Journal of Computers Vol. 33 No. 6, December 2022, pp. 1-19
doi: 10.53106/199115992022123306001

An Improved Chicken Swarm Optimization Algorithm Based on Adaptive Mutation Learning Strategy

Xin-Xin Zhou*, Zhi-Rui Gao, Xue-Ting Yi

School of Computer Science, Northeast Electric Power University, Jilin City, Jilin, China
20100466@neepu.edu.cn, gzr96306116@163.com, 2089654374@qq.com

Received 18 October 2021; Revised 12 March 2022; Accepted 12 April 2022

Abstract. To solve the problem that the Chicken swarm optimization (CSO) has low solution accuracy and tends to fall into the local optimum on later stages of iteration, an adaptive mutation learning Chicken swarm optimization (AMLCSO) is proposed in this paper. Firstly, to solve the problem of uneven initial distribution and improve the algorithm's stability, a good-point set is introduced. Secondly, according to the difference between the current individual position and the optimal individual position, the nonlinear adaptive adjustment of weight is realized and the position update step is dynamically adjusted. This strategy improves the algorithm's convergence. Thirdly, the learning update strategies of Gaussian mutation and normal distribution are introduced to improve the probability of selection and solving accuracy and avoid falling into the local optimum. Finally, the AMLCSO is compared with other standard algorithms and improved Chicken swarm optimization algorithms on twenty benchmark test functions. The experimental results show the AMLCSO has faster convergence and higher solution accuracy.

Keywords: Chicken swarm optimization, Gaussian mutation, learning update strategy, nonlinear adaptive

1 Introduction

The swarm intelligence optimization algorithms are a kind of random optimization algorithm constructed by simulating natural laws or the inherent behavior and living habits of creatures such as foraging and hunting. The swarm intelligence algorithms such as Particle Swarm Optimization(PSO) [1-2], Grey Wolf Optimization (GWO) [3], Ant Colony Optimization (ACO) [4-6], Artificial Fish Swarm Algorithm (AFSA) [7-9] etc. have been applied in the fields of engineering science, computing science and so on. Chicken swarm optimization [10] is proposed by Meng et al. in 2014 to simulate chicken flock hierarchy and foraging behavior. This algorithm has the characteristic of simple implementation [11-12]. It has been widely applied in wireless sensors [12-13], distribution network [14], image processing [15] and other engineering problems [16-21].

In addition, the random initialization of the standard chicken population algorithm will cause uneven distribution of the population, which will affect the stability of the algorithm; The search ability of roosters affects the accuracy of the algorithm. The search ability of hens and chicks will be affected by roosters in the same subgroup, which makes the algorithm easy to fall into local optimization in the later stage of iteration. To solve the above problems, an adaptive mutation learning chicken swarm optimization algorithm is proposed in this paper. It improves the convergence speed and accuracy of the algorithm, and avoids the algorithm falling into local optimization.

The main contributions of this paper are summarized as follows:

(1) In view of the uneven distribution of population initialization, using a good point set to initialize the population. This strategy can improve the ergodicity and stability of the algorithm;

(2) Nonlinear adaptive weight is proposed to adjust the step size of the position update dynamically, which improves the convergence speed of the algorithm;

(3) Gaussian mutation is introduced to perform mutation operations on population individuals to avoid the algorithm falling into the local optimum;

(4) The normal distribution learning update strategy is introduced to improve the location update methods of different levels in the population, which improves the accuracy of the algorithm;

(5) Verify the effectiveness and convergence of the algorithm in this paper.

The rest of this paper is organized as follows: the related work is provided in section 2; Section 3 reviews the chicken swarm algorithm; In section 4, the adaptive mutation learning chicken swarm algorithm (AMLCSO) is proposed; Section 5 verifies the effectiveness and convergence of the adaptive mutation learning algorithm

* Corresponding Author

(AMLCSO); Conclusions and future directions are summarized in section 6.

2 Related Work

However, just like the most intelligent algorithms, the Chicken swarm optimization also has problems of low solution accuracy and slow convergence speed. And it is easy to fall into local optimum in the late stage of the algorithm iteration. In view of these shortcomings, many scholars have improved the Chicken swarm optimization from different aspects. Zhang et al. [22] introduced non-linear inertia weights and learned from rooster into the chicken position update formula, which was used to avoid the algorithm falling into local optimum. And it was applied to the parameter optimization problem of support vector machine. A learning part was added to the chicken position update formula by Wang et al. [23] to solve the problem that the algorithm tends to fall into local optimum. Based on the improved Chicken swarm optimization, they proposed an interrupt load scheduling model considering the user subsidy rate, which was used to reduce the system peak load and operating costs. Sanchari Deb et al. [24] combined the Chicken swarm optimization with the teaching and learning optimization algorithm to avoid falling into local optimum. And it was used to solve the problem of charging station placement. Wang et al. [25] improved the rooster's location update formula from the perspective of balancing exploration and development. The threshold and stimulus of the performing formula were created based on the degree of population aggregation and average improvement. The location was also updated as part of the stimulus-response mechanism. Based on chaos theory, Li et al. [26] improved the Chicken swarm optimization. This method avoided falling into local optimum and improved the optimization ability of the algorithm. Moreover, it was applied to forecast the range of wind power.

Levy flight and random forest were introduced by Zhang et al. [27] to solve the problem of low accuracy and it optimize the radio frequency. Levy flight and non-linear decline strategy were introduced by Liang et al. [28] to improve the convergence accuracy and stability of the algorithm. It was applied in the robot path planning. Han et al. [29] carry out the mutation strategy for the individuals with low fitness to solve the problem of poor optimization accuracy of the algorithm and solve the 0-1 knapsack problem.

To improve the convergence speed of the algorithm, mutation operations, accelerated search methods and unified mutation operators were introduced by Bo et al. [30]. And it was applied to the optimization problem of hypersonic flight ascent trajectory. A dynamic inertia strategy was introduced by Xue et al. [31] to improve the overall convergence speed. It was used to establish a short-term wind power prediction model to predict the short-term wind speed of the wind farm. The cosine inertia weight and Cauchy mutation operator were introduced by Liu et al. [32] to improve the convergence of the algorithm. Furthermore, the improved CSO algorithm was used to optimize the weight and threshold of the extreme learning machine to improve the prediction accuracy. Osamy et al. [33] combined the deer hunting optimization algorithm with the Chicken swarm optimization, which can greatly improve the convergence speed and optimization accuracy of the hybrid algorithm. And it was applied to the human-computer interaction. Lin et al. [34] improved location update formula of rooster and transformed the sequential iteration process into a parallel iterative process, thereby improving the convergence speed. And it applied to optimize image processing unit.

It should be noted that each of problems has been improved in the above literature. The problems in the optimization process of the standard algorithm were solved and the optimization ability was improved. But the algorithms still have shortcomings. In order to improve the convergence speed and solution accuracy, and to avoid falling into the local optimum, an adaptive mutation learning Chicken swarm optimization algorithm (AMLCSO) is proposed in this paper.

3 Chicken Swarm Optimization

In the Chicken swarm optimization, the entire chicken flock is divided into roosters, hens, and chicks according to the hierarchy strictly. Chickens of different levels have different responsibilities. The roosters will look for food actively and grab food from other roosters randomly; The hens will follow the roosters in the same subgroup for foraging; And the chicks will follow their mothers for foraging. The rules are summarized as follows:

(1) The entire chicken flock is divided into subgroups, each consists of a rooster, multiple hens and chicks.

(2) The flock is divided according to the fitness value, and the better the fitness value, the higher the grades. The individuals with the best fitness in the population are regarded as roosters, the worst individuals are regarded as chickens, and the remaining individuals are regarded as hens.

(3) The rooster has the highest grade in the population and plays a leading role during foraging. And it selects

another rooster to compete randomly to expand the search range; The hen is second only to the roosters and randomly follows the roosters in the same subgroup to search for food. And it occasionally steals food from other subgroups. The chicks are the lowest-level and have a large search range. The chicks in each subgroup randomly follow their mother for food and establish a mother-child relationship.

(4) In each subgroup, the hierarchy, dominance relationship and mother-child relationship between each individual will remain unchanged after determination. Until a few generations later, the hierarchy is re-classified and the relationship is established again according to the fitness value.

At the beginning of the algorithm, the number of roosters, hens, hens with chicks and the number of chicks are set as Nr, Nh, Nm, and Nc in the D-dimensional space. N is the size of the chicken flock. G is the number of updates of the hierarchy. $x_{i,j}(t)$ is the position of the $i(i \in [1, N])$ chicken in the $j(j \in [1, N])$-dimensional search space in the tth iteration.

$$Nc = N - Nr - Nh \ . \tag{1}$$

When solving the optimization problem, the chicken swarm individual adopts different position update methods according to different grades.

The rooster position update formula is as follows.

$$x_{i,j}(t+1) = x_{i,j}(t) * (1 + randn(0, \sigma^2)) \ . \tag{2}$$

$$\sigma^2 = \begin{cases} 1 & f_i < f_a \\ \exp\left(\left(f_a - f_i\right)/\left(\left|f_i\right| + \varepsilon\right)\right) & a \in [1, N_r], a \neq i \quad f_i \geq f_a \end{cases} \ . \tag{3}$$

Where $randn(0, \sigma^2)$ is a normal distribution with a mean of 0 and a variance of σ^2; f_i is the fitness value of individual i; a is the any individual of roosters which satisfies $a \neq i$; ε is the smallest constant; f represents the fitness value.

The hen position update formula is as follows.

$$x_{i,j}(t+1) = x_{i,j}(t) + C_1 * Rand * \left(x_{r_1,j}(t) - x_{i,j}(t)\right) + C_2 * Rand * \left(x_{r_2,j}(t) - x_{i,j}(t)\right) \ . \tag{4}$$

$$C_1 = \exp\left(\left(f_i - f_{r_1}\right) / \left(abs\left(f_i\right) + \varepsilon\right)\right) \ . \tag{5}$$

$$C_2 = \exp\left(f_{r_2} - f_i\right) \ . \tag{6}$$

Where *Rand* is a random number in [0,1]; $r_1 \in [1, Nr]$ is a randomly selected number of rooster; $r_2 \in [1, Nr+Nh]$ is a random number of rooster or hen. But r_1 and r_2 are not equal. C_1 C_2 is the following coefficient.

The chicks position update formula is as follows.

$$x_{i,j}(t+1) = x_{i,j}(t) + F * \left(x_{m,j}(t) - x_{i,j}(t)\right) \ . \tag{7}$$

Where $x_{m,j}(t)(m \in [1, Nh])$ is the position of chick's mother which the current chicks are following; $F \in [0, 2]$ is the impact factor of mother hens on chicks.

