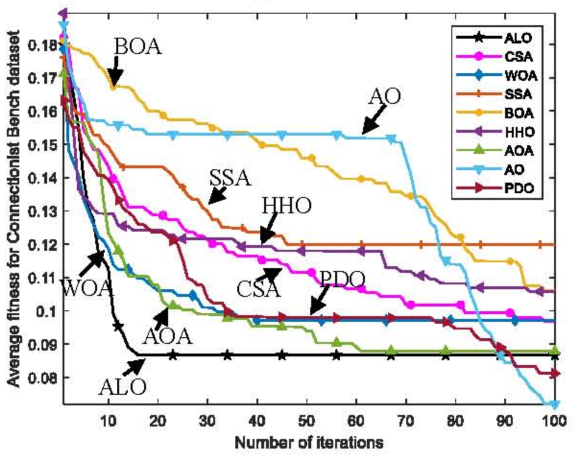
(d) Zoo



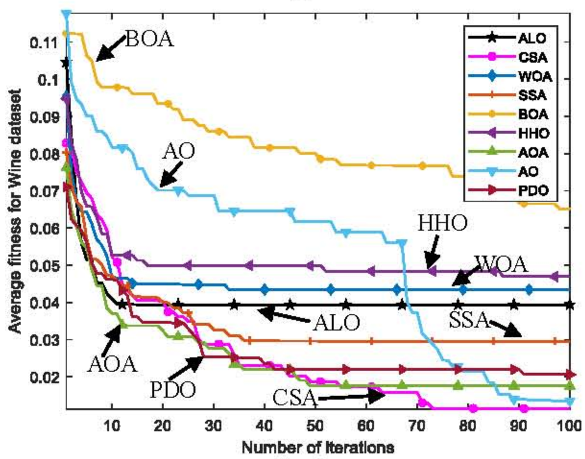
(e) Clean1



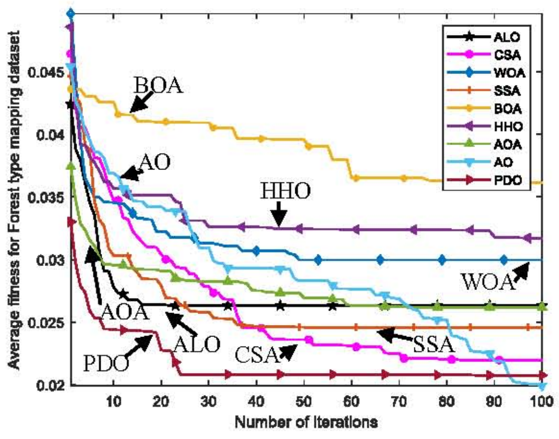
(f) Climate



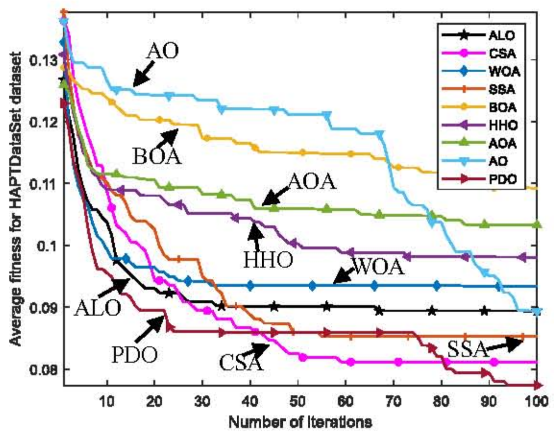
(g) Connectionist



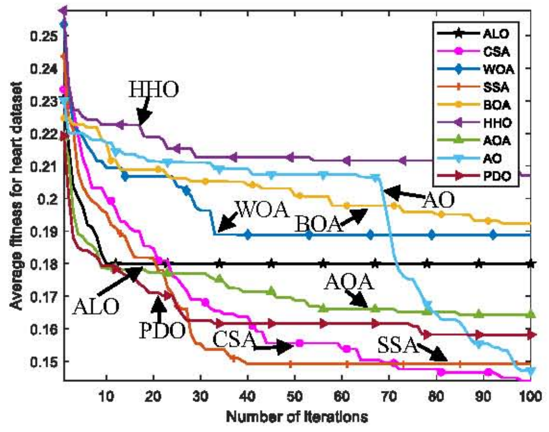
(h) Wine



