Chaos Gray Wolf global optimization algorithm based on Opposition- based Learning

Zhiyong Luo
Chongqing University of Posts and Telecommunications
Mingxiang Tan
Chongqing University of Posts and Telecommunications
Zhengwen Huang
Brunel University London
Guoquan Li (ligq@cqupt.edu.cn)
Chongqing University of Posts and Telecommunications

Research Article

Keywords: Gray wolf optimizer, Opposition-based Learning, tent chaotic map, Polynomial decay function

Posted Date: December 3rd, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2327934/v1

Additional Declarations: No competing interests reported.

Chaos Gray Wolf global optimization algorithm based on Opposition-based Learning

[1]Zhiyong Luo ,[1]Mingxiang Tan ,[2]Zhengwen Huang,[3]Guoquan Li
[1]School of Advanced Manufacturing Engineering,Chongqing University of Posts and Telecommunications, Chongqing, 400065, China
[2]Department of Electronic and Electrical Engineering, Brunel University London, London UB8 3PH, UK
[3]School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing, 400065, China

luozy@cqupt.edu.cn, s202131002@stu.cqupt.edu.cn, Zhengwen.Huang@brunel.ac.uk, ligq@cqupt.edu.cn

Abstract: Gray wolf optimizer (GWO) is a new heuristic algorithm. It has few parameters and strong optimization ability and is used in many fields. However, when solving complex and multimodal functions, it is also easy to trap into the local optimum and premature convergence. In order to boost the performance of GWO, a tent chaotic map and opposition-based learning Grey Wolf Optimizer (CO-GWO) is proposed. Firstly, some better values of the population in the current generation are retained to avoid deterioration in the next generation. Secondly, tent chaotic map and opposition-based（OBL）are introduced to generate values that can traverse the whole feasible region as much as possible, which is conducive to jumping out of local optimization and accelerating convergence. Then, the coefficient $\vec{a}$ is dynamically adjusted by the polynomial attenuation function of the 2-decay method. Finally, the proposed algorithm is tested on 23 benchmark functions. The results show that the proposed algorithm is superior to the conventional heuristic algorithms, GWO and its variants in search-efficiency, solution accuracy and convergence rate.

Keywords*: Gray wolf optimizer, Opposition-based Learning, tent chaotic map, Polynomial decay function*

1. Introduction

Bionics was founded in the mid-1950s. Many scientists seek new inspiration from biology for optimizing systems. The simulated evolutionary algorithm, which is suitable for the optimization of complex problems in the real world, has been developed by some scientists from the mechanism of biological evolution. For example, simulated annealing (SA)[1] was first used to optimize combinatorial problems by Kirkprtricrk *et al*. It overcomes the shortcoming that the hill-climbing (HC) method is very easy to fall into local solutions. In recent years, the main development direction of SA is to combine with other algorithms to form new hybrid algorithms to take full advantage of jumping and avoiding local solutions. Ant Colony Algorithm (ACA) was first proposed by Italian scholar Dirgo *et al*[2]. It is a new simulated evolutionary algorithm. The method is also used to

solve the Traveling Salesman Problem (TSP), Assignment Problem, and Scheduling Problem, and a series of better experimental results are obtained. Influenced by it, ACA gradually attracted the attention of other researchers and used the algorithm to solve some practical problems. Later, more and more biomimetic algorithms were proposed and used in more fields.

Heuristic algorithms are becoming more and more mature, and it is an algorithm with global optimization performance and strong versatility that is suitable for parallel processing. It is used in engineering, aerospace, medicine, management, economics and many other fields. Heuristic algorithms are inspired by human intelligence, the social nature of biological groups and the laws of natural phenomena, and can be divided into evolutionary algorithms and group intelligence algorithms. Including Genetic Algorithm (GA)[3], Differential Evolution algorithm (DE)[4], Immune Algorithm (IA)[5], Ant Colony Optimization algorithm (ACO)[6], Particle Swarm Optimization (PSO)[7], Artificial Bee Colony (ABC)[8], Firefly Algorithm (FA)[9], Bat Algorithm[10], Grey Wolf Optimization (GWO)[11], Shuffled frog-leaping algorithm(SFLA)[12] , etc. According to different search capabilities, heuristic algorithms are divided into global search and local search . Therefore, different heuristic algorithms should be used for different problems. The optimal value or approximate optimal value can be found by a heuristic algorithm in a certain period of time. Optimization problems can be divided into two categories: (1) Solving the optimal problem of a function. (2) A combinatorial optimization problem with optimal objective function value in a solution space. Typical combinatorial optimization problems include the Traveling Salesman Problem (TSP), Scheduling Problem, Knapsack Problem, Bin Packing Problem, etc.

Most search algorithms have problems such as falling into local optimal and slow convergence speeds when dealing with complex problems.For example, the GWO search method is simple and difficult to deal with complex issues.The GA search method is complicated and the search time is long. So this paper proposes an CO-GWO. The main contributions of this paper are as follows:

- Increase the probability of the population spreading over the feasible region, which is conducive to quickly finding the optimal value and accelerating the convergence speed.
- the gray wolf can exploration as much as possible in the early stage to avoid falling into local optimization.
- Through comparative experiments with DE, PSO, ABC, etc, it is proved that CO-GWO performs well in solving simple and complex problems.

The rest of the paper is organized as follows:

Section 2 presents a literature review of GWO. Section 3 gives an overview of GWO, OBL and tent map. Section 4 outlines the proposed GWO algorithm. The results and discussion of benchmark functions are in Section 5. Finally, Section 6 concludes the work.

2. Literature review and Related work

2.1 Literature review

GWO was developed based on the social class and hunting mechanism of the gray wolf. It has been successfully applied to solve the NP-hard problem, such as economic scheduling issues[13], time forcases[14]. Like traditional heuristic algorithms, when GWO is used to optimize complex

multimodal functions, its performance will be worse and it is easy to fall into local optimization. And its performance for some unimodal functions is not as good as other conventional heuristic algorithms. Therefore, in order to improve the performance, many improvement methods had been proposed. According to the references, it is divided into the following four types:

1) Improve population

The population generated by the iteration affects the current optimal value and the convergence speed. OBL is introduced into GWO for the initial population and the iteratively generated population to generate opposing populations so that the generated values are as close to the optimal values as possible[15]. For high-dimensional space, if each dimension is opposed, it may generate bad values, so selective use of OBL can generate better values [16]. In addition, chaotic mapping enables the population to be uniformly distributed, laying the foundation for a diverse global search[17].

2) Improve the balance between exploration and exploitation

In the conventional GWO, the parameter $\vec{a}$ is a crucial coefficient, which implements the process from exploration to exploitation. When the parameter $\vec{a}$ decay rate is slow, the proportion of gray wolf exploration will be greater than the development proportion, which is conducive to global search. On the contrary, it is conducive to local search. In order to balance exploration and exploitation, an exponential function is used to decay $\vec{a}$ throughout iterations, exploitation and exploration are 30% and 70%, respectively[18]. To balance exploitation and exploration more flexibly, an adaptive-based nonlinear function is used to adjust the parameters $\vec{a}$ [15][19][20]. These improvements perform well to some extent, but will still fall into local optimization, so other improvements are needed.

3) Improve the position-updated strategy

Improved position updating has a positive effect on the performance of the algorithm. Various improvement methods have been proposed. In the memory-based Grey Wolf Optimizer (mGWO), the search mechanism of the wolves is modified based on the personal best history of each wolf, crossover and greedy selection [21]. In the GWO based on the extended model (E-GWO), the next wolves select and update their positions according to the previous and the first three wolves in each iteration [22]. In the GWO based on the incremental model (I-GWO), each wolf updates its position based on all the wolves selected before it [22]. In the GWO based on the Group-based Synchronous-Asynchronous, the method incorporates a synchronous-asynchronous processing scheme, a set of different nonlinear functions and an operation to increase diversity [23].

4) Hybrid algorithms

In addition, the mixed-use of the heuristic optimization algorithm is an important means to improve the algorithm. The improvement of this method will often play to the strengths of their respective algorithms to avoid shortcomings. For example, mixing PSO with GWO[24][25], DE and GWO mixing [26], GA and GWO hybrid[27], *etc*.

2.2. Related Work

2.2.1 Tent map

In the heuristic optimization algorithm, the randomness and uniformity of the initial

population are very important to the optimal solution. When the initial population spreads over the whole feasible region, the generated initial population likely contains the optimal solution or the value near the optimal solution, which improves the convergence speed and the accuracy of the optimal value. Meanwhile, chaotic maps are used to generate chaotic sequences, which are nonlinear, ergodic, random, and sensitive to the initial value. Therefore, it can be used as an alternative to pseudo-random number generators in the field of optimization, and often achieves better results than pseudo-random numbers [17]。

Reference [28] states that tent map has better ergodicity, regularity and faster speed than a logistic map. And it is also proved by rigorous mathematical reasoning that the tent map has a prerequisite for optimizing the chaotic sequence of algorithms. Tent map is a piecewise linear mapping, named for its shape like a tent. Moreover, it is a 2D chaotic mapping, which is widely used in chaotic encryption systems.

