machines

MDPI

Article

Improved Chimpanzee Search Algorithm with Multi-Strategy Fusion and Its Application

Hongda Wu, Fuxing Zhang and Teng Gao *

School of Mechanical Engineering and Automation, Dalian Polytechnic University, Dalian 116034, China
* Correspondence: gaoteng@dlpu.edu.cn

Abstract: An improved chimpanzee optimization algorithm incorporating multiple strategies (IMSChoA) is proposed to address the problems of initialized population boundary aggregation distribution, slow convergence speed, low precision, and proneness to fall into local optimality of the chimpanzee search algorithm. Firstly, the improved sine chaotic mapping is used to initialize the population to solve the population boundary aggregation distribution problem. Secondly, a linear weighting factor and an adaptive acceleration factor are added to join the particle swarm idea and cooperate with the improved nonlinear convergence factor to balance the global search ability of the algorithm, accelerate the convergence of the algorithm, and improve the convergence accuracy. Finally, the sparrow elite mutation and Bernoulli chaos mapping strategy improved by adaptive change water wave factor are added to improve the ability of individuals to jump out of the local optimum. Through the comparative analysis of benchmark functions seeking optimization and the comparison of Wilcoxon rank sum statistical test seeking results, it can be seen that the IMSChoA optimization algorithm has stronger robustness and applicability. Further, the IMSChoA optimization algorithm is applied to two engineering examples to verify the superiority of the IMSChoA optimization algorithm in dealing with mechanical structure optimization design problems.

Keywords: improved sine chaos mapping; nonlinear decay factor; sparrow elite mutation

check for updates

Citation: Wu, H.; Zhang, F.; Gao, T. Improved Chimpanzee Search Algorithm with Multi-Strategy Fusion and Its Application. *Machines* **2023**, *11*, 250. https://doi.org/10.3390/machines11020250

Academic Editor: Dan Zhang

Received: 14 December 2022
Revised: 30 January 2023
Accepted: 6 February 2023
Published: 8 February 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Meta-heuristic algorithms are widely used in path planning [1], image detection [2], system control [3], and shop floor scheduling [4] due to their excellent flexibility, practicality, and robustness. Common meta-heuristic algorithms include the genetic algorithm (GA) [5,6], the particle swarm optimization algorithm (PSO) [7,8], the gray wolf optimization algorithm (GWO) [9,10], the chicken flock optimization algorithm (CSO) [11], the sparrow optimization algorithm (CSA) [12], the whale optimization algorithm (WOA) [13], etc.

Different intelligent optimization algorithms exist with different search approaches, but most of them aim at the balance between population diversity and search ability and avoid premature maturity while ensuring convergence accuracy and speed [14]. In response to the above ideas, numerous scholars have proposed improvements to the intelligent algorithms they studied. For example, Zhi-jun Teng et al. [15] introduced the idea of PSO on the basis of the gray wolf optimization algorithm, which preserved the individual optimum while improving the ability of the algorithm to jump out of the local optimum; Hussien A. G. et al. [16] proposed two transfer functions (S-shaped and V-shaped) to map the continuous search space to the binary space, which improved the search accuracy and speed of the whale optimization algorithm; Wang et al. [17] introduced a fuzzy system in the process of chicken flock optimization algorithm, which adaptively adjusted the number of individuals in the algorithm, as well as random factors to balance the local exploitation performance and global search ability of the algorithm; Tian et al. [18] used logistic chaotic mapping to improve the initial population quality of the particle swarm algorithm while applying the auxiliary speed mechanism to the global optimal particles, which effectively

improved the convergence of the algorithm; Li et al. [19] integrated two strategies, Levy flight and dimension-by-dimension evaluation, in the mothballing algorithm to improve the global search capability and to enhance the effectiveness of the algorithm.

The chimpanzee optimization algorithm (ChoA) is a heuristic optimization algorithm based on the social behavior of chimpanzee populations proposed by Khishe et al. [20] in 2020. Compared with traditional algorithms, ChoA has the advantages of fewer parameters, being prone to understand, and high stability. However, it also has the problems of initialized population boundary aggregation distribution, slow convergence speed, low accuracy, and being prone to fall into the local optimum. To address these problems, many researchers have proposed different improvement methods. Du et al. [21] introduced a somersault foraging strategy in ChoA to avoid the algorithmic population from easily falling into the local optimum, as well as to improve the diversity of the pre-population. However, this relatively single improvement leads to its less obvious improvement effect; Kumari et al. [22] combined the SHO algorithm with the ChoA algorithm, which improved the convergence accuracy of the ChoA algorithm itself and enhanced its local exploitation ability to deal with high-dimensional problems; Houssein et al. [23] proposed to extend the population diversity in the search space of the ChoA algorithm in relation to the algorithm initialization phase by applying opposition-based learning (OBL).

In summary, there are numerous improvements to the chimpanzee optimization algorithm, and the improved algorithms are suitable for the optimization of some single problems but reveal shortcomings for others. Therefore, in order to improve the performance of ChoA optimization, an improved chimpanzee optimization algorithm incorporating multiple strategies (IMSChoA) is proposed in this paper. Firstly, an improved sine chaotic mapping is used to initialize the population and solve the phenomenon of population boundary aggregation distribution. Secondly, a linear weight factor and an adaptive acceleration factor are introduced to add to the particle swarm algorithm and cooperate with the improved nonlinear convergence factor to balance the search ability of the algorithm, accelerate the convergence of the algorithm, and improve the convergence accuracy. Finally, the sparrow elite mutation and Bernoulli chaos mapping strategy improved by adaptive change water wave factor are introduced to improve the ability of individuals to jump out of the local optimum. After 21 standard test functions for the optimization search test, and with the help of the Wilcoxon rank sum statistical test for the optimization results, the robustness and applicability of the improved algorithm are verified. Finally, the IMSChoA optimization algorithm is applied to two engineering examples to further verify the superiority of the IMSChoA optimization algorithm in dealing with mechanical structure optimization design problems.

The other sections of the article are organized as follows: in Section 2, the mathematical model of the traditional ChoA algorithm is presented. Section 3 presents the specific improvement strategies incorporated on top of the ChoA algorithm. Section 4 shows the comparison and analysis of the results of IMSChoA with the other four optimization algorithms after 21 standard test function search tests. Section 5 applies the IMSChoA algorithm to two engineering examples and analyzes their optimization results accordingly. Finally, the full text is summarized in Section 6 for discussion.

2. Basic Chimpanzee Algorithm

The ChoA algorithm is an intelligent algorithm proposed by simulating the prey-hunting behavior of chimpanzee groups. According to the abilities shown in the chimpanzee hunting process, individual chimpanzees are classified into driver, barrier, chaser, and attacker. The chimpanzee group hunting process is mainly divided into exploratory phases, i.e., repelling, blocking, and chasing prey. The development stage involves attacking the prey. Each type of chimpanzee has the ability to think independently and search for the location of prey in its own way, while chimpanzees are also affected by sexual behavior, making them appear to confuse individual hunting behavior in the final stage. It is assumed that the first driver, barrier, chaser, and attacker are able to predict the location of prey and the others update their

position according to the closest chimpanzee to the prey. The equation model for chimpanzee repelling and chasing prey is shown in Equations (1) and (2).

$$d(t) = |cX_P(t) - mX_E(t)| \tag{1}$$

$$X_E(t+1) = X_P(t) - a * d(t) \tag{2}$$

where X_P is the position of the prey, X_E is the position of the chimpanzee, m is the chaotic vector, t is the number of iterations, $d(t)$ is the distance of the chimpanzee from the prey, and a and c are the coefficient vectors. a and c are calculated by Equations (3) and (4), respectively. When $|a| < 1$, the chimpanzee individual tends to the prey, and when $|a| > 1$, it means the chimpanzee has deviated from the prey position and expanded the search range.

$$a = 2 * f * r_1 - f \tag{3}$$

$$c = 2 * r_2 \tag{4}$$

where r_1 and r_2 are random numbers taking values of [0, 1], a is a random variable between $[-2f, 2f]$, f is a linear convergence factor, and the calculation of the f formula is Equation (5).

$$f = 2 - \frac{2 * t}{t_{\max}} \tag{5}$$

During the iterations, f decays linearly from 2 to 0, and $t_{\max}$ is the maximum number of iterations.

The position of chimpanzees in the population is co-determined by the position of the driver, barrier, chaser, and attacker. The mathematical model of chimpanzee attack on prey is shown in Equations (6)–(8).

$$\begin{aligned} D_A &= |c * X_A - m * X| \\ D_B &= |c * X_B - m * X| \\ D_C &= |c * X_C - m * X| \\ D_D &= |c * X_D - m * X| \end{aligned} \tag{6}$$

$$\begin{aligned} X_1 &= X_A - a * X_A \\ X_2 &= X_B - a * X_B \\ X_3 &= X_C - a * X_C \\ X_4 &= X_D - a * X_D \end{aligned} \tag{7}$$

$$X(t+1) = \frac{1}{4} * (X_1 + X_2 + X_3 + X_4) \tag{8}$$

From Equations (6)–(8), the position of the prey is estimated from the position of the driver, barrier, chaser, and attacker. Other chimpanzees update their position in the direction of the prey.

In the final stages of a population's predation, when individuals obtain food satisfaction, chimpanzees unleash their natural instinct to force chaotic access to food. The chaotic behavior of chimpanzees in the final stage helps to further alleviate the two problems of local optimal traps and slow convergence when the problem is high-dimensional. To simulate the chimpanzee's chaotic behavior, it is assumed that there is a 50% probability of choosing one of the update positions in either the normal update position mechanism or the chaotic model, and the model formulation is shown in Equation (9) [24].

$$X_E(t+1) = \begin{cases} X_P(t) - a * d, \mu < 0.5 \\ Chaotic, \quad \mu > 0.5 \end{cases} \tag{9}$$

where μ takes the value of [0, 1] random number, and Chaotic is the chaotic mapping used to update the position.

3. Improving the Chimpanzee Algorithm

Firstly, for ChoA, the population initialization is performed by the random distribution method. This approach leads to population diversity, poor uniformity, easy boundary aggregation phenomenon, and large blindness of individual search for the best result. Secondly, the convergence factor of linear decay of the algorithm balancing local search and global search does not conform to the nonlinear merit-seeking characteristics of the algorithm, and finally, the algorithm jumps out of the local optimum with low chaotic perturbation trigger probability, which has great instability.

In summary, the corresponding improvement strategies are introduced for the problems of the ChoA algorithm, as follows.