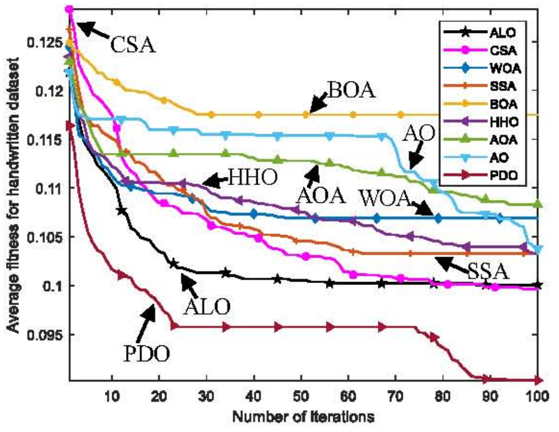
(i) Forest



(j) HAPTDataSet

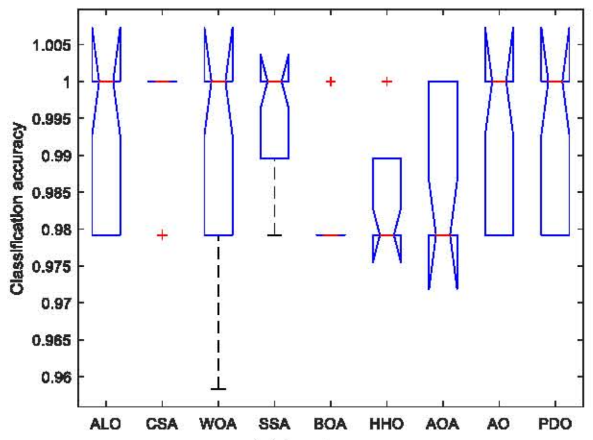


(k) Heart

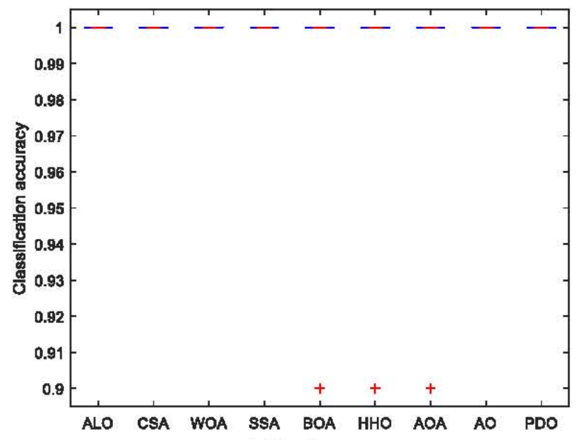


(l) Handwritten

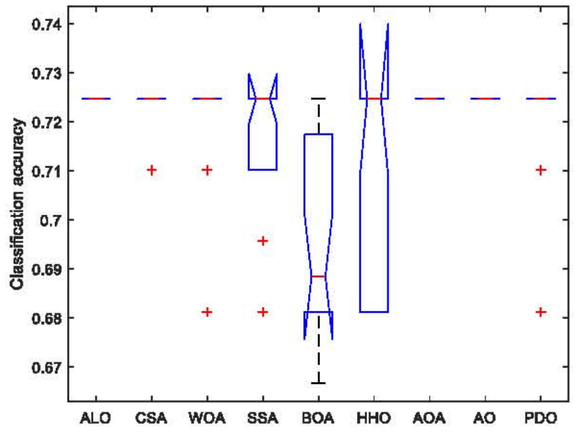
Fig. 2 Convergence curves of 9 nature-inspired algorithms on 12 datasets.