4 Chicken Swarm Optimization Algorithm Based on Adaptive Mutation Learning

The position update formulas of different levels in the standard Chicken swarm optimization are interrelated, which is easy to cause the convergence of the algorithm to decrease; When the highest-level individual falls into the local optimum on the later stage of the iteration, it is also easy to cause the algorithm to fall into the local optimum, thereby leading to a decrease in solution accuracy. Therefore, an adaptive mutation learning Chicken

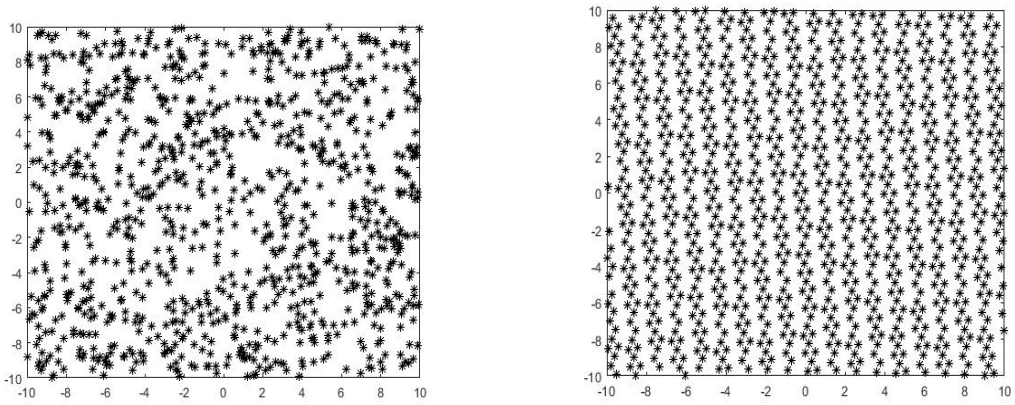
swarm optimization (AMLCSO) is proposed in this paper. The adaptive weighting strategy, Gaussian mutation and normal distribution learning update strategy are introduced to improve the position update method owned by different levels in the population. The way improves the problem that the algorithm is easy to fall into local optimum. And the solving accuracy and convergence of the algorithm are improved. By introducing a good point set, the stability of the algorithm is further improved.

4.1 Population Initialization Based on a Good-Point Set

In the swarm intelligence algorithm, the distribution of the initial population affects the diversity and stability of the algorithm. The standard Chicken swarm optimization uses the method of random initialization to generate the initial population, which will cause the problems of uneven initial population distribution and low ergodicity. And it causes the initial positions of the individuals to gather around the localized extremes. It is not conducive to searching for the global optimum and affects the optimization performance of the algorithm. In the Chicken swarm optimization, individuals with high grade only search in their own group, which tends to reduce the searching range of individuals gradually. And finally the algorithm falls into a local optimum eventually. Therefore, a population initialization strategy based on the good point set is proposed. This method can make the initial value of population distribute evenly in the entire solution space. It is not affected by the spatial dimension and is conducive to finding the most global optimum in the search.

Basic definition of the good point set [35] is as follow: Pop points are uniformly selected in the J-dimensional European space unit cube. If $r \in G_s$, then $P_{pop}(k)=\{(\{r_1*k\},\{r_2*k\},\ldots,\{r_J*k\}),1\le k\le pop\}$.$P_{pop}(k)$ is a set of good points; *pop* is the population number; *r* is the best point; The deviation of the point set satisfies $\varphi(pop)=C(r,\varepsilon)*pop^{-1+\varepsilon}$,where $C(r,\varepsilon)$ is a constant related to r, ε ; ε is an arbitrary constant in the computer. And the value of *r* is $\{2*\cos(2\pi k/p),1\le k\le J\}$, where *P* satisfies the smallest prime number of (p-J)/2≥J.

Good point set and random initialization are used to initialize the population in a two-dimensional space with size of 500, as shown in Fig. 1 shows that the good point set strategy distributes more uniformly than the random method in the same population size. Therefore, the good point set strategy is adopted in this paper to initialize population and generate an uniform distribution, no-overlapping points and richly diverse population in the solution space. The method enhances the ergodicity and stability of the initial population and then the performance of algorithm is improved.



(a) Random method (b) Good point set strategy

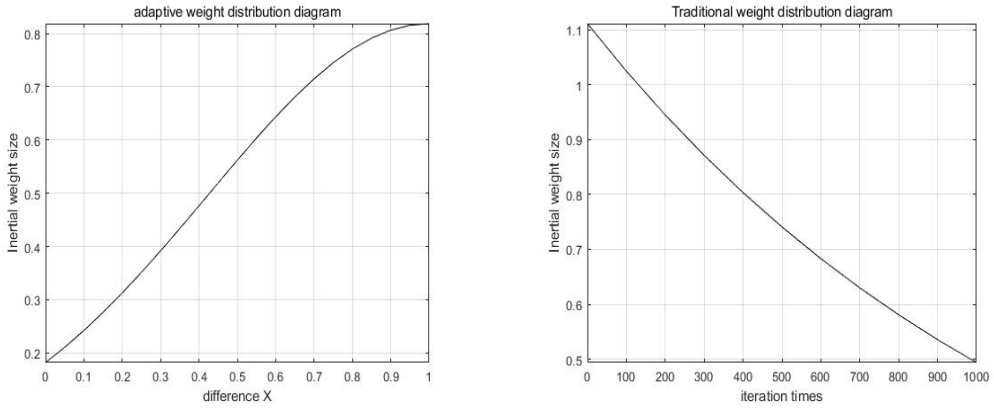
Fig. 1. Initialized population distribution diagram

4.2 The Position Update Strategy of Rooster Based on Adaptive Weight

In the standard Chicken swarm optimization, the rooster population which has the highest grade in the entire population can lead other individuals in the group to find the optimal solution. The position update of rooster is optimized only in the rooster population as presented in Eq. (2). In this way, the algorithm has limitations and blindness in the iterative process and causes the convergence speed of the algorithm to decrease.

In order to solve the above problems, this paper introduces a nonlinear adaptive weight strategy. In Fig. 2(a) the traditional weight values mostly change with the number of iterations. The way does not take the character-

istic of each individual into account in the iteration process and the weight values lack guide. In this paper, the difference between the current individual position and the optimal individual position of the population is used as a guide. The weight is nonlinearly adjusted according to the degree of difference. When the degree becomes larger, the individual is far away from the optimal position and w will increase. At this time, the individual has great global search ability, which can improve the development ability of the algorithm in the early stage. On the contrary, w will decrease and the local search ability of the individual will increase, which can improve the exploration ability of the algorithm on the later stage, as shown in Fig. 2(b) This weighting strategy based on nonlinear adaptive adjustment can adjust the local search ability and global search ability of the algorithm dynamically. And it also balances the development and exploration of the algorithm before and after iteration. Therefore, the convergence and stability of the algorithm is improved.



(a) Adaptive weight strategy (b) Traditional weight

Fig. 2. Distribution of inertia weights of different strategies

The improved roosters position update formula is given as follows.

$$x_{i,j}(t+1) = \left(x_{i,j}(t) * (1 + randn(0, \sigma^2))\right) * w(t) \ . \tag{8}$$

$$X = 1/\left(x_{\max} - x_{\min}\right) * 1/M * \sum_{i=1}^{n} \left|g - x\right| \ . \tag{9}$$

$$w(t) = \exp(-(w_{start} - \left(\log(w_{start}) - \log(w_{end})\right) * \left(X - 1\right)^2)) \ . \tag{10}$$

Where n is the dimension of solution space; $w(t)$ is the adaptive weight of the moment t; $w_{start} w_{end}$ are the initial values and end values. $x_{\max}$ and $x_{\min}$ are the maximum and minimum values of the population. g is the optimal position of the population at time t, and x is the current individual position at time t.

4.3 The Position Update Strategy of Hen Based on Gaussian Mutation and Normal Distribution Learning

In the Chicken swarm optimization, the number of hens accounts for a large proportion and their grade is second only to roosters. It has a great impact on the performance of the algorithm. According to the Eq. (4) , the search direction and distance of the algorithm are controlled by the variable parameters $C_1 C_2$ in the formula of hen position update. And it makes the algorithm not easy to fall into the local optimum. But the search ability of the hen will be affected by the roosters of the same subgroup. And the hen will also randomly steal food from other subgroups. On the later iteration of the algorithm, when the roosters gather and the algorithm fall into the local optimum, it will indirectly cause the hens to fall into the local optimum. And the diversity of population will dete-

riorate, thereby reducing the solution accuracy and convergence speed of the algorithm.

To solve the problems, Gaussian mutation operator is introduced in this paper to exchange information between the optimal individual and the current individual of the population. The searching method of hens is changed and the searching range is expanded. As a result, the diversity of the population and the searching ability of hen are increased and the precocious phenomenon is prevented. Finally, the normal distribution learning update strategy is introduced. From the beginning to the middle of the iteration, w is continuously increasing and the algorithm's global search ability is continuously enhancing. And the global optimum can be searched. From the middle to the late of the iteration, w is continuously decreasing and the local search capability of the algorithm is gradually increasing, which can avoid the algorithm falling into the local optimum. The global optimum can be accurately determined and the convergence speed of the algorithm is improved.

The improved formula of hens position update is given as follows.

$$x_{i,j}(t+1)=\begin{pmatrix} x_{i,j}(t)+\left(x_{best,j}(t)-x_{i,j}(t)\right)*M+C_1*Rand*\left(x_{r_1,j}(t)-x_{i,j}(t)\right)+ \\ C_2*Rand*\left(x_{r_2,j}(t)-x_{i,j}(t)\right) \end{pmatrix}*w \ . \quad (11)$$

$$M=\left(p_{\min}+p_{\max}\right)/2+\left(p_{\max}-p_{\min}\right)/6*\left(\sum_{i=1}^{\dim} r_i-6\right) \ . \quad (12)$$

$$w=w_{\min}*w_{\max}/w_{\min}^{1/(1+10*t/T)}+Randn() \ . \quad (13)$$

Where M is the Gaussian mutation operator. $p_{\min}$ and $p_{\max}$ are the minimum and maximum values taken at the Gaussian variation point; r_i is a random number in the range of [0,1]; dim is the maximum dimension of the population; $x_{\text{best,j}}$ is the position of the optimal solution in the population; w is the random learning coefficient of the hen; T is the total number of iterations, and t is the current iteration number of the algorithm. Randn() is a random number of normal distribution.

4.4 Chicken Position Update Strategy Based on Normal Distribution Learning

In the standard Chicken swarm optimization, chicks only follow hen who has a mother-child relationship with them. As a result, when the hen falls into the local optimum, the following chick will also fall into the local optimum in the later period of the algorithm.

Therefore, the normal distribution learning update strategy (Eq.(13)) is introduced to avoid the chickens falling into the local optimum. In addition, while the chicks learn from their mother, they also randomly follow other hens or roosters in the population to learn. By improving the search method of the chicks, the search range of the chicks is enhanced and it avoids the chicks falling into the local optimum. Then the search ability of the algorithm is improved.