3.1. Improved Sine Chaotic Mapping for Initializing Populations

Because the size of each dimension of chimpanzee individuals is randomly generated in the initialization stage, which leads to poor population diversity, serious boundary aggregation, and low individual variability. Chaotic searches are based on non-repetition and ergodicity, which are different from stochastic search methods, which are based on probabilities [25]. The common chaotic mappings include circle chaotic mapping, tent chaotic mapping, iteration chaotic mapping, logistic chaotic mapping, and sine chaotic mapping. Among them, sine mapping has good stability and high coverage, but it still has uneven distribution and boundary aggregation phenomena.

The expression of the original sine chaos mapping is:

$$X(t) = \mu * \sin(\pi X(t-1)) \tag{10}$$

Therefore, the sine mapping is improved by introducing the Chebyshev mapping for the above problem. At the same time, a high-dimensional chaotic mapping is established to make it better represent the chaotic property based on the original one.

The expression of the improved sine chaos mapping is:

$$P(t) = \mu * \sin(\pi X(t-1)) + \lambda \cos(i * \cos^{-1}(X(t-1))) \tag{11}$$

$$\begin{cases} W(t) = \mu \sin(\pi W(t-1)) + \lambda \cos(t * \cos^{-1}(W(t-1)) \\ O(t) = \mu \sin(\pi O(t-1)) + \lambda \cos(t * \cos^{-1}(O(t-1)) \\ E(t) = \mu \sin(\pi E(t-1)) + \lambda \cos(t * \cos^{-1}(E(t-1)) \\ H(t) = \mu \sin(\pi H(t-1)) + \lambda \cos(t * \cos^{-1}(H(t-1)) \\ X(t) = \mathrm{mod}(W(t) + O(t) + E(t) + H(t), 1) \end{cases} \tag{12}$$

where λ and μ are random numbers between [0, 1] and satisfy the sum of λ and μ as 1.

The dimensional distribution map and dimensional distribution histograms of both initial solutions before and after the improvement are shown in Figure 1. Here, Figure 1a,c is the original sine mapping, and Figure 1b,d is the improved sine mapping. Comparing Figure 1a,b and Figure 1c,d, it was found that the value distribution of the improved sine mapping chaos is more uniform, and the boundary aggregation problem is effectively solved.

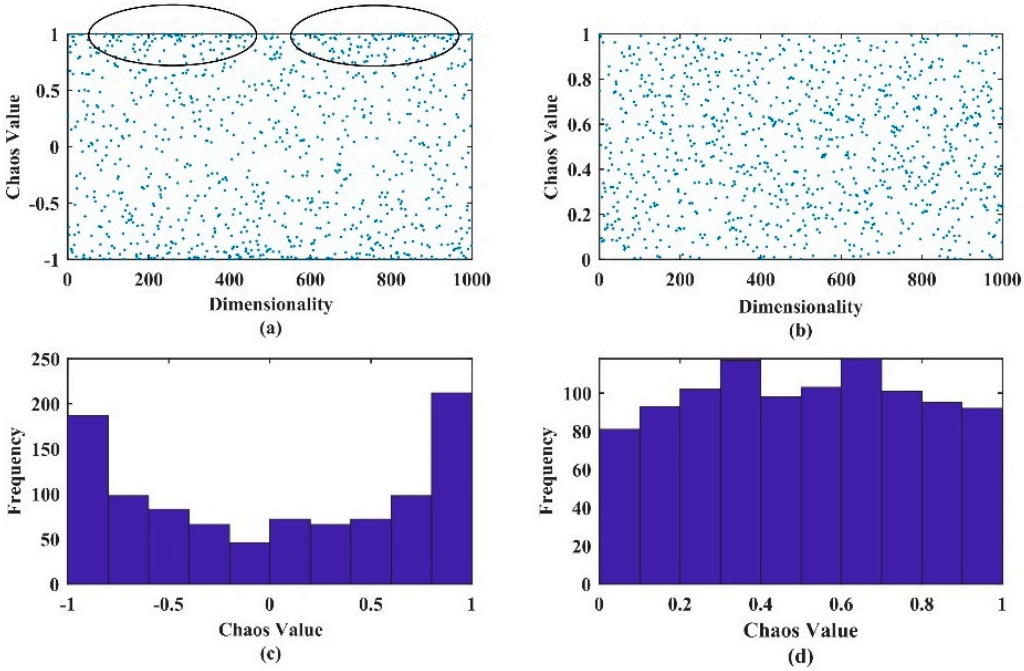


Figure 1. (**a**) Dimensional distribution map of the solutions generated by the original sine mapping. The circle in (**a**) is the focused observation area, demonstrating the boundary aggregation phenomenon; (**b**) dimensional distribution map of the solutions generated by the improved sine mapping; (**c**) dimensional distribution histogram of the solutions generated by the original sine mapping; (**d**) dimensional distribution histogram of the solutions generated by the improved Sine mapping.

3.2. PSO Idea and Nonlinear Convergence Factor

3.2.1. PSO Idea

In order to regulate the balance of global and local search ability in the early and late stages of the ChoA algorithm, the particle swarm idea is introduced to improve the position updating method of the ChoA algorithm. The best position information experienced by the particle itself and the best position information of the population are used to update the current position of the particle and realize the information exchange between individual chimpanzees and the population [26]. The position update formula is Equation (13).

$$X(t+1) = C * (rand * w * (X_1 + X_2 + X_3 + X_4) + rand * (X_1 - X(t))) \quad (13)$$

where w is the inertia weight coefficient and C is the acceleration factor. The inertia weight w and acceleration factor C take values related to the influence of the past motion state of the particle on the present motion state, when w and C become large, the search space of the particle will be expanded, and, when w and C become small, the direction of particle motion produces many changes, and the search space is relatively small, which will lead the algorithm to fall into the local optimum. The convergence speed of the algorithm is accelerated by adjusting the size of w, C to regulate the ability of local search and global search of the algorithm [27].

In order to improve the global search ability of the algorithm in the early stage while strengthening the local optimization ability of the particle swarm in the later stage, this paper introduces an adaptive linear acceleration factor: at the beginning of the algorithm iteration, a larger value of C is given to complete a wide range of search, which is conducive to the algorithm to quickly search for the global optimal position, and, as the number of iterations increases, the algorithm gradually converges, and individuals search for the optimal solution locally, and, at this time, a smaller C value is given to achieve accurate exploration of the optimal position in small steps size so as to improve the convergence accuracy of the algorithm. At the same time, in order to prevent the algorithm from falling

into the local optimum during the iteration process, the cosine function is introduced to correct the acceleration factor and to keep the acceleration factor fluctuating at all times. The adaptive acceleration factor mathematical model is shown in Equation (14), and the variation of C with the number of iterations is shown in Figure 2.

$$C = g * \frac{e^{-0.1\sqrt{t}}}{t_{\max}} * \cos(0.1t + 24.5) + 1.1 \tag{14}$$

where g is the adjustment trade-off factor, and $t_{\max}$ is the maximum number of iterations.

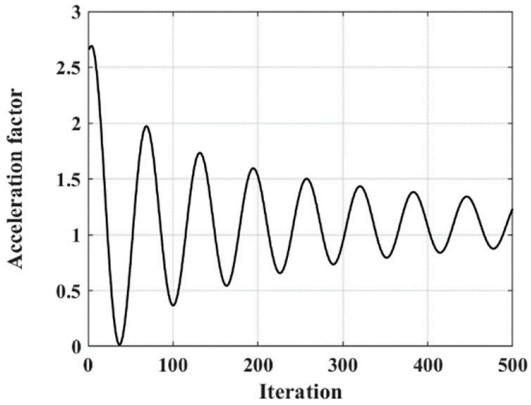


Figure 2. Change curve of acceleration factor.

At the same time, in order to accelerate the convergence of the algorithm and improve the convergence accuracy, the linear inertia weight model is introduced. At the beginning of the algorithm iteration, a large weight is given to make the population search the solution space extensively in large steps and to search the global optimal position quickly. With an increase in iteration number, the algorithm converges gradually at this time. At this time, the inertia weight coefficient gradually becomes smaller to facilitate the fine search of the optimal position in small steps and to improve the convergence accuracy of the algorithm. The linear inertia weight mathematical model is shown in Equation (15), and w varies with the number of iterations, as in Figure 3.

$$w = 0.5 * \left(w_{\max} - \frac{t * (w_{\max} - w_{\min})}{t_{\max}} * \cos^2(0.005 * t + 10) \right) \tag{15}$$

where $w_{\max}$ and $w_{\min}$ are the maximum weight coefficient and minimum weight coefficient, respectively, t is the number of iterations, and $t_{\max}$ is the maximum number of iterations.

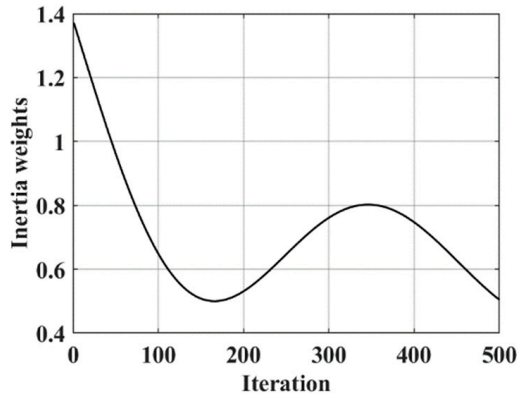


Figure 3. Curve of inertia weight change.

3.2.2. Nonlinear Decay Convergence Factor

One of the important factors in evaluating the performance of heuristic algorithms is the ability to balance the algorithm's global search ability and local search ability. From the analysis of the chimpanzee algorithm, it is known that, when $|a| < 1$, the chimpanzee individual converges to the prey, and, when $|a| > 1$, this means that the chimpanzee has deviated from the prey position and expanded the search range. Therefore, the change in the convergence factor determines the global and local search ability of the algorithm. According to the above description, this paper introduces a nonlinear decay variation model, which cooperates with the adaptive acceleration factor in the particle swarm idea to jointly balance the global search ability and local search ability of the algorithm. Meanwhile, a control factor б is introduced to control the decay amplitude. The nonlinear decay convergence factor mathematical model is described as Equation (16).

$$f = f_g * \left[1 - \left(\frac{e^{\frac{t}{t_{\max}}} - 1}{e - 1}\right)^{\delta}\right] \tag{16}$$

where t is the number of iterations, $t_{\max}$ is the maximum number of iterations, and f_g is the initial convergence factor. б∈[1, 10], and, the larger the б, the slower the decay rate, as shown in Figure 4.