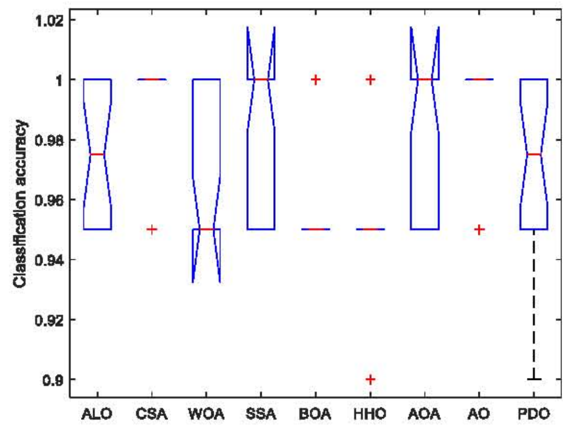
(a) Agerian



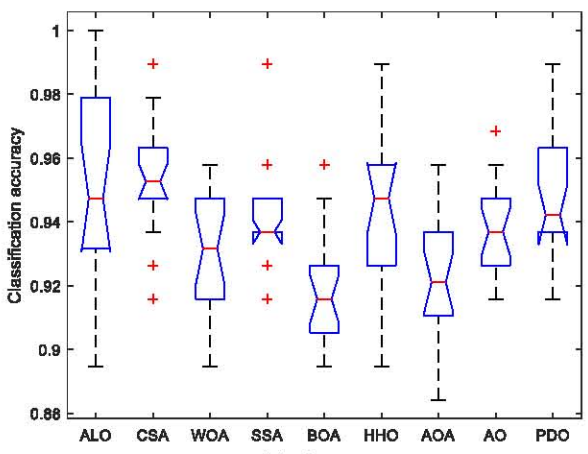
(b) Brain



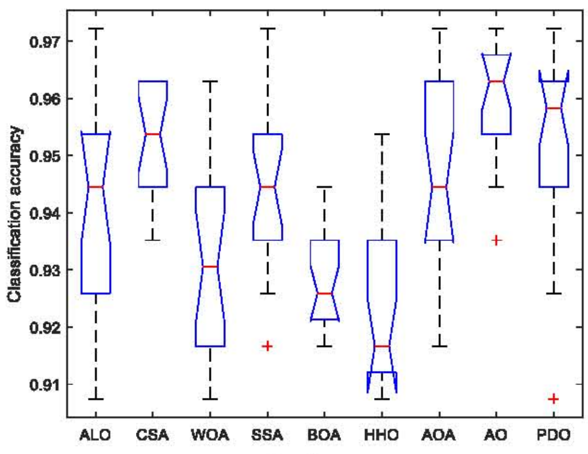
(c) Bupa



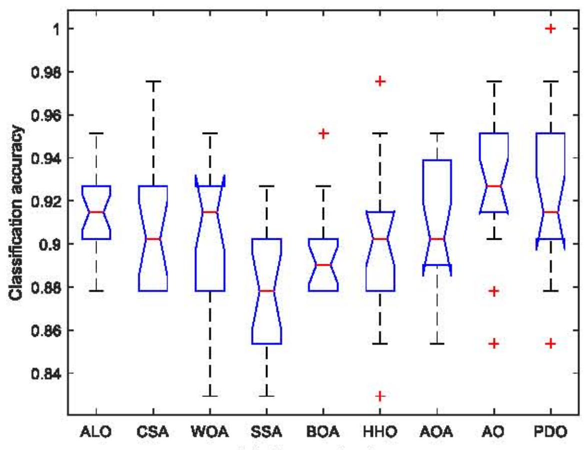
(d) Zoo



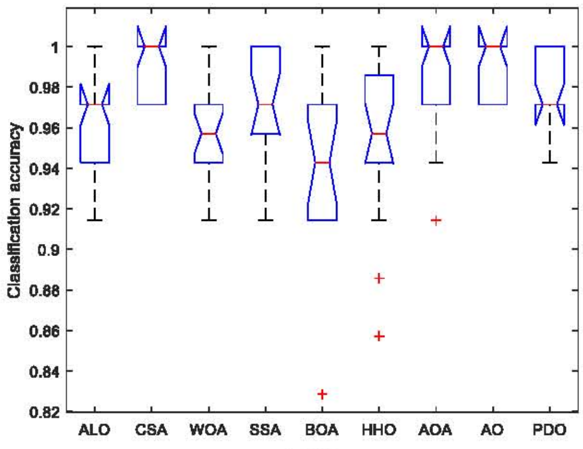
(e) Clean1



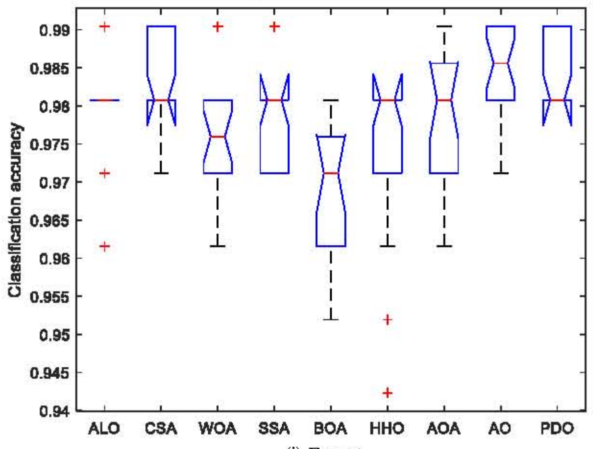
