Hybrid Strategy improves Crayfish Optimization Algorithm

Tong Ding

School of Big Data and Statistics, Guizhou University of Finance and Economics, Guiyang, Guizhou Province

Received: 01 October 2024/ Revised: 11 October 2024/ Accepted: 18 October 2024/ Published: 31-10-2024

Abstract— *In response to the deficiencies of the Crayfish Optimization Algorithm (COA), such as slow convergence speed and susceptibility to local optima when solving complex optimization problems, an improved Crayfish Optimization Algorithm (ICOA) with a hybrid strategy is proposed. This improvement incorporates five strategies: an enhanced position update function for followers from the Sparrow Algorithm, a refined spiral position update function from the Whale Optimization Algorithm, modifications to the criteria and methods for assessing food size, the golden ratio coefficient, and an information accumulation function. Optimization tests conducted on 12 benchmark functions demonstrate that the improved algorithm shows enhancements in both convergence speed and optimization performance. Finally, experiments applying the improved optimization algorithm to engineering application problems further validate the superiority of the enhanced Crayfish Optimization Algorithm.*

Keywords— Classical test functions, COA, Engineering Constraint Problems, Information Accumulation.

I. INTRODUCTION

Traditional optimization algorithms usually target structured problems, which have relatively clear descriptions in terms of the problems and conditions, such as linear programming [1], integer programming [2], mixed programming [3], and constrained or unconstrained conditions [4]. These algorithms possess clear structural information, stable structures and parameters, and unique and clear global optima. Traditional optimization algorithms can be theoretically analyzed in terms of computational complexity and convergence. While they are generally effective for single-extremum problems, they still exhibit many deficiencies when solving multi-extremum problems. In recent years, the increasing complexity of research problems and the ambiguity of computational results have rendered traditional mathematical models increasingly insufficient for these types of problems, creating a growing demand for superior optimization algorithms.

Optimization problems can be expressed as a continuous or combinatorial search space for designs, representing a process of finding the maximum or minimum of a function f(x). The parameter x represents the solution within the search range. The optimization problem goes through multiple iterations to find the optimal solution, ultimately yielding the best solution:

$$x_{best}\left\{x_1, x_2, x_3 \ldots\ldots, x_D\right\}.$$

Here, D denotes the dimensionality of the solution. In each iteration, a new solution

$x_{new}\left\{x_1, x_2, x_3 \ldots\ldots, x_D\right\}$ is generated. If x_{new} is better than x_{best}, then $x_{best} = x_{new}$. The iterations continue until the found solution meets the preset criteria. This final solution (close to the optimal solution) is referred to as the optimal solution. This is the process of meta-heuristic optimization algorithms [5].

Unlike traditional optimization methods, meta-heuristic algorithms do not require consideration of gradient information in the solution space and can search for optimal or suboptimal solutions using random variables and natural heuristic rules[6], [7]. Thus, meta-heuristic algorithms exhibit good resistance to premature convergence, making them easy to program for solving various optimization problems. They have been widely used by scholars in fields such as computer science and medicine (citation). Commonly used meta-heuristic algorithms include the Artificial Bee Colony Algorithm (ABC) [8], Butterfly

Optimization Algorithm (BOA) [9], Grasshopper Optimization Algorithm (GOA) [10], Golden Sine Algorithm (GSA) [11], Moth Flame Optimization Algorithm (MFO) [12], Slime Mold Algorithm (SMA) [13], Teaching-Learning-Based Optimization Algorithm (TLBO)[14], Whale Optimization Algorithm (WOA) [15], and Crayfish Optimization Algorithm (COA)[16], among others. Meta-heuristic algorithms excel in identifying global optimal solutions for optimization problems, and their optimization results are independent of initial conditions while exhibiting strong robustness in the solution domain. As a result, these algorithms have found broad applications in practical scenarios [17][19]. They have demonstrated practical value across various fields, including but not limited to solving the multi-traveling salesman problem [20], multi-level threshold image segmentation [21], ship routing and scheduling problems [22], feature selection [18], [19], and multi-level image segmentation [23]. Currently, scholars have proposed various meta-heuristic algorithms based on characteristics found in nature, which can be broadly categorized into swarm intelligence algorithms, physics-based algorithms, evolutionary algorithms, and human-inspired algorithms [24].

However, according to the no free lunch theorem [25], no single algorithm can solve all optimization problems. Each optimization algorithm is effective for certain optimization problems but ineffective for others. As a result, researchers continuously propose various novel or improved optimization algorithms to address different optimization problems. In 2023, COA introduced a new meta-heuristic optimization algorithm— the Crayfish Optimization Algorithm (COA)— which simulates different strategic behaviors in response to varying environments and food sources. COA presents a method that mimics the foraging, thermoregulating, and competitive behaviors of crayfish. By adjusting temperature, the algorithm balances exploration and exploitation capabilities. However, COA still encounters issues such as slow convergence speed, vulnerability to local optima, and low optimization accuracy. To address these issues, this paper proposes an improved COA operator by integrating an improved position update function from the Sparrow Algorithm, an enhanced spiral position update function from the Whale Optimization Algorithm, the golden ratio coefficient, and an information accumulation function, thereby modifying the factors and methods for assessing food size. This improvement facilitates the transition between global exploration and local exploitation phases. During the global exploration phase, two distinct walking mechanisms are adopted based on the position of followers, enhancing the algorithm's ability to escape local optima and avoiding premature convergence. In the local exploitation phase, four different neighborhood mechanisms are employed to enhance the optimization capability of the algorithm.

II. CRAYFISH OPTIMIZATION ALGORITHM

The COA simulates the foraging, thermoregulating, and competitive behaviors of crayfish in different scenarios by constructing a mathematical model. This algorithm consists of two phases: the thermoregulation phase and the foraging phase. Changes in temperature affect the behavior of crayfish, causing them to enter different phases. Temperature is defined by Formula 1. When the temperature exceeds 30°C, crayfish will choose a cool place to thermoregulate. At appropriate temperatures, crayfish will engage in foraging behavior. Additionally, the amount of food intake for crayfish is influenced by temperature. The optimal feeding range for crayfish is between 20°C and 30°C. Consequently, the feeding amount of crayfish can be approximated as a normal distribution. Therefore, the temperature range for COA is from 20°C to 35°C, and the mathematical model is as follows:

$$temp = rand \times 15 + 20 \tag{1}$$

Where temp represents the current environmental temperature, and rand is a random number between 0 and 1.

$$p = C_1 \times \left(\frac{1}{\sqrt{2 \times \pi} \times \delta} \times e^{\left(-\frac{(temp-\mu)^2}{2\delta^2} \right)} \right) \tag{2}$$

Here, p is the food intake amount, $C_1 = 0.2$, $\mu = 25$, and $\delta = 3$.

2.1 Thermoregulation phase:

When the temperature exceeds 30°C, the temperature is too high. At this point, crayfish will choose to enter a burrow to thermoregulate. Rapidly seeking out a burrow is a random process. When $rand < 0.5$, it indicates that no other crayfish are competing for the burrow, and the crayfish will directly enter the burrow to cool off. When $rand \geq 0.5$, it signifies that other

crayfish are also interested in this burrow. In this case, they will compete for it. Therefore, the burrow is first defined with the following mathematical formula:

$$X_{shade} = \left(X_G + X_L \right) / 2 \tag{3}$$

Where X_{shade} denotes the burrow position, X_G represents the current optimal position, and X_L is the optimal position of the current population.

Entering the Burrow:

$$X_{i,j}^{t+1} = X_{i,j}^{t} + C_2 \times rand \times \left(X_{shade} - X_{i,j}^{t} \right) \tag{4}$$

Where $X_{i,j}^{t+1}$ signifies the j-th dimensional position of the i-th crayfish in the $t+1$-th iteration, and C_2 is a decreasing curve.

$$C_2 = 2 - \left(t / T \right) \tag{5}$$

Where t indicates the current iteration count, and T is the maximum iteration count.