The improved chicks position updating formulas is as follows.

$$x_{i,j}(t+1)=w*\left(x_{i,j}(t)+F*\left(x_{m,j}(t)-x_{i,j}(t)\right)+\left(x_{s,j}(t)-x_{i,j}(t)\right)\right) \ . \quad (14)$$

Where $x_{m,j}(t)(m \in [1, Nh])$ is the position of the chick's mother that the current chick is following, F is the following coefficient, $x_{s,j} \in [1, Nr+Nh]$ is the position of the random s rooster or hen in the population, and ω is the random learning coefficient of the chick.

4.5 AMLCSO Algorithm Description

The steps of the AMLCSO algorithm are outlined as follows:

Step 1: Initialize parameters. Set relevant parameters w_{min} w_{max}; the chicken population size N; the number of roosters Nr; the number of hens Nh; the number of hens with chicks Nm; the number of chicks Nc; parameter dimensions D; the number of grade system updates G and the total number of iterations T.

Step 2: The good point set strategy is used to initialize the individuals of the population;

Step 3: Calculate the fitness values of all the individuals in the chicken group and sort them in order from small to large. Among them, with good fitness values in the top Nr individuals are defined as roosters, with poor fitness values in the bottom Nc individuals are defined as chicks and others were hens.

Step 4: According to the fitness value of each particle at the t iteration in the chicken group, when mod(t,G)=1, all individuals are reordered and a new hierarchy is established.

Step 5: Gaussian mutation operator, learning coefficient and adaptive inertia weight are calculated according to Eq. (9), (10), (12), (13).

Step 6: According to Eq. (8), (11), (14), the positions of roosters, hens and chicks are updated sequentially and the optimal food position is finally updated.

Step 7: If the individual fitness value is better than the original value, update it, otherwise give up updating;

Step 8: If the number of algorithm iterations reaches the upper limit or a candidate solution that satisfies the termination condition has been obtained, the algorithm terminates. Otherwise, it returns to step 4.

Step 9: Finally, the optimal position and fitness value are recorded in the chicken flock.

The pseudo code of the AMLCSO algorithm is shown by Algorithm 1.

Algorithm 1. Outlines the pseudocode of the AMLCSO algorithm

```
Begin
  Step 1. Initialize parameters w_min , w_max , Nr, Nh, Nm, Nc, N F, G, etc
  Step 2. Use the good point set strategy to initialize the population
  Step 3. Calculate fitness, divided into rooster, hen and chicken
  Step 4. while ( t < T ) do
    For mod ( t, G ) == 1
      Update the population hierarchy
    End for
    IF ( i <=Nr) do
      Calculate the adaptive weighting strategy using Eq. (9)(10)
      Update rooster position using Eq. (8)
    End if
    If(Nr+1<i<=Nr+Nh)do
      Calculate the Gaussian mutation operator using Ep. (12)(13)
      Update hen position using Eq. (11)
    End if
    If (Nr+Nh+1< i <=N) do
      Update chick position using Eq. (14)
    End if
      Update the individual's best fitness vlaue and the global best fitness value
      t = t + 1
  End while
    Output the best solution found
End
```

5 Simulation Experiment and Result Analysis

In order to verify the performance of the algorithm in this paper, a simulation was carried out. All algorithms in this paper were written using MatlabR2018a and the simulation experiments were performed under Intel(R) Core (TM) i5-4258U CPU @ 2.40GHz 2.10GHz, memory 12.00GB, and Windows 10.

5.1 Evaluation Criteria

This section tests the improved chicken swarm algorithm on unimodal and multimodal functions to verify AMLCSO from the following aspects: (1) Compare AMLCSO with the chicken swarm algorithm with a single strategy to verify the effectiveness of the overall strategy. (2) Compared with other improved chicken swarm algorithms, the convergence accuracy of the algorithm is verified. (3) By comparing with other advanced algorithms, the convergence speed of the algorithm is verified. (4) The differences between AMLCSO and other algorithms are compared by Wilcoxon's rank sum test.

For all the algorithms involved in this paper, the population size (N) is set 100. To reduce the algorithm's error, each algorithm runs 30 times independently. For each algorithm, the performance of the algorithm is tested from three aspects: mean, standard deviation and Wilcoxon's rank sum test.

The average value is the mean of N operation results. The formula is given as follow:

$$Mean = \frac{1}{N}\sum_{i=1}^{N} f_i \ . \tag{15}$$

Where fi is the result obtained after the *ith* operation of an algorithm, and N is 30. The standard deviation is the arithmetic square root of the variance of N operation results, which can reflect the dispersion degree of N operation results. The formula is:

$$Std = \sqrt{\frac{1}{N-1}\sum_{i-1}^{N}\left(f_i - Mean\right)^2} \ . \tag{16}$$

The Wilcoxon's rank sum test is a non-parametric test used in the hypothesis testing scenarios of two independence groups. It is used to detect a significant difference between the behaviors of the two algorithms.

5.2 Benchmark Test Function

In this paper, 20 benchmark functions with different dimensions and peak values [36-37] are selected. As shown in Table 1, F_1-F_{11} are high-dimensional unimodal function, which are used to test the convergence speed and optimization accuracy of the algorithm. The value obtained at the end of the iteration is closer to the theoretical optimal value, the better; The functions of Table 2 are high-dimensional multi-modal functions, which are used to test the global search performance of the algorithm and the ability to avoid premature maturity. In the search process, the effect of convergence is great and the optimization ability is better. And the global optimum can be found. Among these test functions, F_1 - $F_{15}F_{17}F_{18}F_{20}$ is CEC2017 classic functions.

Table 1. High-dimensional unimodal functions

Function	Range	$f_{\min}$
$F_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100,100]$	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10,10]$	0
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	$[-100,100]$	0
$F_4(x) = \max\left\{\lvert x_i \rvert, 1 \le i \le n\right\}$	$[-100,100]$	0
$F_5(x) = \sum_{i=1}^{n} i x_i^2$	$[-10,10]$	0
$F_6(x) = \sum_{i=1}^{n} \left(\left[x_i + 0.5\right]\right)^2$	$[-100,100]$	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	$[-1.28,1.28]$	0
$F_8(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{(i+1)}$	$[-1,1]$	0
$F_9(x) = \sum_{i=1}^{n} i x_i^4$	$[-1.28,1.28]$	0
$F_{10}(x) = x_1^2 + 10^6 \sum_{i=2}^{n} x_i^6$	$[-100,100]$	0
$F_{11}(x) = 10^6 x_1^2 + \sum_{i=2}^{n} x_i^6$	$[-100,100]$	0

Table 2. High-dimensional multimodal functions

Function	Range	f_{min}
$F_{12}(x)=\sum_{i=1}^{n}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	[-500,500]	-418.9829*dim
$F_{13}(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	[-5.12,5.12]	0
$F_{14}(x)=-20\exp\left(-0.2\sqrt{1/n\sum_{i=1}^{n}x_i^2}\right)-\exp\left(1/n\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)+20+e$	[-32,32]	0
$F_{15}(x)=1/4000\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(x_i/\sqrt{i}\right)+1$	[-600,600]	0
$F_{16}(x)=\frac{\pi}{n}\left\{10\sin\left(\pi y_1\right)+\sum_{i=1}^{n-1}\left(y_i-1\right)^2\left[1+10\sin^2\left(\pi y_{i+1}\right)\right]+\left(y_n-1\right)^2\right\}+\sum_{i=1}^{n}u\left(x_i,10,100,4\right)$	[-50,50]	0
$F_{17}(x)=0.1\left(\sin^2\left(3\pi x_1\right)+\sum_{i=1}^{n}\left(x_i-1\right)^2\left[1+\sin^2\left(3\pi x_i+1\right)\right]+\left(x_n-1\right)^2\left[1+\sin^2\left(2\pi x_n\right)\right]\right)$	[-50,50]	0
$F_{18}=\sum_{i=1}^{n}\lvert x_i\sin(x_i)+0.1x_i\rvert$	[-10,10]	0
$F_{19}=\sum_{i=1}^{n}\left(0.2x_i^2+0.1x_i^2\sin(2x_i)\right)$	[-10,10]	0
$F_{20}=\sum_{i=1}^{n}x_i^6\left(2+\sin(1/x_i)\right)$	[-1,1]	0

5.3 Comparison Algorithm and Algorithm Parameter Setting

In order to verify the effectiveness of the proposed algorithm, the effectiveness of the proposed strategy is verified at first. The algorithms in this paper are named AMLCSO1, AMLCSO2, AMLCSO3 and AMLCSO4 respectively under single strategy. The specific description is shown in Table 3.

Table 3. Improved strategy algorithm

Algorithm	Good point set strategy	Gaussian mutation strategy	Learning update strategy	Adaptive weighting strategy
AMLCSO1	√			
AMLCSO2		√		
AMLCSO3			√	
AMLCSO4				√

In order to verify the effectiveness of the algorithm, three improved ICSO algorithms [44-46] and other excellent intelligent algorithms [38-43], such as pigeon swarm algorithm, wolf swarm algorithm, particle swarm optimization algorithm, genetic algorithm and geographic biological algorithm, are selected to conduct simulation experiments with AMLCSO algorithm proposed in this paper. The relevant parameter settings are shown in Table 4.

Table 4. Algorithm parameter settings used

Algorithm	Related parameters
AMLCSO	G=10, N=100, Nr=0.3N, Nh=0.5N, Nm=0.2Nh, F=(0,2) ω_{min}=0.2, ω_{max}=0.9
CSO [38]	G=10, N=100, Nr=0.3N, Nh=0.5N, Nm=0.2Nh, F=(0,2)
PIO [39]	N=100, Vmax=0.5; R=0.2
GWO [40]	M=1000, N=100
PSO [41]	N=100, w=[0.2, 0.9]
GA [42]	N=100, M=0.1
BBO [43]	N=100, I=1

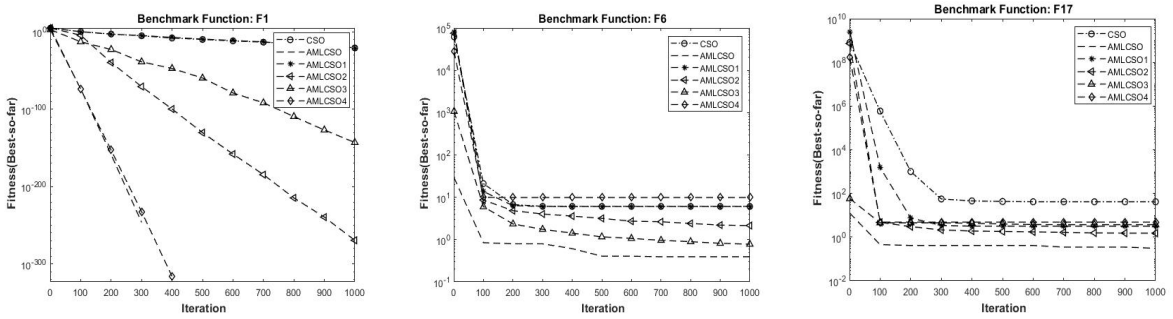
5.4 Analysis of the Effectiveness of the Strategy

Policy Validity Verification. To verify the effectiveness of the proposed improvement strategies in this paper, this section compares the proposed AMLCSO with AMLCSO1, AMLCSO2, AMLCSO3, and AMLCSO4 of the separate strategies. Simulation experiments are carried out in a 30-dimensional, 1000-degree operation environment. The operating parameters of the algorithm are the same as those of AMLCSO in Table 4.