In this paper, tent mapping is used to generate the initial population. The Tent map function is given by Eq (2.1)：

$$x_{n+1} = \begin{cases} \dfrac{x_n}{a}, & 0 \le x_n < a \\ \dfrac{(1-x_n)}{1-a}, & \mathrm{a} \le x_n \le 1 \end{cases} \qquad (2.1)$$

where x_n is the value of the variable x in the nth iteration, n is the number of the iterative step, and a is the system parameter and takes 0.5 generally. In addition, $x_{n+1} \in [0,1]$.

2.2.2 OBL

OBL is to generate an opposition population based on the current population, which helps to jump out of the local optimum and improve the probability of finding the global optimum. In heuristic algorithms, the initial population is generally generated randomly. If the randomly generated initial population is far from the optimal value, the convergence is slow and may fall into the local optimal value. However, the position of the opposing population generated by the OBL is the opposite position of the original population, so that the population is more evenly distributed in the feasible region, it will be beneficial to the global search and accelerate the convergence speed. The opposition value is generated by Eq. (2.2) as follows[15].

$$\hat{x} = a + b - x \qquad (2.2)$$

Where x is the original value, $\hat{x}$ is the opposition value, a and b are the upper and lower bounds of x, respectively.

Of course, for a D-dimensional search space, $i \in (1, D)$, where OBL is also used to generate opposing populations, there is an Eq. (2.3)

$$\hat{x}_i = a_i + b_i - x_i \qquad (2.3)$$

Where $\hat{x}_i$ is the opposition value of x_i in the ith dimension, a_i and b_i are the upper and lower bounds of the ith dimension respectively.

2.2.3 Polynomial decay function based on the 2-Decay method

Polynomial decay function is a commonly used learning rate decay method. Its general form is given by the following Eq. (2.4) as follows[29]：

$$L(t)=(L_0-L_e)\left(1-\frac{t}{T}\right)^N+L_e \tag{2.4}$$

where L_0 is the initial learning rate, L_e is the termination learning rate, t is the number of current iterations, T is the total number of iterations, and N is the exponent of the decay function($N\neq 0; N\in$). When $N>1$, the overall trend of changes in the learning rate is steep first and then flat. When $N<1$, the overall trend of changes in the learning rate is gentle first and then steep, which means that the learning rate in the early stage has not changed much. When $N=1$, a constant rate of change is maintained.

When $N<1$, in order to increase the decay rate of its early learning rate, there is a polynomial decay function based on the 2-decay method[29]. Its general form is given by the following Eq. (2.5)：

$$L'(t)=(L_0-L_e)\left(1-\frac{t}{T}\right)^N+L_e+\frac{\omega}{2}(t^2-Tt) \tag{2.5}$$

Where ω is the amount of translation, in order to meet the decay condition ω should be taken within a certain range, otherwise it will exceed the upper and lower bounds L_0 and L_e. For example, $\omega=4-e^7$, $L_0=2$, $L_e=0$, $T=100$, $N=0.3$, the attenuation trend is shown in Fig.1 below. It can be seen from the figure that the learning rate of the first 900 times decays very slowly, and that of the next 100 times decays very fast.

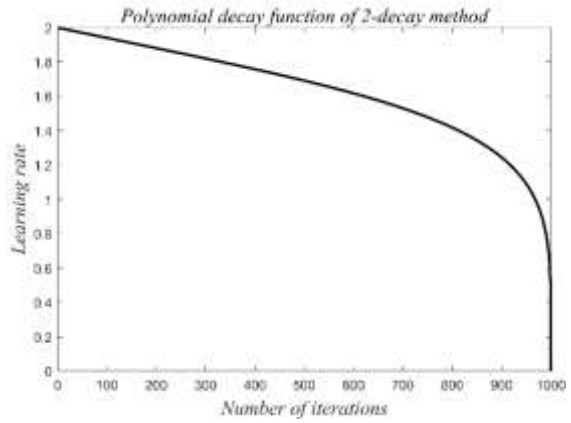


Fig.1 Polynomial decay function trend chart of the 2-decay method

2.2.4Grey Wolf Optimizer

GWO is a heuristic optimization algorithm proposed by Seyedali Mirjalili *et al*. in 2014 [11]. The core idea of the algorithm is the behavior of gray wolves in searching and capturing prey. According to the behavior of gray wolves in hunting, gray wolves are divided into four

categories:alpha wolf (α), beta wolf (β), delta wolf (δ) and omega wolf (ω). α is the head wolf of the whole wolf pack, with the best position advantage. β and δ are the followers of α, and the location advantages are second and third, respectively. ω is the worst position.

During the capturing prey , the gray wolf knows the specific location of the prey and surrounds the prey . The following Eqs. (2.6)-(2.10) will describe this process：

$$\vec{D} = \left| \vec{C} * \vec{X}_P(t) - \vec{X}(t) \right| \quad (2.6)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} * \vec{D} \quad (2.7)$$

$$\vec{A} = 2\vec{a} * \vec{r}_1 - \vec{a} \quad (2.8)$$

$$\vec{C} = 2 * \vec{r}_2 \quad (2.9)$$

where t indicates the current iteration, $\vec{x}$ is the position vector of a grey wolf and $\vec{X}_P$ indicates the position vector of the prey, $\vec{A}$ and $\vec{C}$ are two coefficient vectors, both $\vec{r}_1$ and $\vec{r}_2$ are random vectors in [0,1], the coefficient $\vec{a}$ is decreased from 2 to 0 in Eq(2.10).

$$\vec{a} = 2(1 - \frac{t}{Iter_{\max}}) \quad (2.10)$$

Where t is the current iteration, $Iter_{\max}$ is the total number of iterations.

When the prey position is known, the gray wolf $\vec{X}(t)$ is based on the equivalent (2.6) and (2.7) to update its position according to the target prey $\vec{X}_P$. Different places around the best agent can be reached with respect to the current position by adjusting the value of $\vec{A}$ and $\vec{C}$ vectors according to equations (2.8) and (2.9). As the position of the gray wolf is updated, $\vec{A}$ and $\vec{C}$ gradually become smaller, making the position of the final gray wolf $\vec{X}(t+1)$ gradually close to the target prey $\vec{X}_P$.

When searching for prey, the location of the prey is unknown. The position of ω in the pack will update the position according to the three elite wolves α, β and δ, making the updated position better than the current position. The three elite wolves are the best three wolves after each position update, so α, β and δ are subject to change. The update method is shown in the following Eqs. (2.11)-(2.13).

$$\vec{D}_\alpha = \left| \vec{C}_1 * \vec{X}_\alpha - \vec{X} \right|, \vec{D}_\beta = \left| \vec{C}_2 * \vec{X}_\beta - \vec{X} \right|, \vec{D}_\delta = \left| \vec{C}_3 * \vec{X}_\delta - \vec{X} \right| \quad (2.11)$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 * (\vec{D}_\alpha), \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 * (\vec{D}_\beta), \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 * (\vec{D}_\delta) \quad (2.12)$$

$$\vec{X}(\mathrm{t}+1)=\frac{\vec{X}_1+\vec{X}_2+\vec{X}_3}{3} \qquad (2.13)$$

where $[\vec{C}_1,\vec{C}_2,\vec{C}_3]$ and $[\vec{A}_1,\vec{A}_2,\vec{A}_3]$ are three coefficient vectors respectively, $\vec{X}_\alpha$, $\vec{X}_\beta$, $\vec{X}_\delta$ are positions of the alpha wolf (α), beta wolf (β), and delta wolf (δ) in the search space. $\vec{X}$ is the position vector of a current grey wolf. $\vec{X}(\mathrm{t}+1)$ is the updated position vector.

The pseudo code of the GWO algorithm is presented in Fig.2.

Initialize the grey wolf population $X_i(i=1,2,3,...,n)$ *and initialize* a, A *and* C
Calculate the fitness of each search agent and find X_α, X_β and X_δ
X_α *=the best search agent*
X_β *=the second best search agent*
X_δ *=the third best search agent*
while (t<Iter$_{max}$*)*
  for ((i=0; i<N_1*+*N_2*+*N_3*; i++)*
    Update the position of the current search agent by equation (2.11), (2.12) and (2.13)
  end
  Update a, A, *and* C
  Calculate the fitness of all search agentsUpdate X_α, X_β *and* X_δ,
  $t=t+1$
end
return X_α

Fig.2. Pseudo code of the GWO algorithm

3 The proposed algorithm

In this paper, CO-GWO is proposed, which improves the population generation mode and the balance between exploitation and exploration. The population is divided into two parts. Part A is to keep the better values of the current population without any treatment, and directly use it for subsequent iterative updates. Part B is generated by the tent map and OBL. And the two parts add up to a population size equal to the initial population. Then the polynomial decay function of the 2-decay method is used to update the parameter $\vec{a}$ to balance the relationship between development and exploration.