Competing for the Burrow:

$$X_{i,j}^{t+1} = X_{i,j}^{t} - X_{z,j}^{t} + X_{shade} \tag{6}$$

Where z signifies a randomly selected crayfish individual, as indicated by the following equation:

$$z = int\left(rand \times \left(N - 1 \right) \right) + 1 \tag{7}$$

Where int denotes the integer part.

2.2 Foraging phase:

When the temperature is below 30°C, the conditions are suitable for crayfish to eat. During this time, crayfish will move towards the food. When foraging, crayfish will assess the size of the food. If the food is too large, the crayfish will use their claws to tear off pieces of food before eating; if the food is not large, the crayfish will eat directly. First, the position of the food is defined as follows:

$$X_{food} = X_G \tag{8}$$

Where X_{food} represents the position of the food, which is also the current optimal position of the population.

Food Size:

$$Q = C_3 \times rand \times \left(fitness_i / fitness_{food} \right) \tag{9}$$

Where Q is the food size, C_3 is the food factor representing the maximum food size (constant value of 3), $fitness_i$ denotes the fitness value of the i-th crayfish, and $fitness_{food}$ indicates the fitness value of the food.

Large Food:

The crayfish's judgment of food size comes from the size of the largest food item. When $Q > \left(C_3 + 1 \right) / 2$, it indicates that the food is too large. At this point, the crayfish will use its claws to tear the food before eating. The mathematical equation is as follows:

$$X_{food1} = e^{\left(-\frac{1}{Q} \right)} \times X_{food} \tag{10}$$

When the food is torn into smaller pieces, it is put back into the mouth. To simulate the alternating process, a combination of sine and cosine functions is used. Additionally, the amount of food obtained by the crayfish is also related to the food intake, thus the mathematical equation for foraging is:

$$X_{i,j}^{t+1} = X_{i,j}^{t} + X_{food} \times p \times \left(cos\left(2\pi \times rand\right) - sin\left(2\pi \times rand\right) \right) \tag{11}$$

Small Food:

When $Q \le \left(C_3 + 1\right)/2$, the crayfish only needs to move toward the food and eat directly, as expressed in the following equation:

$$X_{i,j}^{t+1} = \left(X_{i,j}^{t} - X_{food} \right) \times p + p \times rand \times X_{i,j}^{t} \tag{12}$$

During the foraging phase, crayfish adopt different feeding methods based on the size of Q.

III. MIXED STRATEGY IMPROVED CRAYFISH OPTIMIZATION ALGORITHM

The COA is a novel and effective optimization algorithm that can be applied to many optimization problems. However, COA faces challenges when dealing with complex optimization issues, such as slow convergence speed and a tendency to get trapped in local optima. To address these shortcomings, this article introduces a mixed strategy to improve COA, particularly for addressing the weaknesses of the COA in solving complex optimization problems.

3.1 Introduction of improved position update function from the sparrow algorithm:

From the crayfish optimization process, it can be seen that during the foraging phase, the historical optimal crayfish leads, and during the thermoregulation phase, there is a 50% chance to approach the optimal values of the current and historical populations. Therefore, during the exploration process, the crayfish population will quickly converge on the position of the optimal crayfish. This can lead to severe population convergence, making it easy to fall into local optimal solutions, hindering the ability to escape these local optima. Thus, this paper introduces and improves the position update operator of the entrants from the Sparrow Algorithm, sorting the population and selecting the less fit crayfish for disturbance, which enhances randomness and increases the search range, thus increasing the likelihood of escaping local optima and enhancing the global optimization ability of the algorithm. The improved position update function from the Sparrow Algorithm is as follows:

$$w_i = \frac{ln\left(N + 0.5\right) - ln\left(i\right)}{\sum_{j=1}^{N}\left(ln\left(N + 0.5\right) - ln\left(j\right)\right)} \tag{13}$$

Where w_i is the weight coefficient of the i-th ranked crayfish, and N is the population size.

$$X_i^{t+1} = \begin{cases} rand\left(N\left(0{,}5 + 10 \times \frac{i}{N} \times \sqrt{\frac{t}{T}} \right) \right) \times e^{\left(\frac{X_{worst} - X_i^t}{i^2} \right)}, & i > \frac{5}{6}N \\ X_{best}^{t} + \left| w_i \times X_i^t - w_1 \times X_{best}^{t} \right| / \left(w_i + \right) w_1 \times A^{+}, & \frac{2}{3}N < i \le \frac{5}{6}N \end{cases} \tag{14}$$

Where $rand\left(\bullet\right)$ represents the random number generated from the probability density or distribution, X_{worst} indicates the worst individual in the current population, and X_{best}^{t} denotes the optimal position achieved so far. A is a one-dimensional multi-element random matrix with elements of either 1 or -1, and $A^{+} = A^{T}\left(AA^{T}\right)^{-1}$.

3.2 Introduction of improved spiral update position function from the whale optimization algorithm:

During the thermoregulation phase of the crayfish, when the temperature exceeds 30°C, they will either enter a burrow or compete for a burrow. In the original algorithm, only a random selection of a crayfish was used to approach during this

competition, which increased the algorithm's randomness but decreased its convergence. Therefore, the improved spiral update position function from the Whale Optimization Algorithm is introduced as follows:

$$X_i^{t+1} = \left(1 - \left(\frac{t}{T}\right)^2\right) \times \left|X_{best}^t - X_i^t\right| \times e^l \times cos\left(2\pi \times l\right) + X_{best}^t \tag{15}$$

$$l = \left(-2 - \frac{t}{T}\right) \times rand + 1 \tag{16}$$

3.3 Change in the criteria and method for judging food size:

In the original algorithm, crayfish exhibit two behaviors based on food size when foraging, but the criteria for determining food size were too random, causing the algorithm's convergence speed to slow down. Thus, changing the food size judgment criteria is crucial. This paper introduces normalization to alter the food size evaluation criteria as follows:

$$IQ_i^t = \left(fitness_i^t - fitness_{best}^t\right) \Big/ \left(fitness_{worst}^t - fitness_{best}^t + 10^{-300}\right) \tag{17}$$

Where IQ_i^t represents the food size judgment factor, and $fitness_{worst}^t$ denotes the fitness value of the least fit crayfish at iteration t.

3.4 Introduction of the golden ratio coefficient:

In the original algorithm, when crayfish forage for large food items, a temperature-controlled normal distribution was introduced; however, the randomness introduced was insufficient, often leading to small update steps and poor convergence speed. Therefore, the Golden Ratio coefficient is introduced to enhance randomness and increase step length as follows:

$$x_{1,1}^1 = a_1^1 + \left(1 - \frac{\sqrt{5}-1}{2}\right) \times \left(b_1^1 - a_1^1\right) \tag{18}$$

Where $x_{1,1}^1$ represents the Golden Ratio coefficient 1 after its first iteration change, with $a_1^1 = -\pi$ and $b_1^1 = \pi$.

$$x_{1,i+1}^t = \begin{cases} a_i^t + \left(1 - \frac{\sqrt{5}-1}{2}\right) \times \left(b_{i+1}^t - a_i^t\right), \ if \ \ fitness_i^t < fitness_{best}^t \\ x_{2,i}^t, \ if \ \ fitness_i^t \geq fitness_{best}^t \\ a_{i+1}^t + \left(1 - \frac{\sqrt{5}-1}{2}\right) \times \left(b_{i+1}^t - a_{i+1}^t\right), \ if \ \ x_{1,i}^t = x_{2,i}^t \end{cases} \tag{19}$$

Where $x_{1,i+1}^t$ represents the coefficient of the Golden Ratio coefficient 1 after the $i+1$-th change in the t-th iteration, $x_{2,i}^t$ represents the coefficient of the Golden Ratio coefficient 2 after the i-th change in the t-th iteration, a_{i+1}^t represents the iteration change constant 1, and b_{i+1}^t represents the iteration change constant 2, as follows.