Table 5 shows the mean (Mean) and standard deviation (std) results of the unimodal functions and the multimodal functions. Fig. 3 shows some of the algorithm convergence graphs of the unimodal and the multimodal functions under different strategies.

Table 5. Test function results of AMLCSO algorithm under different strategies

Fun		CSO	AMLCSO1	AMLCSO2	AMLCSO3	AMLCSO4	AMLCSO
F_1	Mean	5.4119e-47	3.6416e-46	3.0779e-139	3.1353e-89	**0**	**0**
	Std	2.2673e-46	1.3012e-45	1.5040e-138	4.8924e-89	**0**	**0**
F_2	Mean	1.0657e-39	3.6343e-40	8.1931e-95	8.5113e-52	**0**	**0**
	Std	2.4925e-39	8.3831e-40	4.2703e-94	6.2720e-52	**0**	**0**
F_3	Mean	1.6330e+03	1.2994e+03	4.3803e-73	1.2229e-42	**0**	**0**
	Std	892.4371	813.3543	2.3577e-72	2.6117e-42	**0**	**0**
F_4	Mean	4.5603	4.2602	1.7290e-59	1.3417e-37	**0**	**0**
	Std	3.4103	5.5018	6.3311e-59	9.9448e-38	**0**	**0**
F_5	Mean	7.2790e-49	4.4695e-48	1.7915e-143	5.2435e-90	**0**	**0**
	Std	2.6358e-48	1.9653e-47	9.6471e-143	1.1080e-89	**0**	**0**
F_6	Mean	2.2558	2.1068	2.1702	1.4004	4.1603	**1.0205**
	Std	0.3324	0.3781	0.2887	0.3278	**0.3092**	**0.4402**
F_7	Mean	0.0041	0.0021	2.4212e-05	1.3737e-04	4.4979e-05	**1.2008e-05**
	Std	0.0026	0.0013	1.6006e-05	5.1899e-05	2.6193e-05	**9.7240e-06**
F_8	Mean	2.5309e-40	2.3761e-40	1.0785e-98	2.4450e-52	**0**	**0**
	Std	5.5103e-40	9.6625e-40	1.1264e-98	1.7276e-52	**0**	**0**
F_9	Mean	3.2436e-22	2.9031e-27	2.7279e-234	3.5003e-159	**0**	**0**
	Std	1.7405e-21	1.5553e-26	0	1.0533e-158	**0**	**0**
F_{10}	Mean	1.6561e+10	1.4581e+14	2.9993e-140	3.9160e-169	**0**	**0**
	Std	8.9182e+10	5.4678e+14	1.6152e-139	0	**0**	**0**
F_{11}	Mean	1.3393e-06	1.5404e+08	2.7554e-200	2.1898e-176	**0**	**0**
	Std	7.2122e-06	3.0600e+08	0	0	**0**	**0**
F_{12}	Mean	-5.8099e+03	-9.007e+03	-9.0425e+03	-8.7309e+03	-5.3290e+03	**-9.8115e+03**
	Std	533.6261	348.0928	1.1455e+03	524.0823	**500.1422**	744.6764
F_{13}	Mean	0	0	0	0	0	**0**
	Std	0	0	0	0	0	**0**
F_{14}	Mean	4.6777e-15	4.6777e-15	1.0066e-15	8.8818e-16	8.8818e-16	**8.8818e-16**
	Std	8.8620e-16	8.8620e-16	6.3773e-16	0	0	**0**
F_{15}	Mean	0	0	0	0	0	**0**
	Std	0	0	0	0	0	**0**
F_{16}	Mean	0.1625	0.1379	0.1264	0.0511	0.5121	**0.0175**
	Std	0.0455	0.0420	0.0276	0.0155	0.0791	**0.0064**
F_{17}	Mean	1.2931	1.3052	1.8517	0.8687	2.7648	**0.1679**
	Std	0.1681	0.2230	0.1579	0.1273	0.0912	**0.0473**
F_{18}	Mean	2.8362e-39	6.1200e-38	1.0878e-77	3.9801e-51	0	**0**
	Std	5.6898e-39	2.6788e-37	5.8570e-77	4.5058e-51	0	**0**
F_{19}	Mean	5.7665e-49	1.3277e-47	7.8323e-148	2.4453e-92	0	**0**
	Std	2.0705e-48	7.0557e-47	2.5187e-147	4.9660e-92	0	**0**
F_{20}	Mean	3.0755e-11	6.7180e-12	0	2.9713e-232	0	**0**
	Std	1.5412e-10	2.3480e-11	0	0	0	**0**



(a) F1 (b) F6 (c) F17

Fig. 3. Convergence diagram of partial test functions

Analysis of Results. In Table 5, the standard deviation of 15 functions is 0 in these 20 functions. The results clearly show that the AMLCSO algorithm combined with all strategies has good convergence and stability. Although the mean and standard deviation of the functions $F_6F_7F_{12}F_{16}F_{17}$ are not 0, they have little improvement in optimizing ability compared with other strategy algorithms. But the final optimal value is better than other algorithms. Therefore, it can be seen that the combined improvement strategy can improve the algorithm's optimization ability more effectively. In Fig. 3, the optimization effect under different improvement strategies is different. AMLCSO2, AMLCSO3 and AMLCSO4 have the best optimization results, but AMLCSO under all strategies is better than a single strategy.

5.5 Algorithm Accuracy Analysis

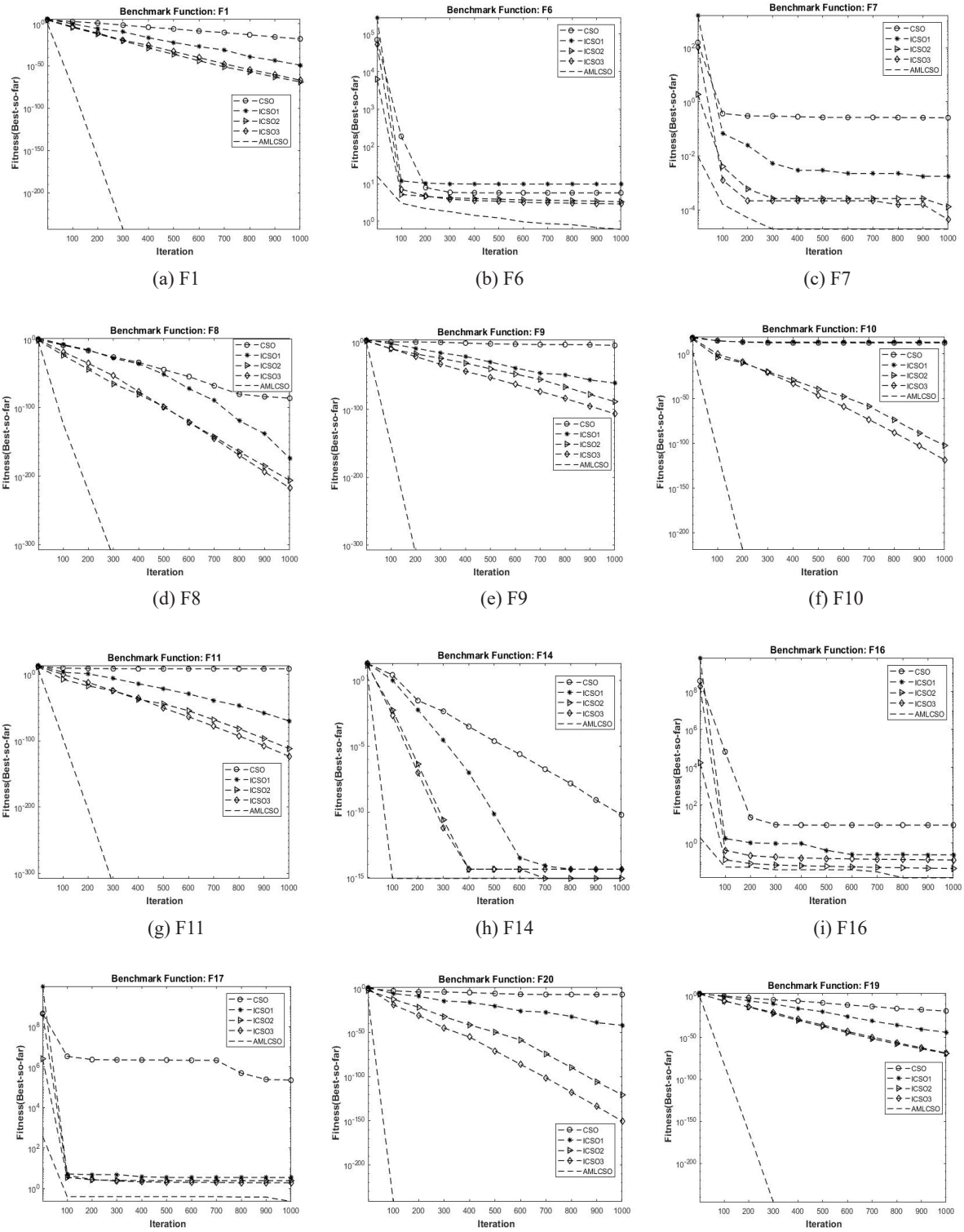
Experimental Results of the Algorithm. In order to verify the convergence accuracy performance of the AMLCSO algorithm proposed in this paper, AMLCSO and the improved CSO algorithm [38] are tested. The population dimension (dim) is 30, 50, and 100 and the number of population iterations is 1000. In this way, the test function F_1-F_{20} is tested. The specific experimental results are shown in Table 6 and some effects are shown in Fig. 4.

Experimental Results of the Algorithm. In Table 6, we can see that the mean and standard deviation of 9 benchmark functions in the unimodal function are zero except the function F_6F_7. It indicates that AMLCSO algorithm has higher optimization accuracy and stability than other improved CSO algorithms in the same series; The standard deviation of six benchmark functions in the multimodal function are zero expect the function $F_{12}F_{16}F_{17}$. So the global search performance of AMLCSO algorithm is higher than other algorithms in the same series; With the continuous improvement of dimension, the mean and standard deviation of AMLCSO algorithm in multiple benchmark functions are always zero. It indicates that the optimization ability, stability and accuracy of the algorithm have not been affected. For other algorithms, the average and standard deviation of multiple functions decrease by multiple orders of magnitude with the improvement of dimensionality. For unimodal function F_9, the average value of all improved CSO algorithms of the same series differs by 40 orders of magnitude between dimension 30 and dimension 100. For the multi-modal function F_{20}, both the average and standard deviation of ICSO [46] differ by 80 orders of magnitude between dimension 30 and dimension 100. And both the average and standard deviation of ICSO [44] differ by 20 orders of magnitude between dimension 30 and dimension 100. It can be seen that the AMLCSO algorithm proposed in this paper has higher accuracy and better stability than the improved CSO algorithm of the same series.

