Hindawi
Scientific Programming
Volume 2022, Article ID 4623980, 12 pages
https://doi.org/10.1155/2022/4623980

Research Article

Marine Predators Algorithm Based on Adaptive Weight and Chaos Factor and Its Application

Shujun Liang,[1] Youmei Pan,[2] Huanlong Zhang,[2] Jie Zhang,[2] Fengxian Wang,[2] and Zhiwu Chen[2]

[1]*Software Engineering College, Zhengzhou University of Light Industry, Zhengzhou 450002, China*
[2]*College of Electric and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China*

Correspondence should be addressed to Zhiwu Chen; 1980652007@qq.com

Received 11 January 2022; Revised 20 July 2022; Accepted 16 August 2022; Published 29 September 2022

Academic Editor: Pengwei Wang

Aiming at the problems of uneven distribution of initialized populations and unbalanced exploration and exploitation leading to slow convergence, low convergence accuracy, and easy to fall into local optimality of marine predators algorithm (MPA), a marine predators algorithm based on adaptive weight and chaos factor is proposed (ACMPA), the algorithm is applied to the traveling salesman problem (TSP), and the shortest path planning and research are carried out for the traveling salesman problem. Firstly, the improved adaptive weight strategy is used to balance the exploration and exploitation stage of the algorithm and improve the convergence accuracy of the algorithm. Secondly, the chaos factor is used to replace the random factor, and the ergodicity of the chaos factor is used to make it easier for predators to jump out of local optimization and enhance the optimization ability of the algorithm. Finally, 10 benchmark test functions, the CEC2015 test set, and the CEC2017 test set are used to evaluate the effectiveness of the ACMPA. The results show that, compared with the other four intelligent optimization algorithms, the improved ACMPA achieves better results in both mean and standard deviation, and the algorithm has a better effect on the shortest path problem.

1. Introduction

Travel salesman problem [1] is a typical combinatorial optimization problem, which can be simply described as finding an optimal closed path that can traverse all locations in the shortest time given the distance between some locations, and each location is visited only once. For the TSP problem, the emerging intelligent optimization algorithm has been well applied to this problem. The core of the intelligent optimization algorithm comes from the simulation of natural biological behavior, and the optimal solution is obtained by searching for the optimal value in global exploration and local exploitation [2, 3]. At the same time, more and more scholars have improved the algorithm and applied it to the traveling salesman problem [4, 5].

Metaheuristic algorithms proposed in recent years, such as Monarch Butterfly Optimization (MBO) [6], mainly simulate the migration and adaptation behavior of monarch butterflies. The position of each butterfly represents a feasible solution. The butterfly population is distributed in two continents, migrating and adapting to the environment. Zheng et al. proposed a slime mold algorithm (SMA) [7], which simulated the behavior and morphological changes of slime mold in the foraging process. Slime mold would determine the weight of each individual's location according to the objective conditions of the current location and update the location through the weight index. Oliva et al. proposed the Moth Swarm Algorithm (MSA) [8], which simulates the behavior of moths flying toward the moon at night in nature, divides the moths into different types of moths, and carries out group cooperation, which balances the detection and mining capabilities of the algorithm to some extent. Alabool et al. proposed the Harris Hawks Optimization (HHO) [9], which simulates the behavior of Harris Hawks hunting rabbits and carries out unique cooperative foraging activities with other family members in the group.

The marine predators algorithm is a novel metainspired optimization algorithm proposed by Faramarzi et al. in 2020, which simulates the behavior of predators attacking prey through Brownian motion and Lévy motion, and the optimization process is divided into three stages [10]. Predator and prey will move in two different ways according to different speed ratios in these three stages. Although the MPA has made great achievements in optimization [11, 12], it still has the problems of slow convergence speed and easy to fall into local optimization. Mohamed Abd Elaziz et al. [13] proposed an enhanced marine predators algorithm, which introduces a differential evolution operator into the original marine predators algorithm, enhances the global exploration ability of the MPA, and improves the ability of the algorithm to jump out of the local optimal solution. Ma et al. [14] proposed an improved marine predators algorithm (MSIMPA), which uses a chaotic opposition learning strategy to generate a high-quality initial population and introduces an adaptive t-distribution mutation operator to update population to improve population quality and optimization accuracy. Oszust [15] proposed a new local escape operator (LEO-MPA), which uses new candidate solutions to replace part of the worst solutions to deal with premature convergence and significantly improve the optimization. Abdel-basset et al. [16] proposed an improved marine predators algorithm (IMPA). By introducing the ranking-based diversity reduction (RDR) strategy, the IMPA searched for the optimal individual position to replace the particles with poor effect many consecutive times so as to improve the convergence performance of the algorithm. But the convergence performance of the marine predators algorithm still needs to be improved. Chu et al. [17] proposed a whale optimization algorithm based on adaptive weight and simulated annealing, which adaptively adjusted the disturbance value according to the distance between the individual whale and the optimal one so as to balance the exploration and exploitation of the algorithm. Deng et al. [18] proposed a particle swarm optimization algorithm based on the neighborhood velocity imitation strategy, and the algorithm imitates the velocity of the optimal particle in the neighborhood, adaptively adjusts the convergence performance of the algorithm, and adopts the centroid mutation strategy to enhance the ability of the algorithm to jump out of the local extremum. Zhang et al. [19] proposed an improved whale optimization algorithm, which added an adaptive weight strategy in the later stage of local exploitation to improve the local search ability of the algorithm so as to improve the convergence accuracy of the algorithm. Although the adaptive weight strategy has been well applied in the optimization algorithm, it is seldom applied in the marine predators algorithm, and the convergence accuracy is low.

In order to improve the convergence speed and jump out of the local optimal solution of the MPA, a marine predators algorithm based on adaptive weight and chaos factor strategy is proposed in this paper. Firstly, the adaptive weight is combined with the step length of the algorithm, which gives a larger weight to the step length in the early stage of optimization to improve the global search ability of the algorithm and gives smaller weight to the step length in the later stage of exploitation to achieve rapid convergence. Secondly, the ergodicity of the logistic chaos function is used to replace the random vector in the process of optimization to improve the ability to jump out of the local optimal solution. Thirdly, the simulation results of 10 benchmark test functions, the CEC2015 test set, and the CEC2017 test set show that the experimental results demonstrate the effectiveness of the ACMPA. Finally, the ACMPA is applied to the TSP problem, and the superiority of the ACMPA compared with the MPA is verified.

In this paper, we use chaotic mapping instead of random initialization so that the initial population distribution has the same probability at each location in each search region, the distribution is more uniform, which is conducive to the next iteration, and it is easier for the algorithm to find the global optimum. The introduction of the adaptive weight strategy makes it easier to find the global optimum in the exploration phase with larger step lengths and in the exploitation phase with smaller step lengths. By the above two methods, the convergence speed and convergence accuracy of the algorithm are improved.

In general, this paper proposes a marine predators algorithm based on adaptive weight and chaos factor. In Section 2, the marine predators algorithm (MPA) is introduced, in Section 3, the method to improve MPA is proposed, in Section 4, the results of the test function are introduced and analyzed, and finally, in Section 5, the work is summarized.

2. Marine Predators Algorithm

The marine predators algorithm is a new metaheuristic optimization algorithm, which simulates the behavior of predators preying on prey in nature and searches for the best proxy position through Brownian motion and Lévy motion [20]. The MPA optimization process is as follows:

(1) Initialization phase:

The MPA is similar to the majority heuristic optimization algorithm in the initialization stage. Its initial solution is distributed in the whole solution space by random strategy, and the specific expressions are as follows:

$$X_0 = X_{\min} + \mathrm{rand}\left(\mathrm{X}_{\max} - \mathrm{X}_{\min}\right), \tag{1}$$

where $\mathrm{X}_{\max}$ and $X_{\min}$ are the maximum and the minimum bounds of variables and *ran d* is a uniform random number between 0 and 1.