$$a_{i+1}^t = \begin{cases} x_{1,i}^t, \ if \ \ fitness_i^t \geq fitness_{best}^t \\ -\pi \times rand, \ if \ \ x_{1,i}^t = x_{2,i}^t \end{cases} \tag{20}$$

$$b_{i+1}^t = \begin{cases} x_{2,i}^t, \ if \ \ fitness_i^t < fitness_{best}^t \\ -\pi \times rand, \ if \ \ x_{1,i}^t = x_{2,i}^t \end{cases} \tag{21}$$

$$x_{2,1}^{1} = a_1^1 + \left(\frac{\sqrt{5}-1}{2}\right) \times \left(b_1^1 - a_1^1\right) \tag{22}$$

Where $x_{2,1}^{1}$ represents the coefficient of the Golden Ratio coefficient 2 after the first change in the first iteration.

$$x_{2,i+1}^{t} = \begin{cases} x_{1,i}^{t}, \ if \ \ fitness_i^t < fitness_{best}^t \\ a_{i+1}^t + \left(\frac{\sqrt{5}-1}{2}\right) \times \left(b_i^t - a_{i+1}^t\right), \ if \ \ fitness_i^t \geq fitness_{best}^t \\ a_{i+1}^t + \left(\frac{\sqrt{5}-1}{2}\right) \times \left(b_{i+1}^t - a_{i+1}^t\right), \ if \ \ x_{1,i}^t = x_{2,i}^t \end{cases} \tag{23}$$

Where $x_{2,i+1}^{t}$ represents the coefficient of the Golden Ratio coefficient 2 after the $i+1$-th change in the t-th iteration.

$$x_1^{t+1} = x_{1,N}^t \tag{24}$$

Where $x_1^1 = x_{1,1}^1$, that is, the coefficient of the Golden Ratio coefficient 1 in the first iteration is equal to the coefficient of the Golden Ratio coefficient 1 after the first change in the first iteration. x_1^{t+1} represents the coefficient of the Golden Ratio coefficient 1 in the $t+1$-th iteration, and $x_{1,N}^{t}$ represents the coefficient of the Golden Ratio coefficient 1 after the last change in the t-th iteration.

$$x_2^{t+1} = x_{2,N}^t \tag{25}$$

Where $x_2^1 = x_{2,1}^1$, that is, the coefficient of the Golden Ratio coefficient 2 in the first iteration is equal to the coefficient of the Golden Ratio coefficient 2 after the first change in the first iteration. x_2^{t+1} represents the coefficient of the Golden Ratio coefficient 2 in the $t+1$-th iteration, and $x_{2,N}^{t}$ represents the coefficient of the Golden Ratio coefficient 2 after the last change in the t-th iteration.

$$X_{i,j}^{t+1} = x_{2,i}^t \cdot X_{i,j}^t \tag{26}$$

3.5 Information accumulation function:

In the original algorithm, the update state of the sorted crayfish upon entering the burrow was not considered. An information accumulation function can be introduced to gradually accumulate self-information. On one hand, this can enhance the approach speed of the leading crayfish, while on the other hand, it can enhance the randomness of the trailing crayfish. The information accumulation function is utilized in the foraging phase of the original algorithm to gradually strengthen self-information attributes, as follows:

$$cos\left(\frac{\pi}{2} \times \left(1 - \left(\frac{3(i-1)}{2N}\right)\right)\right) \tag{27}$$

Application during the cooling phase of entering the burrow:

$$X_{i,j}^{t+1} = cos\left(\frac{\pi}{2} \times \left(1 - \left(\frac{3(i-1)}{2N}\right)\right)\right) \times \left(X_{i,j}^t + C_2 \times rand \times \left(X_{shade} - X_{i,j}^t\right)\right) \tag{28}$$

3.6 Icoa algorithm process:

The algorithm process is as follows:

Step 1: Input: Population size N, initial iteration $t=1$, maximum iteration count T, initial Golden Ratio coefficients 1 and 2, objective function, search space L_b, L_u.

Step 2: Randomly generate the initial population N_i, where $i=1,2,3,\ldots\ldots,N$, and calculate the fitness of the population.

Step 3: Sort based on fitness values, record the optimal individual and target value. Split according to the sorted sequence of crayfish, entering the position update phase for joiners or entering various phases of the original algorithm.

Step 4: Cooling phase and competition phase.

1) When $i \le N/3$, the temperature > 30°C, and $rand < 0.5$, ICOA enters the cooling phase. At this point, COA obtains a new position $X_{i,j}^{t+1}$ based on the cave location (X_{shade}) and the crayfish position ($X_{i,j}^{t}$), and then proceeds to Step 7.

2) When $i \le N/3$, temperature > 30°C, and $rand \ge 0.5$, ICOA enters the competition phase. According to the equation, these two crayfish will compete for the burrow, updating the position X_i^{t+1} based on the optimal position (X_{best}^{t}) using a spiral update. Then, proceed to Step 7.

Step 5: Foraging phase.

1) When $i \le N/3$ and temperature < 30°C, ICOA enters the foraging phase, defining food intake p, food size Q, and food judgment factor IQ.

2) If $IQ > 0.2$, then the food is ground before eating, and continue to Step 7.

Step 6: Joiner position update phase.

1) When $N/3 < i \le 2N/3$, ICOA performs the joiner position update phase, approaching significantly based on the optimal position (X_{best}^{t}), updating to get the new position X_i^{t+1}.

2) When $i > 2N/3$, ICOA performs the joiner position update phase, indicating that the seeker is in a very hungry state, randomly perturbing based on the worst position, and updating to get the new position X_i^{t+1}.

Step 7: Evaluate whether to stop the current generation cycle based on the population and determine whether to exit the cycle. If not, return to Step 3. Otherwise, the algorithm iteration ends, outputting the global optimal value result and the optimal path.

The specific flowchart is as follows, where the red dashed boxes indicate newly added modules, the red formulas represent new formulas, the orange formulas indicate changed original formulas, the green formulas represent additional operators based on the original formulas, and the black formulas indicate formulas that have not been changed.