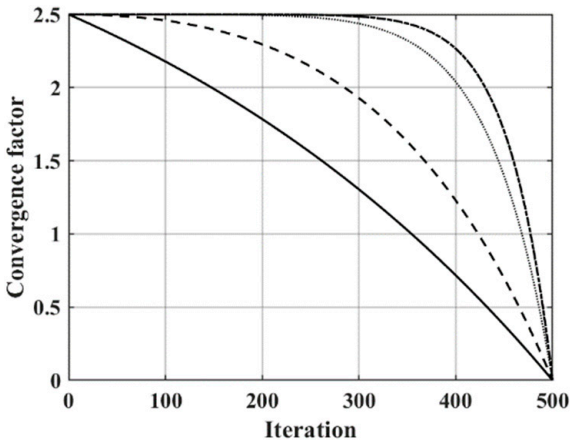


Figure 4. Comparison curve of convergence factors.

3.3. Improved Sparrow Elite Variation and Logistic Chaos Mapping

In the ChoA algorithm, the individual update is affected by the last optimal individual in each iteration, so the ChoA algorithm is easy to converge to the local optimum during the iterative process. To address the above problems, an optimization strategy combining adaptive water wave factor improved sparrow elite mutation and Bernoulli chaotic mapping is proposed.

3.3.1. Improved Sparrow Elite Variation and Logistic Chaos Mapping

The sparrow search algorithm is an efficient population intelligence optimization algorithm, which divides the search population into three parts: explorers, followers, and early warners, whose work is divided among themselves to find the optimal value [28]. Sparrow elite mutation is used to assign the capabilities of individuals with higher search performance to the current optimal individual. At each ChoA iteration, the individuals with the top 40% of the current fitness value are given a stronger optimization ability, and an adaptive water wave factor is added to the mutant individual update formula [29] to

further improve the optimization ability of mutant individuals. The sparrow elite mutation mathematical description is shown in Equation (17).

$$X(t+1)_{0.4} = \begin{cases} X(t)_{0.4} \cdot v \cdot \exp\left(-\frac{t}{\alpha \cdot t_{\max}}\right) & R < ST \\ X(t)_{0.4} + v \cdot Q \cdot L & R \geq ST \end{cases} \tag{17}$$

where $X(t)_{0.4}$ is for the top 40% of the current fitness value of the individual, Q is a random number obeying a normal distribution of [0, 1], L is a 1 × d matrix with all elements of 1, ST is the warning value, taken as 0.6, and v is the water wave factor, which varies adaptively with the number of iterations. The mathematical model of the adaptive water wave factor is shown in Equation (18).

$$v = 1 - \sin\left(\frac{\pi \cdot t}{2 \cdot t_{\max}} + 2 \cdot \pi\right) \tag{18}$$

As the iterations increase, the uncertainty in the iterative process and the dramatic abrupt changes in the water wave factor enhance the ability of individuals to jump out of the local optimum. The water wave factor changes are shown in Figure 5.

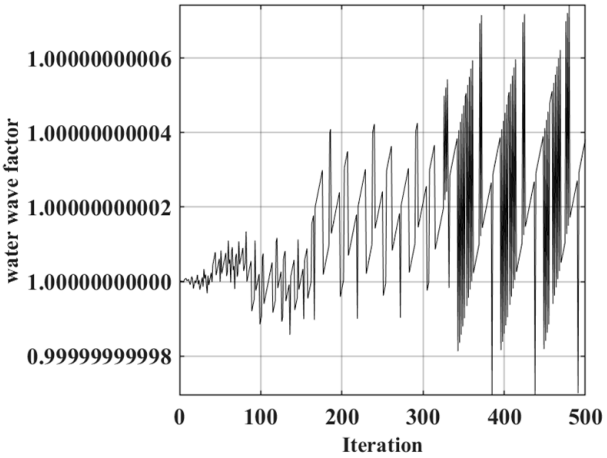


Figure 5. Adaptive water wave factor distribution with 500 iterations.

3.3.2. Bernoulli Chaotic Mappings

Bernoulli chaotic mapping is a classical representative of chaotic mapping and is more widely used [30]. Its mathematical expression is shown in Equation (19).

$$Z_{t+1} = \begin{cases} Z_t/(1-\lambda) & Z_t \in (0, 1-\lambda] \\ (Z_t - 1 + \lambda)/\lambda & Z_t \in (0, 1-\lambda] \end{cases} \tag{19}$$

where t is the number of chaotic iterations and λ is the conditioning factor, generally taken as 0.4. The resulting new chaotic sequence roots are mapped into the search space of the solution as follows.

$$X_{td} = X_L + (X_U - X_L) * Z_{td} \tag{20}$$

where X_{td} is the position of the tth element in d dimensions, X_U and X_L are analyzed as the upper and lower bounds of the search space, and Z_{td} is the chaotic value generated by Equation (19).

3.4. IMSChoA Algorithm Flow

The specific implementation steps of the IMSChoA algorithm are as follows.

Step 1: Initialize the population using the improved sine chaotic mapping, including the number of population individuals N, the maximum number of iterations $t_{\max}$, the

dimension d, the search boundary u_b, and l_b, the maximum and minimum weight factors, and the adjustment trade-off factor g, and set the relevant parameters.
Step 2: Update the acceleration factor, inertia weight, convergence factor, and water wave factor.
Step 3: Calculate the position of each chimpanzee.
Step 4: Update the positions of repellers, blockers, pursuers, and attackers.
Step 5: Calculate the adaptation degree value and the average value of the adaptation degree to find the global optimum and individual optimum.
Step 6: Compare the individual adaptation degree value f with the average value of adaptation degree $favg$. If $f < favg$, perform Brenoylli perturbation to determine whether the perturbed individual is better than the original individual, and update if better. Otherwise, keep the original individual unchanged; if $f > favg$, perform sparrow elite variation, and replace it if it is better than the original individual, otherwise keep it.
Step 7: Update the global optimal value of the population and the individual optimal value.
Step 8: Determine whether the condition is satisfied, and output the result if satisfied, otherwise return to step 2 for execution.

The flow chart is shown in Figure 6.

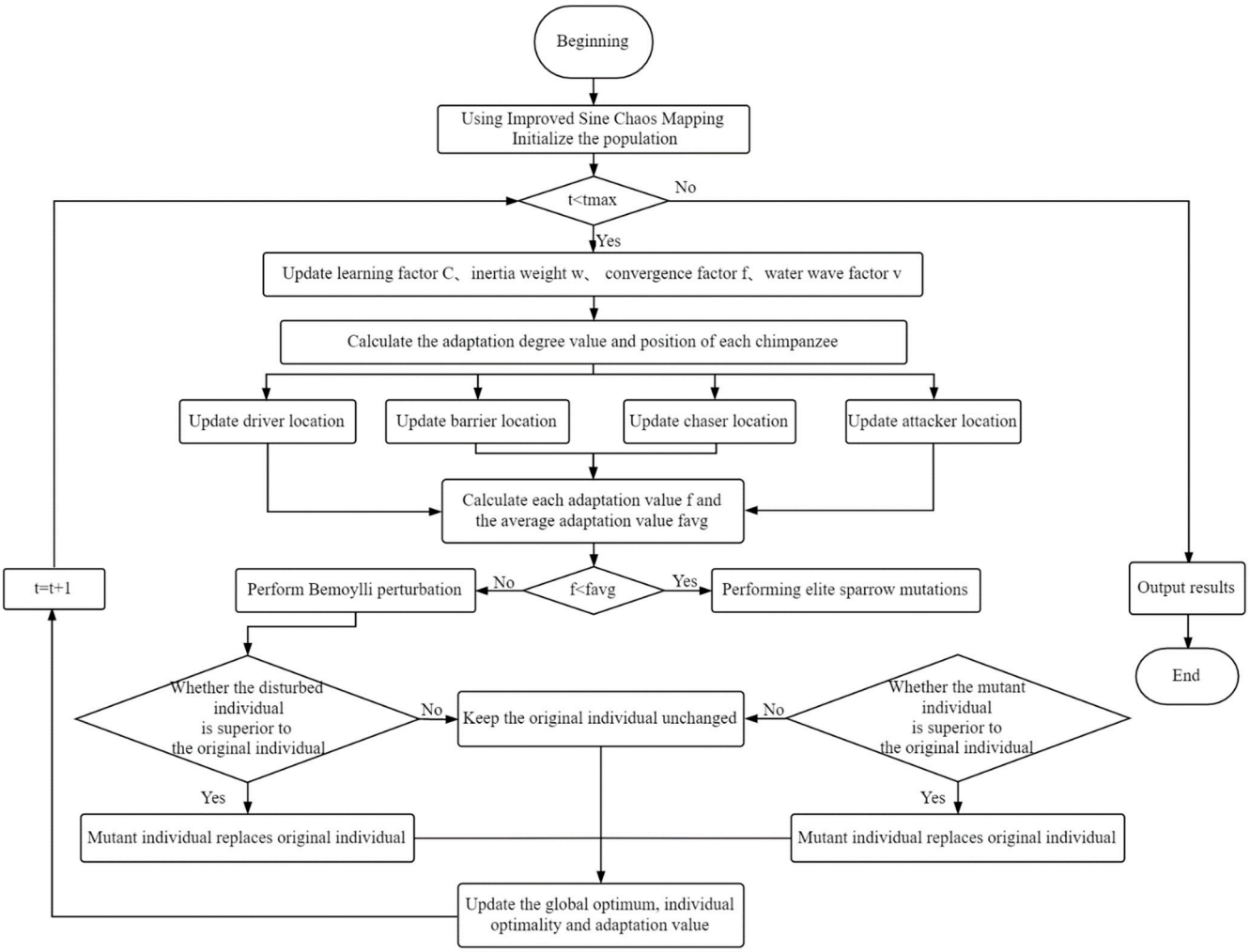


Figure 6. Flow chart of IMSChoA algorithm optimization.