(f) Climate



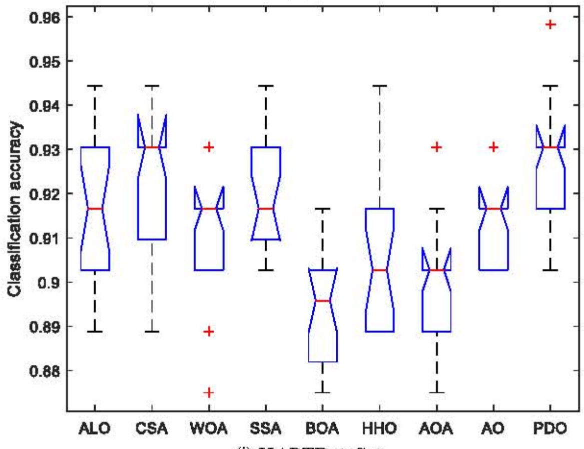
(g) Connectionist



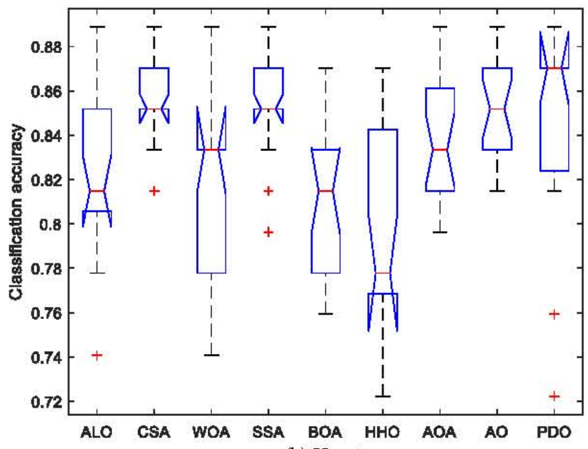
(h) Wine



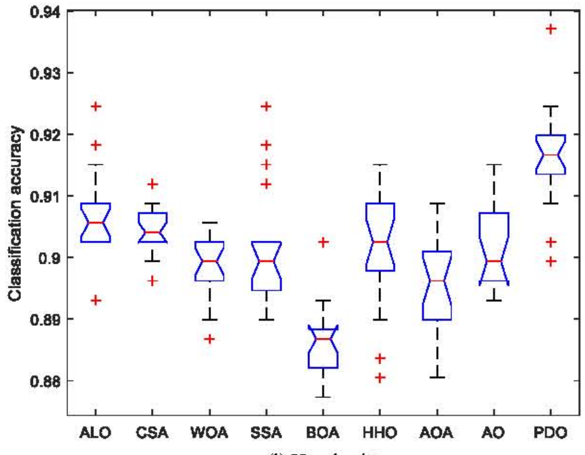
(i) Forest



(j) HAPTDataSet



(k) Heart



(l) Handwritten

Fig. 3 Box plots of fitness values for 9 nature-inspired algorithms.