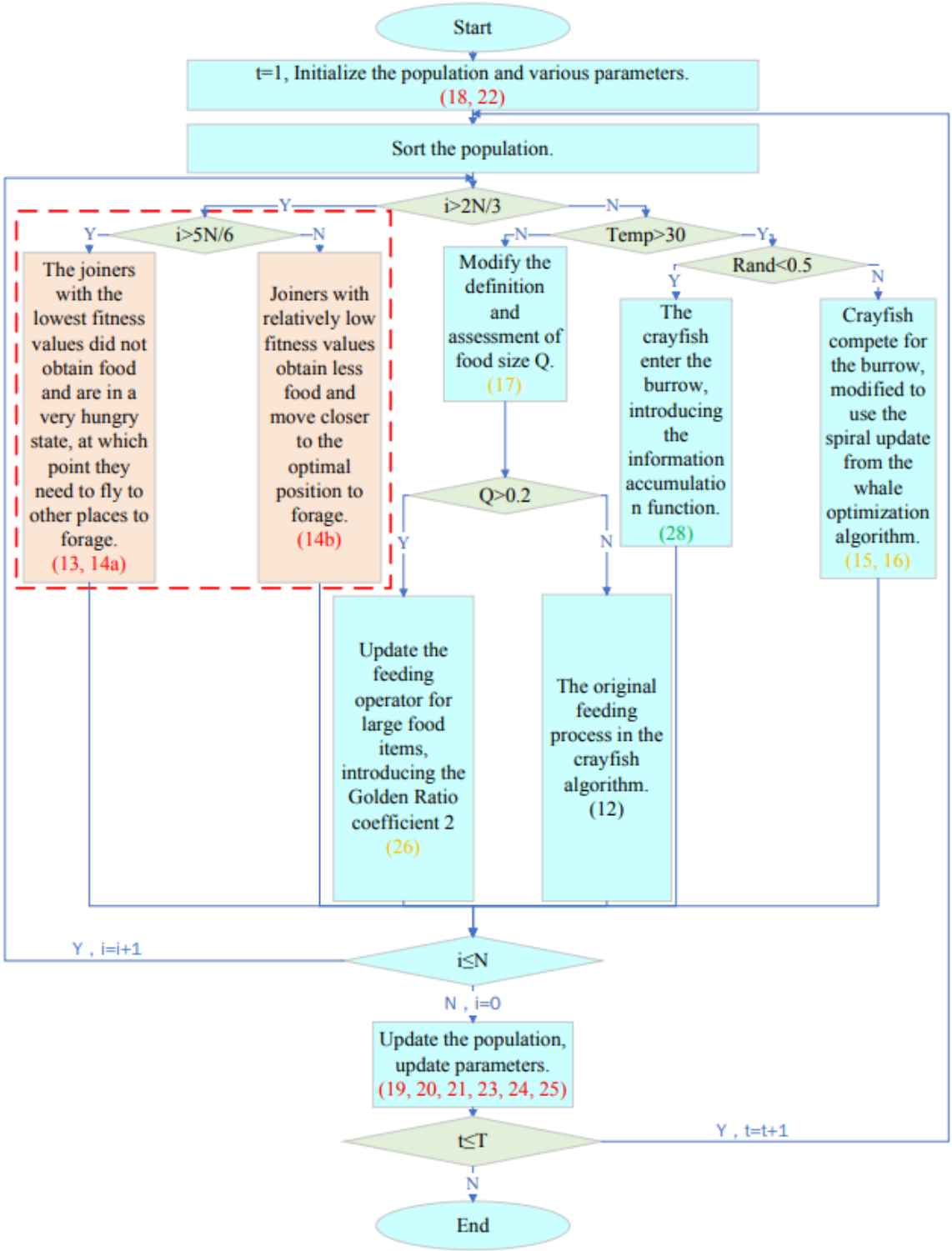


FIGURE 1: ICOA Algorithm Flowchart

IV. SIMULATION EXPERIMENTS AND RESULT ANALYSIS

4.1 Test functions and initialization parameters:

To verify the superiority of the ICOA algorithm compared to other optimization algorithms, this study selects 12 standard test functions that are widely researched to study ICOA. Among these, F1, F2, F3, F4, F6, F7, and F10 are unimodal functions with a single global optimal solution, used to validate the algorithm's local search capability. F5, F8, F9, and F11 are multimodal functions, mainly used to test the algorithm's ability to escape local optima. F12 is a composite function that can be used to test

the algorithm's potential to solve complex optimization problems in the real world. Table 1 presents the functions and their global optima. The results show that ICOA has a significant advantage compared to existing meta-heuristic algorithms and demonstrates good optimization effects and convergence performance. This study uses 12 standard test functions to validate the optimization capabilities of ICOA. This section can be divided into two parts. The first part presents the experimental results and analysis of the 12 standard reference functions. The second part involves comparisons with various classic algorithms. This study compares ICOA with nine popular and modern algorithms: Artificial Bee Colony (ABC), Butterfly Optimization Algorithm (BOA), Grasshopper Optimization Algorithm (GOA), Golden Sine Algorithm (GSA), Moth Flame Optimization Algorithm (MFO), Slime Mold Algorithm (SMA), Teaching-Learning-Based Optimization (TLBO), Whale Optimization Algorithm (WOA), and Crayfish Optimization Algorithm (COA). These algorithms have received considerable attention in swarm intelligence optimization, showing good performance and representativeness. Compared to these algorithms, ICOA can demonstrate its superiority.

This study validates the optimization performance of ICOA using the 12 standard test functions and conducts comparative experiments with nine algorithms. The expressions of the 12 standard test functions can be found in Table 2. The 3D view is shown in Fig 1. The distribution of ICOA across the 12 standard test functions is illustrated in Fig. 2.

TABLE 1
12 STANDARD TEST FUNCTIONS

Function formulation	Dim	Search range	$f_{\min}$
$f_1(x)=\sum_{i=1}^{D}x_i^2$	30	$[-100,100]^D$	0
$f_2(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	30	$[-10,10]^D$	0
$f_3(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2$	30	$[-100,100]^D$	0
$f_4(x)=max_i\left(\lvert x_i\rvert,1\le i\le D\right)$	30	$[-100,100]^D$	0
$f_5(x)=\sum_{i=1}^{D}\left(\lfloor x_i+0.5\rfloor\right)^2$	30	$[-100,100]^D$	0
$f_6(x)=\sum_{i=1}^{D}ix_i^4+random[0,1)$	30	$[-1.28,1.28]^D$	0
$f_7(x)=\sum_{i=1}^{D-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	30	$[-10,10]^D$	0
$f_8(x)=\sum_{i=1}^{D}\left(-x_i sin\left(\sqrt{\lvert x_i\rvert}\right)\right)+418.9829\times D$	30	$[-500,500]^D$	0
$f_9(x)=\sum_{i=1}^{D}\left[x_i^2-10cos\left(2\pi x_i\right)+10\right]$	30	$[-5.12,5.12]^D$	0
$f_{10}(x)=-20e^{\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)}-e^{\left(\frac{1}{D}\sum_{i=1}^{D}cos(2\pi x_i)\right)}+20+e$	30	$[-32,32]^D$	0
$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	$[-600,600]^D$	0
$f_{12}(x)=\frac{\pi}{D}\left\{10sin^2(\pi y_i)+\sum_{i=1}^{D-1}(y_i-1)^2\left[1+10sin^2(\pi y_{i+1})\right]+(y_D-1)^2\right\}+\sum_{i=1}^{D}u(x_i,10,100,4)$ $y_i=1+\frac{1}{4}(x_i+1),u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\le x_i\le a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	30	$[-50,50]^D$	0

To enhance the fairness of the comparative experiments, this study set the population size to $N = 50$, the maximum number of iterations to $T = 100$, and the dimensionality to $dim = 30$.

4.1.1 Analysis of statistical results for the 12 standard test functions:

Table 2 shows the statistical results obtained after independently running ICOA and the nine comparative algorithms 20 times. The bold parts in the table denote the best results among the 10 algorithms.