3.5. Time Complexity Analysis

Time complexity is an important index reflecting the performance of the algorithm [31]. Assuming that the chimpanzee population size is N, the search space dimension is n, the initialization time is t_1, the update time of individual chimpanzee positions is t_2, and the

time to solve for the value of the target fitness function is $f(n)$, the time complexity of the ChoA algorithm is:

$$T_1(n) = O(t_1 + N(nt_2 + f(n))) = O(n + f(n)) \tag{21}$$

In the IMSChoA algorithm, the time required to initialize the parameters is kept consistent with the standard ChoA. The time used to initialize the population using the modified sine is t_3, which is employed in the loop phase, assuming that the time required to introduce the particle swarm idea, the nonlinear convergence factor, the modified sparrow elite variation, and the logistic chaos mapping are t_4, t_5, and t_6, respectively. Then, the time complexity of IMSChoA is:

$$T_2(n) = O(t_1 + t_3 + N(nt_2 + t_4 + t_5 + nt_6 + f(n))) = O(n + f(n)) \tag{22}$$

The time complexity of SPWChoA and ChoA is the same by Equations (21) and (22). It is shown in Equation (23).

$$T_1(n) = T_2(n) = O(n + f(n)) \tag{23}$$

In summary, the improvement strategy proposed in this paper for the ChoA defect does not increase the time complexity.

4. Algorithm Performance Testing

4.1. Experimental Parameter Settings

In this paper, the PSO algorithm, GWO algorithm, IMSChoA algorithm, ChoA algorithm, and MFO algorithm are selected for the optimization search comparison. The basic parameters were uniformly set as follows: population size N = 30, the maximum number of iterations t_{max} = 500, and the internal parameters of the algorithm are shown in Table 1.

Table 1. Parameter table of the algorithm.

Algorithm	Parameters
PSO	C_1 = 1.445; C_2 = 1.445; w_{max} = 2.0; w_{min} = 0.5
GWO	a decreases linearly from 1.5 to 0; r_1,r_2∈[0, 1]
IMSChoA	w_{max} = 2.5; w_{min} = 0.05; f_g = 2.5; λ = 0.4; g = 1000
ChoA	m = chaos (3,1,1)
MFO	t∈[k, 1]; k varies linearly between −1 and −2; b = 1

4.2. Benchmark Test Functions

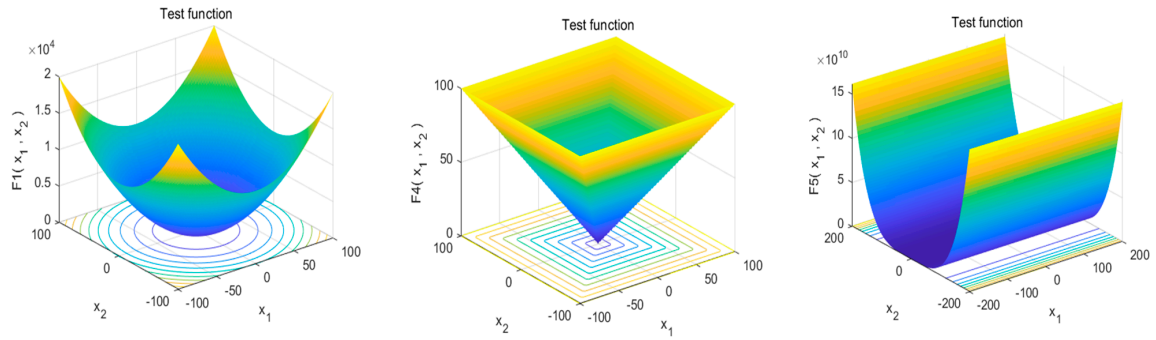
To verify the effectiveness of the improved chimpanzee algorithm (IMSChoA), 21 benchmark test functions used in the literature [32] were experimentally selected for the optimization test, as shown in Table 2. F1~F7 are continuous unimodal test functions, F8~F12 are continuous multimodal test functions, and F13~F21 are fixed multimodal test functions. Figures 7–9 show some of the several continuous unimodal test functions, continuous multimodal test functions, and fixed multimodal test functions function value distributions, respectively.

Table 2. Benchmark functions.

No.	Function Name	Definition Domain	Dimensionality	Optimal Value	Absolute Accuracy Error
F1	Sphere	[−100, 100]	30	0	1.00×10^{-3}
F2	Schwefel' problem 2.22	[−10, 10]	30	0	1.00×10^{-3}
F3	Schwefel' problem 1.2	[−100, 100]	30	0	1.00×10^{-3}

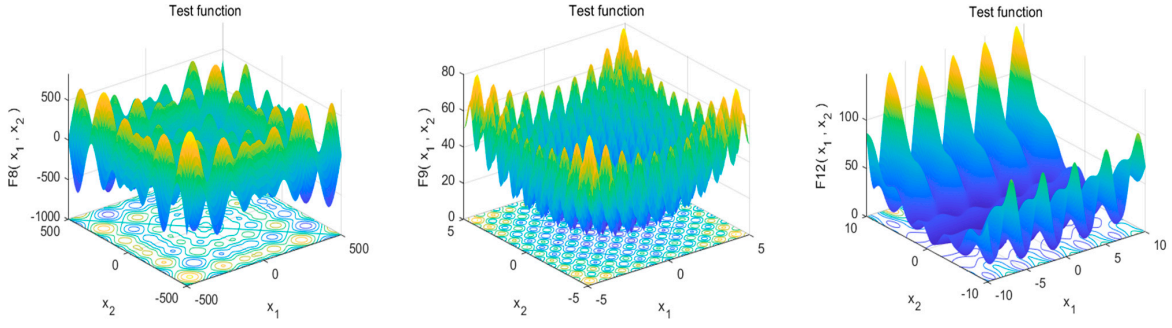
Table 2. *Cont.*

No.	Function Name	Definition Domain	Dimensionality	Optimal Value	Absolute Accuracy Error
F4	Schwefel' problem 2.21	[−100, 100]	30	0	1.00×10^{-3}
F5	Generalized Rosenbrock's Function	[−30, 30]	30	0	1.00×10^{-2}
F6	Step Function	[−100, 100]	30	0	1.00×10^{-2}
F7	Quartic Function	[−1.28, 1.28]	30	0	1.00×10^{-2}
F8	Generalized Schwefel's problem	[−500, 500]	30	−12,569.5	1.00×10^{2}
F9	Generalized Rastrigin's Function	[−5.12, 5.12]	30	0	1.00×10^{-2}
F10	Ackley's Function	[−32, 32]	30	0	1.00×10^{-2}
F11	Ceneralized Criewank Function	[−600, 600]	30	0	1.00×10^{-2}
F12	Ceneralized Penalized Function	[−50, 50]	30	0	1.00×10^{-2}
F13	Branin Function	[−5, 5]	2	0.398	1.00×10^{-2}
F14	Shekell's Foxholes Function	[−65, 65]	2	1	1.00×10^{-2}
F15	Kowalik's Function	[−5, 5]	4	0.0003	1.00×10^{-2}
F16	Six-Hump Camel-Back Function	[−5, 5]	2	−1.03	1.00×10^{-2}
F17	Goldstein-Price Function	[−2, 2]	2	3	1.00×10^{-2}
F18	Hatman's Function1	[0, 1]	3	−3.86	1.00×10^{-2}
F19	Hatman's Function2	[0, 1]	6	−3.32	1.00×10^{-2}
F20	Shekel's Family 1	[0, 10]	4	−10	1.00×10^{-2}
F21	Shekel's Family 2	[0, 10]	4	−10	1.00×10^{-2}



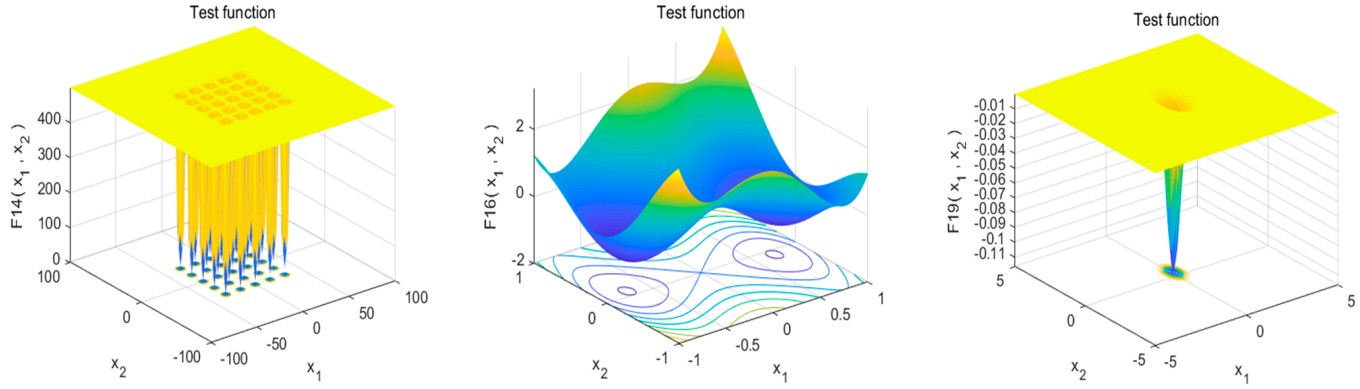
(**a**) (**b**) (**c**)

Figure 7. Distribution diagram of different continuous unimodal test function values. (**a**) F1 function; (**b**) F4 function; (**c**) F5 function.



(**a**) (**b**) (**c**)

Figure 8. Distribution diagram of different continuous multimodal test function values. (**a**) F8 function; (**b**) F9 function; (**c**) F12 function.



(**a**) (**b**) (**c**)

Figure 9. Distribution diagram of different fixed multimodal test function values. (**a**) F14 function; (**b**) F16 function; (**c**) F19 function.

4.3. Comparison of the IMSChoA Algorithm with Other Algorithms

Twenty-one basic test functions are selected to perform the optimization search test for the algorithm mentioned in the summary of 4.1, and the iteration curves are shown in Figure 10. Among them, F1~F7 are continuous unimodal test functions with only one global optimum and no local optimum, which are used to test the global search ability and convergence speed of the algorithm. From Figure 10a–g, it can be seen that the IMSChoA optimization algorithm has a fast convergence speed and strong global search capability, which is faster and stronger than other group intelligence optimization algorithms in terms of iteration speed and preliminary global search capability. F8~F13 are continuous multimodal test functions with multiple local optima, which are used to test the ability of the algorithm to jump out of the local optimum. From Figure 10h–l, it can be seen that the IMSChoA optimization algorithm falls into the local optimum when calculating F8, F9, F11, and F12 functions, but it quickly jumps out of the current state and continues to search for the optimal iteration, which proves that the IMSChoA optimization algorithm has a strong ability to jump out of the local optimum, which is further improved compared with the ChoA optimization algorithm. F13~F21 are fixed multimodal test functions, which are used to test the equilibrium development capability and stability of the system. As can be seen from Figure 10m–u, the IMSChoA optimization algorithm can well balance the iterative ability of the algorithm and quickly complete the optimization search test, and the algorithm has significantly improved in terms of convergence accuracy and stability. However, it can be seen in Figure 10t,u that there are still some functions that still fall into the local optimum in the iterative process, and although the final completion jumps out of the local optimum in relation to finding the optimal solution, the iteration time is longer. For the overall function iteration graph, we can conclude that the algorithm is better for the high-dimensional, large range of optimal function search, while for the low-dimensional one, a small range of optimal search still has a certain disadvantage, although compared with the other optimization algorithms, they still have some inadequacies regarding the iteration speed and jumping out of the local optimum problem. There is still room for improvement.