V. Conclusions

Based on the wrapper-based feature selection method, the Crow Search Algorithm (CSA) was compared with eight swarm intelligence optimization algorithms: Ant Lion Optimizer (ALO), Arithmetic Optimization Algorithm (AOA), Butterfly Optimization Algorithm (BOA), Harris Hawks Optimization (HHO), Sparrow Search Algorithm (SSA), Whale Optimization Algorithm (WOA), and Population-Based Optimization Algorithm (PDO). The convergence curves and box plots of precision values for the nine nature-inspired algorithms were presented across 12 datasets. Comprehensive performance metrics were compared. CSA achieved the highest average fitness values on most datasets, with an absolute advantage in the number of selected features, while also maintaining superior accuracy. The evaluation of the proposed nature-inspired algorithms was based on the mean and standard deviation of fitness, the number of selected features, and accuracy, with optimal values highlighted in bold. The comparison revealed that CSA consistently obtained the highest average fitness values on most datasets and demonstrated a clear advantage in the number of selected features and accuracy. The results, including mean and standard deviation of fitness, number of selected features, accuracy, convergence curves, and box plots of precision values across 12 datasets for the nine algorithms, provide significant reference value for future research.

References

[1] X. Zhang, C. Mei, D. Chen, Y. Yang, and J. Li, "Active Incremental Feature Selection Using a Fuzzy-rough-set-based Information Entropy," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 5, pp. 901-915, 2019.

[2] M. Z. Fidakar, and D. J. Mussa, "Brain Tumour Classification Using BoF-SURF with Filter-Based Feature Selection Methods," *Multimedia Tools and Applications*, vol. 83, pp. 65833-65855, 2024.

[3] H. Kema, K. Meena, and R. Pandian, "Analyze the Impact of Feature Selection Techniques in the Early Prediction of CKD," *International Journal of Cognitive Computing in Engineering*, vol. 5, no. 1, pp. 66-77, 2024.

[4] X. Xu, F. Wei, T. Jia, L. Zhuo, H. Zhang, X. Li, and X. Wu, "Embedded EEG Feature Selection for Multi-Dimension Emotion Recognition via Local and Global Label Relevance," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 32, pp. 514-526, 2024.

[5] T. Zhao, Y. Zheng, and Z. Wu, "Feature Selection-Based Machine Learning Modeling for Distributed Model Predictive Control of Nonlinear Processes," *Computers & Chemical Engineering*, vol. 169, pp. 108074, 2023.

[6] J. Tang, G. Liu, and Q. Pan, "A Review on Representative Swarm Intelligence Algorithms for Solving Optimization Problems: Applications and Trends," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 10, pp. 1627-1643, 2021.

[7] F. A. Hashim, E. H. Houssein, K. Hussain, M. S. Mabrouk, and W. Al-Atabany, "Honey Badger Algorithm: New Metaheuristic Algorithm for Solving Optimization Problems," *Mathematics and Computers in Simulation*, vol. 192, pp. 84-110, 2022.

[8] W. Zhao, L. Wang, and S. Mirjalili, "Artificial Hummingbird Algorithm: A New Bio-Inspired Optimizer with Its Engineering Applications," *Computer Methods in Applied Mechanics and Engineering*, vol. 388, pp. 114194, 2022.

[9] L. M. Abualigah, A. T. Diabat, S. Mirjalili, M. E. Elaziz, and A. H. Gandomi, "The Arithmetic Optimization Algorithm," *Computer Methods in Applied Mechanics and Engineering*, vol. 376, pp. 113609, 2021.

[10] H. Karami, M. V. Anaraki, S. Farzin, and S. M. Mirjalili, "Flow Direction Algorithm (FDA): A Novel Optimization Approach for Solving Optimization Problems," *Computers & Industrial Engineering*, vol. 156, 107224, 2021.