(2) The optimization process of the original MPA is divided into three parts according to different speed ratios, and each part has different tasks to simulate the movement process of predators and prey in space, which is summarized as follows:

Stage 1. When the number of iterations is in the first third of the maximum number of iterations (when the predator is faster than the prey), this stage belongs to the exploration stage. At this time, the best strategy of the predator is not to move. The specific expression is as follows:

$$\text{While Iter} < \frac{1}{3}\text{Max\_iter}$$

$$\text{stepsize}_i = R_B \otimes \left(\text{Elite}_i - R_B \otimes \text{Prey}_i\right)\ (i = 1, 2, \ldots, N),$$

$$\text{Prey}_i = \text{Prey}_i + P \cdot R \otimes \text{stepsize}_i \tag{2}$$

where Iter is the current iteration number, Max_iter is the maximum iteration number, stepsice$_i$ is the moving step size, R_B is the random vector generated by Brownian motion, Elite$_i$ is the elite matrix constructed by the top predator with the best fitness, Prey$_i$ is the prey matrix with the same dimension as the elite matrix, $N$is the population number, P is a constant, the recommended value is 0.5 [21], and R is a random vector evenly distributed between [0,1].

Stage 2. When the number of iterations is between one-third and two-thirds of the maximum number of iterations (when the speed of the predator is the same as that of the prey), this stage is a process of trying to change from the exploration stage to the exploitation stage. At this time, the population is divided into two parts, one for global exploration, the predator for exploration by Brownian motion, the other for local exploitation, and the prey for exploitation by Lévy motion.

The first half of the population motion expression is as follows:

$$\text{While } \frac{1}{3}\text{Max\_iter} < \text{Iter} < \frac{2}{3}\text{Max\_iter}$$

$$\text{stepsize}_i = R_L \otimes \left(\text{Elite}_i - R_L \otimes \text{Prey}_i\right)\left(i = 1, 2, \ldots, \frac{N}{2}\right).$$

$$\text{Prey}_i = \text{Prey}_i + P \cdot R \otimes \text{stepsize}_i \tag{3}$$

The last half of the population motion expression is as follows:

$$\textit{While } \frac{1}{3}\text{Max\_iter} < \text{Iter} < \frac{2}{3}\text{Max\_iter}$$

$$\text{stepsize}_i = R_B \otimes \left(R_B \otimes \text{Elite}_i - \text{Prey}_i\right)\left(i = \frac{N}{2}, \ldots, N\right), \tag{4}$$

$$\text{Prey}_i = \text{Elite}_i + P \cdot CF \otimes \text{stepsize}_i$$

where R_L is the random vector generated by Lévy motion, CF is the adaptive parameter controlling the predator step size, and the formula is as follows:

$$CF = \left(1 - \frac{\text{Iter}}{\text{Max\_iter}}\right)^{\left(2*\text{Iter}/\text{Max}_{\text{iter}}\right)}. \tag{5}$$

Stage 3. When the number of iterations is in the last two-thirds of the maximum number of iterations (when the predator is slower than the prey), this stage is the exploitation stage, and the predator adopts the Lévy migration strategy. The specific expression is as follows:

$$\text{While Iter} > \frac{2}{3}\text{Max\_iter}$$

$$\text{stepsize}_i = R_L \otimes \left(R_L \otimes \text{Elite}_i - \text{Prey}_i\right)\ (i = 1, 2, \ldots, N).$$

$$\text{Prey}_i = \text{Elite}_i + P \cdot CF \otimes \text{stepsize}_i \tag{6}$$

(3) In addition to the above stages, the actions of predators will also be affected by the environment, such as eddy current effect or fish gather device *FADs*, and transferred to other hunting areas; that is, predators will have 80% of the time looking for prey, while 20% of their time will be transferred to other hunting areas to increase their ability to jump out of the local optimal solution, a strategy to avoid the premature convergence problem; specific expression is as follows:

$$\text{Prey}_i \begin{cases} \text{Prey}_i + CF\left[X_{\min} + R \otimes \left(X_{\max} - X_{\min}\right)\right] \otimes U & if\, r \le FADs \\ \text{Prey}_i + \left[\text{FAD}s(1 - r)\right]\left(\text{Prey}_{r1} - \text{Prey}_{r2}\right) & if\, r > FADs \end{cases}, \tag{7}$$

where $FADs = 0.2$ is the probability of being affected in the optimization process. U is a binary vector group containing 0 and 1. If the generated random vector is less than 0.2, the array is 0; if the random vector is greater than 0.2, the array is 1. $r1$ and $r2$ are random indexes of the prey matrix.

3. Improved Marine Predators Algorithms

Aiming at the problems of easy imbalance in exploitation and exploration and easy to fall into local optimization of marine predators algorithm, an improved algorithm of adaptive weight and chaos factor is proposed. The specific improvement method is as follows.

3.1. Adaptive Weight.

3.1. Adaptive Weight. The weight factor balances the important parameters of global exploration and local exploitation in the optimization algorithm [22]. In the early stage of the algorithm, the particles in the space are scattered and chaotic. When the weight is given a large value, it will enhance the global search ability of the algorithm and quickly find the optimal area in a large range. In the late stage of the algorithm, the particles in the space will be concentrated in a certain area. When the weight is given a small value, the local search ability of the algorithm can be improved, and the convergence speed can be accelerated [23]. Therefore, appropriate weight parameters can improve the optimization performance of the algorithm.

Inspired by literature [24, 25], a new adaptive weight strategy is proposed to replace the original step size formula, and a larger step size is given in the early stage of the algorithm, which is beneficial to increasing the search step size and realizing global search. In the later stage of the algorithm, the algorithm gradually converges, and individuals begin to conduct local search. At this time, a small step is given to reduce the search distance, which is beneficial to the local detailed search and improves the convergence accuracy of the algorithm. From formula (5), MPA step length formula, it can be seen that, in the early stage of the algorithm, the step distance is small, which is not conducive to the predator's search for the whole world. It is in the process of trying to change from the exploration stage to the exploitation stage, and the sudden increase in step size is not conducive to the balance between exploration and exploitation; in the later stage of the algorithm, the algorithm is in a period of rapid convergence, but the step distance is reduced, which is not conducive to the convergence of the algorithm. Therefore, the improved step length formula is

$$CF = \omega_{\max} - \left(\omega_{\max} - \omega_{\min}\right) \cdot \left(1 - \frac{t}{\mathrm{Max}_{\mathrm{iter}}}\right)^{\left(2.5 - t/\mathrm{Max}_{\mathrm{iter}}\right)}, \tag{8}$$

where $\omega_{\max}$ and $\omega_{\min}$ are the maximum and minimum values of inertia weight. When the inertia weight $\omega_{\max} = 1$ and $\omega_{\min} = 0.001$, the algorithm can maintain good performance, make the step size curve conform to the previous step growth, increase the global search ability, and gradually reduce the step size spacing with the increase of the number of iterations to achieve rapid convergence.

3.2. Logistic Chaos Factor. Chaos mapping is an aperiodic function with ergodicity and randomness. Due to its unique advantages, it has been widely used as a global optimization processing mechanism [26]. Many scholars have applied chaos theory to intelligent optimization algorithms and achieved good results [27, 28]. (1) Replace some random parameters in the algorithm with sequences generated by the chaotic system. (2) The results of the chaotic system are applied to search space to improve the local search ability of the algorithm.

