electronics

MDPI

Article

An Improved Nonlinear Tuna Swarm Optimization Algorithm Based on Circle Chaos Map and Levy Flight Operator

Wentao Wang and Jun Tian *

College of Software, Nankai University, Tianjin 300071, China
* Correspondence: jtian@nankai.edu.cn

Abstract: The tuna swarm optimization algorithm (TSO) is a new heuristic algorithm proposed by observing the foraging behavior of tuna populations. The advantages of TSO are a simple structure and fewer parameters. Although TSO converges faster than some classical meta-heuristics algorithms, it can still be further accelerated. When TSO solves complex and challenging problems, it often easily falls into local optima. To overcome the above issue, this article proposed an improved nonlinear tuna swarm optimization algorithm based on Circle chaos map and levy flight operator (CLTSO). In order to compare it with some advanced heuristic algorithms, the performance of CLTSO is tested with unimodal functions, multimodal functions, and some CEC2014 benchmark functions. The test results of these benchmark functions are statistically analyzed using Wilcoxon, Friedman test, and MAE analysis. The experimental results and statistical analysis results indicate that CLTSO is more competitive than other advanced algorithms. Finally, this paper uses CLTSO to optimize a BP neural network in the field of artificial intelligence. A CLTSO-BP neural network model is proposed. Three popular datasets from the UCI Machine Learning and Intelligent System Center are selected to test the classification performance of the new model. The comparison result indicates that the new model has higher classification accuracy than the original BP model.

Keywords: artificial intelligence; circle chaotic map; Levy flight; nonlinear adaptive weight; tuna swarm optimization

Citation: Wang, W.; Tian, J. An Improved Nonlinear Tuna Swarm Optimization Algorithm Based on Circle Chaos Map and Levy Flight Operator. *Electronics* **2022**, *11*, 3678. https://doi.org/10.3390/electronics11223678

Academic Editor: Javid Taheri

Received: 23 October 2022
Accepted: 7 November 2022
Published: 10 November 2022

1. Introduction

Nowadays, many engineering problems in real life have become more and more complex and challenging. High-quality solutions can help people effectively reduce resource investment. Because most production practice problems are multivariate, nonlinear, and have many complex constraints, the traditional branch and bound algorithm [1], conjugate gradient method [2], and dynamic programming method [3] cannot achieve remarkable results regarding these problems. The meta-heuristic algorithm has the characteristics of strong global search ability, no dependence on gradient information, and wide adaptability. It can effectively overcome the shortcomings of traditional optimization algorithms. Much of the research on meta-heuristic algorithms has shown that these algorithms are able to solve nonlinear optimization problems [4,5]. Many researchers tend to use meta-heuristic algorithms to solve complex engineering problems. Now, meta-heuristic algorithms are applied in various fields, such as workshop scheduling [6], task optimization [7], engineering management [8–10], and others.

The meta-heuristic algorithm is a mathematical method inspired by biological behavior and some physical phenomena in nature. These methods are used to solve complex problems in real life [11]. The meta-heuristic algorithm has the advantages of a simple structure, fewer hyperparameters, and being easy to understand. Based on these advantages, it has become an important method for solving optimization problems today. Meta-heuristic algorithms can be divided into four categories: swarm intelligence algorithms [12], evolutionary algorithms [13], human-based algorithms [14], and physical and chemical-based

algorithms [15]. The swarm intelligence algorithms simulate the behavior of animal populations. Each individual in the population is a candidate solution. They are randomly explored in the search space, which effectively avoids the possibility of entering the local optimum. Some classic and newly proposed swarm intelligence algorithms include Golden Jackal Optimization (GJO) [16], the Gray Wolf Optimization Algorithm (GWO) [17], and the Poplar Optimization Algorithm (POA) [18]. Some classical evolutionary algorithms include Genetic Algorithms [19] and the Biogeographic-Based Optimization Algorithm (BBO) [20], etc. Meta-heuristic algorithms can effectively enhance the efficiency of engineering practice. This has attracted more and more scholars' attention.

In many industrial problems, specific solution functions can be established with mathematical models. How to solve complex function optimization problems has become a focus of current research. For the optimization problems with fewer constraints and dimensions, the traditional mathematical methods can achieve outstanding results. Although meta-heuristic algorithms have very good performance in dealing with complex and high dimensional optimization problems, the convergence speed of simple meta-heuristic algorithms still needs to be improved. Sometimes with a single meta-heuristic algorithm, it is difficult to get rid of the attraction of local extremum. To further enhance the optimization capability of meta-heuristic algorithms, many experts try to use different strategies to improve them. Zhongzhou Du introduced Levy flight in the iterative process of PSO, which accelerated the optimization speed of PSO [21]. Hang Yu used a chaotic mapping strategy to improve the GWO initialization method, which improved the accuracy of the GWO solution [22]. Xiaoling Yuan introduced adaptive weight into the PSO algorithm, which greatly strengthened its global search capability [23]. So-Youn Park combined CS with oppositional learning, making the CS converge faster [24]. W. Xie used the golden sine operator to improve the Black Hole algorithm (BH) [25], giving it better exploration performance [26].