[11] M. Azizi, U. Aickelin, H. A. Khorshidi, and M. Baghalzadeh Shishehgarkhaneh, "Energy Valley Optimizer: a Novel Metaheuristic Algorithm for Global and Engineering Optimization," *Scientific Reports*, vol. 13, no. 1, pp. 226, 2023.

[12] R. K. Agrawal, B. Kaur, and S. Sharma, "Quantum Based Whale Optimization Algorithm for Wrapper Feature Selection," *Applied Soft Computing*, vol. 89, pp. 106092, 2020.

[13] O. Gokalp, E. Tasci, and A. Ugur, "A Novel Wrapper Feature Selection Algorithm Based on Iterated Greedy Metaheuristic for Sentiment Classification," *Expert Systems with Applications*, vol. 146, pp. 113176, 2020.

[14] T. M. Le, T. M. Vo, T. N. Pham, and S. V. T. Dao, "A Novel Wrapper-Based Feature Selection for Early Diabetes Prediction Enhanced With a Metaheuristic," *IEEE Access*, vol. 9, pp. 7869-7884, 2021.

[15] J. Hu, W. Gui, A. Heidari, Z. Cai, G. Liang, H. Chen, and Z. Pan, "Dispersed Foraging Slime Mould Algorithm: Continuous and Binary Variants for Global Optimization and Wrapper-based Feature Selection," *Knowledge-Based Systems*, vol. 237, pp. 107761, 2021.

[16] M. Alzaqebah, N. Alrefai, E. A. Ahmed, S. Jawarneh, and M. K. Alsmadi, "Neighborhood Search Methods with Moth Optimization Algorithm as a Wrapper Method for Feature Selection Problems," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 4, pp. 2088-2098, 2020.

[17] S. Arora, and P. Anand, "Binary Butterfly Optimization Approaches for Feature Selection," *Expert Systems with Applications*, vol. 116, pp. 147-160, 2019.

[18] S. Mirjalili, "The Ant Lion Optimizer," *Advances in Engineering Software*, vol. 83, pp. 80-98, 2015.

[19] Y. Ling, Y. Zhou, and Q. Luo, "Lévy Flight Trajectory-Based Whale Optimization Algorithm for Global Optimization," *IEEE Access*, vol. 5, pp. 6168-6186, 2017.

[20] J. Xue, B. Shen, and A. Pan, "A Hierarchical Sparrow Search Algorithm to Solve Numerical Optimization and Estimate Parameters of Carbon Fiber Drawing Process," *Artificial Intelligence Review: An International Science and Engineering Journal*, vol. 56, no. Suppl. 1, pp. 1113-1148, 2023.

[21] S. Arora, and S. Singh, "Butterfly Optimization Algorithm: A Novel Approach for Global Optimization," *Soft Computing*, vol. 23, no. 3, pp. 715-734, 2019.

[22] A. A. Heidari, S. M. Mirjalili, H. Faris, I. Aljarah, M. M. Mafarja, and H. Chen, "Harris Hawks Optimization: Algorithm and Applications," *Future Generation Computer Systems*, vol. 97, pp. 849-872, 2019.

[23] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, "The Arithmetic Optimization Algorithm," *Computer Methods in Applied Mechanics & Engineering*, vol. 376, pp. 113609, 2021.

[24] C. Liu, and J. Zhen, "Diagonal Loading Beamforming Based on Aquila Optimizer," *IEEE Access*, vol. 11, pp. 69091-69100, 2023.

[25] G. Nithyanandam, C. Ambiyaram, and S. Prabathkumar, "An Intelligent Hybrid Prairie Dog Optimization Algorithm-based Stable Cluster Reliable Routing Scheme for VANETs," *International Journal of Communication Systems*, vol. 36, pp. 14-36, 2023.

[26] A. Askarzadeh, "A Novel Metaheuristic Method for Solving Constrained Engineering Optimization Problems: Crow Search Algorithm," *Computers & Structures*, vol. 169, pp. 1-12, 2016.