3.1 The population generation mode improvement

For the conventional GWO, the initial population $X_i=\{x_{i1},x_{i2},...,x_{ij},...,x_{iD}\}$, $i\in(1,N)$, $j\in(1,D)$ is randomly generated in the search space, where N is the population size, D is

the dimension of the problem. Here, we divide the population into two parts, as follows:

1) Part A：

The population fitness values are sorted, and most of the better individuals of the population are preserved. Here, the number of better values to be retained is determined according to Eqs. (3.1) and (3.2), so that better and better values are retained during the iteration process. Part one of the population is represented as P_1, and its population number is N_1.

$$N_1 = round\left(N * \partial\right) \tag{3.1}$$

$$\partial = \partial_0 + \frac{t}{T}\left(1 - \partial_0\right) \tag{3.2}$$

where N in Eq. (3.1) denotes the population size, ∂ denotes the ratio of the number of retained better values to the population size, ∂_0 in Eq. (3.2) is the initial percentage of retained better values to the population, t is the current iteration, T is the total number of iterations, and $round$ denotes rounding to the nearest whole number.

2) Part B：

A value X_0 is randomly selected as the initial value of the tent map in the population P1. And the tent map is performed using Eq. (3.3), where the number of populations generated by chaos is N_2. N_2 will decreases gradually with the iteration, so that the values generated in the later stage will no longer diverge. And the later population is composed almost entirely of the best individuals of each iteration, making the value of fitness gradually approach the optimal value, which is conducive to convergence to the optimal value at a later stage, as calculated in Eq. (3.4). Then the opposite population of the mapped population is generated by Eq. (2.3), and N_3 opposite values are obtained, where $N_2 = N_3$. Therefore, the size of the population is $P_2 = N_2 + N_3$.

$$x_{ij}^{p+1} = \begin{cases} \dfrac{x_{ij}^{p}}{a}, & 0 \le x_{ij}^{p} < a \\ \dfrac{\left(1 - x_{ij}^{p}\right)}{1 - a}, & \mathrm{a} \le x_{ij}^{p} \le 1 \end{cases} \tag{3.3}$$

$$N_2 = \frac{1 - N_1}{2} \tag{3.4}$$

In Eq. (3.3), $p \in [1, N_2]$. In addition, to avoid scale-inconsistent data leading to instability of the algorithm, the initial Tent map value by selected randomly needs to be normalized to [0,1] using eq (3.5), $x_{ij}^{p+1} \in [0,1]$. And the mapped value needs to be inversely normalized to the upper and lower bounds of x_{ij} by Eq. (3.6).

$$Y_{ij}^{p+1} = \frac{x_{ij}^{p+1} - a_j}{\left|a_j - b_j\right|} \tag{3.5}$$

$$X_{ij}^{p+1} = Y_{ij}^{p+1} * \left|a_j - b_j\right| + a_j \tag{3.6}$$

where a_j and b_j are the lower and upper bounds of x_{ij} in the jth dimension respectively.

Finally, the complete population $P = [P1, P2]$ is used for later iterations.

3.2 Improve the balance between exploration and exploitation

In GWO, the value of $\vec{A}$ determines the exploration and exploitation behavior of the Gray Wolf. When $\left|\vec{A}\right| \geq 1$, the gray wolf is committed to exploration to conduct a global search, which easily jumps out of the local optimum. When $\left|\vec{A}\right| < 1$, the gray wolf is committed to exploitation to conduct a local search to find the optimum at the current location. Therefore, in order to increase the number of explorations in the early stage, the polynomial decay function of the 2-decay method is used to update parameter $\vec{a}$ following Eq. (3.7). And then the parameter $\vec{A}$ is controlled according to Eq. (2.8), so that there are enough exploration times in the early stage of the algorithm. This method is conducive to a comprehensive search in the whole feasible region to avoid getting trapped in local optimum and locating the optimal value more accurately.

$$\vec{a} = \left(B_0 - B_e\right)\left(1 - \frac{t}{T}\right)^N + B_e + \frac{\omega}{2}\left(t^2 - Tt\right) \tag{3.7}$$

where B_0 is the upper bound of $\vec{a}$, B_e is the lower bound of $\vec{a}$, t is the number of current iterations, T is the total number of iterations, and N is the exponent of the decay function($N \neq 0; N \in$).Where ω is the amount of translation, in order to meet the decay condition ω should be taken within a certain range, otherwise it will exceed the upper and lower bounds B_0 and B_e.

3.3 Overall process of CO-GWO

Based on the above analysis, the pseudo code of the proposed CO-GWO can be presented as follows in Fig.3 and the main flowchart of the proposed algorithm is demonstrated in Fig.4.

Initialize the population randomly P0 in the search space and initialize parameters；
While*(t<Iter$_{max}$)*
Calculate fitness values and arrange them
P1:N_1 better value in P0 is retained By Eqs. (3.1) and (3.2)
Randomly select an individual x_0 in P1 as the initial value of tent map
Generate N_2 chaotic populations by Eq. (3.3)
Generate the N_3 opposite population of chaotic populations by Eq. (2.3)
Combine N_2 and N_3 into P2

Combine P1 and P2 as the current population P

Find the X_α *= the best search agent*

Find the X_β *= the second best search agent*

Find the X_δ *= the third best search agent*

Update $\vec{a}$ *according to Eq. (3.7)*

Update $\vec{A}$ *and* $\vec{C}$ *according to Eqs. (2.8) and (2.9)*

For *(i=0; i<N_1+N_2+N_3; i++)*

Update the position of wolves according to Eqs. (2.11), (2.12) and (2.13)

end

return X_α

t = t + 1;

end

return X_α

Fig.3 pseudo code of theCO- GWO algorithm

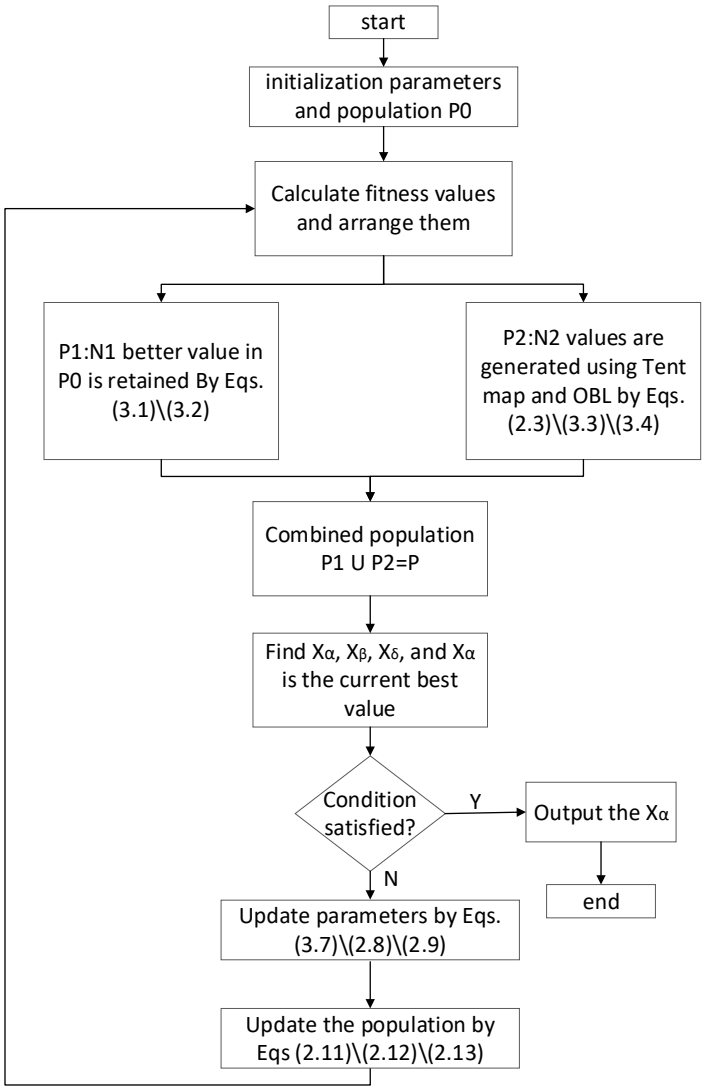


Fig.4 The main flowchart of CO-GWO

After calculating fitness and sorting, CO-GWO retains the population P1 corresponding to the better fitness according to Section 3.1 of the article, uses tent chaotic mapping and OBL to generate population P2, and combines population P1 and P2 to obtain population P. Find the best three values $X_\alpha, X_\beta, X_\delta$ of the current generation in the population P. Use part 3.2 of the article to control parameter $\vec{a}$, and update the population according to part 2.2.4 of the article until the conditions are met to output the optimal value. The improved algorithm enables the population to traverse the entire feasible region as much as possible without changing the bad condition, and at the same time increases the gray wolf's exploration opportunities, which is conducive to jumping out of the local optimum and speeding up the search and convergence speed.