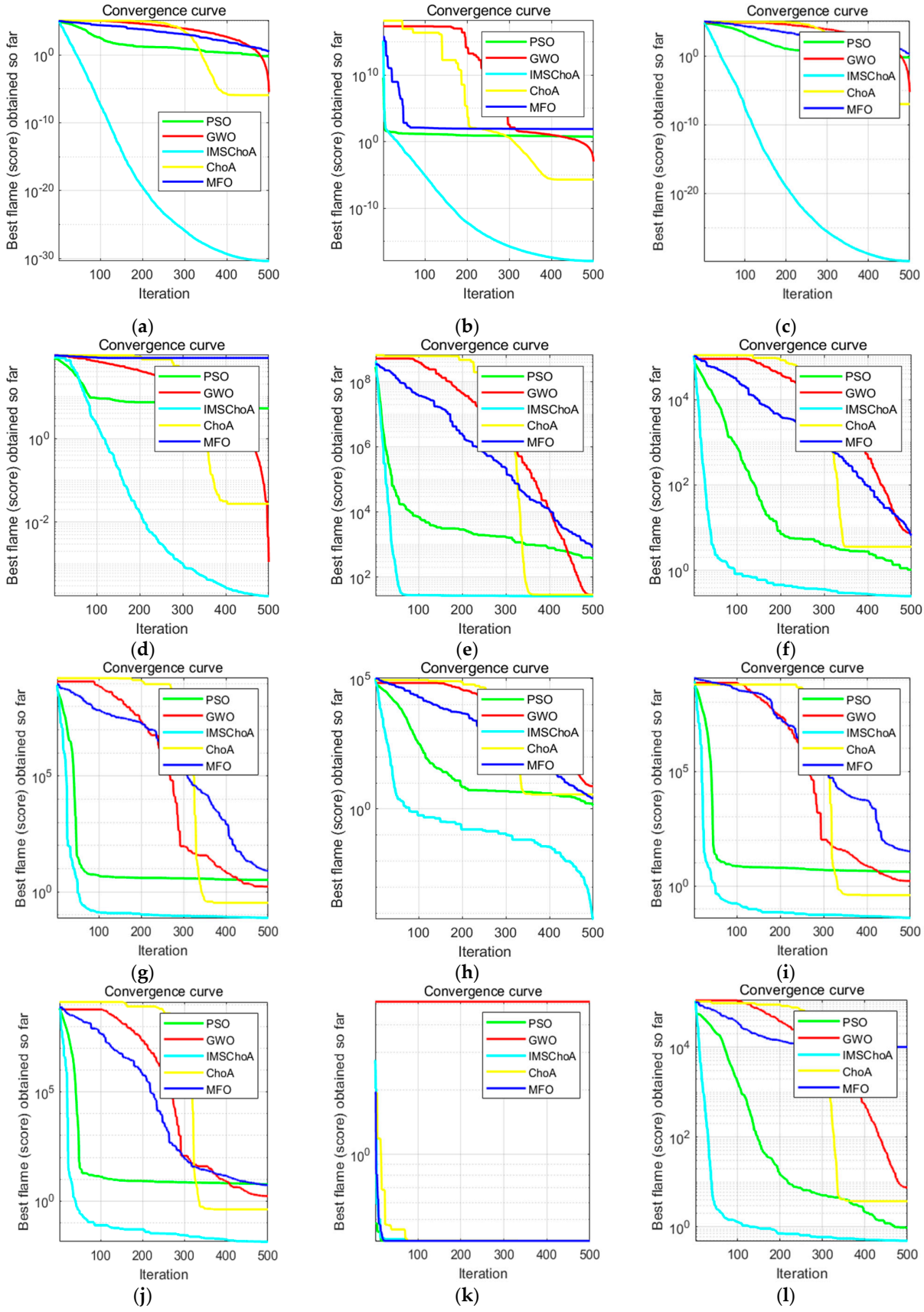


Figure 10. *Cont.*

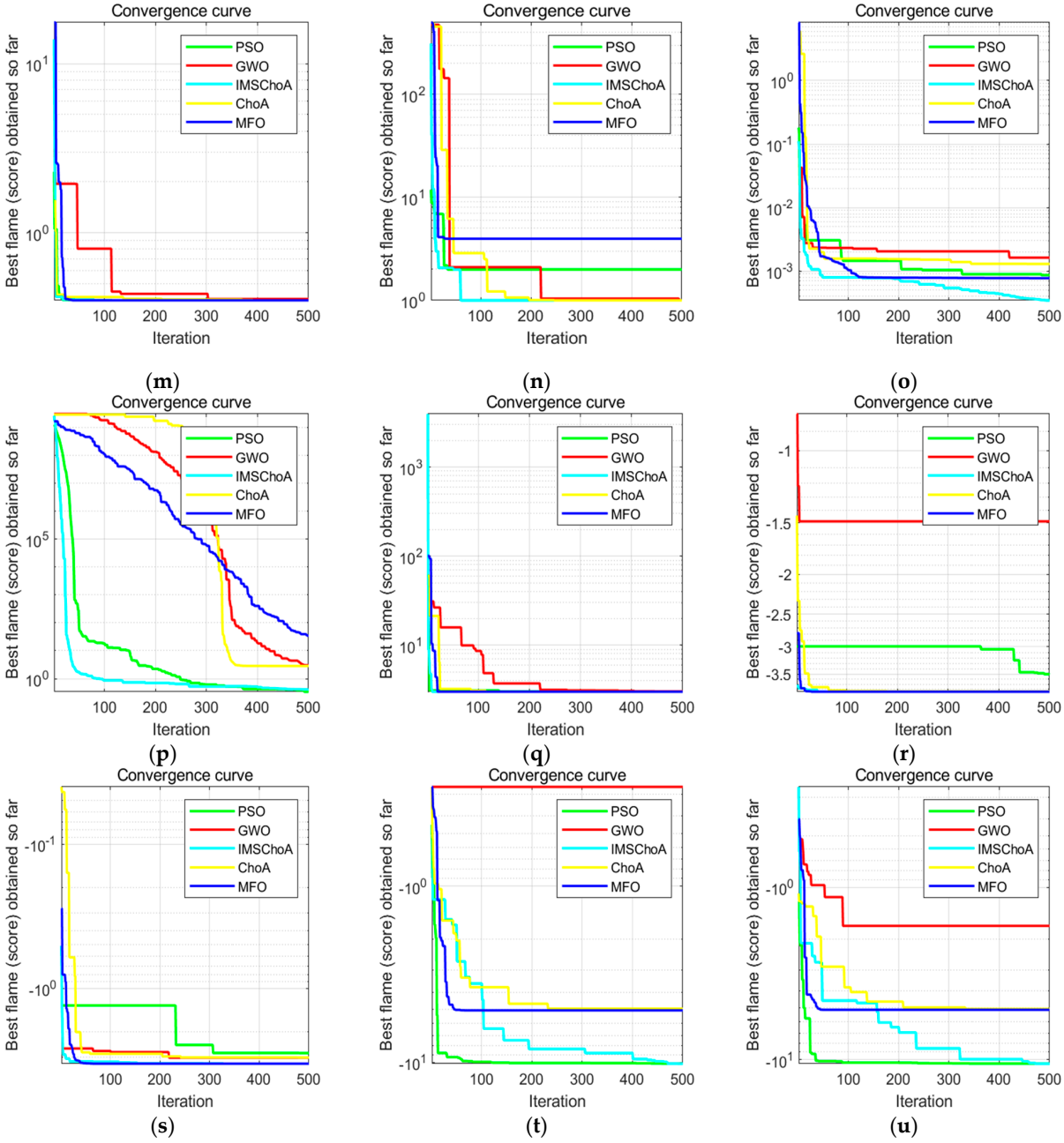


Figure 10. Convergence curves of different functions. (**a**) F1 function; (**b**) F2 function; (**c**) F3 function; (**d**) F4 function; (**e**) F5 function; (**f**) F6 function; (**g**) F7 function; (**h**) F8 function; (**i**) F9 function; (**j**) F10 function; (**k**) F11 function; (**l**) F12 function; (**m**) F13 function; (**n**) F14 function; (**o**) F15 function; (**p**) F16 function; (**q**) F17 function; (**r**) F18 function; (**s**) F19 function; (**t**) F20 function; (**u**) F21 function.

In order to maintain the fairness of the test environment, twenty-one basic test functions are selected, and each algorithm is run 50 times independently, and the test results are shown in Table 3. The optimal value, mean value, and standard deviation reflect the convergence accuracy, convergence speed, and optimality-seeking stability of the algorithms, respectively. Compared with other algorithms, the IMSChoA algorithm can find a fixed optimal value in each function, and the computational performance of all functions is better than that of the PSO algorithm, except for F20 and F21, which are slightly worse than the PSO algorithm in terms of finding speed. Compared with the MFO algorithm, IMSChoA outperforms the MFO algorithm in terms of computational performance for all functions, except for the F18 function, which is slightly less stable than the MFO algorithm. Compared with the ChoA algorithm

and the GWO algorithm, IMSChoA outperforms both of them in all aspects. This proves that the IMSChoA optimization algorithm has certain advantages in convergence accuracy, convergence speed, and stability of the optimization search.

Table 3. Experimental results of function test (30 dimensions).