However, there are limitations of a single chaotic mapping, and speaking of multiple chaotic maps incorporated into the algorithm separately, randomly, in parallel, or selectively, the effect will be greatly improved [29]. In this paper, we combine four chaotic mappings in parallel instead of one chaotic mapping; compared with random numbers, chaos can form a nonrepetitive and uniform state within a certain range [30]. Random number R is in the early stage of the algorithm. If uneven particle distribution and overlap are explored in the early stage, the global nonconvergence of the algorithm will be affected, and the convergence accuracy of the algorithm will be affected. Therefore, mixed chaos is used to replace the random number R, and its expression is as follows:

$$\begin{aligned}
R_{1n+1} &= \mu \cdot R_{1,n}\left(1 - R_{1,n}\right)\\
R_{2n+1} &= \beta \cdot R_{2,n}\left(1 - R_{2,n} \cdot R_{2,n}\right)\\
R_{3n+1} &= \frac{\alpha \cdot \sin\left(R_{3,n}\right)}{4}\\
R_{4n+1} &= 1.07 \cdot (7.86 \cdot R_{4,n} -\\
&\quad 23.31 \cdot R_{4,n}\hat{}2 + 28.75 \cdot R_{4,n}\hat{}3\\
&\quad - 13.302857 \cdot R_{4,n}\hat{}4,
\end{aligned} \tag{9}$$

where $R_n \in [0, 1]$, μ belongs to the chaos parameter, and $\mu \in (0, 4]$. The experiment shows that when μ value is 4, β value is 2.59, and α value is 3.8.

3.3. ACMPA Steps. First, the initialization is performed by chaotic mapping to determine the position of each individual in the population. Second, when the number of iterations is less than 1/3 of the total number of iterations, the position is updated by replacing the R value in (2) with the R value in (9); when the number of iterations is greater than 1/3 of the total number of iterations and less than 2/3 of the total number of iterations, half of the populations are updated by (3) (bringing the result of (9) into (3)), and others are updated by (4) (bringing the result of (8) into 3); when the number of iterations is greater than 2/3 of the total number of iterations, it is updated by (6) (bringing the result of (8) into (6)). Finally, the number of iterations is judged to be satisfied; otherwise, the iterations are repeated. The relevant flowchart of the algorithm is shown in Figure 1.

4. Simulation Experiment and Result Analysis

4.1. Parameter Settings. In this paper, the population size of marine predators is set as 100, dimension $D = 10$, and the maximum iteration number is 1000. The simulation experiment environment is Windows 10 system, 16 G memory, and MATLAB R2019a. In this paper, four optimization algorithms, including the marine predators algorithm (MPA) [20], Moth-Flame Algorithm (MFO) [30], Chimp Optimization Algorithm (ChoA) [31], and Sine Cosine Algorithm (SCA) [32], are selected to compare the effectiveness of ACMPA. Parameter settings of various algorithms are shown in Table 1. To make the experimental results more authoritative, all hyperparameters in Table 1 are those presented in the original paper [20, 31–33].

4.2. Complexity Analysis. In optimization algorithms, complexity is often used to analyze the quality of algorithm operation time (time complexity) and space (space complexity). Among them, we usually use the time complexity to evaluate the efficiency of the algorithm and use the space complexity to evaluate the quality of the running space. Since the variables of the algorithm are not added or reduced in this paper, the space complexity has not changed, so it will

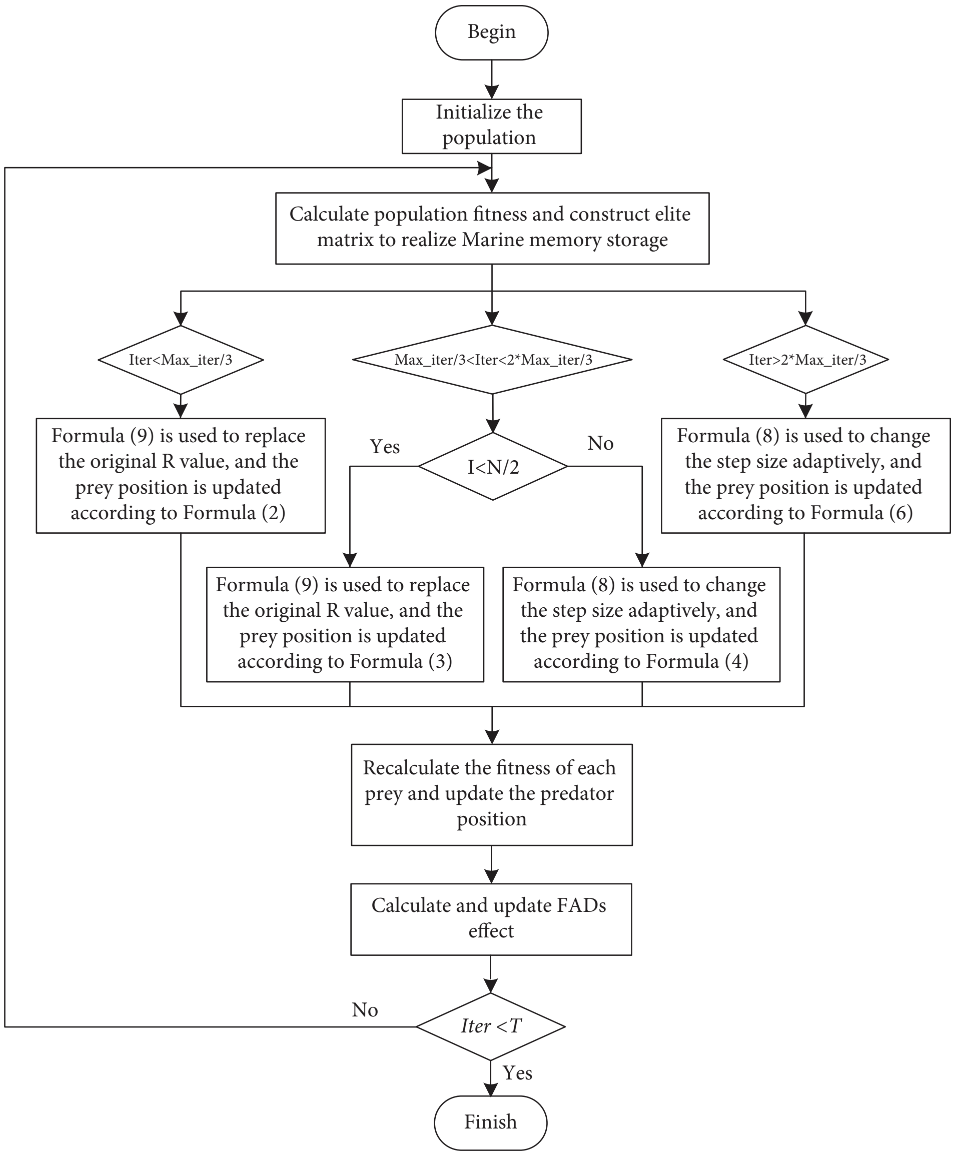
Begin
Initialize the population
Calculate population fitness and construct elite matrix to realize Marine memory storage
Iter<Max_iter/3
Max_iter/3<Iter<2*Max_iter/3
Iter>2*Max_iter/3
Formula (9) is used to replace the original R value, and the prey position is updated according to Formula (2)
Yes
I<N/2
No
Formula (8) is used to change the step size adaptively, and the prey position is updated according to Formula (6)
Formula (9) is used to replace the original R value, and the prey position is updated according to Formula (3)
Formula (8) is used to change the step size adaptively, and the prey position is updated according to Formula (4)
Recalculate the fitness of each prey and update the predator position
Calculate and update FADs effect
No
Iter <T
Yes
Finish

Figure 1: Algorithm flowchart.