TABLE 2
ACCURACY COMPARISON RESULTS OF 10 ALGORITHMS IN 30 DIMENSIONS

Function	Metric	COA	ABC	BOA	GOA	GSA	MFO	SMA	TLBO	WOA	ICOA
$f_1(x)$	Mean	8.777E-151	6021.049	6.60495E-4	241.352	7.849E-58	64926.864	1.049E-54	1246.242	2.463E-1	**0**
	Std	2.633E-150	1399.963	4.525E-05	155.351	2.347E-57	7463.328	3.147E-54	460.114	3.866E-1	**0**
$f_2(x)$	Mean	1.407E-72	46.369	2.623E-2	11.529	4.176E-26	1336026430	5.705E-37	11.788	2.618E-2	**4.627E-284**
	Std	4.220E-72	7.597	7.196E-3	3.182	1.105E-25	2368723731	1.711E-36	2.840	0.016	**0**
$f_3(x)$	Mean	1.160E-167	48427.933	5.964E-4	3552.626	6.295E-52	95461.519	2.969E-33	8907.340	3160.286	**0**
	Std	0	4483.031	4.257E-05	1912.723	1.847E-51	13466.888	8.908E-33	3285.498	3552.179	**0**
$f_4(x)$	Mean	5.287E-85	78.178	1.829E-2	11.366	7.769E-29	85.614	1.476E-26	22.173	7.742	**0**
	Std	1.267E-84	3.960	1.979E-3	2.022	2.307E-28	3.210	4.427E-26	3.882	4.767	**0**
$f_5(x)$	Mean	**0**	6313.100	**0**	210.800	**0**	65373.300	**0**	1600.900	1.100	**0**
	Std	**0**	932.201	**0**	74.917	**0**	6060.215	**0**	694.481	1.578	**0**
$f_6(x)$	Mean	4.767E-4	5.735	**4.642E-4**	0.213	5.843E-4	110.699	8.330E-4	4.137E-1	1.912E-2	5.879E-4
	Std	**3.539E-4**	1.068	4.264E-4	7.636E-2	1.032E-3	25.631	8.754E-4	2.340E-1	8.046E-3	4.678E-4
$f_7(x)$	Mean	28.422	148560.303	28.907	1485.492	4.500E-2	2721211.944	28.695	3319.250	29.235	**2.906E-3**
	Std	3.344E-1	56395.206	2.437E-2	747.658	5.378E-2	386367.247	1.426E-1	2020.238	1.037	**7.017E-3**
$f_8(x)$	Mean	5737.629	8397.505	9224.488	5531.058	**7.904E-1**	8503.890	4660.388	6894.501	5916.009	903.815
	Std	926.593	377.341	319.899	593.694	**4.196E-1**	534.569	422.346	814.259	428.221	1461.886
$f_9(x)$	Mean	**0**	279.883	139.074	149.946	**0**	437.480	**0**	86.645	11.970	**0**
	Std	**0**	16.703	89.153	42.967	**0**	22.486	**0**	14.515	6.181	**0**
$f_{10}(x)$	Mean	**4.441E-16**	16.254	1.305E-2	5.206	**4.441E-16**	20.200	**4.441E-16**	9.594	2.466	**4.441E-16**
	Std	**0**	8.271E-1	1.046E-3	1.080	**0**	1.117E-1	**0**	1.010	5.839	**0**
$f_{11}(x)$	Mean	**0**	51.898	3.801E-3	2.053	**0**	552.423	**0**	12.529	1.965E-1	**0**
	Std	**0**	6.713	6.662E-4	5.224E-1	**0**	40.906	**0**	4.982	1.257E-1	**0**
$f_{12}(x)$	Mean	4.475E-2	135.906	5.418E-1	14.153	2.95597E-4	505.386	1.840E-1	10.471	2.107E-1	**2.150E-06**
	Std	2.287E-2	24.057	9.322E-2	5.230	5.946E-4	52.257	5.151E-2	3.596	1.917E-1	**5.901E-06**

According to the data in Table 2, the ICOA algorithm demonstrates significant superiority in terms of mean values. Specifically, for functions F1 to F5, the ICOA algorithm achieves the theoretical optimal value, while BOA, GSA, SMA, and the original COA only reach the theoretical optimal value for function F5. Additionally, the performance of ICOA in functions F1 to F4 far exceeds that of other algorithms, while the remaining algorithms do not come close to the optimal value for several functions in the above five function tests and even exhibit extremely high values. Not only that, but both COA and ICOA also have

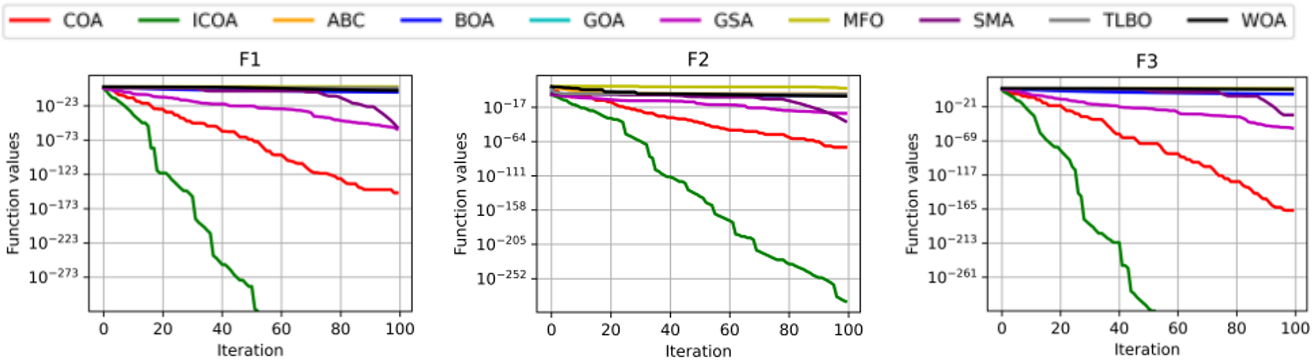
values that are far below those of the other algorithms for these five functions. In functions F7 and F12, the ICOA algorithm also outperforms the other nine algorithms, with results much closer to the optimal solution. It is noteworthy that in the test for function F7, COA fails to escape from local optima, while the improved ICOA algorithm successfully does so, enhancing its optimization results and bringing it closer to the optimal value. In functions F9 to F11, except for ICOA, COA, BOA, GSA, and SMA also reached the theoretical optimal value. However, for function F6, the ICOA algorithm did not outperform the other algorithms, even though its numerical results were very close to those of the best algorithms. In function F8, although ICOA was trapped in a local optimum, it still performed better than the other eight algorithms except for GSA.

In terms of standard deviation, Table 2 shows that the stability of the ICOA algorithm is very reliable. For functions F1 to F5, the ICOA algorithm achieved the theoretical optimal value in all 20 runs, with a standard deviation of 0, whereas BOA, GSA, SMA, and the original COA only exhibited stability in function F5. In functions F1 to F4, these four algorithms displayed some variability, while the other algorithms exhibited significant volatility and performed poorly. Notably, like in terms of mean values, COA and ICOA also have standard deviations that far outperform those of the other algorithms in the aforementioned five functions. In functions F7 and F12, ICOA demonstrated optimal stability in terms of standard deviation in addition to its mean performance. However, the COA algorithm did not show superiority compared to the other algorithms. In functions F9 to F11, COA, BOA, GSA, and SMA also reliably found the optimal solution, with no fluctuations observed over 20 runs, excluding ICOA. In function F6, although the stability of the ICOA algorithm did not reach optimal levels, its standard deviation was not significantly different from the optimal standard deviation, showing only a slight difference. In function F8, although ICOA exhibited strong volatility, this also indicated its ability to escape local optima, making it more likely to find the optimal solution compared to the other eight algorithms, aside from GSA.

In summary, the ICOA algorithm demonstrated clear superiority in terms of mean values, particularly for functions F1 to F5, where its performance far exceeded that of other algorithms, achieving the theoretical optimal value. In functions F7 and F12, the ICOA algorithm also outperformed the other nine algorithms, yielding results that were closer to the optimal solution. Furthermore, regarding standard deviation, the ICOA algorithm exhibited very reliable stability, demonstrating optimal stability in most functions. Although the ICOA algorithm's stability and mean performance did not reach the best levels for certain functions, the difference from the optimal standard deviation was minimal, and it showed the ability to escape local optima in function F8. Therefore, considering the performance of the ICOA algorithm across multiple functions, it is evident that ICOA possesses high optimization performance, demonstrating strong global search ability and good stability.

4.1.2 Analysis of convergence curves for 12 standard test functions:

Fig. 1 presents the convergence curves of the ICOA algorithm and nine other comparative algorithms on 12 standard benchmark functions. By comparing the plots, it can be observed that in functions F1 to F5 and F9 to F11, the ICOA algorithm demonstrates good convergence capability in a 30-dimensional space. Except for function F8, the ICOA algorithm exhibits the fastest convergence speed across the remaining 12 test functions, showcasing excellent convergence ability. Although the convergence speed of the ICOA algorithm in function F8 is slower than that of the GSA algorithm, it still surpasses that of the other eight algorithms. Compared to the original COA algorithm, the ICOA algorithm has made significant improvements in both the optimal values achieved and the speed of convergence. These results indicate that the ICOA algorithm exhibits outstanding convergence capabilities in multidimensional spaces, particularly excelling in high-dimensional scenarios.