4. Tests and analysis

In this section, the proposed algorithm is tested on 23 benchmark functions. Although the function is simple, it can compare our results. Table 1 shows unimodal benchmark functions, Table

2 shows multimodal benchmark functions, and Table 3 shows fixed-dimension multimodal benchmark functions. Where Dim denotes the dimension, $Range$ denotes the range of values, and $f_{\min}$ denotes the minimum value of the function. In addition, we also choose some classical heuristic optimization algorithms for comparisons, such as DE, PSO, ABC, GSA, and Whale Algorithm (WOA)[30]. Meanwhile, others improved GWO are also compared, for example, GWO[11], OGWO[15] and IGWO[17].

The population size of each algorithm is 50 and the number of iterations is 1000. In order to make the comparison more accurate, each algorithm is run 100 times to calculate the average value and variance and then compared. The parameters of each algorithm are shown in Table 4.

In addition, let coefficients of the polynomial decay function of the 2-decay method as follows $\omega = 4 - e^7$, $B_0 = 2$, $B_e = 0$, $T = 100$, $N = 0.3$.

Function	Dim	$Range$	$f_{\min}$
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[-100,100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \Pi_{i=1}^{n} \lvert x_i \rvert$	30	[-10,10]	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	30	[-100,100]	0
$f_4(x) = \max_i \left\{ \lvert x_i \rvert, 1 \le i \le n \right\}$	30	[-100,100]	0
$f_5(x) = \sum_{i=1}^{n-1} [100 \left(x_{i+1} - x_i^2 \right)^2 + \left(x_i - 1 \right)^2]$	30	[-30,30]	0
$f_6(x) = \sum_{i=1}^{n} \left([x_i + 0.5] \right)^2$	30	[-100,100]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	[-1.28,1.28]	0

Table .1. Unimodal benchmark functions

Function	Dim	$Range$	$f_{\min}$
$f_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert} \right)$	30	[-500,500]	-418.9829 5
$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos\left(2\pi x_i \right) + 10]$	30	[-5.12,5.12]	0
$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + e$	30	[-32,32]	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \Pi_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	30	[-600,600]	0

$f_{12}(x)=\frac{\pi}{n}\left\{\begin{array}{l}10\sin(\pi y_1)\\+\sum_{i=1}^{n-1}(yi-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]\\+(y_n-1)^2\end{array}\right\}$ $+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & xi>a\\0 & -a<xi<a\\k(-x_i-a)^m & xi<-a\end{cases}$	30	[-50,50]	0
$f_{13}(x)=0.1\left\{\begin{array}{l}\sin^2(3\pi x_1)\\+\sum_{i=1}^{n}(x_i-1)^2\left[1+\sin^2(3\pi x_i+1)\right]\\+(x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\end{array}\right\}$ $+\sum_{i=1}^{n}u(x_i,5,100,4)$	30	[-50,50]	0

Table. 2. Multimodal benchmark functions

Function	*Dim*	*Range*	$f_{\min}$
$f_{14}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	2	[-65,65]	1
$f_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	4	[-5,5]	0.00030
$f_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	[-5,5]	-1.0316
$f_{17}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$	2	[-5,5]	0.398
$f_{18}(x)=1+(x_1+x_2+1)^2\cdot$ $(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\cdot$ $\left[30+(2x_1-3x_2)^2\cdot\right.$ $\left.(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right]$	2	[-2,2]	3
$f_{19}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right)$	3	[1,3]	-3.86
$f_{20}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right)$	6	[0,1]	-3.32
$f_{21}(x)=-\sum_{i=1}^{5}\left[(X-a_i)(X-a_i)^T+c_i\right]^{-1}$	4	[0,10]	-10.1532

$f_{22}(x)=-\sum_{i=1}^{7}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	4	[0,10]	-10.4028
$f_{23}(x)=-\sum_{i=1}^{10}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	4	[0,10]	-10.5363

Table .3. Fixed-dimension multimodal benchmark functions

Algorithms	The parameters
PSO	$C_1=C_2=2, \omega=0.5$
DE	$F=0.5,\ CR=0.3$
ABC	$T=60,\ ER=0.2$
GSA	$G_0=100$
WOA	$a=2,\ b=1$

Table. 4. The parameters of algorithms.

4.1 Compared with other classical heuristic algorithms

Table 5, Table 6 and Table 7 show the means and variances of the 23 benchmark functions in different heuristic algorithms, respectively. As can be seen from Table 5, the CO-GWO is very friendly to unimodal benchmark functions for finding the optimal value. Only F6 has the best performance of GSA. Among them, the CO-GWO is particularly prominent in F1, F2, F3, F4, F5 and F7, and has the fastest convergence, as can be seen in Fig. 5.

		DE	GSA	ABC	PSO	WOA	CO-GWO
F1	Mean	9.166e-12	4.273e-16	3.566e-01	3.778e-01	2.320e-16	**2.067e-129**
	Std	5.180e-12	1.169e-16	1.506e-01	1.424e-01	6.177e-16	**1.032e-128**
F2	Mean	1.147e-07	2.523e-02	4.697e-02	2.327	6.306e-10	**1.928e-75**
	Std	3.094e-08	2.523e-01	1.137e-02	6.577e-01	3.825e-09	**2.854e-75**
F3	Mean	2.797e+04	3.568e+02	3.497e+04	8.073e-01	1.817e+01	**3.881e-30**
	Std	3.428e+03	1.197e+02	4.603e+03	3.294e-01	1.875e+01	**2.612e-29**
F4	Mean	1.293	5.864e-02	1.946e+01	3.294e-01	6.204	**7.921e-33**
	Std	2.355e-01	2.204e-01	6.119	6.049e-02	3.370	**5.593e-32**
F5	Mean	2.976e+01	4.506e+01	3.851e+02	6.889e+01	3.693e+01	**2.654e+01**
	Std	1.302e+01	6.866e+01	1.656e+02	1.577e+01	4.554e+01	**5.164e-01**
F6	Mean	9.544e-12	**4.479e-16**	3.579-01	1.618	8.814e-06	4.609e-01
	Std	4.729e-12	**1.074e-16**	1.190e-01	0.569e-01	3.563e-05	2.634e-01
F7	Mean	2.150e-02	3.357e-02	1.022e-01	1.147	2.563e-02	**3.380e-04**
	Std	5.232e-03	2.262e-02	2.642e-02	4.378e-01	9.336e-03	**2.092e-04**