Table 1: Algorithm parameter settings.

Algorithm	Parameter
ACMPA	$FADs = 0.2$; $P = 0.5$
MPA	$FADs = 0.2$, $P = 0.5$, and $R =$ rand
MFO	$b = 1$
ChoA	$r =$ rand; $\mu \in [0, 1]$
SCA	$a = 2$; $r4 =$ rand

not be discussed here. Big O is used to represent the time complexity of the improved algorithm, in which the added improved strategy only changes the detection step of the algorithm and hardly affects the time complexity of the algorithm; that is, O (ACMPA) = O (MPA). In this improvement, although the time complexity does not change, the accuracy of the optimal value is greatly improved compared with the original algorithm. By analysis, the complexity of the algorithm after improvement is $O(\text{Max\_iter} * (\text{n} * D + fit * n))$, where fit is the cost of function evaluation.

4.3. Optimization Comparison of 10 Benchmark Test Functions. In order to better test the effectiveness of the improved marine predator algorithm ACMPA, 10 different test functions in the benchmark test function were selected to conduct experiments on the algorithm. Among the 10 test functions, $f1 - f4$ are unimodal functions, and the peak value of this function has only one optimal value, which can be used to test the convergence speed of different algorithms. $f5 - f8$ are multimodal functions, which have a large number of local optimal values. If the tested function can effectively achieve good convergence accuracy, it indicates that the algorithm has the ability to jump out of the local optimal solution. $f9 - f10$ are multichannel low-dimensional functions with a small number of local minima, which

Table 2: Introduction of benchmark functions.

Function	Dim	Scope	Minimum
$f1(x) = \sum_{i=1}^{D} x_i^2$	30	$[-100, 100]$	0
$f2(x) = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$	10	$[-10, 10]$	0
$f3(x) = \sum_{i=1}^{D} (\sum_{j-1}^{i} x_j)^2$	10	$[-100, 100]$	0
$f4(x) = \max\{\lvert x_i\rvert, 1 \leq i \leq n\}$	10	$[-100, 100]$	0
$f5 = \sum_{i=1}^{D} ix_i^4 + \text{random}[0, 1)$	10	$[-1.28, 1.28]$	0
$f6(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i)] + 10D$	10	$[-5.12, 5.12]$	0
$f7(x) = -20\exp(-0.2\sqrt{1/D\sum_{i=1}^{D} x_i^2}) - \exp(1/D\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$	10	$[-32, 32]$	0
$f8(x) = 1/4000\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(x_i/\sqrt{i}) + 1\sum_{i=1}^{D} x_i^2$	10	$[-600, 600]$	0
$f9(x) = (1/500 + \sum_{i=1}^{25} 1/j + \sum_{i=1}^{2} (x_{i-}a_{ij})^6)^{-1}$	2	$[-65.536, 65.536]$	0
$f10(x) = \sum_{i=1}^{11} [a_i - x_1(b_i^2 + b_i x_2)/b_i^2 + b_i x_3 + x_4]^2$	4	$[-5, 5]$	0

Table 3: Comparison of optimization results of benchmark test functions.

Function	Result	ACMPA	ChoA	MFO	MPA	SCA
$f1$	Ave	**1.4547e-84**	3.3747e-75	1.2929e-32	4.4914e-65	3.3693e-35
	Std	**2.3378e-84**	1.7719e-74	2.2985e-32	1.6501e-64	1.3574e-34
$f2$	Ave	**6.6421e-57**	3.6967e-47	1.8681e-19	8.0445e-39	1.0473e-24
	Std	**2.4495e-56**	1.9534e-48	1.7940e-19	1.3648e-38	1.5119e-24
$f3$	Ave	**1.08424e-50**	1.1830e-37	8.0371e-12	1.5264e-34	9.0355e-17
	Std	**3.71256e-60**	6.4715e-37	1.5424e-11	5.7895e-34	3.3919e-16
$f4$	Ave	**1.7589e-39**	1.2554e-24	3.0230e-10	2.3507e-27	5.3163e-13
	Std	**5.7318e-39**	3.7076e-24	6.7830e-10	2.9474e-27	1.2047e-12
$f5$	Ave	**1.1536e-04**	1.1885e-04	1.3902e-03	1.9046e-04	3.0483e-04
	Std	**1.0551e-04**	7.8796e-05	5.5616e-04	1.2141e-04	2.8847e-04
$f6$	Ave	**0**	5.5816e-09	13.4674	**0**	**0**
	Std	**0**	3.0572e-08	11.5827	**0**	**0**
$f7$	Ave	**8.8817e-16**	6.0527	4.5593e-15	**8.8817e-16**	3.8487e-15
	Std	**0**	9.1719	6.4863e-16	**0**	1.3466e-15
$f8$	Ave	**0**	0.0282	0.1265	**0**	0.0363
	Std	**0**	0.0602	0.0619	**0**	0.0945
$f9$	Ave	**0.9980**	0.9980	1.0311	**0.9980**	1.0641
	Std	**0**	1.3084e-06	0.1814	**0**	0.3622
$f10$	Ave	**3.0748e-04**	1.2484e-03	8.9213e-04	**3.0748e-04**	6.8870e-04
	Std	**8.5518e-19**	1.7323e-05	2.9622e-04	3.8762e-18	4.0040e-04

can be used to test the comprehensive ability of the function. The details of each function are listed in Table 2.

In order to ensure the rationality of the algorithm, the ACMPA and other comparison algorithms were independently run 30 times, and their average value and standard deviation were recorded. The best results were marked in a deepened font. Table 2 shows the average and standard deviation of the five algorithms. Figure 1 shows the curve convergence of each algorithm under 10 test functions.

It can be seen from Table 3 that the improved ACMPA generally shows good results, ACMPA has been greatly improved on unimodal functions $f1, f2, f3$, and $f4$, and the highest solution accuracy reaches 1.17788e-82, which shows that, after adding adaptive weight and chaos factor, the algorithm step size gradually decreases with the increase of iteration times, and the particles in the space are distributed evenly, which balances the exploration and exploitation stages of the algorithm. For multimodal functions $f5, f6, f7$, and $f8$, the algorithm basically reaches the optimal value, which shows that the algorithm can jump out of the local optimal solution. For multichannel low-dimensional functions $f9$ and $f10$, the improved algorithm and the original algorithm are both optimal in terms of average value, and ACMPA is better in standard deviation.

As shown in Figure 2, on unimodal functions $f1, f2, f3$, and $f4$, the convergence speed and convergence accuracy of the ACMPA are greatly improved compared with MPA, which indicates that the global search ability of the algorithm is improved in the global search space. For multimodal functions, $f5, f6$, an d $f7$, ACMPA, MPA, ChOA, and SCA have good convergence accuracy, but ACMPA is faster than other algorithms in convergence speed, which indicates that the algorithm has the ability to jump out of the local optimum after falling into the local optimum. For multichannel low-dimensional functions $f8$ and $f9$, the convergence performance of ACMPA is also optimal. Generally speaking, ACMPA has a certain improvement in convergence speed and convergence accuracy.