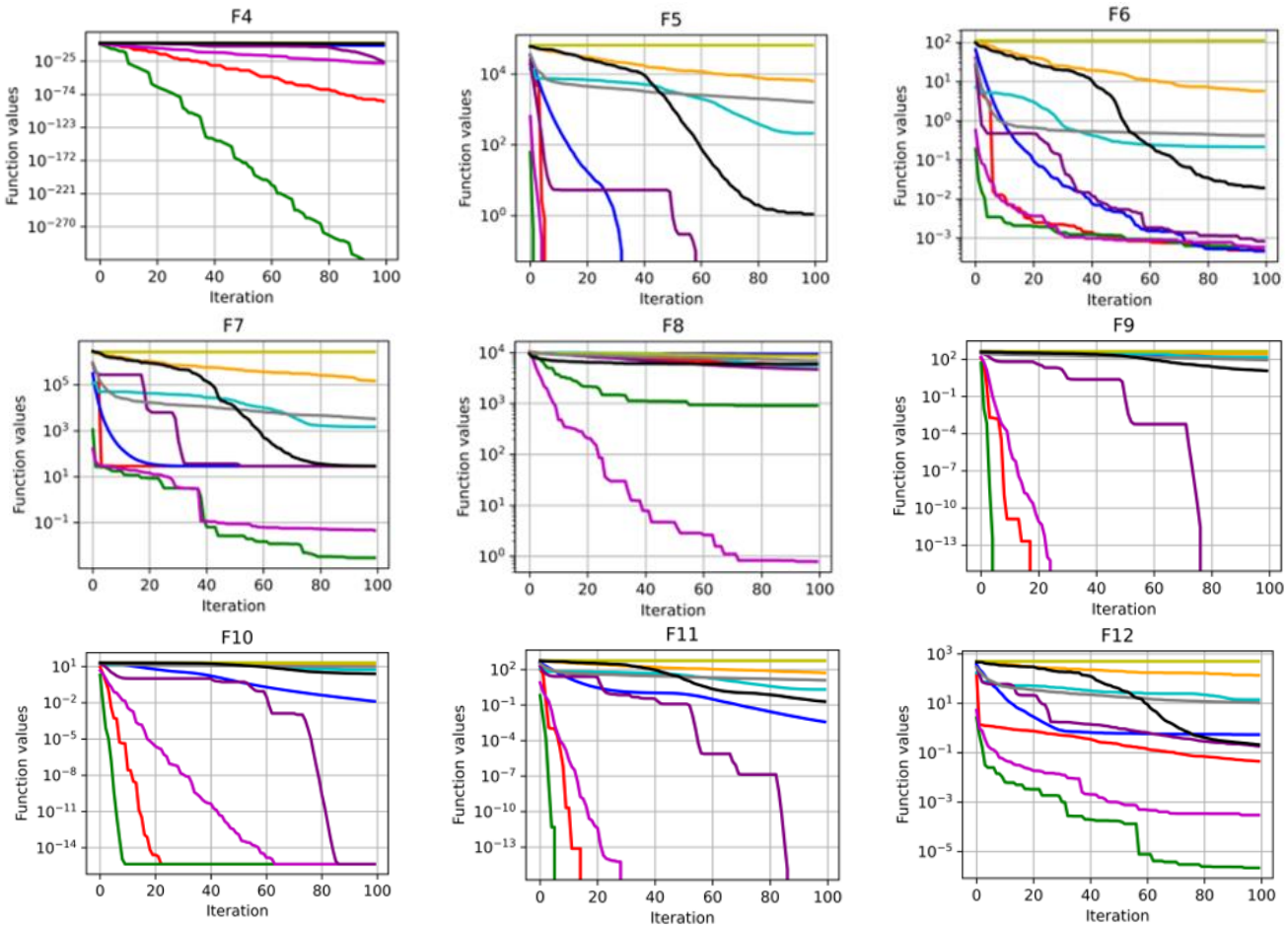


FIGURE 2. Convergence speed of 10 algorithms in 30 dimensions

4.2 Tension/compression spring design problem:

Pressure vessel design is a complex and important engineering field that involves several key issues and challenges. Here are some common research directions:

- **Structural Optimization:** This involves considering the mechanical properties of materials and the geometric shapes of the vessels to reduce weight or costs while meeting strength and stiffness requirements. The goal of structural optimization is to design the most efficient structure that satisfies the functional needs of the vessel while minimizing the use of materials.
- **Material Selection:** Material selection is crucial in pressure vessel design. Researchers need to comprehensively consider factors such as material strength, corrosion resistance, and heat resistance to choose the most suitable materials. The choice of different materials directly impacts the performance and reliability of the vessel.
- **Fatigue Life Prediction:** Given that vessels are subject to cyclic loads during use, fatigue life prediction is an essential research direction in this field. Researchers need to use relevant fatigue analysis methods to consider the fatigue strength of materials and the stress conditions of the vessel to predict its lifespan and take measures to extend its service life.
- **Safety Analysis:** Safety analysis is another critical research direction in pressure vessel design. Researchers must conduct comprehensive analyses and assessments of the vessel's stress state, stress distribution, and deformation to ensure safe operation under various working conditions and to detect and prevent potential safety hazards in a timely manner.
- **Application of Optimization Algorithms:** In recent years, the application of intelligent optimization algorithms in pressure vessel design has received widespread attention. By employing intelligent optimization algorithms such as

genetic algorithms and particle swarm optimization, researchers can more broadly explore design space and find more optimized solutions.

These are some common directions in the research of pressure vessel design, and researchers may choose more specific research topics based on particular needs and challenges. In actual research processes, it is important to integrate knowledge from multiple disciplines, including materials science, structural engineering, mechanics, and thermodynamics, to enhance the performance and reliability of pressure vessels.

The spring design problem addressed in this paper focuses on the tension/compression spring design, aiming to obtain the minimum spring mass under three variables and four constraint conditions. The schematic of the spring is shown in Fig. 3. The variables in the design problem include coil diameter d, mean coil diameter D, and the number of effective coils N. The constraint conditions include minimum deviation (g1), shear stress (g2), impact frequency (g3), and outer diameter limits (g4). By incorporating each variable into the constraint conditions, the minimum spring mass f(x) is derived.

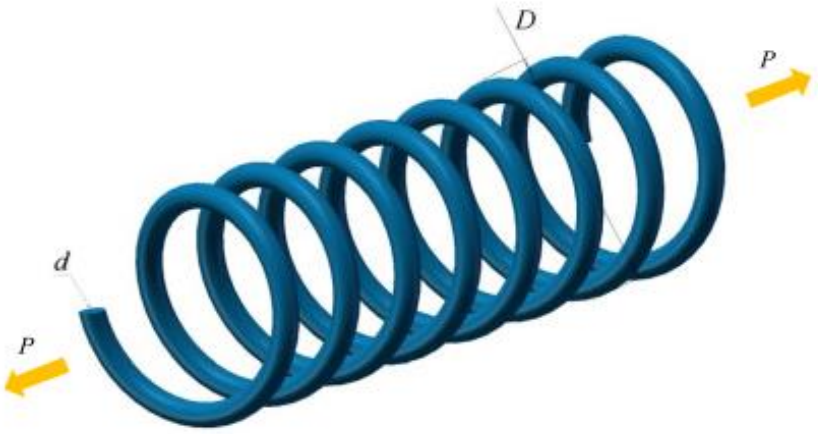


FIGURE 3: Schematic diagram of tension/compression spring design (a)