Table. 5. Results of unimodal benchmark functions

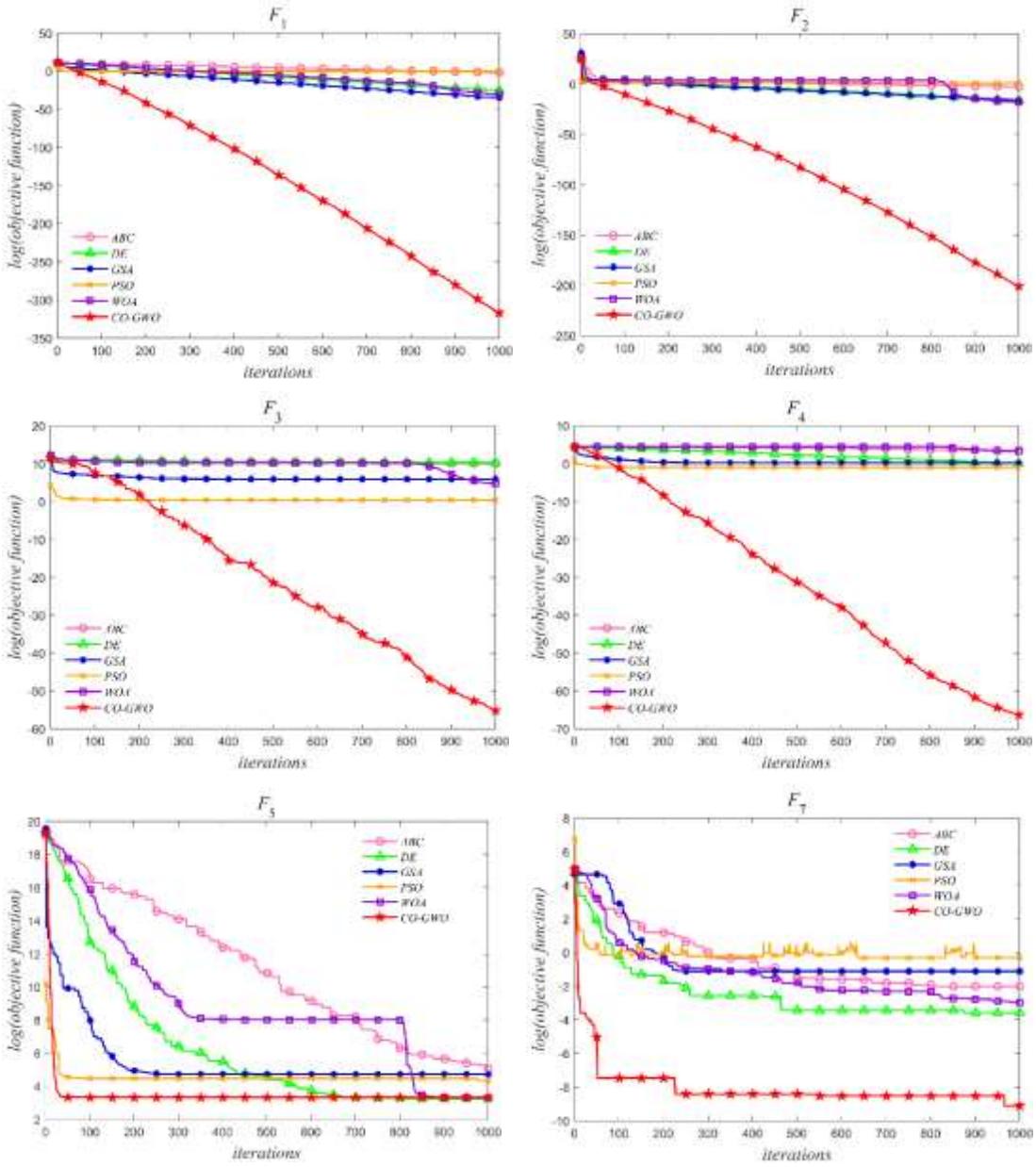


Fig.5. Convergence diagram of F1, F2, F3, F4, F5 and F7

Table 6 shows the multimodal benchmark functions in each heuristic optimization algorithm. CO-GWO is relatively friendly to most multimodal functions. And F9 and F11 are directly their optimal values 0. It can be seen that CO-GWO has a very strong optimization ability, can well jump out of local optimization and converge quickly. Figure 6 shows the convergence of F9, F10 and F11, which shows that the convergence of CO-GWO is much faster than other optimization algorithms. However, F12 and f13 are more prominent in DE.

		DE	GSA	ABC	PSO	WOA	CO-GWO
F8	Mean	**-9.530e+03**	-2.736e+03	-7.548e-03	-1.285e+03	-5.677e+03	-4.504e+03
	Std	5.892e+02	4.043e+02	1.222e+03	1.867e+02	1.459e+02	2.850e+02

F9	Mean	8.598e+01	1.682e+01	2.153e+02	7.306+01	9.430e+01	**0**
	Std	7.766	5.228	1.877e+01	1.730e+01	6.461e+01	**0**
F10	Mean	8.494e-07	1.344e-08	2.444e-01	1.168	1.963e+01	**7.638e-15**
	Std	2.110e-07	1.601e-09	7.334e-02	3.030e-01	2.282	**1.184e-15**
F11	Mean	3.693e-10	4.893	6.488e-01	1.750e-02	1.227e-02	**0**
	Std	1.584e-09	2.047	1.188e-01	7.127e-03	1.680e-02	**0**
F12	Mean	**3.158e-12**	1.153e-01	1.116	1.132e-01	4.138e-01	2.153e-02
	Std	**1.846e-12**	2.064e-01	9.338e-01	5.618e-02	6.167e-01	1.351e-02
F13	Mean	**1.136e-11**	2.053e-02	3.445	1.528	7.961e-03	2.795e-01
	Std	**7.163e-12**	1.604e-01	2.591	4.518e-01	1.668e-02	1.718e-01

Table. 6. Results of multimodal benchmark functions

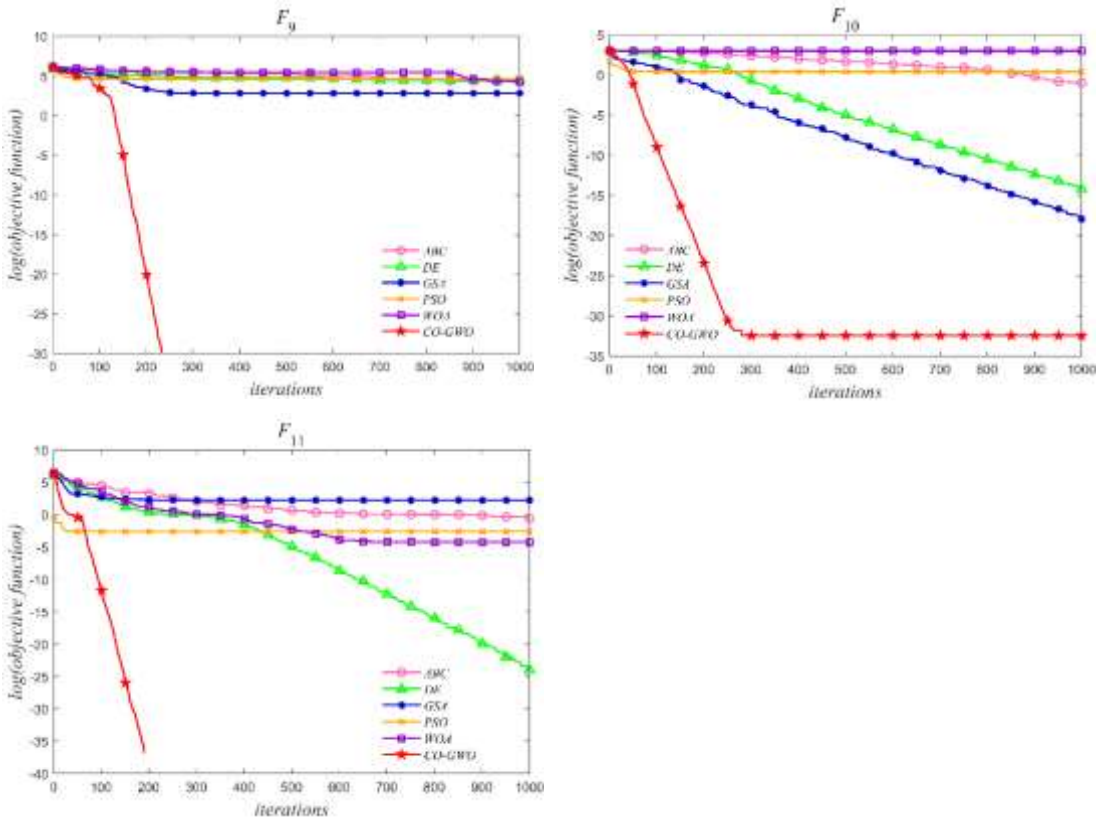
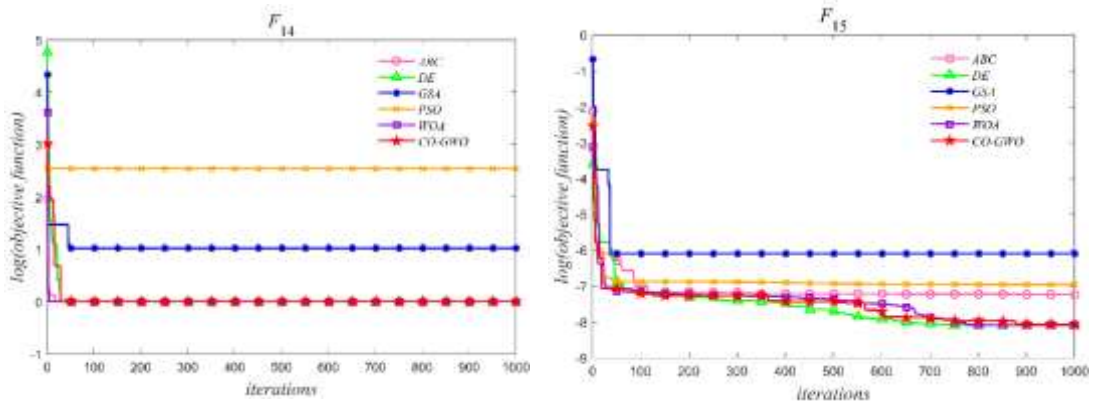


Fig.6. Convergence diagram of F9, F10 and F11

As can be seen from Table 7, for fixed-dimension multimodal benchmark functions, the performance of each heuristic algorithm is similar. In the end, they converge near the optimal value. However, CO-GWO converges faster than the other algorithms, especially the functions F14, F15, F17 and F18, as shown in Figure 7.