Function Name	Algorithm	Optimum Value	Average Value	Standard Deviation
F1	PSO	1.7739×10^{0}	4.2242×10^{3}	1.3399×10^{4}
	GWO	3.8341×10^{-6}	3.8284×10^{4}	4.0132×10^{4}
	IMSChoA	2.3526×10^{-30}	9.8425×10^{2}	7.5512×10^{3}
	ChoA	6.2740×10^{-11}	4.1370×10^{4}	3.9220×10^{4}
	MFO	7.8368×10^{1}	1.0170×10^{4}	1.8759×10^{4}
F2	PSO	5.8589×10^{0}	9.9897×10^{8}	2.2020×10^{10}
	GWO	0.0008×10^{0}	5.5000×10^{16}	7.3352×10^{16}
	IMSChoA	3.3714×10^{-18}	8.1379×10^{3}	1.8164×10^{8}
	ChoA	9.2287×10^{-6}	1.4540×10^{13}	43.4356×10^{13}
	MFO	5.0048×10^{1}	2.4481×10^{16}	3.4463×10^{17}
F3	PSO	0.6379×10^{0}	5.2282×10^{3}	1.3308×10^{4}
	GWO	5.7064×10^{-6}	3.2449×10^{4}	3.1888×10^{4}
	IMSChoA	1.2559×10^{-30}	7.7739×10^{2}	6.0601×10^{3}
	ChoA	9.8877×10^{-8}	4.5275×10^{4}	4.3573×10^{4}
	MFO	2.0501×10^{0}	1.4254×10^{4}	2.5940×10^{4}
F4	PSO	2.8173×10^{0}	1.2901×10^{1}	1.6815×10^{1}
	GWO	4.9114×10^{-4}	3.6630×10^{1}	3.0889×10^{1}
	IMSChoA	2.4606×10^{-4}	1.2484×10^{1}	2.8149×10^{1}
	ChoA	1.8419×10^{-2}	6.0327×10^{1}	4.3718×10^{1}
	MFO	9.2480×10^{1}	9.2986×10^{1}	1.3286×10^{2}
F5	PSO	6.7185×10^{2}	2.4067×10^{6}	1.6751×10^{7}
	GWO	2.9001×10^{1}	1.4677×10^{8}	2.1058×10^{8}
	IMSChoA	2.6094×10^{1}	3.8713×10^{6}	3.9511×10^{7}
	ChoA	2.8961×10^{1}	3.2706×10^{7}	3.2469×10^{6}
	MFO	9.0475×10^{4}	4.8562×10^{7}	1.0463×10^{8}
F6	PSO	1.2471×10^{0}	4.1923×10^{3}	1.3160×10^{4}
	GWO	7.5015×10^{0}	4.0337×10^{4}	4.3493×10^{4}
	IMSChoA	0.4743×10^{0}	9.2983×10^{2}	8.0821×10^{3}
	ChoA	4.0870×10^{0}	4.5433×10^{4}	4.1956×10^{4}
	MFO	7.7287×10^{0}	2.6025×10^{4}	3.1180×10^{4}
F7	PSO	3.2543×10^{0}	8.8942×10^{6}	5.3271×10^{7}
	GWO	1.6692×10^{0}	2.8689×10^{8}	4.6735×10^{8}
	IMSChoA	0.7521×10^{-1}	5.5112×10^{6}	5.1999×10^{7}
	ChoA	0.3327×10^{0}	8.0941×10^{8}	2.5548×10^{8}
	MFO	7.918×10^{0}	7.7442×10^{7}	1.7117×10^{8}
F8	PSO	1.2435×10^{1}	2.1457×10^{2}	3.2457×10^{3}
	GWO	4.5876×10^{-3}	2.1475×10^{0}	2.4785×10^{1}
	IMSChoA	8.7812×10^{-5}	7.7865×10^{-1}	1.5782×10^{1}
	ChoA	4.7852×10^{-4}	1.4231×10^{2}	2.4785×10^{3}
	MFO	7.2145×10^{0}	2.1452×10^{1}	7.8452×10^{3}
F9	PSO	4.2127×10^{0}	1.0856×10^{7}	5.8853×10^{7}
	GWO	1.6693×10^{0}	1.9264×10^{8}	2.9353×10^{8}
	IMSChoA	0.4157×10^{-1}	7.6690×10^{6}	7.0082×10^{7}
	ChoA	0.4032×10^{0}	3.1662×10^{8}	2.8858×10^{8}
	MFO	3.1148×10^{1}	1.8874×10^{8}	3.0155×10^{8}

Table 3. *Cont.*

Function Name	Algorithm	Optimum Value	Average Value	Standard Deviation
F10	PSO	2.3462×10^{2}	1.1810×10^{7}	6.7967×10^{7}
	GWO	1.6697×10^{0}	1.7130×10^{8}	2.5750×10^{8}
	IMSChoA	0.1330×10^{-1}	6.1348×10^{6}	6.4309×10^{7}
	ChoA	0.4086×10^{0}	6.1276×10^{8}	6.0678×10^{8}
	MFO	5.2633×10^{0}	5.8286×10^{7}	1.4566×10^{8}
F11	PSO	3.9789×10^{-1}	3.9946×10^{-1}	9.5817×10^{-3}
	GWO	5.1327×10^{0}	5.2427×10^{0}	2.1338×10^{-14}
	IMSChoA	3.9789×10^{-1}	4.1277×10^{-2}	1.4810×10^{-1}
	ChoA	3.9814×10^{-1}	4.3905×10^{-1}	1.8777×10^{-1}
	MFO	7.8961×10^{-1}	4.0922×10^{-1}	1.0551×10^{-1}
F12	PSO	9.5104×10^{-1}	4.9024×10^{3}	1.2576×10^{4}
	GWO	7.5020×10^{0}	4.0944×10^{4}	4.4157×10^{4}
	IMSChoA	4.9153×10^{-1}	9.6161×10^{2}	7.6567×10^{3}
	ChoA	3.7205×10^{0}	5.0133×10^{4}	4.4403×10^{4}
	MFO	1.0101×10^{4}	2.3015×10^{4}	2.2183×10^{4}
F13	PSO	3.9789×10^{-1}	4.1037×10^{-1}	1.1297×10^{-1}
	GWO	4.0002×10^{-1}	4.6941×10^{-1}	2.4898×10^{-1}
	IMSChoA	3.6787×10^{-1}	3.1971×10^{-1}	1.3684×10^{-1}
	ChoA	3.9810×10^{-1}	4.2046×10^{-1}	5.9657×10^{-1}
	MFO	3.9787×10^{-1}	5.0134×10^{-1}	9.5415×10^{-1}
F14	PSO	1.9920×10^{0}	2.2788×10^{0}	1.2463×10^{2}
	GWO	9.9954×10^{-1}	2.3876×10^{1}	9.0365×10^{1}
	IMSChoA	9.9800×10^{-1}	2.0056×10^{0}	1.4190×10^{1}
	ChoA	9.9961×10^{-1}	2.1719×10^{1}	9.3934×10^{1}
	MFO	3.9683×10^{0}	1.1003×10^{1}	5.5468×10^{1}
F15	PSO	9.1133×10^{-4}	6.6071×10^{-3}	1.0972×10^{-2}
	GWO	1.8780×10^{-3}	4.1942×10^{-2}	7.6213×10^{-1}
	IMSChoA	3.9751×10^{-4}	2.2051×10^{-3}	1.0951×10^{-2}
	ChoA	1.3171×10^{-3}	1.6130×10^{-3}	7.6465×10^{-4}
	MFO	1.4888×10^{-3}	1.6207×10^{-3}	9.3492×10^{-4}
F16	PSO	3.3912×10^{0}	1.9896×10^{7}	1.2035×10^{8}
	GWO	3.0040×10^{0}	6.8273×10^{8}	1.0925×10^{9}
	IMSChoA	4.0537×10^{-1}	1.5265×10^{7}	1.6062×10^{7}
	ChoA	2.8445×10^{0}	1.2423×10^{9}	1.2502×10^{9}
	MFO	3.4422×10^{1}	1.3698×10^{8}	3.1584×10^{8}
F17	PSO	3.0287×10^{0}	3.6754×10^{0}	1.4510×10^{0}
	GWO	3.0159×10^{0}	3.5667×10^{0}	3.8514×10^{-1}
	IMSChoA	3.0000×10^{0}	3.3619×10^{0}	9.2576×10^{-2}
	ChoA	3.1006×10^{0}	4.0996×10^{0}	4.3993×10^{0}
	MFO	3.0505×10^{0}	6.6287×10^{0}	3.8731×10^{1}
F18	PSO	-3.8538×10^{0}	-3.0897×10^{-0}	4.8218×10^{-1}
	GWO	-3.7439×10^{0}	-3.7220×10^{0}	1.4091×10^{-1}
	IMSChoA	-3.8628×10^{0}	-3.8573×10^{0}	3.4272×10^{-2}
	ChoA	-3.8549×10^{0}	-3.8158×10^{0}	2.0926×10^{-1}
	MFO	-3.7436×10^{0}	-3.5870×10^{0}	3.2694×10^{-2}
F19	PSO	-2.8067×10^{0}	-1.7476×10^{0}	8.7338×10^{-1}
	GWO	-2.0159×10^{-1}	-2.0159×10^{-1}	1.1669×10^{-1}
	IMSChoA	-3.3220×10^{0}	-3.2524×10^{0}	2.6905×10^{-2}
	ChoA	-3.0124×10^{0}	-2.8667×10^{0}	4.4520×10^{-1}
	MFO	-3.3220×10^{0}	-3.0003×10^{0}	3.7082×10^{-1}

Table 3. *Cont.*

Function Name	Algorithm	Optimum Value	Average Value	Standard Deviation
F20	PSO	-1.0138×10^{1}	-9.7699×10^{0}	1.2788×10^{-1}
	GWO	-2.7312×10^{-1}	-2.7312×10^{-1}	1.6670×10^{0}
	IMSChoA	-1.0152×10^{1}	-7.2606×10^{0}	2.7155×10^{-1}
	ChoA	-4.9481×10^{-1}	-4.2491×10^{0}	1.1362×10^{0}
	MFO	-5.0552×10^{0}	-4.8248×10^{0}	8.9123×10^{-1}
F21	PSO	-1.0518×10^{1}	-1.0113×10^{1}	1.3731×10^{0}
	GWO	-1.6697×10^{0}	-1.5444×10^{0}	2.8352×10^{-1}
	IMSChoA	-1.0536×10^{1}	-7.3240×10^{0}	2.6989×10^{-1}
	ChoA	-5.0690×10^{0}	-4.3478×10^{-1}	1.1266×10^{0}
	MFO	-5.1285×10^{0}	-4.9702×10^{0}	7.2547×10^{0}

4.4. Wilcoxon Rank Sum Test

In order to reflect the effectiveness of the improved algorithm, the literature suggests that a statistical test should be performed for the evaluation of the performance of the improved algorithm, and the effectiveness of the improved algorithm should be proved by the results of the statistical test. In this paper, the Wilcoxon rank sum test is used at the 5% significance level to determine whether the results of each iteration of IMSChoA are significantly different from PSO, GWO, ChoA, and MFO. The Wilcoxon rank sum test is a nonparametric statistical test that can detect more complex data distributions, and the general data analysis is only for the current data mean and standard deviation and does not compare with the data from multiple runs of the algorithm, so this data comparison analysis is not scientific. To demonstrate the superiority of the IMSChoA optimization algorithm, the results of the 12 runs of the test function were selected, and the results of the PSO, GWO, ChoA, and MFO algorithm runs were subjected to then Wilcoxon rank sum test, and the p-value was calculated, and, when $p < 5\%$, they can be considered as a strong verification of the rejection of the null hypothesis [33]. The results of the Wilcoxon rank sum test are shown in Table 4. The symbols "+", "−", and "=" indicate that IMSChoA outperforms, underperforms, and cannot make significant judgments of other algorithms, respectively. From the results in Table 4, the p-values of the Wilcoxon rank sum test for IMSChoA are basically less than 5%, indicating that, statistically speaking, IMSChoA has a significant advantage in the performance of the basic function search, which further reflects the robustness of IMSChoA.