We can see that the AMLCSO algorithm can obtain the optimal solution the optimum within 400 iterations, whether it is a unimodal function or a multimodal function from Fig. 4. It indicates that the solution accuracy of this algorithm is significantly higher than other algorithms.

Table 6. Test function results of the same series of Chicken swarm optimizations

Fun		CSO			ICSO [45]			ICSO [46]			ICSO [44]			AMLC-SO		
	dim	30	50	100	30	50	100	30	50	100	30	50	100	30	50	100
F_1	Mean	5.41e-47	3.77e-25	32.3996	1.51e-57	7.04e-45	5.30e-32	8.31e-75	3.26e-68	3.09e-65	2.25e-75	1.36e-65	4.21e-61	**0**	**0**	**0**
	Std	2.26e-46	1.07e-17	131.4004	3.14e-57	2.25e-44	2.78e-31	2.75e-74	1.43e-67	5.33e-65	3.49e-75	4.56e-65	6.84e-61	**0**	**0**	**0**
F_2	Mean	1.06e-39	1.46e-27	2.92e-15	2.77e-43	6.42e-37	5.56e-31	1.02e-44	1.62e-37	1.78e-34	9.18e-48	1.04e-39	1.50e-35	**0**	**0**	**0**
	Std	2.49e-39	4.09e-27	4.69e-15	5.49e-43	2.18e-36	1.05e-30	9.20e-45	1.39e-37	1.94e-34	7.96e-48	1.48e-38	1.02e-35	**0**	**0**	**0**
F_3	Mean	1.63e+03	6.78e+03	3.12e+04	481.0958	6.93e+03	4.13e+04	5.28e-24	6.96e-22	4.68e-19	2.00e-32	8.03e-28	1.19e-24	**0**	**0**	**0**
	Std	892.4371	3.10e+03	8.81e+03	750.7385	3.88e+03	1.46e+04	1.87e-23	1.73e-21	1.59e-18	6.15e-32	3.09e-27	2.57e-24	**0**	**0**	**0**
F_4	Mean	4.5603	22.5783	4.5603	97.3214	100	100	1.05e-32	8.48e-30	9.51e-28	2.74e-32	4.74e-29	3.58e-26	**0**	**0**	**0**
	Std	3.4103	6.3490	3.4103	7.2919	0	0	1.00e-32	7.98e-30	1.86e-27	2.05e-32	3.56e-29	3.57e-26	**0**	**0**	**0**
F_5	Mean	7.27e-49	1.15e-19	9.1865	1.09e-57	2.11e-45	8.38e-34	5.88e-76	2.23e-69	2.05e-65	3.08e-76	1.87e-66	5.36e-61	**0**	**0**	**0**
	Std	2.63e-48	2.78e-19	21.0914	4.78e-57	7.13e-45	1.32e-33	9.43e-76	4.19e-69	3.03e-65	3.50e-76	2.98e-66	1.59e-60	**0**	**0**	**0**
F_6	Mean	2.2558	6.3949	55.5615	4.4094	9.9157	22.1112	1.3359	4.3411	13.9325	**0.9516**	3.6122	12.2	1.0205	**0.5749**	**1.7030**
	Std	0.3324	0.6251	68.2953	1.3521	0.1620	**0.2344**	0.3954	0.7790	0.99	**0.2750**	0.5163	1.0377	0.4402	**0.1336**	0.4086
F_7	Mean	0.0041	0.1722	11.3568	8.27e-04	0.0012	0.0039	2.57e-04	2.79e-04	3.07e-04	1.13e-04	1.18e-04	1.59e-04	**1.20e-05**	**1.41e-05**	**8.25e-06**
	Std	0.0026	0.1837	6.1229	3.70e-04	6.08e-04	0.0023	1.17e-04	1.29e-04	1.37e-04	6.33e-05	5.46e-05	6.72e-05	**9.72e-06**	**1.49e-05**	**7.13e-06**
F_8	Mean	2.53e-40	1.28e-26	3.91e-14	3.1e-188	7.5e-145	8.13e-28	1.4e-193	2.9e-178	3.8e-165	4.6e-226	8.4e-215	2.1e-212	**0**	**0**	**0**
	Std	5.51e-40	6.81e-26	1.63e-13	0	3.9e-144	4.38e-27	0	0	0	0	0	0	**0**	**0**	**0**
F_9	Mean	3.24e-22	0.0017	9.9474	3.27e-79	4.55e-57	6.05e-22	1.0e-102	4.69e-83	4.95e-73	2.0e-116	1.3e-102	2.69e-96	**0**	**0**	**0**
	Std	1.74e-21	0.0065	5.82	1.48e-78	2.26e-56	3.25e-21	5.4e-102	1.78e-82	2.12e-72	8.1e-116	5.7e-102	4.92e-96	**0**	**0**	**0**
F_{10}	Mean	1.65e+10	3.55e+13	7.20e+15	2.7e-116	7.55e+12	2.65e+14	2.14e-59	3.40e-85	2.35e-63	1.3e-134	3.7e-117	5.7e-107	**0**	**0**	**0**
	Std	8.91e+10	1.36e+14	6.67e+15	1.3e-115	2.28e+13	4.15e+14	1.15e-58	1.81e-84	1.26e-62	3.5e-134	1.4e-116	2.0e-106	**0**	**0**	**0**
F_{11}	Mean	1.33e-06	7.31e+07	4.92e+09	9.29e-82	1.47e+07	2.19e+08	4.8e-126	4.66e-95	8.53e-77	3.39e-72	1.5e-123	1.8e-112	**0**	**0**	**0**
	Std	7.21e-06	2.46e+08	5.24e+09	5.00e-81	5.50e+07	2.52e+08	2.5e-125	2.51e-94	4.59e-76	1.83e-71	8.0e-123	9.6e-112	**0**	**0**	**0**
F_{12}	Mean	-5.8e+03	-7.7e+03	-1.1e+04	-1.4e+04	-2.1e+04	-3.5e+04	-5.5e+03	-7.5e+03	-1.1e+04	-1.4e+04	-1.2e+04	-2.5e+04	**-9.8e+03**	**-2.05e+04**	**-4.0e+04**
	Std	**533.6261**	838.8484	1.49e+03	1.22e+03	2.52e+03	2.85e+03	628.5586	889.1198	1.81e+03	1.31e+03	1.28e+03	1.73e+03	744.6764	**462.6124**	**1.10e+03**
F_{13}	Mean	0	2.84e-14	1.05e-05	0	0	0	0	0	0	0	0	0	**0**	**0**	**0**
	Std	0	5.63e-14	1.81e-05	0	0	0	0	0	0	0	0	0	**0**	**0**	**0**
F_{14}	Mean	4.67e-15	4.52e-10	4.1683	0.6643	1.9947	9.9511	3.84e-15	3.13e-15	3.37e-15	4.44e-15	4.32e-15	4.44e-15	**8.88e-16**	**8.88e-16**	**8.88e-16**
	Std	8.86e-16	7.66e-10	3.4739	3.5774	5.9790	9.9515	1.32e-15	1.71e-15	1.62e-15	0	6.37e-16	0	**0**	**0**	**0**
F_{15}	Mean	0	0.0227	0.3822	0	0	0	0	0	0	0	0	0	**0**	**0**	**0**
	Std	0	0.0856	0.6628	0	0	0	0	0	0	0	0	0	**0**	**0**	**0**
F_{16}	Mean	0.1625	125.3347	4.87e+05	0.1466	0.3234	4.1574	0.0703	0.1666	0.3641	0.0555	0.1463	0.3147	**0.0175**	**0.0179**	**0.0184**
	Std	0.0455	622.9528	4.22e+05	0.0448	0.0731	6.4431	0.0257	0.0407	0.0642	0.0151	0.0445	0.0532	**0.0064**	**0.0057**	**0.0048**
F_{17}	Mean	1.2931	8.82e+04	9.73e+06	1.2688	3.5648	1.06e+04	0.8196	2.3411	7.7509	0.6512	2.1446	7.6863	**0.1679**	**0.5407**	**1.3400**
	Std	0.1681	2.05e+05	7.91e+06	0.2220	0.5856	5.74e+04	0.2211	0.4513	0.8894	0.1418	0.2826	0.4295	**0.0473**	**0.2838**	**0.2820**
F_{18}	Mean	2.83e-39	0.0051	0.0786	1.28e-41	8.40e-34	1.30e-27	9.73e-45	9.50e-38	3.29e-35	4.04e-48	3.03e-40	3.46e-36	**0**	**0**	**0**
	Std	5.68e-39	0.0132	0.0816	2.89e-41	3.19e-33	2.28e-27	1.38e-44	1.11e-37	2.50e-35	3.69e-48	2.42e-40	2.55e-36	**0**	**0**	**0**
F_{19}	Mean	5.76e-49	1.05e-19	0.0836	1.22e-59	3.81e-46	2.58e-34	3.81e-77	5.44e-70	1.55e-66	3.10e-77	5.64e-68	1.00e-62	**0**	**0**	**0**
	Std	2.07e-48	3.29e-19	0.2949	3.95e-59	1.73e-45	5.73e-34	6.49e-77	9.90e-70	3.27e-66	5.00e-77	8.00e-68	3.84e-62	**0**	**0**	**0**
F_{20}	Mean	3.07e-11	5.42e-05	0.0064	2.58e-96	3.35e-09	1.31e-05	1.2e-135	6.0e-108	3.08e-87	6.8e-162	1.2e-145	5.5e-139	**0**	**0**	**0**
	Std	1.54e-10	1.16e-04	0.0028	1.26e-95	1.80e-08	2.00e-05	3.29e-19	3.2e-107	1.65e-86	2.6e-161	4.8e-145	1.6e-138	**0**	**0**	**0**