		DE	GSA	ABC	PSO	WOA	CO-GWO
F14	Mean	9.980e-01	3.539	9.980e-01	1.267e+01	9.980e-01	**9.980e-01**
	Std	2.566e-15	2.291	2.566e-15	1.492e-13	9.383e-16	6.147e-07
F15	Mean	3.744e-04	4.491e-03	7.117e-04	3.535e-04	2.388e-03	**3.149e-04**
	Std	1.788e-04	2.564e-03	7.151e-05	1.455e-04	5.345e-03	**2.719e-05**
F16	Mean	-1.031	-1.031	-1.031	-1.031	-1.031	-1.031
	Std	1.562e-15	1.337e-15	1.562e-15	1.560e-15	1.522e-15	5.181e-06
F17	Mean	3.978e-01	3.978e-01	3.978e-01	3.978e-01	3.978e-01	**3.978e-01**
	Std	1.060e-15	1.060e-15	1.060e-15	1.060e-15	1.060e-15	6.730e-05
F18	Mean	3	3	3	3	3	**3**
	Std	1.338e-15	6.657e-15	9.004e-16	8.993e-16	1.329e-15	2.388e-06
F19	Mean	-3.004e-01	-0.300	-3.004e-01	-3.187	-3.004e-01	-3.005e-01
	Std	3.905e-16	3.905e-16	3.905e-16	5.513e-01	3.905e-16	3.905e-16
F20	Mean	-3.318	-1.973	-3.318	-3.255	-3.251	-3.223
	Std	1.892e-2	5.587e-01	1.783e-02	5.931e-02	7.517	5.315e-02
F21	Mean	-1.007e+01	-4.885	-8.883	-5.157	-7.237	-9.823
	Std	7.470e-01	7.695	2.273	7.173e-01	2.837	7.267e-01
F22	Mean	-1.040e+01	-7.650	-9.737	-5.193	-7.371	-1.025e+01
	Std	1.543e-14	2.743	1.651	7.478e-01	3.010	5.929e-01
F23	Mean	-1.053e+01	-1.053e+01	-1.023e+01	-5.290	-7.170	-1.0203e+01
	Std	1.426e-14	1.145e-14	1.173	9.271e-01	2.952	5.883e-01

Table .7. Results of fixed-dimension multimodal benchmark functions



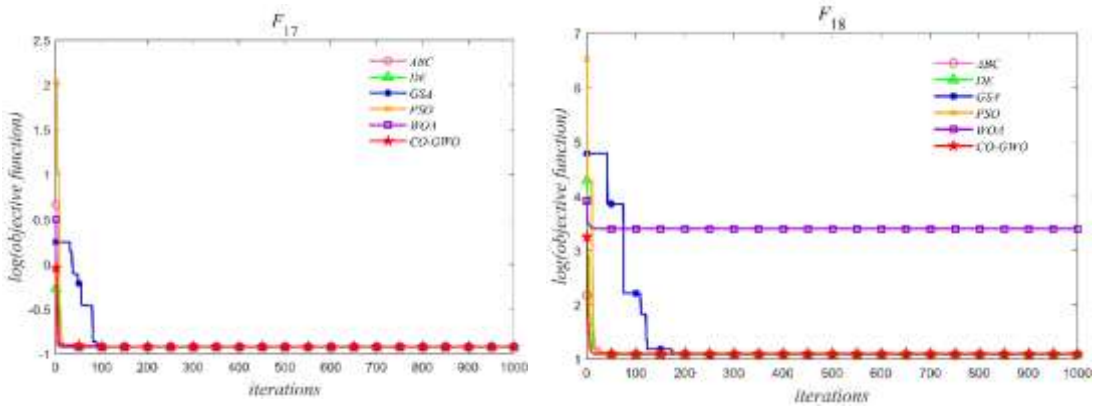


Fig.7. Convergence diagram of F14, F15, F17and F18

F1~F7 are unimodal benchmark functions, which have no obvious local solution. Many optimization algorithms can converge to the optimal value, but CO-GWO can converge to a better optimal value. F8~F13 are multimodal benchmark functions, which have multiple local solutions. It is easy to fall into local optimal values when optimizing such functions. However, CO-GWO can always jump out of the local optimal solution and converge to the overall optimal solution, which has great advantages. F14~F23 are fixed-dimension multimodal benchmark functions. For such functions, many optimization algorithms can converge to the optimal value, but CO-GWO converges faster in most cases.

4.2 Compared with other improved gray wolf algorithms

At present, there are some improved gray wolf algorithms, such as OGWO and IGWO. And they also performed well in their respective periods. Therefore, the current CO-GWO algorithm is compared with other improved gray wolf algorithms.

As can be seen from Table 8, each of the improved algorithms performs very well in the unimodal benchmark functions for finding the best. However, CO-GWO performs the most outstandingly and shows an amazing result in finding the optimum. Both in the optimal value and convergence speed, it far exceeds the other improved algorithms. Figure 8 shows the convergence of unimodal benchmark functions in each improved gray wolf algorithm.

		GWO	OGWO	IGWO	CO-GWO
F1	Mean	3.435e-70	7.223e-92	1.061e-97	**2.066e-129**
	Std	9.357e-70	2.148e-91	3.708e-97	**1.031e-128**
F2	Mean	4.961e-41	5.434e-54	1.936e-56	**1.928e-75**
	Std	7.120e-41	8.305e-54	3.354e-56	**2.853e-75**
F3	Mean	1.253e-19	8.157e-24	3.080e-27	**3.881e-30**
	Std	4.972e-19	3.786e-23	1.675e-26	**2.612e-29**
F4	Mean	1.850e-17	1.405e-23	2.591e-29	**7.921e-33**

	Std	2.653e-17	3.095e-23	7.275e-29	**5.592e-32**
F5	Mean	2.649e+01	2.652e+01	2.674e+01	**2.653e+01**
	Std	6.977e-01	6.604e-01	7.130e-01	**5.163e-01**
F6	Mean	3.380e-01	3.495e-01	1.305	4.609e-01
	Std	2.590e-01	2.641e-01	4.249e-01	2.634e-01
F7	Mean	5.412e-04	3.543e-04	3.391e-04	**3.380e-04**
	Std	3.072e-04	2.871e-04	2.627e-04	**2.092e-04**