Xie et al. proposed a new meta-heuristic algorithm called the tuna swarm optimization algorithm (TSO) [27] in 2021 after observing the foraging behavior of tuna swarms. There are two common foraging strategies for tuna swarms: spiral foraging strategy and parabolic foraging strategy. TSO searches for the global optimal value by simulating the common individual in the tuna swarm to follow the optimal individual in the swarm to attack the prey. Comparing TSO with the Whale Optimization Algorithm (WOA) [28], the Salp Swarm Algorithm (SSA) [29], and some other advanced algorithms, the comparison results indicate that TSO outperforms the competitors. The tuna swarm optimization algorithm has the advantages of less parameters and easy realization. Therefore, after it was proposed, TSO has been widely studied and applied to engineering practice. Although TSO performs very well in many engineering practices, it still has some shortcomings. Firstly, TSO cannot efficiently search for the global optimal value. It is easily attracted by local extremum. Secondly, TSO does not converge fast enough. Finally, the followers of the optimal individual blindly follow the later. There is a lack of local exploitation. At present, Hu et al. have used Gaussian mutation to improve the TSO algorithm, and have applied the improved algorithm to photovoltaic power prediction [30]. Kumara et al. improved the TSO algorithm by using chaotic maps to increase the diversity of the algorithm population [31]. This paper proposes an improved tuna swarm optimization algorithm (CLTSO) based on the Circle chaotic map [32], Levy flight operator, and nonlinear adaptive operator. The innovations made in this article are summarized as follows:

(1) At the CLTSO initialization stage, this paper introduces the Circle chaotic map to uniformly generate individual positions. Because the initial positions of tuna individuals are randomly generated, the initial tuna individuals are likely to cluster together. In this paper, the emergence of the initial individual aggregation problem can be effectively solved by introducing the Circle chaotic map.
(2) In CLTSO, the optimal individual and its follower positions are updated by using Levy flight strategy. Because Levy flight uses a combination of long and short steps, it can significantly enlarge the search scope of CLTSO.

(3) In the iterative process of CLTSO, a nonlinear convergence factor is introduced to balance the exploration and the exploitation. In CLTSO, a large convergence factor in the initial iteration can bring the common individuals closer to the optimal individuals. A smaller convergence factor at the end of iteration increases the capability of followers to explore local scope.

This article covers the following aspects: Section 1 introduces some related content of the meta-heuristic algorithm and the tuna swarm optimization algorithm. Section 2 reviews the two foraging strategies of the original tuna swarm optimization algorithm. Section 3 introduces the improved Circle chaotic map strategy, the Levy flight operator, and the nonlinear adaptive weight operator, and the usage of these operators to improve TSO. Section 4 compares CLTSO with some classical and advanced meta-heuristics and makes some experimental analysis. Section 5 modifies the BP neural network based on CLTSO, and then tests the new model by using three popular datasets. Finally, Section 6 summarizes the content of the article.

The main mathematical symbols mentioned in this paper are shown in Table 1.

Table 1. Explanation of symbols.

Symbol	Meaning
X_i^{int}	Tuna individual in TSO
ub	The upper boundary of the search space of TSO
lb	The lower boundary of the search space of TSO
NP	Population size of TSO
τ	Distance parameter
α_1	Weight parameters of tuna following the best individual
α_2	Weight parameters of tuna following the front individual
p	Weight parameters in parabolic foraging strategy
s	The step length of Levy flight
t	Current number of iterations of the algorithm
T_{Max}	Maximum number of iterations of the algorithm
α_{1i}	Improved version of α_1
α_{2i}	Improved version of α_2
p_i	Improved version of p

2. An Overview of Tuna Optimization Algorithms

Tuna is the top predator in the ocean. Although tuna swim very fast, some small prey are more flexible than tuna. Therefore, in the process of predation, tuna often choose group cooperation to capture prey. The tuna swarm has two efficient predatory strategies, namely, the spiral foraging strategy and the parabolic foraging strategy. When the tuna swarm uses the parabolic foraging strategy, each tuna will follow the previous individual closely. The tuna swarm forms a parabola to surround the prey. When the tuna swarm adopts the spiral foraging strategy, the tuna swarm will aggregate into spiral shapes and drive prey to shallow water areas. Prey is more likely to be captured. By observing these two foraging behaviors of tuna swarm, researchers proposed a new swarm intelligence optimization called TSO.

2.1. Population Initialization

There are NP tunas in a tuna swarm. At the swarm initialization phase, the tuna swarm optimization algorithm randomly generates the initial swarm in the search space. The mathematical formulas for initializing tuna individuals are as follows:

$$\begin{aligned} X_i^{\text{int}} &= rand \cdot (ub - lb) + lb \\ &= \begin{bmatrix} x_i^1 & x_i^2 & \cdots & x_i^j \end{bmatrix} \\ &\begin{cases} i = 1, 2, \ldots, NP \\ j = 1, 2, \ldots, Dim \end{cases} \end{aligned} \tag{1}$$

where X_i^{int} is the i-th tuna, ub and lb are the upper and lower boundaries of the range of tuna exploration, and $rand$ is a random variable with uniform distribution from 0 to 1. In particular, each individual, X_i^{int}, in the tuna swarm represents a candidate solution for TSO. Each individual tuna consists of a set of Dim-dimensional numbers.