Table 4. Wilcoxon rank-sum test results.

No.	PSO	GWO	ChoA	MFO
F1	3.04×10^{-20}	3.04×10^{-20}	3.04×10^{-20}	3.04×10^{-20}
F2	3.04×10^{-20}	3.04×10^{-20}	3.04×10^{-20}	3.04×10^{-20}
F3	3.04×10^{-20}	3.04×10^{-20}	3.04×10^{-20}	3.04×10^{-20}
F4	3.04×10^{-20}	3.04×10^{-20}	3.04×10^{-20}	3.04×10^{-20}
F5	1.29×10^{-17}	2.69×10^{-17}	2.33×10^{-10}	1.06×10^{-10}
F6	7.24×10^{-18}	7.11×10^{-18}	1.39×10^{-17}	1.43×10^{-10}
F7	4.28×10^{-17}	7.09×10^{-18}	1.38×10^{-17}	6.77×10^{-14}
F8	7.09×10^{-18}	7.09×10^{-18}	7.24×10^{-18}	2.24×10^{-10}
F9	3.45×10^{-20}	3.33×10^{-20}	1.23×10^{-19}	NaN
F10	3.45×10^{-20}	3.33×10^{-20}	2.98×10^{-20}	2.42×10^{-16}
F11	3.49×10^{-20}	3.31×10^{-20}	2.69×10^{-04}	3.65×10^{-14}
F12	7.07×10^{-20}	9.55×10^{-11}	8.01×10^{-10}	1.85×10^{-17}
+/=/−	12/0/0	12/0/0	12/0/0	11/0/1

5. Application Analysis of IMSChoA Algorithm Engineering Calculations

5.1. Spring Optimization Design Case Study

The optimization goal of the extension and compression spring design problem is to reduce the weight of the spring. The spring mechanism design is shown in Figure 11. The constraints of this design are shear stress, vibration frequency, and minimum vibration deflection. The variables y_1, y_2, and y_3 represent the coil diameter d, spring coil diameter D, and the number of coils N, respectively, and $f(x)$ is the minimum spring weight. The spring stretching mathematical model is described as follows.

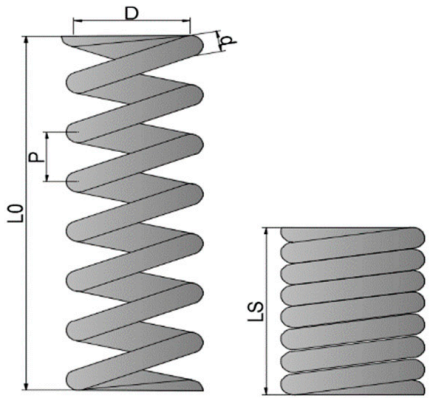


Figure 11. Schematic diagram of spring extension/compression structure.

Objective function:

$$\min f(x) = y_1^2 y_2 (2 + y_3) \tag{24}$$

Constraints:

$$\begin{cases} s_1(y) = 1 - \frac{y_2^3 y_3}{71785 y_1^4} \le 0 \\ s_2(y) = \frac{4y_2^2 - y_1 y_2}{12566(y_2 y_1^3 - y_1^4)} + \frac{1}{5108 y_1^2} - 1 \le 0 \\ s_3(y) = 1 - \frac{140.45 y_1}{y_2^3 y_3} \le 0 \\ s_4(y) = \frac{y_1 + y_2}{1.5} - 1 \le 0 \end{cases} \tag{25}$$

Among them: $0.05 \le y_1 \le 2$, $0.05 \le y_2 \le 2$, $0.05 \le y_3 \le 2$, $0.05 \le y_4 \le 2$.

The PSO algorithm, GWO algorithm, IMSChoA algorithm, ChoA algorithm, and MFO algorithm proposed in this paper were compared experimentally, where the data of the compared algorithms were obtained from the literature [24,34]. The experiments were selected with a population size of 50 and a maximum number of iterations of 500, and each algorithm was run 100 times independently to take the average value. The optimization results are shown in Table 5.

Table 5. The optimal solutions of each algorithm in the stretching/compression spring design problem.

Algorithm	Kp	Ki	Kd	Adaptability Value
PSO	0.5232	0.0603	0.5821	64.7664
GWO	0.5114	0.3415	0.0734	55.3211
IMSChoA	0.2615	0.0215	0.4115	43.1124
ChoA	0.4562	0.4754	0.4214	53.1451
MFO	0.6533	0.3147	0.4315	57.5521

As shown in Table 5, the IMSChoA algorithm obtains the optimal solution of the function $[y_1, y_2, y_3]$ = [0.0615, 0.7215, 5.5122] and the optimal solution $f(x)$ = 0.0124. IMSChoA has good optimization results for the extension/compression spring design problem, and the optimization results for the spring coil diameter, spring coil diameter, and spring

coil number are better than other algorithms. This shows that IMSChoA obtains the best solution for reducing the weight of the spring.

5.2. Optimization Experiments of the Fully Automatic Piston Manometer Control System

The ultimate goal of the optimization of the manometer control system is to check that the piston quickly and stably reaches the equilibrium position and achieves pressure measurement. The PID controller optimized by a group intelligence algorithm is generally used in engineering for regulation to achieve fast, stable, and accurate control. the PID control expression is shown in Equation (26), where k_p is a proportional coefficient; k_i is an integral coefficient; and k_d is a differential coefficient. The structure of the manometer control system is shown in Figure 12.

$$u(k) = k_p e(k) + k_i \sum_{n=0}^{k} e(k) + k_d[e(k) - e(k-1)] \tag{26}$$

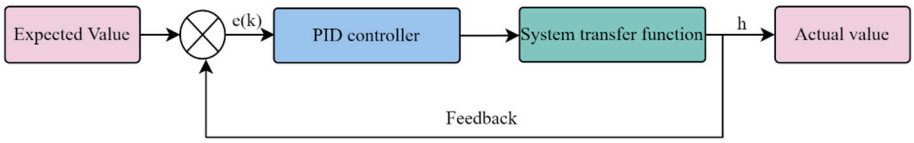


Figure 12. Structure of PID control system.

The experiment uses a 500 Mpa fully automatic piston manometer, as shown in Figure 13. The device uses STM32F429IGT6 as the control core, equipped with a series of control circuits. By controlling the pneumatic solenoid valve to control the weight configuration, the servo motor carries out pressure to make the piston float to the balance position to complete the pressure check. The range of piston movement is set from −2 mm to 2 mm, and 0 mm is the equilibrium position.

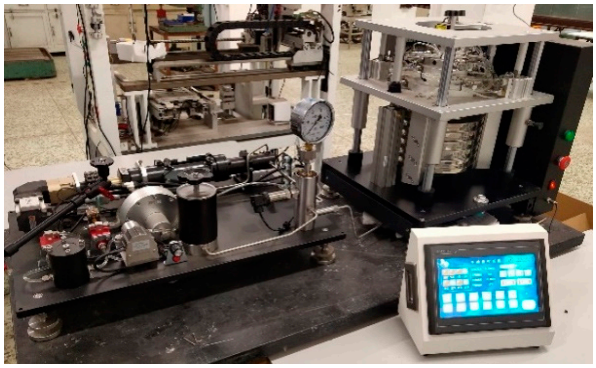


Figure 13. 500 Mpa automatic piston manometer.

The experiment uses a 500 Mpa fully automatic piston manometer, as shown in Figure 13. The device uses STM32F429IGT6 as the control core, equipped with a series of control circuits. By controlling the pneumatic solenoid valve to control the weight configuration, the servo motor carries out pressure to make the piston float to the balance position to complete the pressure check. The range of piston movement is set from −2 mm to 2 mm, and 0 mm is the equilibrium position.

The PSO algorithm, GWO algorithm, IMSChoA algorithm, ChoA algorithm, and MFO algorithm were used to adjust the parameters of the PID controller and to compare the results of the manometer operation, respectively. The initial conditions of each algorithm are the same. Each algorithm is run 50 times independently, and the average value is taken. The results of PID parameter adjustment were obtained in Table 6, and the displacement curve and velocity curve of the check piston movement are shown in Figures 14 and 15.

Table 6. Tuning parameters of PID controller optimized by different algorithms.

Algorithm	Kp	Ki	Kd	Adaptability Value
PSO	0.5232	0.0603	0.5821	64.7664
GWO	0.5114	0.3415	0.0734	55.3211
IMSChoA	0.2615	0.0215	0.4115	43.1124
ChoA	0.4562	0.4754	0.4214	53.1451
MFO	0.6533	0.3147	0.4315	57.5521

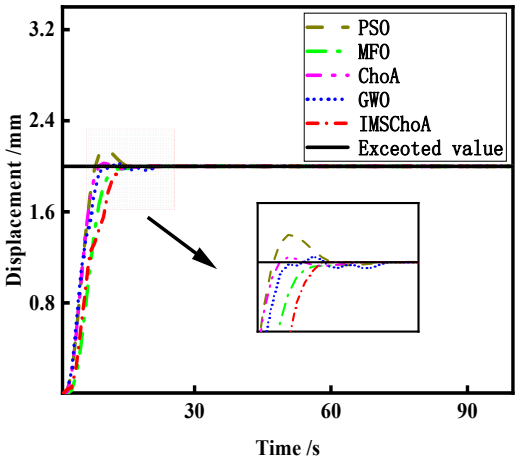


Figure 14. Displacement curve.