Table .8. Results of unimodal benchmark functions

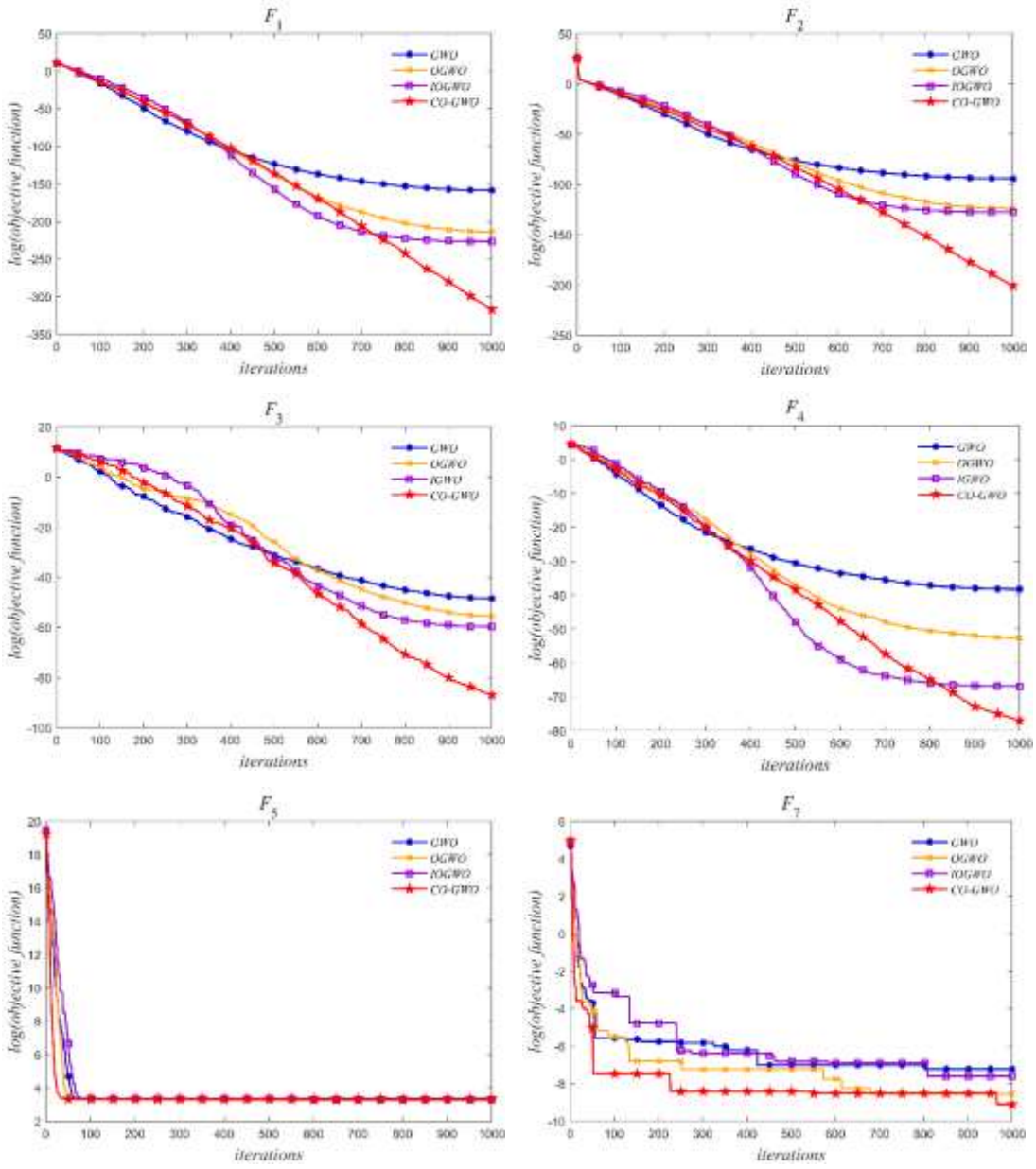
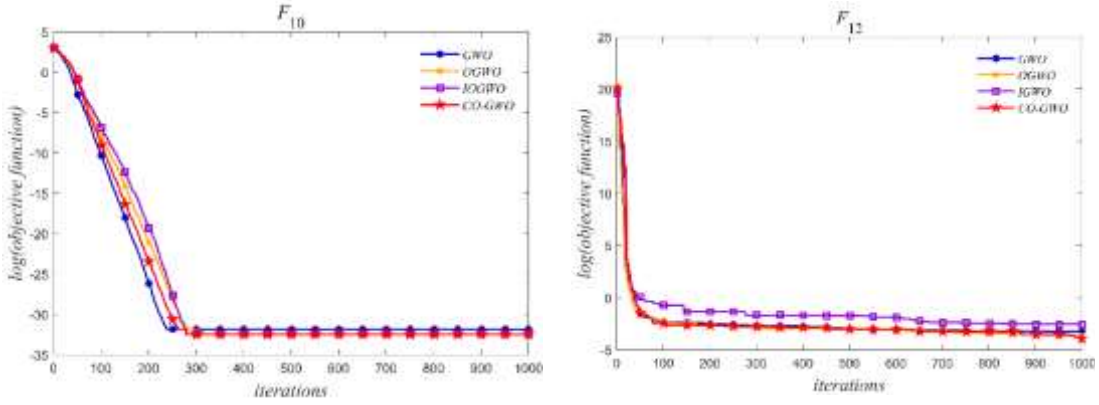


Fig. 8. Convergence diagram of F1, F2, F3, F4, F5 and F7

Similarly, it can be seen from Table 9 that the optimization results of CO-GWO in most multimodal functions are better than other improved gray wolf optimization algorithms. Only F8 is slightly worse. In addition, the convergence rate of CO-GWO is also fast. Figure 9 shows the convergence of F10, F12 and f13.

		GWO	OGWO	IGWO	CO-GWO
F8	Mean	**-6.260e+03**	-5.271e+03	-4.835e+03	-4.504e+03
	Std	9.322e+02	1.493e+03	1.353e+03	2.850e+02
F9	Mean	0	0	0	**0**
	Std	0	0	0	**0**
F10	Mean	1.342e-14	8.277e-15	8.252e-15	**7.638e-15**
	Std	2.786e-15	1.399e-15	1.784e-15	**1.184e-15**
F11	Mean	1.325e-03	0	0	**0**
	Std	4.645e-03	0	0	**0**
F12	Mean	2.363e-02	2.538e-02	7.320e-02	**2.153e-02**
	Std	1.552e-02	**1.340e-02**	3.262e-02	1.351e-02
F13	Mean	3.380e-01	2.879e-01	7.964e-01	**2.795e-01**
	Std	1.717e-01	**1.636e-01**	1.866e-01	1.718e-01

Table. 9. Results of multimodal benchmark functions



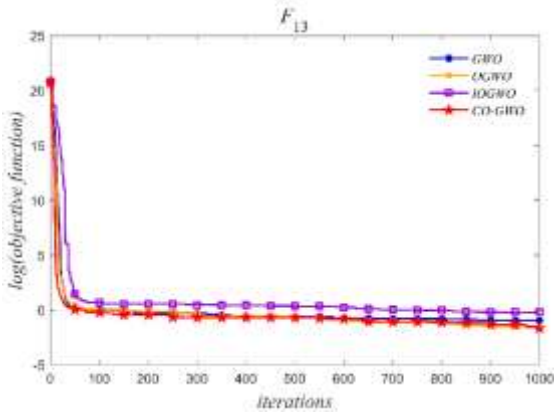


Fig. 9. Convergence diagram of F10, F12 and F13

It can be seen from Table 10 that the optimization results of each improved gray wolf optimization algorithm on the fixed-dimension multimodal benchmark functions are similar. CO-GWO is only slightly outstanding in F14, F15 and F18. Moreover, CO-GWO is exploitated further at the current optimal value only after traversing the entire feasible region in the early stage, so it will hardly jump out of the current position almost during the exploitation process, so the convergence rate of CO-GWO is fast. Fig. 10 shows the convergence of some functions.

		GWO	OGWO	IGWO	CO-GWO
F14	Mean	2.991	2.395	2.803	**9.980e-01**
	Std	3.154	2.490	3.147	**6.147e-07**
F15	Mean	3.262e-03	2.742e-03	3.228e-04	**3.149e-04**
	Std	7.001e-03	6.541e-03	9.379e-05	**2.719e-05**
F16	Mean	-1.031	-1.031	-1.031	-1.031
	Std	3.841e-09	1.457e-06	2.027e-05	5.181e-06
F17	Mean	3.978e-01	3.978e-01	3.978e-01	3.978e-01
	Std	1.896e-07	4.306e-07	4.464e-11	6.730e-05
F18	Mean	3	3	3	3
	Std	5.381e-06	2.493e-06	4.558e-06	**2.388e-06**
F19	Mean	-3.004e-01	-3.004e-01	-3.004e-01	-3.004e-01
	Std	3.905e-16	3.905e-16	3.905e-16	3.905e-16
F20	Mean	-3.261	-3.247	-3.201	-3.223
	Std	7.363e-02	6.935e-02	9.254e-02	5.315e-02
F21	Mean	-9.482	-9.643	-8.407	-9.823
	Std	1.745	1.531	2.450	7.267e-01

F22	Mean	-1.034e+01	-1.024e+01	-10.030	-1.025e+01
	Std	5.273e-01	9.065e-01	1.363	5.929e-01
F23	Mean	-1.037e+01	-1.042e+01	-10.152	-1.020e+01
	Std	1.141	7.607e-01	1.409	5.883e-01