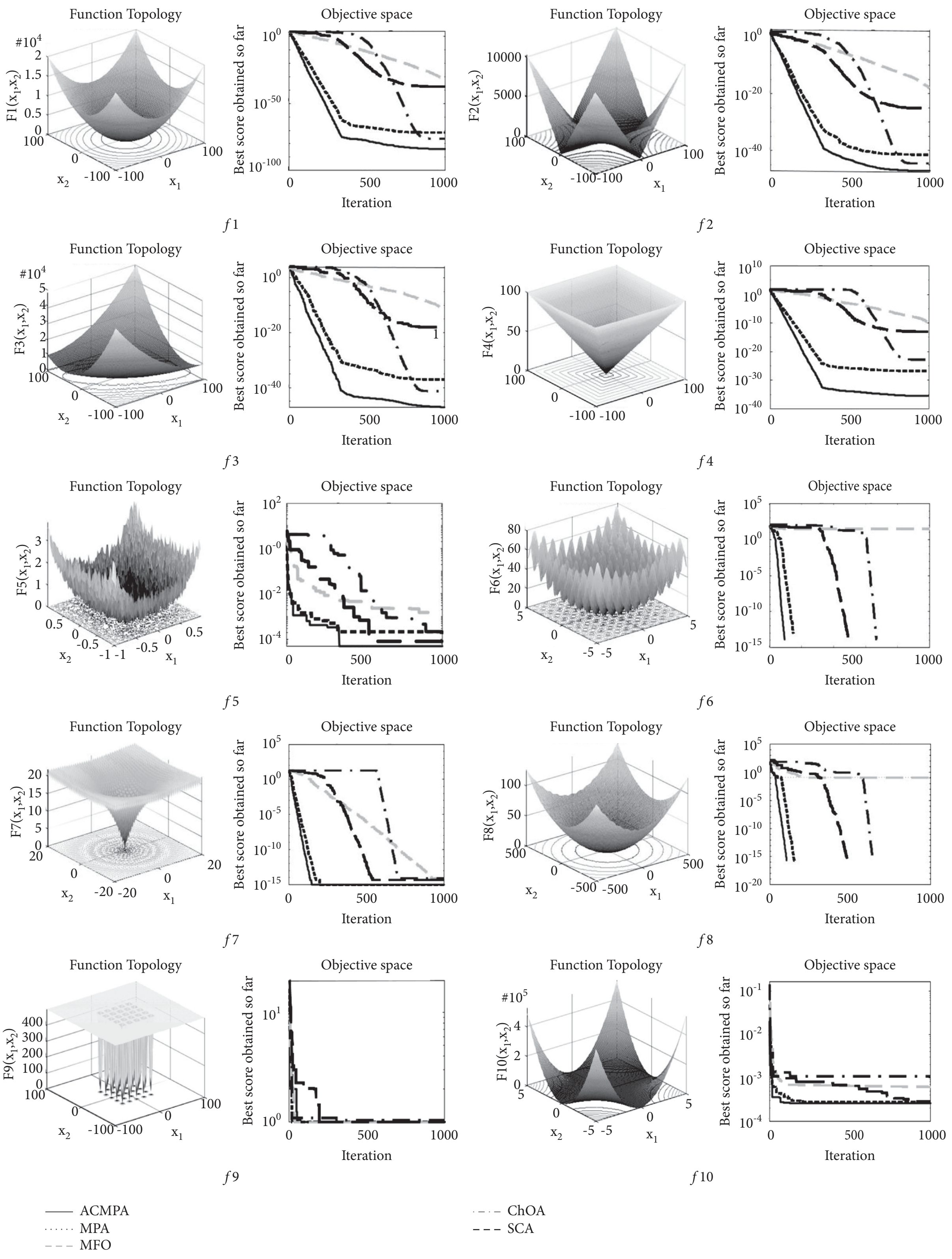

Figure 2: Convergence curves of various algorithms.

In order to further test the effectiveness of the improved algorithm, the Wilcoxon signed-rank test is used to test the algorithm. The Wilcoxon signed-rank test is a nonparametric test method, which can analyze the statistical parameters of the optimized algorithm. In this experiment, the p-value index is used to judge whether there is a statistically significant difference between the two algorithms. When the p-value is less than 0.05, it indicates that there is a significant difference

Table 4: Wilcoxon signed-rank test results.

Function	p-value
ACMPA to CHoA	0.0077
ACMPA to MFO	0.0051
ACMPA to MPA	0.0431
ACMPA to SCA	0.0077

Table 5: Comparison of CEC2015 test function algorithms.

Function	Result	ACMPA	MPA	MFO	ChOA	SCA
C1	Ave	**100**	**100**	1.4367e + 02	1.9867e + 02	1.9406e + 02
	Std	**6.2236e-12**	8.3599e-10	31.3966	15.6369	24.4470
C2	Ave	**200**	**200**	2.0133e + 02	2.2343e + 02	2.2912e + 02
	Std	**3.5954e-09**	4.1162e-09	7.3185	15.1857	15.6238
C3	Ave	**300**	**300**	3.0004e + 02	3.0004e + 02	3.0003e + 02
	Std	5.5409e-13	1.2952e-12	0.1830	0.5332	0.1304
C4	Ave	**400.0102**	400.0524	400.6603	401.7285	401.8974
	Std	**0.0064**	0.0460	0.3867	0.5680	0.3179
C5	Ave	**500**	**500**	**500**	500.3483	500.1672
	Std	3.9467e-13	8.8980e-13	**0.00e + 00**	0.7670	0.0808
C6	Ave	**600**	**600**	601.3449	608.7733	612.8731
	Std	**1.2219e-10**	1.4504e-10	2.6639	4.4299	5.1286
C7	Ave	**699.9999**	**699.9999**	**700.5513**	705.1753	701.6148
	Std	**4.8709e-11**	1.2783e-10	1.1859	5.3929	0.9748
C8	Ave	**800**	**800**	800.0203	802.0496	800.4230
	Std	**2.3476e-10**	1.1634e-09	4.3201e-02	2.2464	0.07425
C9	Ave	**907.0017**	907.6713	933.6712	949.6334	922.3368
	Std	**1.2232**	1.4662	3.9071	4.7009	14.8927
C10	Ave	**1.0701e + 03**	1.0703e + 03	1074	1.2937e + 04	8.6890e + 03
	Std	**1.0522**	1.8241	1.9033e + 03	5.6000e + 02	1.5992e + 03
C11	Ave	**1100.4948**	1100.6903	1102.2503	1105.5146	1104.4510
	Std	**0.6012**	0.9643	1.0535	3.2987	0.7902
C12	Ave	**1200.2276**	1200.2466	1306.1335	1708.0688	1414.3597
	Std	**5.1716e-02**	0.0631	87.7573	1.9049e + 02	24.9596
C13	Ave	**1353.8668**	1358.6005	1516.5927	2091.4406	1685.0403
	Std	**56.9207**	59.8232	67.9113	3.3472e + 02	36.5153
C14	Ave	**1538.0583**	1586.8077	2473.4767	2698.8810	2526.0234
	Std	**142.4978**	189.4950	1.9160e + 02	1.9410e + 02	138.0687
C15	Ave	**1640.0000**	1640.0013	1683.5216	2152.7014	1816.9235
	Std	**1.5346e-05**	8.5698e-04	75.7575	1.6498e + 02	40.3826

between the proposed algorithm and the comparison algorithm. Table 4 shows the comparison of experimental results between the ACMPA and other optimization algorithms.

Table 4 shows the simulation results of the Wilcoxon signed-rank test. It can be seen from the table that the significance level p values are all less than 0.05, which means that the ACMPA has significant advantages over other algorithms.