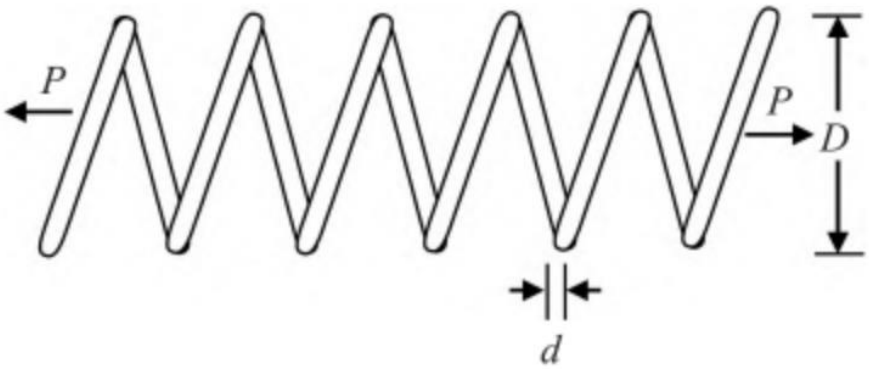


FIGURE 4: Schematic diagram of tension/compression spring design (b)

The mathematical model for this problem is as follows:

$$x=\left[x_1 \;\; x_2 \;\; x_3\right]=\left[d \;\; D \;\; N\right] \tag{29}$$

$$f\left(x\right)=\left(x_3+2\right)\times x_2\times x_1^{\,2} \tag{30}$$

$$g_1\left(x\right)=1-\frac{x_3\times x_2^3}{71785\times x_1^4}\le 0 \tag{31}$$

$$g_2\left(x\right)=\frac{4\times x_2^2-x_1\times x_2}{71785\times x_1^4}+\frac{1}{5108\times x_1^{\,2}}-1\le 0 \tag{32}$$

$$g_3(x) = 1 - \frac{140.45 \times x_1}{x_2^{\,2} \times x_3} \le 0 \tag{33}$$

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \le 0 \tag{34}$$

$$0.05 \le x_1 \le 2,\ 0.25 \le x_2 \le 1.3,\ 2.0 \le x_3 \le 15 \tag{35}$$

Table 3 shows the results of the ICOA and comparative algorithms in the tension/compression spring design problem. From the final data, it is evident that ICOA achieved the optimal value, while other algorithms did not perform as well as ICOA. Fig. 3 and 4 present comparative graphs of the optimal values among various algorithms. They also demonstrate ICOA's convergence speed and optimization capability, confirming that COA effectively addresses the tension/compression spring design problem.

TABLE 3
EXPERIMENTAL RESULTS OF TENSION/COMPRESSION SPRING DESIGN

Optimization Method	d	D	N	Optimal Value
COA	0.06233343	0.655097585	4.335902363	0.01612711
ABC	0.06036433	0.432369438	11.90176535	0.021902094
BOA	1.99279927	0.515235722	5.200026858	1.00E+33
GOA	0.05	0.25	2.35185811	1.00E+33
GSA	0.05	0.25	2	1.00E+33
MFO	0.05	0.315042555	15	0.013389309
SMA	0.05	0.25	2	1.00E+33
TLBO	0.05	0.310725658	15	0.01320584
WOA	0.05	0.31248667	14.73713678	0.01307533
ICOA	**0.05**	**0.3139422**	**14.58918**	**0.013020108**

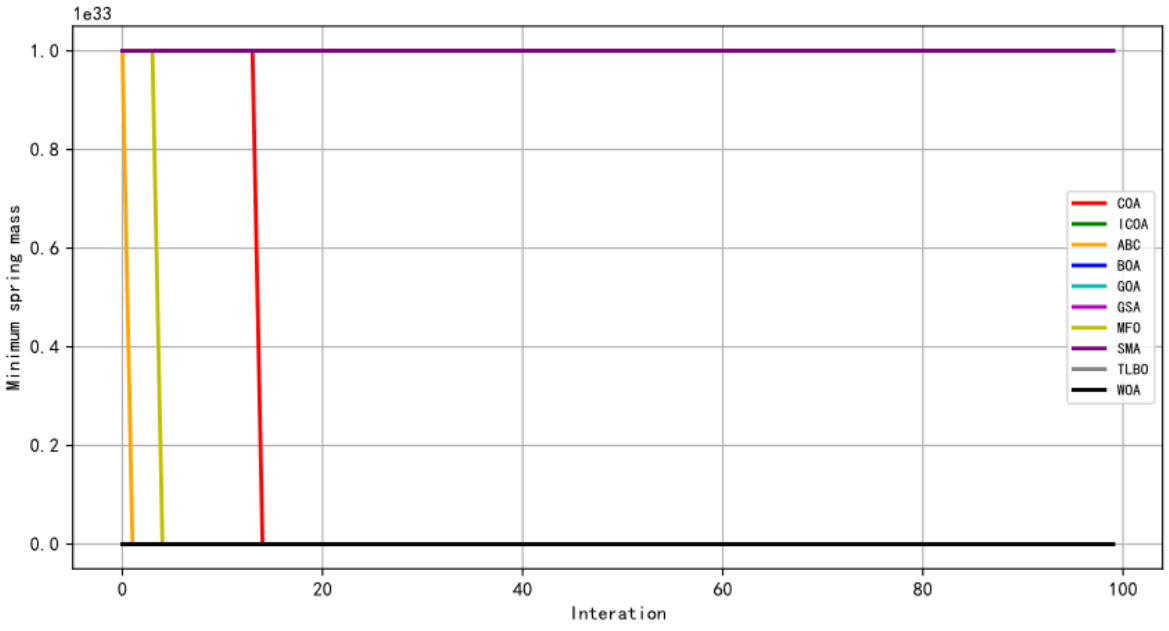


FIGURE 5: Pressure spring design (a)

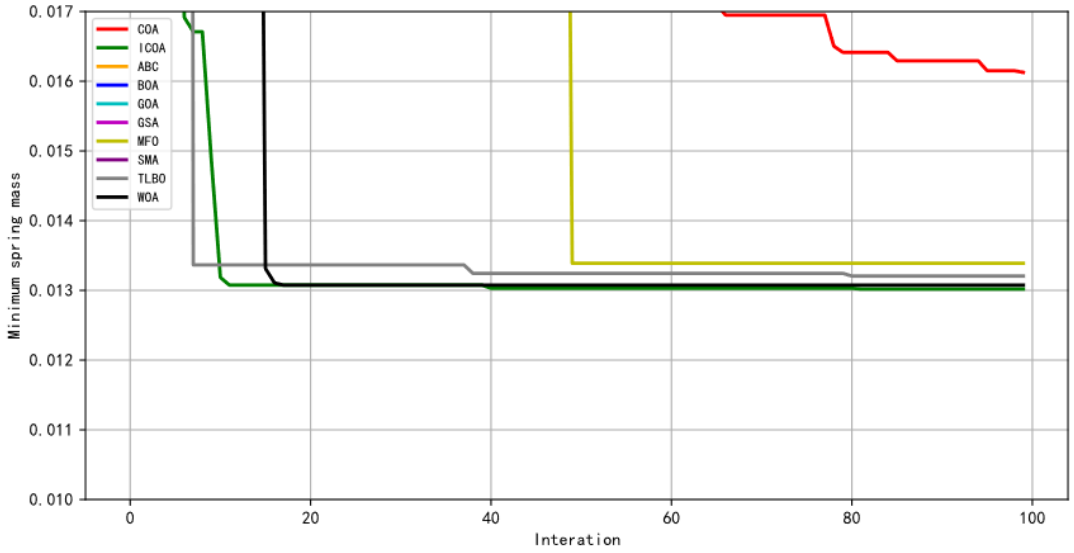


FIGURE 6. Pressure spring design (b)

V. ANALYSIS OF THE CHARACTERISTICS AND OPTIMIZATION ABILITY OF ICOA

5.1 Characteristics of ICOA:

- In ICOA, the algorithm divides the population through sorting and enters different phases based on temperature changes. This method balances the exploration and exploitation capabilities of ICOA more effectively.
- During the updating of participants, the reptiles follow either the optimal or the worst individuals. The update of participants represents the best solution, bringing each individual closer to the optimal or worst values.
- In the cooling phase, the reptiles approach the cave. The cave represents the best solution, enhancing the proximity of each individual to the optimal solution.
- In the competition phase, an improved spiral update position function from the whale optimization algorithm is introduced, enhancing the convergence of the algorithm and speeding up the convergence rate.
- In the foraging phase, the criteria for food size judgment are altered. While maintaining the randomness of the algorithm, the convergence of the algorithm is enhanced. When foraging for larger food sources, the update step size is increased to enhance randomness, and an information accumulation function is introduced to gradually aggregate individual information. This approach increases the speed at which the leading individuals converge and enhances the random abilities of the trailing individuals. The information accumulation function is also introduced when foraging for smaller food sources.