Table .10. Results of fixed-dimension multimodal benchmark functions

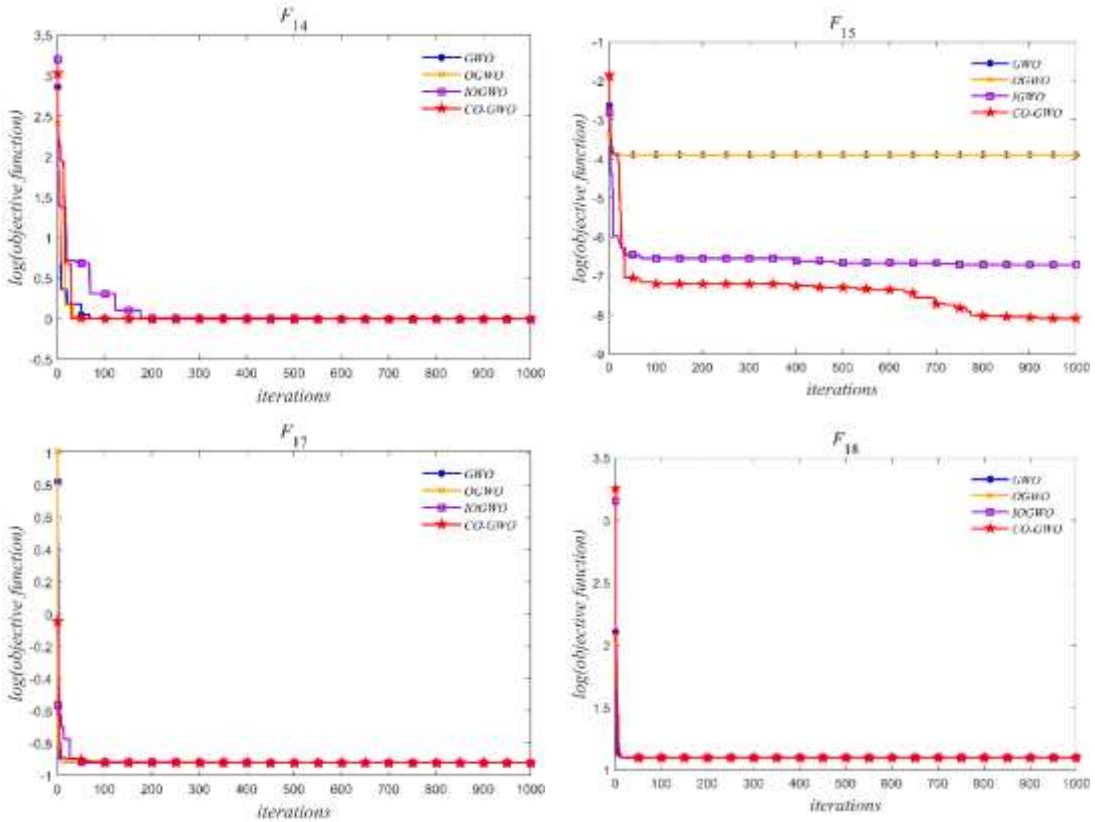


Fig.10. Convergence diagram of F14, F15, F17 and F18

Compared with other improved gray wolf optimization algorithms, CO-GWO performs well in unimodal benchmark functions, multimodal benchmark functions and fixed-dimension multimodal benchmark functions. For unimodal benchmark functions, it can always converge to a better optimal value quickly. For multimodal benchmark functions, it can easily jump out of the use of OBL and local optimal value and quickly converge to the global optimal value. For the fixed-dimension multimodal benchmark functions, although the optimal value is similar to that of other improved algorithms, its convergence speed is better than that of other improved algorithms.

In the previous period, tTent mapping enables the individuals in the population to traverse the entire feasible region, resulting in the optimal value of each generation being found from the global range. And the individuals of the next iteration will only be the better part of the individuals in this iteration. This makes the next-generation individuals always better than the previous-generation individuals, which helps to accelerate the convergence speed, so CO-GWO excels in unimodal benchmark functions. In addition, the algorithm combines the polynomial decay function of 2-Decay method, which makes the parameters $\vec{a}$ decay slowly and increases the

number of exploration of gray wolves, which is good for jumping out of local optimum, so CO-GWO performs well in multimodal benchmark functions.

5. Conclusions

CO-GWO is proposed to ensure that the local optimization can be jumped out in time and a better optimal value can be obtained. The population of the algorithm is composed of two parts, one is to retain the better part of each iteration, and the other is generated by tent map and OBL. This makes the value of each iteration not get worse, but also traverses the whole feasible region to search for a better value of the next generation. It accelerates the search speed and optimization ability. In addition, in order to better balance the relationship between exploration and exploitation, the polynomial decay function based on the 2-decay method is used to control parameter $\vec{a}$.

23 benchmark functions are tested, and CO-GWO is compared with five classical heuristic optimization algorithms, GWO and its variants. Experimental results show that the optimization performance and stability of CO-GWO are better than other optimization algorithms.

References

[1] Steinbrunn, Michael, Guido Moerkotte, and Alfons Kemper. "Heuristic and randomized optimization for the join ordering problem." *The VLDB Journal* 6.3 (1997): 191-208.

[2] Dorigo, Marco, Vittorio Maniezzo, and Alberto Colorni. "Ant system: optimization by a colony of cooperating agents." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1 (1996): 29-41.

[3] Holland, John H. "Genetic algorithms." *Scientific american* 267.1 (1992): 66-73.

[4] Storn, Rainer, and Kenneth Price. "Minimizing the real functions of the ICEC'96 contest by differential evolution." *Proceedings of IEEE international conference on evolutionary computation*. IEEE, 1996.

[5] Wang, Lei, Jin Pan, and Li-cheng Jiao. "The immune algorithm." *ACTA ELECTONICA SINICA* 28.7 (2000): 96.

[6] Dorigo, Marco, Vittorio Maniezzo, and Alberto Colorni. "Ant system: optimization by a colony of cooperating agents." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1 (1996): 29-41.

[7] Eberhart, Russell, and James Kennedy. "Particle swarm optimization." *Proceedings of the IEEE international conference on neural networks*. Vol. 4. 1995.

[8] Karaboga, Dervis, and Bahriye Basturk. "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm." *Journal of global optimization* 39.3 (2007): 459-471.

[9] Yang, Xin-She. "Firefly algorithm, stochastic test functions and design optimisation." *International journal of bio-inspired computation* 2.2 (2010): 78-84.

[10] Yang, Xin-She. "A new metaheuristic bat-inspired algorithm." *Nature inspired cooperative strategies for optimization (NICSO 2010)*. Springer, Berlin, Heidelberg, 2010. 65-74.

[11] Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis. "Grey wolf optimizer." *Advances in engineering software* 69 (2014): 46-61.

[12] Eusuff, Muzaffar, Kevin Lansey, and Fayzul Pasha. "Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization." *Engineering optimization* 38.2 (2006): 129-154.

[13] Gandomi, Amir Hossein, and Amir Hossein Alavi. "Krill herd: a new bio-inspired optimization

algorithm." *Communications in nonlinear science and numerical simulation* 17.12 (2012): 4831-4845.

[14] Yusof, Yuhanis, and Zuriani Mustaffa. "Time series forecasting of energy commodity using grey wolf optimizer." (2015): 25-30.

[15] Yu, Xiaobing, WangYing Xu, and ChenLiang Li. "Opposition-based learning grey wolf optimizer for global optimization." *Knowledge-Based Systems* 226 (2021): 107139.

[16] Dhargupta, Souvik, et al. "Selective opposition based grey wolf optimization." *Expert Systems with Applications* 151 (2020): 113389.

[17] Li, Yu, Xiaoxiao Lin, and Jingsen Liu. "An improved gray wolf optimization algorithm to solve engineering problems." *Sustainability* 13.6 (2021): 3208.

[18] Mittal, Nitin, Urvinder Singh, and Balwinder Singh Sohi. "Modified grey wolf optimizer for global engineering optimization." *Applied Computational Intelligence and Soft Computing* 2016 (2016).

[19] Long, Wen, et al. "An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization." *Engineering Applications of Artificial Intelligence* 68 (2018): 63-80.

[20] Long, Wen, et al. "A modified augmented Lagrangian with improved grey wolf optimization to constrained optimization problems." *Neural Computing and Applications* 28.1 (2017): 421-438.

[21] Gupta, Shubham, and Kusum Deep. "A memory-based grey wolf optimizer for global optimization tasks." *Applied Soft Computing* 93 (2020): 106367.

[22] Seyyedabbasi, Amir, and Farzad Kiani. "I-GWO and Ex-GWO: improved algorithms of the Grey Wolf Optimizer to solve global optimization problems." *Engineering with Computers* 37.1 (2021): 509-532.

[23] Rodríguez, Alma, et al. "Group-based synchronous-asynchronous grey wolf optimizer." *Applied Mathematical Modelling* 93 (2021): 226-243.

[24] Tawhid, Mohamed A., and Abdelmonem M. Ibrahim. "A hybridization of grey wolf optimizer and differential evolution for solving nonlinear systems." *Evolving Systems* 11.1 (2020): 65-87.

[25] Sundaramurthy, Shanmugam, and Preethi Jayavel. "A hybrid grey wolf optimization and particle swarm optimization with C4. 5 approach for prediction of rheumatoid arthritis." *Applied Soft Computing* 94 (2020): 106500.

[26] Jayabarathi, T., et al. "Economic dispatch using hybrid grey wolf optimizer." *Energy* 111 (2016): 630-641.

[27] Daniel, Ebenezer. "Optimum wavelet-based homomorphic medical image fusion using hybrid genetic–grey wolf optimization algorithm." *IEEE Sensors Journal* 18.16 (2018): 6804-6811.

[28] Kennedy, James, and Russell Eberhart. "Particle swarm optimization." *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. IEEE, 1995.

[29] Zhang, Tao, and Wei Li. "k-decay: A new method for learning rate schedule." *arXiv preprint arXiv:2004.05909* (2020).

[30] Mirjalili, Seyedali, and Andrew Lewis. "The whale optimization algorithm." *Advances in engineering software* 95 (2016): 51-67.