(a) F1 (b) F6 (c) F7

(d) F8 (e) F9 (f) F10

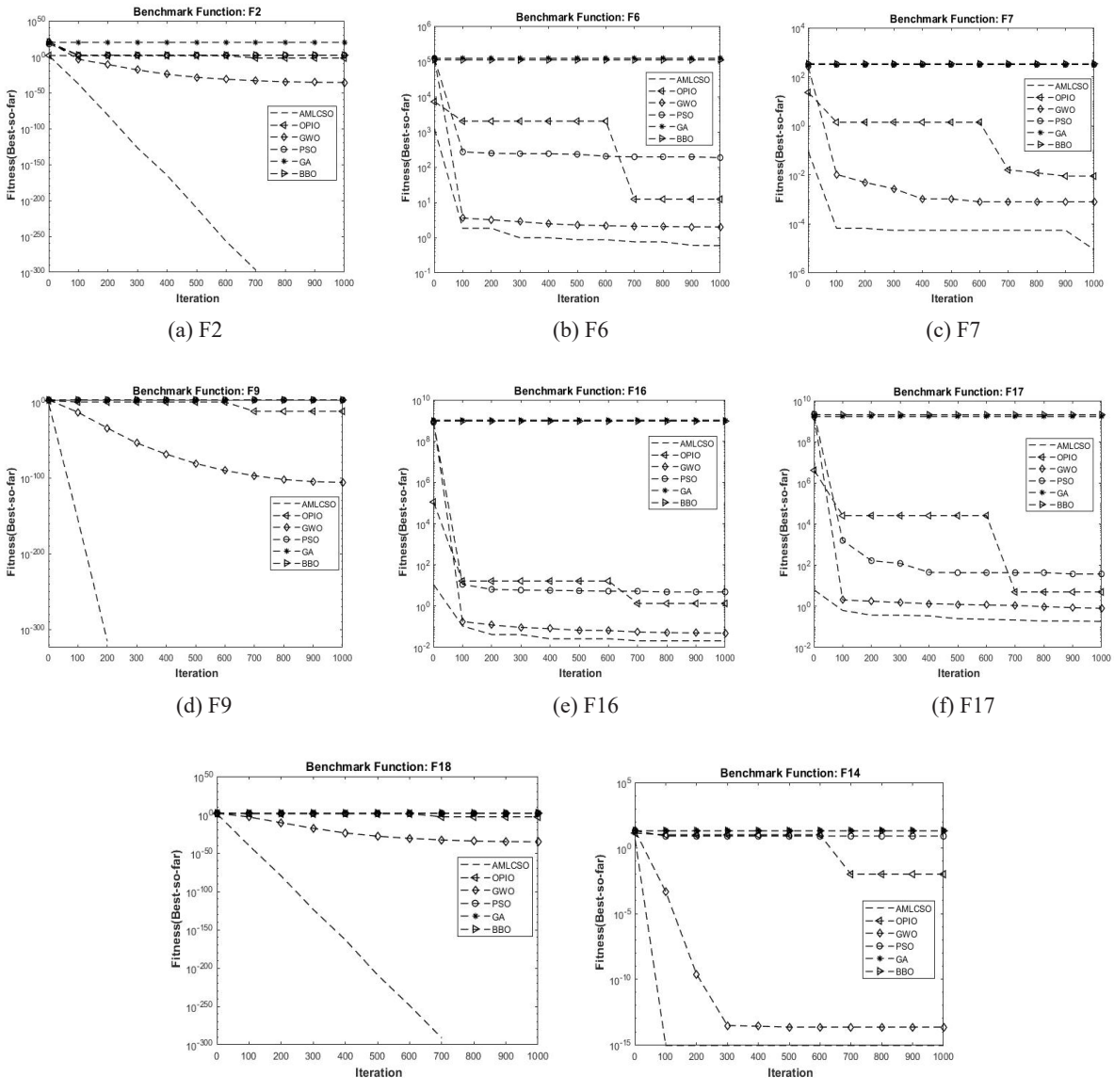
(g) F11 (h) F14 (i) F16

(j) F17 (k) F20 (l) F19

Fig. 4. Part of the test function image

5.6 Analysis of Algorithm Convergence Speed

Experimental Results of the Algorithm. In order to further verify the convergence speed of AMLCSO, the functions F_1 - F_{20} are tested with population iteration times (M) of 100, 500 and 1000 and dimension of 50. The AMLCSO in this paper is compared with pigeon swarm algorithm, wolf pack algorithm, particle swarm optimization algorithm, genetic algorithm and geographic bio-algorithm. The experimental results are shown in Table 7 and some experimental results are shown in Fig. 5.



(a) F2 (b) F6 (c) F7

(d) F9 (e) F16 (f) F17

(g) F18 (h) F14

Fig. 5. 100-dimensional partial function image

Table 7. Test results of functions with iterations of 100, 500, and 1000

Fun		PIO			GWO			PSO			GA			BBO			AMLCSO		
	M	100	500	1000	100	500	1000	100	500	1000	100	500	1000	100	500	1000	100	500	1000
F_1	Mean	0.0014	0.0015	0.0019	0.0110	4.23e-29	2.88e-62	278.6975	204.5226	183.2289	1.16e+05	1.17e+05	1.15e+05	1.14e+05	1.16e+05	1.00e+05	**6.40e-73**	**0**	**0**
	Std	0.0010	0.0017	0.0027	0.0045	3.98e-29	3.00e-62	37.0624	17.4411	20.1611	5.66e+03	7.79e+03	8.16e+03	6.61e+03	6.33e+03	6.52e+03	**2.29e-72**	**0**	**0**
F_2	Mean	0.0244	0.0336	0.0280	0.0257	1.44e-17	2.15e-36	7.35e+04	532.1669	132.5603	2.78e+21	8.66e+21	2.46e+21	3.11e+09	3.03e+10	179.3570	**1.00e-38**	**8.7e-201**	**0**
	Std	0.0121	0.0163	0.0177	0.0064	5.94e-18	1.69e-36	2.05e+05	1.56e+03	32.5422	5.74e+21	3.67e+22	4.70e+21	1.66e+10	1.63e+11	12.1706	**1.53e-38**	**0**	**0**
F_3	Mean	0.0071	0.0064	0.0039	760.6208	2.69e-05	1.40e-14	4.97e+03	1.96e+03	1.25e+03	2.78e+05	2.92e+05	2.69e+05	2.25e+05	2.06e+05	1.72e+05	**5.09e-68**	**0**	**0**
	Std	0.0109	0.0169	0.0047	429.1075	4.18e-05	4.35e-14	1.34e+03	389.3442	267.0240	6.16e+04	7.11e+04	6.74e+04	5.20e+04	4.44e+04	4.71e+04	**2.08e-67**	**0**	**0**
F_4	Mean	0.0179	0.0185	0.0207	1.5758	1.56e-06	6.10e-14	9.1821	5.6562	4.9963	89.8709	90.0727	90.7049	90.2072	90.2398	88.4510	**9.93e-36**	**1.1e-196**	**0**
	Std	0.0078	0.0103	0.0111	0.3918	1.37e-06	1.34e-13	1.2862	0.5279	0.4004	2.9516	2.2231	1.9836	1.9875	2.1289	3.24	**1.80e-35**	**0**	**0**
F_5	Mean	6.53e-04	5.16e-04	8.24e-04	0.0024	7.70e-30	6.38e-63	5.94e+03	4.82e+03	4.53e+03	2.73e+04	2.72e+04	2.82e+04	2.81e+04	2.88e+04	2.76e+04	**1.41e-72**	**0**	**0**
	Std	6.38e-04	6.88e-04	0.0011	0.0012	9.23e-30	9.71e-63	709.6784	446.6580	636.8489	2.45e+03	2.15e+03	2.41e+03	2.19e+03	2.37e+03	1.99e+03	**5.24e-72**	**0**	**0**
F_6	Mean	12.3952	12.2164	12.2273	**4.5604**	**1.2065**	1.1377	276.5169	207.1656	188.4768	1.13e+05	1.15e+05	1.16e+05	1.11e+05	1.13e+05	1.00e+05	2.2392	0.9947	**0.5749**
	Std	0.2327	0.4264	0.5980	**0.9770**	**0.3639**	0.4661	32.8325	19.6006	20.9110	9.68e+03	7.99e+03	8.13e+03	1.03e+04	7.30e+03	1.00e+05	1.0177	0.3319	**0.1336**
F_7	Mean	0.0029	0.0025	0.0030	0.0136	9.61e-04	4.79e-04	336.8198	358.5266	339.5036	347.9808	361.7189	354.6302	336.6395	348.7848	344.7198	**1.82e-04**	**3.57e-05**	**1.41e-05**
	Std	0.0034	0.0026	0.0027	0.0046	3.47e-04	2.92e-04	51.9575	39.6555	62.0928	44.0504	40.6652	50.8782	56.2976	48.9768	47.5572	**1.52e-04**	**3.29e-05**	**1.49e-05**
F_8	Mean	9.66e-12	4.39e-12	1.74e-11	3.55e-23	1.4e-128	9.4e-262	0.8261	0.7382	0.7860	0.8209	0.7888	0.7491	0.5376	0.5232	0.2426	**1.3e-105**	**0**	**0**
	Std	2.11e-11	8.54e-12	4.88e-11	1.90e-22	4.7e-128	0	0.3081	0.2236	0.2518	0.2206	0.2610	0.1974	0.2417	0.2106	0.1114	**7.4e-105**	**0**	**0**
F_9	Mean	3.78e-11	1.03e-10	1.94e-11	6.46e-10	5.01e-53	3.2e-107	330.0873	341.2987	323.2461	359.3219	350.6077	278.2440	358.5747	354.7477	329.1141	**6.1e-148**	**0**	**0**
	Std	8.76e-11	3.55e-10	3.13e-11	8.97e-10	9.94e-53	8.0e-107	50.2546	40.0921	59.7572	41.4602	40.9810	40.0200	47.0116	42.4556	45.6	**2.9e-147**	**0**	**0**
F_{10}	Mean	3.14e-05	1.83e-04	1.11e-04	4.92e+03	2.88e-48	2.6e-119	3.80e+11	8.51e+10	5.43e+10	4.80e+18	4.33e+18	4.44e+18	3.03e+18	2.96e+18	2.80e+18	**4.2e-100**	**0**	**0**
	Std	1.31e-04	7.23e-04	5.76e-04	6.84e+03	8.87e-48	7.9e-119	1.40e+11	2.86e+10	1.63e+10	8.28e+17	9.13e+17	7.24e+17	5.19e+17	5.88e+17	4.81e+17	**1.36e-99**	**0**	**0**
F_{11}	Mean	0.7960	1.7435	1.1428	0.0128	8.54e-58	3.7e-125	7.69e+05	1.47e+05	9.30e+04	4.52e+12	4.37e+12	4.44e+12	3.21e+12	3.28e+12	3.00e+12	**4.30e-93**	**0**	**0**
	Std	1.7439	6.0979	3.4621	0.0374	4.06e-57	2.0e-124	3.13e+05	5.46e+04	3.01e+04	7.01e+11	7.42e+11	9.7e +11	6.06e+11	5.03e+11	3.00e+12	**2.19e-92**	**0**	**0**
F_{12}	Mean	-1.76e+4	-1.75e+4	-1.73e+4	-9.46e+3	-9.70e+3	-9.7e+03	-5.51e+3	-1.0e+04	-1.0e+04	-3.4e+03	-3.4e+03	-3.4e+03	-3.6e+03	-3.6e+03	-4.6e+03	**-2.0e+4**	**-2.07e+4**	**-2.05e+4**
	Std	4.72e+03	4.75e+03	6.48e+03	985.4546	920.8386	1.10e+03	633.9779	1.10e+03	1.22e+03	594.4487	547.6802	631.3909	588.4754	532.6166	624.8058	**612.5046**	**371.2471**	**462.6124**
F_{13}	Mean	0.0203	0.0148	0.0170	46.5913	1.1779	0	646.4298	638.7350	632.8055	746.6365	750.0365	747.7274	730.6072	729.5750	667.1654	**0**	**0**	**0**
	Std	0.0279	0.0164	0.0266	17.1129	2.8534	0	28.0412	27.5170	25.7247	26.7575	23.8045	29.7052	28.2511	32.0811	25.5617	**0**	**0**	**0**
F_{14}	Mean	0.0137	0.0114	0.0090	0.0167	8.53e-14	1.93e-14	9.0785	8.3915	8.2549	20.7072	20.7135	20.7061	20.7108	20.6822	20.4931	**8.88e-16**	**8.88e-16**	**8.88e-16**
	Std	0.0092	0.0061	0.0059	0.0038	1.02e-14	3.71e-15	0.4647	0.2825	0.2522	0.0985	0.1381	0.1354	0.1103	0.1064	0.0928	**0**	**0**	**0**
F_{15}	Mean	0.0035	0.0042	0.0090	0.0239	0.0028	2.77e-04	9.2356	1.0544	1.0491	1.04e+03	1.03e+03	1.03e+03	1.02e+03	1.01e+03	925.3332	**0**	**0**	**0**
	Std	0.0038	0.0057	0.0176	0.0309	0.0058	0.0015	2.34	0.0049	0.0044	66.2582	66.0679	64.1701	66.4884	76.5585	76.3722	**0**	**0**	**0**
F_{16}	Mean	1.4436	1.3710	1.3838	0.3010	0.0406	0.0412	11.27	5.7188	4.8926	9.79e+08	9.97e+08	1.00e+09	9.67e+08	9.93e+08	8.78e+08	**0.0723**	**0.0329**	**0.0179**
	Std	0.1004	0.1918	0.1351	0.1493	0.0144	0.0201	3.57	0.8094	0.6004	1.28e+08	1.41e+08	1.23e+08	1.77e+08	1.30e+08	1.07e+08	**0.0317**	**0.0110**	**0.0057**
F_{17}	Mean	5.0179	5.0168	5.0167	3.3561	0.9522	0.8574	1.10e+03	57.2321	44.9071	1.89e+09	1.95e+09	2.04e+09	1.99e+09	1.87e+09	1.72e+09	**1.0348**	**0.5311**	**0.5407**
	Std	0.0280	0.0196	**0.0237**	0.5849	0.2995	0.3260	1.15e+03	17.8639	10.3292	2.94e+08	1.96e+08	2.44e+08	2.19e+08	2.52e+08	2.23e+08	**0.5668**	**0.1389**	0.2838
F_{18}	Mean	0.0057	0.0059	0.0069	0.1425	5.05e-04	4.96e-06	63.4122	59.1136	53.6486	111.0510	112.6113	110.0843	107.8287	108.1979	99.5190	**3.93e-39**	**1.1e-201**	**0**
	Std	0.0026	0.0038	0.0038	0.3681	5.95e-04	9.99e-06	4.5250	4.4406	3.8606	5.8530	6.2860	4.8614	6.8364	5.1036	6.5267	**1.15e-38**	**0**	**0**
F_{19}	Mean	1.48e-05	1.11e-05	1.56e-05	3.11e-05	1.23e-31	9.98e-65	56.7746	41.9466	38.2965	219.7382	221.0341	220.7308	217.0216	221.7592	193.3555	**3.62e-75**	**0**	**0**
	Std	1.35e-05	1.25e-05	1.67e-05	1.60e-05	9.89e-32	1.48e-64	7.3322	4.2268	3.5400	19.6665	16.0285	16.8958	18.5968	13.0863	13.2479	**1.3e-74**	**0**	**0**
F_{20}	Mean	4.07e-18	9.08e-18	1.23e-18	1.26e-14	4.77e-68	1.2e-132	5.9491	5.9600	6.0007	6.6571	6.1452	6.3215	5.9588	6.7037	5.0905	**1.1e-219**	**0**	**0**
	Std	1.77e-17	3.54e-17	4.03e-18	1.14e-14	1.34e-67	5.0e-132	1.1535	0.9311	1.0971	1.0852	1.0895	0.9835	1.1639	0.9173	0.9828	**0**	**0**	**0**