5.2 Optimization ability of ICOA:

Based on the above experimental results and analysis, it can be concluded that ICOA has good optimization performance on 12 standard benchmark functions and 1 engineering problem. In the convergence curves of some test functions, ICOA does not achieve a good purity value at the very beginning of iterations. However, during the iteration process, ICOA can continuously converge to obtain a better optimal solution. This is because the way in which the reptiles approach the cave in the cooling phase demonstrates the algorithm's strong convergence capability. In the participant and foraging phases, the reptiles approach food, improving the algorithm's exploitation ability. Many comparative algorithms tend to get trapped in local optima by the end of iterations, leading to convergence deviation. Although ICOA also experiences local optimum issues in later stages, its exploration and exploitation balance is enhanced by participant updates and temperature adjustments, allowing the algorithm to maintain good convergence capability and achieve satisfactory precision values in subsequent iterations.

VI. CONCLUSION

The improved crayfish optimization algorithm proposed in this paper balances its exploration and exploitation capabilities by filtering participants through sorting and controlling temperature, allowing ICOA to enter different phases. The cooling phase is the exploration phase of ICOA, while the participant and foraging phases represent the development stage of ICOA. The distinct characteristic of this algorithm lies in the exploration and development process based on participant updates and temperature control. During the exploration phase, when a cave is removed, the crayfish enter the cave to escape from high temperatures. The development phase is divided into the participant phase and the foraging phase. In the participant phase, ICOA further splits based on sorting, adopting different cruising strategies. In the foraging phase, ICOA adjusts food size criteria to decide whether to break it up, while foraging is controlled by the amount of food available, allowing for different strategies.

In the experimental section, this paper used 12 standard benchmark functions to verify the optimization effectiveness of ICOA and compared it with 9 different algorithms. The results indicate that ICOA performs well overall, but its exploration capability is somewhat weak, which leads to the algorithm being prone to local optima in later stages. Nevertheless, ICOA maintains good convergence capability in most test functions. After escaping local optima, ICOA can converge rapidly and yield a good solution. The comparative algorithms—ABC, BOA, GOA, GSA, MFO, SMA, TLBO, WOA, and COA—all yield good results in testing functions, but their lack of convergence capability prevents them from escaping local optima in later stages, resulting in lower fitness values. Computational results show that ICOA can achieve better sensitivity values in these engineering problems, indicating its effectiveness in solving practical issues.

This study has validated the effectiveness of ICOA in solving optimization problems using 12 standard benchmark functions and engineering problems. In future work, we will further enhance the exploration capability of ICOA and apply this algorithm to solve practical engineering problems such as three-dimensional path planning for drones, feature selection, image processing, and multi-threshold image segmentation.

REFERENCES

[1] Dantzig G B. Linear programming[J]. Operations research, 2002, 50(1): 42–47.

[2] Chen H, Chen L, Zhang G. Block-structured integer programming: Can we parameterize without the largest coefficient?[J]. Discrete Optimization, 2022, 46: 100743.

[3] Daryalal M, Bodur M, Luedtke J R. Lagrangian dual decision rules for multistage stochastic mixed-integer programming[J]. Operations Research, 2024, 72(2): 717-737.

[4] Gautier A, Granot F. On the equivalence of constrained and unconstrained flows[J]. Discrete applied mathematics, 1994, 55(2): 113-132.

[5] Ezugwu A E, Shukla A K, Nath R, et al. Metaheuristics: a comprehensive overview and classification along with bibliometric analysis[J]. Artificial Intelligence Review, 2021, 54: 4237-4316.

[6] Dréo J. Metaheuristics for hard optimization: methods and case studies[M]. Springer Science & Business Media, 2006.

[7] Holland J H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence[M]. MIT press, 1992.

[8] Karaboga D. Artificial bee colony algorithm[J]. scholarpedia, 2010, 5(3): 6915.

[9] Arora S, Singh S. Butterfly optimization algorithm: a novel approach for global optimization[J]. Soft computing, 2019, 23: 715-734.

[10] Meraihi Y, Gabis A B, Mirjalili S, et al. Grasshopper optimization algorithm: theory, variants, and applications[J]. Ieee Access, 2021, 9: 50001-50024.

[11] Tanyildizi E, Demir G. Golden sine algorithm: a novel math-inspired algorithm[J]. Advances in Electrical & Computer Engineering, 2017, 17(2).

[12] Mirjalili S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm[J]. Knowledge-based systems, 2015, 89: 228-249.

[13] Li S, Chen H, Wang M, et al. Slime mould algorithm: A new method for stochastic optimization[J]. Future generation computer systems, 2020, 111: 300-323.

[14] Rao R V, Savsani V J, Vakharia D P. Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems [J]. Information sciences, 2012, 183(1): 1-15.

[15] Mirjalili S, Lewis A. The whale optimization algorithm[J]. Advances in engineering software, 2016, 95: 51-67.

[16] Jia H, Rao H, Wen C, et al. Crayfish optimization algorithm[J]. Artificial Intelligence Review, 2023, 56(Suppl 2): 1919-1979.

[17] Mzili T, Riffi M E, Mzili I, et al. A novel discrete Rat swarm optimization (DRSO) algorithm for solving the traveling salesman problem[J]. Decision making: applications in management and engineering, 2022, 5(2): 287-299.

[18] Jia H, Sun K, Li Y, et al. Improved marine predators algorithm for feature selection and SVM optimization [J]. KSII Transactions on

Internet and Information Systems (TIIS), 2022, 16(4): 1128-1145.

[19] Jia H, Zhang W, Zheng R, et al. Ensemble mutation slime mould algorithm with restart mechanism for feature selection[J]. International Journal of Intelligent Systems, 2022, 37(3): 2335-2370.

[20] Mzili I, Mzili T, Riffi M E. Efficient routing optimization with discrete penguins search algorithm for MTSP [J]. Decision Making: Applications in Management and Engineering, 2023, 6(1): 730-743.

[21] Liu Q, Li N, Jia H, et al. Modified remora optimization algorithm for global optimization and multilevel thresholding image segmentation[J]. Mathematics, 2022, 10(7): 1014.

[22] Das M, Roy A, Maity S, et al. Solving fuzzy dynamic ship routing and scheduling problem through new genetic algorithm[J]. Decision Making: Applications in Management and Engineering, 2022, 5(2): 329-361.

[23] Qi H, Zhang G, Jia H, et al. A hybrid equilibrium optimizer algorithm for multi-level image segmentation[J]. Math Biosci Eng, 2021, 18(4): 4648-4678.

[24] Wen C, Jia H, Wu D, et al. Modified remora optimization algorithm with multistrategies for global optimization problem[J]. Mathematics, 2022, 10(19): 3604.

[25] Wolpert D H, Macready W G. No free lunch theorems for optimization[J]. IEEE transactions on evolutionary computation, 1997, 1(1): 67-82.