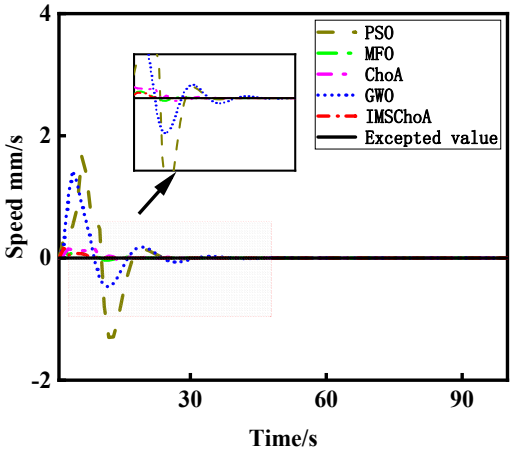


Figure 15. Speed curve.

As shown in Figures 14 and 15, the PID controller optimized by the IMSChoA algorithm has the best control effect on the manometer system, and the check piston can reach the balance position quickly and stably to complete the pressure detection. This further proves the feasibility of IMSChoA in practical engineering applications for the optimal design of mechanical structures.

6. Conclusions

In this paper, we propose an improved chimpanzee search algorithm with multi-strategy fusion, namely, IMSChoA, to address the problems of the ChoA optimization algorithm, such as low convergence accuracy and being prone to fall into local optimality. Firstly, we use improved sine chaotic mapping to initialize the population and solve the phenomenon of population boundary aggregation distribution. Secondly, the particle swarm algorithm idea was added, cooperating with the improved nonlinear convergence factor to balance the searchability of the algorithm, to accelerate the convergence of the algorithm, and to improve the convergence accuracy. Finally, the adaptive water wave factor

improved sparrow elite mutation, and the Bernoulli chaos mapping strategy was added to improve the ability of individuals to jump out of the local optimum. After 21 standard test functions for the optimization search test and analysis with the help of Wilcoxon rank sum statistical test results, the robustness and applicability of the algorithm were verified. Finally, the IMSChoA optimization algorithm was applied to the spring design case study and the optimization analysis of the fully automatic piston manometer control system, and the experimental results showed that the IMSChoA optimization algorithm also has good applicability to mechanical structure optimization design problems, but it has to be said that the comprehensive performance of the algorithm for low-dimensional, small-range high-precision search is still inadequate. Therefore, the next step will be to consider combining the IMSChoA algorithm with deep learning to eliminate the limitations of the algorithm in optimizing high-precision, as well as complex, problems, as well as to use it to solve more practical engineering problems.

Author Contributions: T.G. designed the project and coordinated the work. H.W. checked and discussed the results and the whole manuscript. F.Z. contributed to the discussion of this study. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Scientifific Research Fund Project of the Education Department of Liaoning Province, grant number No. LJKZ0510.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tharwat, A.; Elhoseny, M.; Hassanien, A.E.; Gabel, T.; Kumar, A. Intelligent Bézier curve-based path planning model using Chaotic Particle Swarm Optimization algorithm. *Cluster Comput.* **2019**, *22* (Suppl. S2), 4745–4766. [CrossRef]
2. Cinsdikici, M.G.; Aydın, D. Detection of blood vessels in ophthalmoscope images using MF/ant (matched filter/ant colony) algorithm. *Comput. Methods Programs Biomed.* **2009**, *96*, 85–95. [CrossRef] [PubMed]
3. Ghanamijaber, M. A hybrid fuzzy-PID controller based on gray wolf optimization algorithm in power system. *Evol. Syst.* **2019**, *10*, 273–284. [CrossRef]
4. Maharana, D.; Kotecha, P. Optimization of Job Shop Scheduling Problem with Grey Wolf Optimizer and JAYA Algorithm. In *Smart Innovations in Communication and Computational Sciences*; Springer: Singapore, 2019; pp. 47–58. [CrossRef]
5. Ebrahimi, B.; Rahmani, M.; Ghodsypour, S.H. A new simulation-based genetic algorithm to efficiency measure in IDEA with weight restrictions. *Measurement* **2017**, *108*, 26–33. [CrossRef]
6. Bu, S.J.; Kang, H.B.; Cho, S.B. Ensemble of Deep Convolutional Learning Classifier System Based on Genetic Algorithm for Database Intrusion Detection. *Electronics* **2022**, *11*, 745. [CrossRef]
7. Afzal, A.; Ramis, M.K. Multi-objective optimization of thermal performance in battery system using genetic and particle swarm algorithm combined with fuzzy logics. *J. Energy Storage* **2020**, *32*, 101815. [CrossRef]
8. Xin-gang, Z.; Ji, L.; Jin, M.; Ying, Z. An improved quantum particle swarm optimization algorithm for environmental economic dispatch. *Expert Syst. Appl.* **2020**, *152*, 113370. [CrossRef]
9. Sun, X.; Hu, C.; Lei, G.; Guo, Y.; Zhu, J. State Feedback Control for a PM Hub Motor Based on Gray Wolf Optimization Algorithm. *IEEE Trans. Power Electron.* **2020**, *35*, 1136–1146. [CrossRef]
10. Meidani, K.; Hemmasian, A.; Mirjalili, S.; Farimani, A.B. Adaptive grey wolf optimizer. *Neural Comput. Appl.* **2022**, *34*, 7711–7731. [CrossRef]
11. Meng, X.; Liu, Y.; Gao, X.; Zhang, H. A new bio-inspired algorithm: Chicken swarm optimization. In *International Conference in Swarm Intelligence*; Springer: Cham, Switzerland, 2014; pp. 86–94. [CrossRef]
12. Xue, J.; Shen, B. A novel swarm intelligence optimization approach: Sparrow search algorithm. *Syst. Sci. Control. Eng.* **2020**, *8*, 22–34. [CrossRef]
13. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
14. Lu, C.; Gao, L.; Yi, J. Grey wolf optimizer with cellular topological structure. *Expert Syst. Appl.* **2018**, *107*, 89–114. [CrossRef]
15. Teng, Z.; Lv, J.; Guo, L. An improved hybrid grey wolf optimization algorithm. *Soft Comput.* **2019**, *23*, 6617–6631. [CrossRef]
16. Hussien, A.G.; Hassanien, A.E.; Houssein, E.H.; Amin, M.; Azar, A.T. New binary whale optimization algorithm for discrete optimization problems. *Eng. Optim.* **2020**, *52*, 945–959. [CrossRef]
17. Wang, Z.; Qin, C.; Wan, B.; Song, W.W.; Yang, G. An Adaptive Fuzzy Chicken Swarm Optimization Algorithm. *Math. Probl. Eng.* **2021**, *2021*, 8896794. [CrossRef]
18. Tian, D.; Shi, Z. MPSO: Modified particle swarm optimization and its applications. *Swarm Evol. Comput.* **2018**, *41*, 49–68. [CrossRef]

19. Li, Y.; Zhu, X.; Liu, J. An improved moth-flame optimization algorithm for engineering problems. *Symmetry* **2020**, *12*, 1234. [CrossRef]
20. Khishe, M.; Mosavi, M.R. Chimp optimization algorithm. *Expert Syst. Appl.* **2020**, *149*, 113338. [CrossRef]
21. Du, N.; Zhou, Y.; Deng, W.; Luo, Q. Improved chimp optimization algorithm for three-dimensional path planning problem. *Multimed. Tools Appl.* **2022**, *81*, 27397–27422. [CrossRef]
22. Kumari, C.L.; Kamboj, V.K.; Bath, S.K.; Tripathi, S.L.; Khatri, M.; Sehgal, S. A boosted chimp optimizer for numerical and engineering design optimization challenges. *Eng. Comput.* **2022**, 1–52. [CrossRef]
23. Houssein, E.H.; Emam, M.M.; Ali, A.A. An efficient multilevel thresholding segmentation method for thermography breast cancer imaging based on improved chimp optimization algorithm. *Expert Syst. Appl.* **2021**, *185*, 115651. [CrossRef]
24. Liu, C.; He, Q. Golden sine chimpanzee optimization algorithm integrating multiple strategies. *J. Autom.* **2022**, *47*, 1–14.
25. Hekmatmanesh, A.; Wu, H.; Handroos, H. Largest Lyapunov Exponent Optimization for Control of a Bionic-Hand: A Brain Computer Interface Study. *Front. Rehabil. Sci.* **2022**, *2*, 802070. [CrossRef] [PubMed]
26. Li, Y.; Han, M.; Guo, Q. Modified Whale Optimization Algorithm Based on Tent Chaotic Mapping and Its Application in Structural Optimization. *KSCE J. Civ. Eng.* **2020**, *24*, 3703–3713. [CrossRef]
27. Xiong, X.; Wan, Z. The simulation of double inverted pendulum control based on particle swarm optimization LQR algorithm. In Proceedings of the 2010 IEEE International Conference on Software Engineering and Service Sciences, Beijing, China, 16–18 July 2009; IEEE: Piscataway, NJ, USA, 2010; pp. 253–256. [CrossRef]
28. Liu, X.; Bai, Y.; Yu, C.; Yang, H.; Gao, H.; Wang, J.; Chang, Q.; Wen, X. Multi-Strategy Improved Sparrow Search Algorithm and Application. *Math. Comput. Appl.* **2022**, *27*, 96. [CrossRef]
29. Liu, Z.; Li, M.; Pang, G.; Song, H.; Yu, Q.; Zhang, H. A Multi-Strategy Improved Arithmetic Optimization Algorithm. *Symmetry* **2022**, *14*, 1011. [CrossRef]
30. Wang, P.; Zhang, Y.; Yang, H. Research on economic optimization of microgrid cluster based on chaos sparrow search algorithm. *Comput. Intell. Neurosci.* **2021**, *2021*, 5556780. [CrossRef]
31. Mareli, M.; Twala, B. An adaptive Cuckoo search algorithm for optimisation. *Appl. Comput. Inform.* **2018**, *14*, 107–115. [CrossRef]
32. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]
33. Xinming, Z.; Xia, W.; Qiang, K. Improved grey wolf optimizer and its application to high-dimensional function and FCM optimization. *Control. Decis.* **2019**, *34*, 2073–2084.
34. He, Q.; Luo, S.H.H. Hybrid improvement strategy of chimpanzee optimization algorithm and its mechanical application. *Control. Decis. Mak.* **2022**, 1–11.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.