Experimental Analysis of Convergence Rate. In Table 7, with the number of iterations increasing, the AMLCSO algorithm is better than other standard algorithms in average value and standard deviation. The AMLCSO algorithm finds the optimal solution after 500 iterations while the means and standard deviation of other standard algorithms change little. For example, in the function F_2, only the wolf pack optimization increased by 36 numbers, while other standard algorithms only increased by an order of magnitude within 10. When the AMLCSO algorithm is iterated 500, the number is increased by 160 and the optimal solution is directly obtained after 1000 iterations. This shows that the convergence speed of the AMLCSO algorithm is better than other standard algorithms and the adaptive weights introduced by the Chicken swarm optimization can improve the convergence speed of the algorithm. The standard deviation of the function F_{17} is lower than all standard algorithms and the stability is slightly worse. But for the average value, the AMLCSO algorithm can effectively converge to the global optimal value. Other algorithms are more likely to fall into the local optimum. The result shows that the addition of the learning update strategy with normal distribution and the Gaussian mutation strategy can make the algorithm jump out of the local optimum and find the optimal value.

Fig. 5 shows the partial function convergence diagrams of the six algorithms. In the convergence graph of function F_6, most of the standard algorithms have fallen into the local optimum when the iteration is about 600 times in the middle. At this moment, most of the individuals in the population have gathered at the local optimum, which will eventually lead more individuals to gather here. As a result, the algorithm will not jump out of the local optimum. When the algorithm in this paper iterates to 100, the advantage of accuracy is obvious and the convergence speed is obviously better than other standard algorithms. It reflects that the introduction of adaptive weighting strategy can improve the convergence accuracy of the algorithm. In the subsequent iterative, the algorithm jumped out of the local optimum many times and obtained the optimal solution. The result reflects that the addition of the learning update strategy with the normal distribution and the Gaussian mutation strategy can prevent the algorithm from falling into the local optimum to a certain extent. And the convergence speed and optimization accuracy of the algorithm are improved ultimately.

5.7 Statistical Analysis of Benchmark Function

This section uses Wilcoxon's rank sum test to evaluate statistical performance. The performance of AMLCSO is compared with other algorithms by 5% significance. The experimental results are shown in Table 8.

In Wilcoxon's rank sum test, the statistical performance of AMLCSO and other algorithms are determined by the values of p and h. When the value of p is less than 5% or h is equal to 1, it indicates that there is obvious difference between AMLCSO and the comparison algorithm. We should note that if the results of the two algorithms are the same, the value of P is NaN. But it does not mean that the performance of the two algorithms is identical.

From Table 8, we can see that AMLCSO has obvious difference with CSO, OPIO, PSO, GA and BBO in all performance. For other algorithms, the global optimization has obtained in F_{13} and F_{15}. There are obvious differences between AMLCSO and ICSO [45] in 17 functions except F_{12}. And AMLCSO has obvious differences with ICSO [46], ICSO [44] and GWO in 18 functions. For most functions, AMLCSO is significantly different from CSO, ICSO [46], ICSO [44]. ICSO [45], GWO, OPIO, PSO, GA and BBO.

Table 8. Experimental results of Wilcoxon rank sum test under 50 dimensions

Fun		CSO	ICSO [42]	ICSO [43]	ICSO [41]	OPIO	GWO	PSO	GA	BBO
F_1	p	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12
	h	1	1	1	1	1	1	1	1	1
F_2	p	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12
	h	1	1	1	1	1	1	1	1	1
F_3	p	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12
	h	1	1	1	1	1	1	1	1	1
F_4	p	1.21e-12	1.69e-14	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12
	h	1	1	1	1	1	1	1	1	1
F_5	p	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12
	h	1	1	1	1	1	1	1	1	1
F_6	p	3.02e-11	3.02e-11	3.02e-11	3.02e-11	3.02e-11	3.02e-11	3.02e-11	3.02e-11	3.02e-11
	h	1	1	1	1	1	1	1	1	1
F_7	p	3.02e-11	3.02e-11	3.69e-11	3.16e-10	6.07e-11	3.02e-11	3.02e-11	3.02e-11	3.02e-11
	h	1	1	1	1	1	1	1	1	1
F_8	p	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12
	h	1	1	1	1	1	1	1	1	1
F_9	p	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12
	h	1	1	1	1	1	1	1	1	1

Table 8. Experimental results of Wilcoxon rank sum test under 50 dimensions (continue)

Fun		CSO	ICSO [42]	ICSO [43]	ICSO [41]	OPIO	GWO	PSO	GA	BBO
F_{10}	p	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12
	h	1	1	1	1	1	1	1	1	1
F_{11}	p	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12
	h	1	1	1	1	1	1	1	1	1
F_{12}	p	3.02e-11	0.7845	3.02e-11	3.02e-11	2.96e-05	3.02e-11	3.02e-11	3.02e-11	3.02e-11
	h	1	0	1	1	1	1	1	1	1
F_{13}	p	0.0014	NaN	NaN	NaN	1.21e-12	0.3337	1.21e-12	1.21e-12	1.21e-12
	h	1	0	0	0	1	0	1	1	1
F_{14}	p	1.21e-12	2.07e-13	5.89e-08	1.68e-14	1.21e-12	5.57e-13	1.21e-12	1.21e-12	1.21e-12
	h	1	1	1	1	1	1	1	1	1
F_{15}	p	0.0028	NaN	NaN	NaN	1.21e-12	0.3337	1.21e-12	1.21e-12	1.21e-12
	h	1	0	0	0	1	0	1	1	1
F_{16}	p	3.02e-11	3.02e-11	5.07e-10	3.02e-11	3.02e-11	9.26e-09	3.02e-11	3.02e-11	3.02e-11
	h	1	1	1	1	1	1	1	1	1
F_{17}	p	3.02e-11	3.02e-11	3.02e-11	3.02e-11	3.02e-11	3.02e-11	3.02e-11	3.02e-11	3.02e-11
	h	1	1	1	1	1	1	1	1	1
F_{18}	p	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12
	h	1	1	1	1	1	1	1	1	1
F_{19}	p	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12
	h	1	1	1	1	1	1	1	1	1
F_{20}	p	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12	1.21e-12
	h	1	1	1	1	1	1	1	1	1

6 Conclusion

The standard Chicken swarm optimization has low solution accuracy and tends to fall into the local optimum on the later stage of iteration. To solve the problems, an adaptive mutation learning Chicken swarm optimization is proposed in this paper. Firstly, the good point set is introduced into the initial population to enhance the ergodicity of the initial population; Secondly, an adaptive weight strategy is adopted to improve the convergence speed of the algorithm; Thirdly, the Gaussian mutation strategy is utilized to avoid the algorithm falling into the local optimum; Finally, the learning and updating strategy of normal distribution is introduced to improve the accuracy of the algorithm. In the paper, not only the effectiveness of all strategies is analyzed, but also simulation experiments are carried out on twenty test functions. And the AMLCSO algorithm is simulated with the improved chicken swarm algorithm and multiple standard algorithms of the same series. The results show that the algorithm proposed has strong competitiveness in convergence speed, solution accuracy and stability.