4.4. CEC2015 Test Set Experimental Comparison. In order to verify that the ACMPA has better optimization performance, this paper selects 15 test functions in the CEC2015 test set [33] to compare five optimization algorithms, among which $C1 - C2$ test functions are unimodal functions, $C3 - C9$ are multimodal functions, $C10 - C12$ are hybrid functions, and $C13 - C15$ are composition functions. The convergence speed and accuracy of the function and the overall performance of the algorithm are tested, respectively. In this experiment, the population size $N = 100$ and the maximum iteration $\text{Max}\_iter = 1000$ were set. The experimental results are best expressed in a deepened font. The results of the ACMPA and other algorithms are compared in Table 5.

From Table 5, it can be seen that the test functions $C1 - C15$ and the improved algorithm ACMPA can achieve the best data in average and standard deviation compared with the other four algorithms, which effectively verifies that the MPA can improve the overall performance of the algorithm after adding adaptive weight and chaos factors. So far, the feasibility and effectiveness of the ACMPA have been proved.

4.5. CEC2017 Test Set Experimental Comparison. In order to further verify the effectiveness of the ACMPA, the CEC2017 test set [34] is selected to compare the ACMPA and four algorithms such as MPA, MFO, ChoA, and SCA. The CEC2017 test set has a total of 30 test functions, among which $F1 - F3$ are unimodal functions, to test the

Table 6: Comparison of CEC2017 test function algorithms.

Function	Result	ACMPA	MPA	MFO	ChOA	SCA
$F1$	Ave	100.0028	100.0031	2.3422e + 06	1.1551e + 09	5.6391e + 08
	Std	4.3941E-04	2.3829e-03	1.2789e + 07	1.1074e + 09	2.1562e + 08
$F3$	Ave	300	300	5.8487e + 03	1.8962e + 03	1.0424e + 03
	Std	2.2027e-08	2.7266e-08	8.2107e + 03	6.3536e + 02	3.1214e + 02
$F4$	Ave	400	400	4.1325e + 02	5.3241e + 02	4.3947e + 02
	Std	1.1974e-07	8.4773e-08	24.8459	1.1234e + 02	23.0909
$F5$	Ave	5.0735e + 02	5.0776e + 02	5.2609e + 02	5.5061e + 02	5.4575e + 02
	Std	2.2033	2.4566	8.7353	7.8088	8.0603
$F6$	Ave	6.0000e + 02	6.0007e + 02	6.0072e + 02	6.2374e + 02	6.1572e + 02
	Std	1.0454e-04	1.5006e-04	2.3146	7.1808	3.2753
$F11$	Ave	1.1012e + 03	1.1015e + 03	1.1953e + 03	1.2437e + 03	1.1799e + 03
	Std	0.8453	0.8496	72.8008	78.4126	30.4272
$F12$	Ave	1.2060e + 03	1.2045e + 03	4.9034e + 05	3.9895e + 05	1.0786e + 07
	Std	5.0054	5.6945	1.8664e + 06	3.7667e + 06	8.7799e + 06
$F13$	Ave	1.3036e + 03	1.3040e + 03	1.0345e + 04	1.7719e + 04	2.2340e + 04
	Std	2.0029	2.0985	1.0702e + 04	7.7592e + 03	1.8245e + 04
$F14$	Ave	1.4030e + 03	1.4025e + 03	2.0643e + 03	4.7000e + 03	1.5597e + 03
	Std	2.00654	4.4315	6.7003e + 02	1.4554e + 03	53.5723
$F15$	Ave	1.5002e + 03	1.5005e + 03	4.8513e + 03	8.6770e + 03	2.8796e + 03
	Std	0.3083	0.4553	3.3089e + 03	7.1903e + 03	1.5268e + 03
$F16$	Ave	1.6008e + 03	1.6008e + 03	1.6927e + 03	1.8382e + 03	1.6958e + 03
	Std	0.3727	0.5454	92.0091	80.4113	51.4587
$F17$	Ave	1.7087e + 03	1.7288e + 03	1.7938e + 03	1.7735e + 03	1.7701e + 03
	Std	7.3446	8.1711	19.9024	16.0423	12.5332
$F18$	Ave	1.8010e + 03	1.8035e + 03	2.0567e + 04	2.9153e + 04	8.1438e + 04
	Std	7.4186e-01	2.7993	1.4960e + 04	1.2987e + 04	5.3922e + 04
$F19$	Ave	1.9003e + 03	1.9011e + 03	8.6479e + 03	1.6903e + 04	4.4967e + 03
	Std	0.3909	0.4350	8.8960e + 03	2.6356e + 03	3.8910e + 03
$F20$	Ave	2.0113E + 03	2.0095e + 03	2.0486e + 03	2.1414e + 03	2.0805e + 03
	Std	5.2539	9.0082	45.6560	66.5820	15.5710
$F21$	Ave	2200	2200	2.2630e + 03	2.2824e + 03	2.2187e + 03
	Std	1.1151e-05	1.2767e-05	65.2106	64.8760	33.6048
$F22$	Ave	2.2406e + 03	2.2508e + 03	2.3040e + 03	2.7652e + 03	2.3496e + 03
	Std	49.7553	50.7507	16.5592	5.4232e + 02	31.2560
$F23$	Ave	2.5728e + 03	2.5815e + 03	2.6247e + 03	2.6529e + 03	2.6503e + 03
	Std	38.4263	85.5243	10.0225	5.1754	7.8379
$F24$	Ave	2.4933e + 03	2500	2.7504e + 03	2.7954e + 03	2.7710e + 03
	Std	2.91e-06	25.3706	48.0576	1.8565e + 01	44.2710
$F25$	Ave	2.8382e + 03	2.8778e + 03	2.9309e + 03	2.9838e + 03	2.9563e + 03
	Std	121.12	75.5298	27.5676	54.7609	17.2074
$F26$	Ave	2.6832e + 03	2.6933e + 03	2.9820e + 03	3.5103e + 03	3.0519e + 03
	Std	101.48	101.4840	34.8850	4.1445e + 02	26.8695
$F27$	Ave	3.0885e + 03	3.0887e + 03	3.0924e + 03	3.0991e + 03	3.1019e + 03
	Std	0.6793	0.7179	1.6937	4.3551	2.1492
$F28$	Ave	3.0600e + 03	3.0800e + 03	3.2907e + 03	3.2405e + 03	3.2512e + 03
	Std	54.7722	76.1120	1.0547e + 02	6.7944	51.4405
$F29$	Ave	3.1328e + 03	3.1362e + 03	3.4067e + 03	3.2686e + 03	3.4146e + 03
	Std	4.9434	4.6382	1.0321e + 02	60.9215	21.9284
$F30$	Ave	3.3971e + 03	3.3970e + 03	4.4095e + 05	2.0085e + 06	6.3930e + 05
	Std	4.3737	3.9169	3.1864e + 05	2.6075e + 06	4.2434e + 05

convergence speed and accuracy of the algorithm. The $F2$ function has been ruled out due to instability [35]. $F4 - F10$ multimodal functions test whether the function can effectively jump out of the local optimal solution. $F11 - F20$ are hybrid functions, and $F21 - F30$ are composite functions to test the overall performance of the function.

Table 7: The results of runtime overhead.