In the future research, some directions are worthy of attention. For example, the method can be generalized to all practical problems where other algorithms can be applied, such as the optimal parameter selection problem, the multi-objective optimization problem, the knapsack problem and the image threshold segmentation problem.

7 Acknowledgment

This research is supported by Jilin Science and Technology Innovation Development Program Projects (No. 20190302202).

References

[1] Q.-M. Yan, R.-P. Ma, Y.-X. Ma, J.-J. Wang, Adaptive simulated annealing particle swarm optimization algorithm, Journal of Xidian University 48(4)(2021) 120-127.

[2] Y. Li, W.-S. Mu, X.-Q. Chu, Z.-T. Fu, K-means clustering algorithm based on improved quantum particle swarm optimization and its application, Control and Decision 37(4)(2022) 839-850.

[3] M.A. Al-Betar, M.A. Awadallah, H. Faris, I. Aljarah, A.I. Hammouri, Natural selection methods for Grey Wolf Optimizer, Expert Systems with Applications 113(2018) 481-498.

[4] H.-E. Tseng, C.-C. Chang, S.-C. Lee, Y.-M. Huang, Hybrid bidirectional ant colony optimization (hybrid BACO): An algorithm for disassembly sequence planning, Engineering Applications of Artificial Intelligence 83(2019) 45-56.

[5] J. Li, Y. Xia, B. Li, Z. Zeng, A pseudo-dynamic search ant colony optimization algorithm with improved negative feedback mechanism, Cognitive Systems Research 62(2020) 1-9.

[6] M.G.H. Omran, S. Al-Sharhan, Improved continuous Ant Colony Optimization algorithms for real-world engineering optimization problems, Engineering Applications of Artificial Intelligence 85(2019) 818-829.

[7] X. Huang, G. Xu, F. Xiao, Optimization of a Novel Urban Growth Simulation Model Integrating an Artificial Fish Swarm Algorithm and Cellular Automata for a Smart City, Sustainability 13(4)(2021) 2338.

[8] J. Zhou, G. Qi., C. Liu, A Chaotic Parallel Artificial Fish Swarm Algorithm for Water Quality Monitoring Sensor Networks

3D Coverage Optimization, Journal of Sensors 2021(2021) 1-12
[9] Z. Hua, Y. Xiao, J. Cao, Misalignment Fault Prediction of Wind Turbines Based on Improved Artificial Fish Swarm Algorithm, Entropy 23(6)(2021) 692.
[10]X.B. Meng, Y. Liu, X.Z. Gao, H.Z. Zhang, A New Bio-inspired Algorithm: Chicken Swarm Optimization, Berlin: Springer International Publishing, 2014.
[11]D. Zouache, Y.O. Arby, F. Nouioua, F.B. Abdelaziz, Multi-objective Chicken swarm optimization: A novel algorithm for solving multi-objective optimization problems, Computers & Industrial Engineering 129(2019) 377-391.
[12]D. Wu, S. Xu, F. Kong, Convergence Analysis and Improvement of the Chicken swarm optimization Algorithm, IEEE Access 4(2016) 9400-9412.
[13]X. Yu, L. Zhou, Q. Yu, M. Hu, F. Zhang, Localization Algorithm for Mine Wireless Sensor Network Based on Rigid Cluster and Chicken Swarm Optimization, Journal of Southwest Jiaotong University 54(4)(2019) 870-878.
[14]L. Wang, X. Wang, Y. Zheng, H. Sun, J. Li, M. Zhao, Reactive Power Optimization of Distribution Network Considering Output Correlation of Multiple Wind Turbines, Power System Technology 41(11)(2017) 3463-3469.
[15]B. Niu, X. Mu, Z. Yi, P. Hu, Two-Dimensional Maximum Entropy Infrared Image Fast Segmentation Based on Chicken Swarm Optimization, 2018 10th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2018.
[16]Z. Zhou, G. Chen, A novel weighted multi-point source thermal radiation model based on inversion optimization of heat source weight parameters, Natural Gas Industry 40(10)(2020) 139-147.
[17]Y. Zhang, J. Zhao, L. Wang, H. Wu, R. Zhou, J. Yu, An improved OIF Elman neural network based on CSO algorithm and its applications, Computer Communications 171(2021) 148-156.
[18]L. Xiang, J. Li, A. Hu, Y. Zhang, Deterministic and probabilistic multi-step forecasting for short-term wind speed based on secondary decomposition and a deep learning method, Energy Conversion and Management 220 (2020) 113098.
[19]A. Sharma, R. Pachauri, A. Sharma, N. Raj, in: Proc. Extraction of the solar PV module parameters using chicken swarm optimization technique, 2019 Women Institute of Technology Conference on Electrical and Computer Engineering (WITCON ECE), 2019.
[20]S. Banerjee, S. Chattopadhyay, Improved Serially Concatenated Convolution Turbo Code (SCCTC) using Chicken swarm optimization, in: Proc. 2015 IEEE Power, Communication and Information Technology Conference (PCITC), 2015.
[21]Y. Mu, L. Zhang, X. Chen, X. Gao, Optimal Trajectory Planning for Robotic Manipulators Using Chicken Swarm Optimization, in: Proc. 2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2016.
[22]T. Zhang, M. Yu, B. Li, Z. Liu, Capacity Prediction of Lithium-Ion Batteries Based on Wavelet Noise Reduction and Support Vector Machine, Transactions of China Electrotechnical Society 35(14)(2020) 3126-3136.
[23]J. Wang, F. Zhang, H. Liu, J. Ding, C. Gao, Interruptible load scheduling model based on an improved chicken swarm optimization algorithm, CSEE Journal of Power and Energy Systems 7(2)(2021) 232-240.
[24]S. Deb, X.-Z. Gao, K. Tammi, K. Kalita, P. Mahanta, A novel chicken swarm and teaching learning based algorithm for electric vehicle charging station placement problem, Energy 220(2021) 119645.
[25]Y. C. Wang, C. Liu, Y.F. Wang, Chicken swarm optimization algorithm based on stimulus-response mechanism, Control and Decision (2021) 1-9. DOI: 10.13195/j.kzyjc.2021.1059.
[26]W. Li, B. Wang, Z.J. Cao, H.H. Chen, X.H. Chen, Application of CCSO in Wind Power Interval Prediction, Acta Energiae Solaris Sinica 42(7)(2021) 350-358.
[27]J. Zhang, K. Xia, Z. Yin, Quantum Chicken Swarm Optimization with Levy Flight and Its Application in Parameter optimization of Random Forest, in: Proc. 2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE), 2019.
[28]X. Liang, D. Kou, L. Wen, An Improved Chicken Swarm Optimization Algorithm and its Application in Robot Path Planning, IEEE Access 8(2020) 49543-49550.
[29]M. Han, S. Liu, An Improved Binary Chicken Swarm Optimization Algorithm for Solving 0-1 Knapsack Problem, in: Proc. 2017 13th International Conference on Computational Intelligence and Security (CIS), 2017.
[30]W. Fu, B. Wang, X. Li, L. Liu, Y. Wang, Ascent Trajectory Optimization for Hypersonic Vehicle Based on Improved Chicken Swarm Optimization, IEEE Access 7(2019) 151836-151850.
[31]H. Xue, L. Li, K. Chao, C. Fu, Short-Term Wind Power Prediction Based on Improved Chicken Algorithm and Support Vector Machine, in: Proc. 2018 International Symposium on Computer, Consumer and Control (IS3C), 2018.
[32]Z.F. Liu, L.-L. Li, M.-L. Tseng, M.K. Lim, Prediction short-term photovoltaic power using improved chicken swarm optimizer - Extreme learning machine model, Journal of Cleaner Production, 248(2020) 119272.
[33]W. Osamy, A.A. El-Sawy, A. Salim, CSOCA: Chicken Swarm Optimization Based Clustering Algorithm for Wireless Sensor Networks, IEEE Access 8(2020) 60676-60688.
[34]M. Lin, Y. Zhong, R. Chen, Graphic process units-based chicken swarm optimization algorithm for function optimization problems, Concurrency and Computation: Practice and Experience 33(8)(2021) e5953.
[35]Y. Li, Z. Ni, F. Jin, J. Li, F. Li, Research on Clustering Method of Improved Glowworm Algorithm Based on Good-Point Set, Mathematical Problems in Engineering 2018(2018) 1-8.
[36]W. Long, J. Jiao, X. Liang, T. Wu, M. Xu, S.Cai, Pinhole-imaging-based learning butterfly optimization algorithm for global optimization and feature selection, Applied Soft Computing 103(2021) 107146.

[37]L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, A.H. Gandomi, The Arithmetic Optimization Algorithm, Computer Methods in Applied Mechanics and Engineering 376(2021) 113609.
[38]B. Li, G. Shen, G. Sun, T. Zheng, Improved Chicken Swarm Optimization Algorithm, Journal of Jilin University (Engineering and Technology Edition) 49(4)(2019) 1339-1344.
[39]Y. Hu, Q. Feng, X. Hai, Y. Ren, Improved Pigeon-inspired Optimization Algorithm Based on Adaptive Learning Strategy, Journal of Beijing University of Aeronautics and Astronautics 46(12)(2020) 2348-2356.
[40]X. Chen, F. Meng, J. Wu, Improved Wolf Pack Algorithm for Large-scale Optimization Problems, Systems Engineering-Theory & Practice 41(3)(2021) 790-808.
[41]H. Lu, S. Yin, G. Gong, Y. Liu, G. Chen, A Particle Swarm Optimization Algorithm Based on Deep Deterministic Policy Gradient, Journal of University of Electronic Science and Technology of China 50(2)(2021)199-206.
[42]H. Pan, H. Ding, M. Lei, L. Wang, Maximum Power Tracking Control of Direct Drive Ocean Wave Power Generation System Based on Genetic Algorithm, Acta Energiae Solaris Sinica 42(3)(2021) 221-227.
[43]Q. Zhang, A. Gong, X. Xu, C. Luo, F. Xie, Locating the Critical Slip Surface of Homogeneous Soil Slope Based on Modified Biogeography-Based Optimization Algorithm, Henan Science 39(2)(2021) 276-281.
[44]Y. Sun, Y. Dai, Optimization of Micro-grid Operation Based on Improved CSO, Journal of Gansu Agricultural University 54(3)(2019) 220-228.
[45]Z. Wu, D. Yu, X. Kang, Application of Improved Chicken Swarm Optimization for MPPT in Photovoltaic System, Acta Energiae Solaris Sinica 40(6)(2019) 1589-1598.
[46]J. Zheng, R. Chellali, C. Chen, Y. Xing, Improved Chicken Swarm Optimization Algorithm with Adaptive Features, Microelectronics & Computer 35(10)(2018) 93-98.