Algorithm	Runtime overhead (s)
MPA	40.73
ACMPA	44.41
SCA	15.76
MFO	17.27
ChOA	145.84

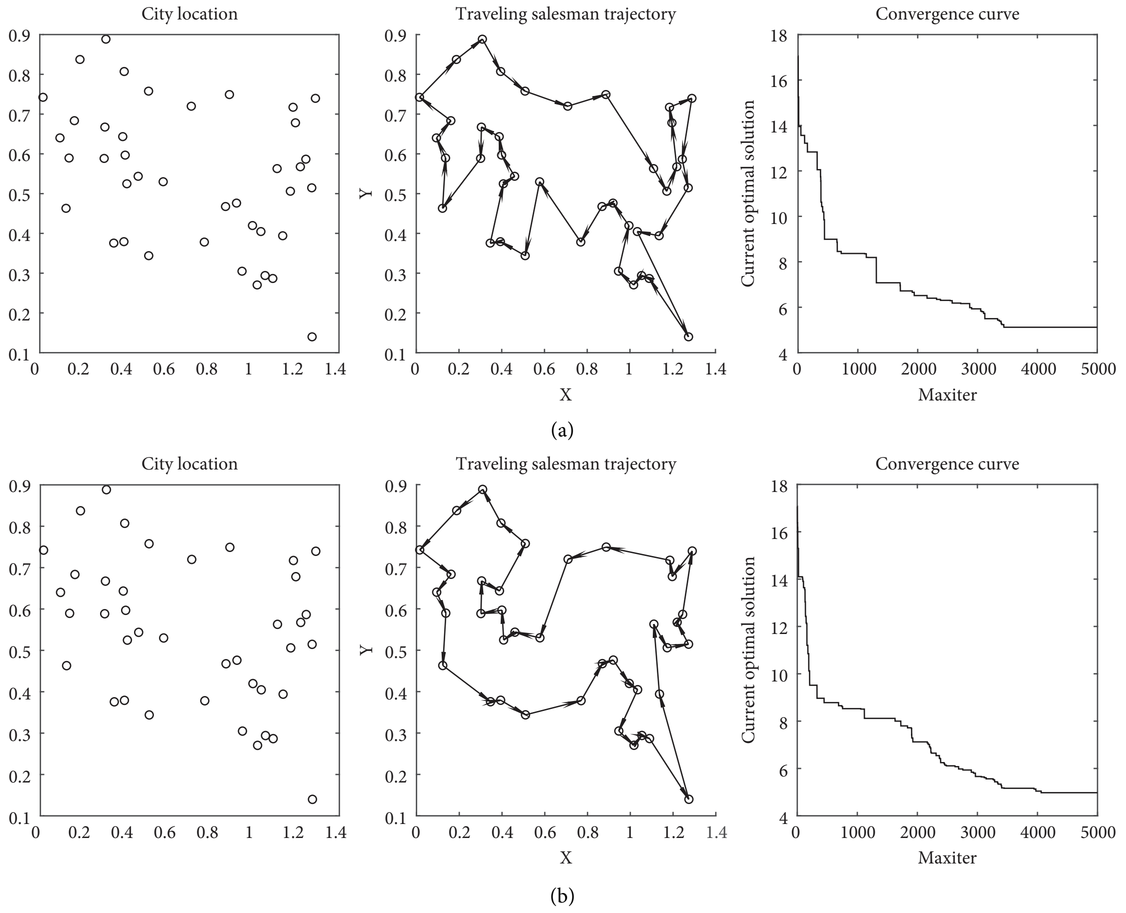

Figure 3: Comparative experimental diagram of the two algorithms. (a) The shortest path length of the MPA. (b) The shortest path length of the AMPA.

Set dimension $D = 10$, the population number $N = 100$, and the maximum number of iterations Max $\_iter = 1000$. The average value and standard deviation of each algorithm were recorded independently 30 times, and the optimal value was marked with a deepened font. The results of the ACMPA and other comparison algorithms are shown in Table 6.

It can be seen from Table 6 that the ACMPA significantly improves the average and standard deviation of 29 test functions. On a few functions $F6, F10, F14, F16, F20$, and $F30$, the average value of the improved function ACMPA fails to reach the optimal value, but in other test functions, the average value is better than other comparison algorithms. It shows that the convergence speed and convergence precision of the marine predators algorithm are improved by adding adaptive weight and chaos factor, which verifies the effectiveness of the improved algorithm.

Table 8: Shortest path length of two algorithms.

Algorithm	MPA	ACMPA
Mean	5.522221	**5.135867**
Std	0.268217	**0.094553**

4.6. Runtime Overhead. The runtime overhead of each algorithm is shown in the table (the algorithm is calculated within the CEC2017 function library, the number of populations is 30, and the number of iterations is 100). From the data in Table 7, we can conclude that the difference in running time between the improved algorithm and the previous algorithm is not significant. But from the analysis above, we can conclude that the effect of the improved algorithm increases significantly.

5. ACMPA Based on Traveling Salesman Problem

5.1. Traveling Salesman Problem Description. The TSP problem, also known as the salesman problem, is mainly described as a salesman going to sell products in several cities. The salesman starts from a certain city and requires to pass through each city, and each city can only be visited so that the total path is the shortest. Set the city node set as $X = \{x_1, x_2, \ldots, x_n\}$ and the distance between cities as $d(x_i, x_{i+1})$, and the objective function of the shortest path is

$$f = \min \sum d_{(x_i,x_{i+1})} + d_{(x_n,x_1)}. \tag{10}$$

5.2. Traveling Salesman Algorithm Test. The number of cities is set as 40, and the maximum iteration is 5000. In this experiment, the ACMPA and MPA are compared and analyzed. Table 8 shows the optimal access path length of 40 cities and the path length calculated by the two algorithms. Figures 3(a) and 3(b) show the shortest path length results of the two algorithms, respectively. For the TSP problem, we conducted 30 experiments and took the mean and variance to compare the degree of superiority of the results, and the results are shown in Table 8. From Table 8, we can get the following conclusions. The improved MPA has a better way to handle this problem.

It can be seen from Table 8 and Figure 3 that the path simulation of 40 city coordinates and the path optimization using the ACMPA are closer to the optimal solution, which has a great improvement in the path optimization compared with the MPA. In general, the ACMPA is effective in solving the traveling salesman problem.

6. Conclusion

Aiming at the problems of the MPA, like unbalanced exploration and exploitation, vulnerability to local extreme points, and so on, this paper proposes a marine predators algorithm based on adaptive weight and chaos factor.

Firstly, the adaptive weight is set as a dynamic step, and a large step length is given in the early stage of the algorithm to carry out an effective global search, which improves the convergence speed of the algorithm. At the same time, with the increase of the number of iterations, the step length is gradually reduced, which is more conducive to local careful search in the later stage and maintains the balance between algorithm exploration and exploitation. Secondly, logistic chaos mapping is used to replace random numbers; due to the ergodicity and randomness of the chaotic system, the scattered particles in space can effectively avoid repeated folding and improve the ability of the algorithm to fall into the local optimal solution. Applying the ACMPA to the TSP problem effectively improves the optimization accuracy of the original algorithm in practical applications. Finally, the application algorithm is compared and tested, which proves the effectiveness of the ACMPA. In the future research, the improved marine predators algorithm will be applied to large-scale engineering practice to further exploit the advantages of the optimization algorithm.

Data Availability

The data used to support the findings of the study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Authors' Contributions

Shujun Liang contributed to conceptualization, methodology, investigation, original draft preparation, and review and editing of the paper. Youmei Pan contributed to investigation and writing the original draft. Huanlong Zhang contributed to methodology and editing and review of the paper. Jie Zhang contributed to conceptualization and methodology. Fengxian Wang contributed to conceptualization, methodology, and investigation. Zhiwu Chen contributed to review and editing of the paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61873246, 62072416, 62006213, 6167241, and 61702462), the Program for Science & Technology Innovation Talents in Universities of Henan Province (21HASTIT028), the Natural Science Foundation of Henan (202300410495), and the Zhongyuan Science and Technology Innovation Leadership Program (214200510026).

References

[1] D. L. Applegate, R. E. Bixby, V. Chvátal, and J. Cook, "The traveling salesman problem," *The Traveling Salesman Problem*, Princeton university press, 2011.

[2] H. Zhang, Y. Pan, J. Zhang, K. Dai, and Y. Feng, "Tent chaos and nonlinear convergence factor whale optimization algorithm," *International Journal of Innovative Computing, Information and Control*, vol. 17, no. 2, pp. 687–700, 2021.

[3] Q. Meng and M. Zhang, "MULTI-RESOURCE EQUILIBRIUM OPTIMIZATION OF SCIENTIFIC research PROJECTS based ON PIGEON COLONY algorithm," *International Journal of Innovative Computing, Information and Control*, vol. 16, no. 5, pp. 1667–1680, 2020.

[4] X. Ran, X. Zhou, M. Lei, W. Tepsan, and W. Deng, "A novel k-means clustering algorithm with a noise algorithm for capturing urban hotspots," *Applied Sciences*, vol. 11, no. 23, Article ID 11202, 2021.

[5] M. M. Krishna, N. Panda, and S. K. Majhi, "Solving traveling salesman problem using hybridization of rider optimization and spotted hyena optimization algorithm," *Expert Systems with Applications*, vol. 183, Article ID 115353, 2021.

[6] G. G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Computing & Applications*, vol. 31, no. 7, pp. 1995–2014, 2019.

[7] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime mould algorithm: a new method for stochastic optimization," *Future Generation Computer Systems*, vol. 111, pp. 300–323, 2020.

[8] G. G. Wang, "Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Computing*, vol. 10, no. 2, pp. 151–164, 2018.

[9] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872, 2019.

[10] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, "Marine predators algorithm: a nature-inspired metaheuristic," *Expert Systems with Applications*, vol. 152, Article ID 113377, 2020.

[11] M. Abdel-Basset, R. Mohamed, M. Elhoseny, R. K. Chakrabortty, and M. Ryan, "A hybrid COVID-19 detection model using an improved marine predators algorithm and a ranking-based diversity reduction strategy," *IEEE Access*, vol. 8, pp. 79521–79540, 2020.

[12] M. Abdel-Basset, R. Mohamed, M. Elhoseny, A. K. Bashir, A. Jolfaei, and N. Kumar, "Energy-aware marine predators algorithm for task scheduling in IoTbased fog computing applications," *IEEE Transactions on Industrial Informatics*, 2020.

[13] M. Abd Elaziz, S. B. Thanikanti, I. A. Ibrahim et al., "Enhanced Marine Predators Algorithm for identifying static and dynamic Photovoltaic models parameters," *Energy Conversion and Management*, vol. 236, Article ID 113971, 2021.

[14] C. Ma, G. H. Zeng, and B. Huang, "Marine Predator Algorithm Based on Chaotic Opposition Learning and Group Learning," *Computer Engineering and Applications*, vol. 58, pp. 1–14, 2021.

[15] M. Oszust, "Enhanced Marine Predators Algorithm with Local Escaping Operator for Global Optimization," *Knowledge-Based Systems*, vol. 232, Article ID 107467, 2021.

[16] M. Abdel-Basset, R. Mohamed, M. Elhoseny, R. K. Chakrabortty, and M. Ryan, "A hybrid COVID-19 detection model using an improved marine predators algorithm and a ranking-based diversity reduction strategy," *IEEE Access*, vol. 8, pp. 79521–79540, 2020.

[17] D. Chu, H. Chen, and G. Wang, "Whale optimization algorithm based on adaptive weight and simulated annealing," *Acta Electronica Sinica*, vol. 47, no. 5, pp. 992–999, 2019.

[18] H. Deng, J. Li, and K. Hu, "Particle swarm optimization based on imitate neighborhood velocity strategy," *Computer Engineering And Design*, vol. 41, no. 10, pp. 2803–2811, 2020.

[19] Y. Zhang and F. Chen, "A modified whale optimization algorithm," *Computer Engineering*, vol. 44, no. 03, pp. 208–219, 2018.

[20] M. A. Soliman, H. M. Hasanien, and A. Alkuhayli, "Marine predators algorithm for parameters identification of triple-diode photovoltaic models," *IEEE Access*, vol. 8, pp. 155832–155842, 2020.

[21] M. Abdel-Basset, R. Mohamed, M. Elhoseny, A. K. Bashir, A. Jolfaei, and N. Kumar, "Energy-aware marine predators algorithm for task scheduling in IoT-based fog computing applications," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 5068–5076, 2021.

[22] M. Weiel, M. Götz, A. Klein, D. Coquelin, R. Floca, and A. Schug, "Dynamic particle swarm optimization of biomolecular simulation parameters with flexible objective functions," *Nature Machine Intelligence*, vol. 3, pp. 727–734, 2021.

[23] X. L. Zhang, Q. F. Wang, and W. L. Ji, "An improved particle swarm optimization algorithm for adaptive inertial weight," *Microelectronics & Computer*, vol. 36, no. 3, pp. 66–70, 2019.

[24] D. M. Zhang, Z. Y. Chen, and Z. Y. Xin, H. Zhang and W. Yan, Salp swarm algorithm based on craziness and adaptive," *Control and Decision*, vol. 35, no. 9, pp. 2112–2120, 2020.

[25] H. G. Zhu, L. Q. Tian, and N. Chen, "Dynamic adaptive inertial weight particle swarm optimization algorithm based on chaos optimization," *Journal of North Institute of Science and Technology*, vol. 18, no. 5, pp. 88–95, 2020.

[26] R. Sheik and A. Kaveh, "A survey of chaos embedded metaheuristic algorithms, International Journal of Optimization in Civil Engineering," *Int.j.optim.civil Eng*, vol. 3, no. 4, pp. 617–633, 2013.

[27] F. B. Demir, T. Tuncer, and A. F. Kocamaz, "A chaotic optimization method based on logistic-sine map for numerical function optimization," *Neural Computing & Applications*, vol. 32, no. 17, pp. 14227–14239, 2020.

[28] H. M. Mohammed, S. U. Umar, and T. A. Rashid, "A systematic and meta-analysis survey of whale optimization algorithm," *Computational Intelligence and Neuroscience*, vol. 2019, p. 1, 2019.

[29] Z. Song, S. Gao, Y. Yu, J. Sun, and Y. Todo, "Multiple chaos embedded gravitational search algorithm," *IEICE Transactions on Information and Systems*, vol. E100.D, no. 4, pp. 888–900, 2017.

[30] R. Zhuo and W. Q. Wang, "Self-adaptive salp swarm algorithm with chaotic mapping and dynamic learning," *Computer Engineering And Design*, vol. 42, no. 7, pp. 1963–1972, 2021.

[31] S. Mirjalili, "Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm," *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.

[32] M. Khishe and M. R. Mosavi, "Chimp optimization algorithm," *Expert Systems with Applications*, vol. 149, Article ID 113338, 2020.

[33] S. Mirjalili, "SCA: a sine cosine algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016.

[34] Q. Chen, B. Liu, Q. Zhang, J. Liang, P. Suganthan, and B. Qu, "Problem Definitions and Evaluation Criteria for CEC 2015 Special Session on Bound Constrained Single-Objective Computationally Expensive Numerical Optimization," *Computational Intelligence Laboratory, Zhengzhou University*, Technical Report", Zhengzhou, China and Nanyang Technological University, Zhengzhou, China, 2014.

[35] A. W. Mohamed, A. A. Hadi, A. M. Fattouh, and K. M. Jambi, "LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems," *IEEE Congress on evolutionary computation*, pp. 145–152, Donostia, Spain, June 2017.