2.2. Parabolic Foraging Strategy

Herring and eel are the main food sources of tuna. When they encounter predators, they will use their speed advantage to constantly change their direction of swimming. It is very difficult for predators to catch them. Because tuna is less agile than their prey, the tuna swarm will take a cooperative approach to attack the prey. The tuna swarm will use the prey as a reference point to keep chasing prey. During predation, each tuna follows the previous individual, and the whole tuna swarm forms a parabola to surround the prey. In addition, the tuna swarm also uses a spiral foraging strategy. Assuming that the probability of the tuna swarm choosing either strategy is 50%, the mathematical model of parabolic foraging of the tuna swarm is as follows:

$$X_i^{t+1} = \begin{cases} X_{best}^t + rand \cdot (X_{best}^t - X_i^t) + TF \cdot p^2 \cdot (X_{best}^t - X_i^t), & \text{if rand } < 0.5 \\ TF \cdot p^2 \cdot X_i^t, & \text{if rand } \geq 0.5 \end{cases} \tag{2}$$

$$p = (1 - \frac{t}{t_{\max}})^{(t/t_{\max})} \tag{3}$$

where t indicates that the tth iteration is currently running and $t_{\max}$ means the maximum number of iterations preset. TF is a random value of 1 or -1.

2.3. Spiral Foraging Strategy

Besides the parabolic foraging strategy, there is another efficient cooperative foraging strategy called the spiral foraging strategy. While chasing the prey, most tuna cannot choose the right direction, but a small number of tuna can guide the swarm to swim in the right direction. When a small group of tuna start chasing the prey, the nearby tuna will follow this small group of individuals. Eventually, the entire tuna swarm will form a spiral formation to catch the prey. When the tuna swarm adopts a spiral foraging strategy, individuals will exchange information with the best to follow individuals or adjacent individuals in the swarm. Sometimes the best individual is not able to lead the swarm to capture prey effectively. The tuna will then select a random individual in the swarm to follow. The mathematical formula of the spiral foraging strategy is as follows:

$$X_i^{t+1} = \begin{cases} \alpha_1 \cdot (X_{rand}^t + \tau \cdot \left| X_{rand}^t - X_i^t \right| + \alpha_2 \cdot X_i^t), & \\ \qquad i = 1 & \\ \alpha_1 \cdot (X_{rand}^t + \tau \cdot \left| X_{rand}^t - X_i^t \right| + \alpha_2 \cdot X_{i-1}^t), & ,if \text{ rand} < \frac{t}{t_{\max}} \\ \qquad i = 2, 3, \ldots, NP & \\ \alpha_1 \cdot (X_{best}^t + \tau \cdot \left| X_{best}^t - X_i^t \right| + \alpha_2 \cdot X_i^t), & \\ \qquad i = 1 & ,if \text{ rand} \geq \frac{t}{t_{\max}} \\ \alpha_1 \cdot (X_{best}^t + \tau \cdot \left| X_{best}^t - X_i^t \right| + \alpha_2 \cdot X_{i-1}^t), & \\ \qquad i = 2, 3, \ldots, NP & \end{cases} \tag{4}$$

where X_i^{t+1} denotes the i-th tuna in the $t+1$ iteration. The current best individual is X_{best}^t. X_{rand}^t is the reference point randomly selected in the tuna swarm. α_1 is the trend weight coefficient to control the tuna individual swimming to the optimal individual or randomly selected adjacent individuals. α_2 is the trend weight coefficient to control the tuna individual swimming to the individual in front of it. τ is the distance parameter that

controls the distance between the tuna individual and the optimal individual or a randomly selected reference individual. Their mathematical calculation model is as follows:

$$\alpha_1 = a + (1 - a) \cdot \frac{t}{t_{\max}} \tag{5}$$

$$\alpha_2 = (1-a) - (1-a) \cdot \frac{t}{t_{\max}} \tag{6}$$

$$\tau = e^{bl} \cdot \cos(2\pi b) \tag{7}$$

$$l = e^{3\cos(((t_{\max}+1/t)-1)\pi)} \tag{8}$$

where a is a constant to measure the degree of tuna following and b is a random number uniformly distributed in the range of [0, 1].

2.4. Pseudocode of TSO

The pseudocode of the original TSO is displayed in Algorithm 1. The flow chart of TSO is displayed in Figure 1.

Algorithm 1 Pseudocode of TSO Algorithm

Initialization: Set parameters *NP*, *Dim*, a, z and T_{Max}
Initialize the position of tuna X_i (i = 1, 2, . . . , NP) by (1)
Counter $t = 0$
while $T < T_{\text{Max}}$ **do**
  Calculate the fitness value of all tuna
  Update the position and value of the best tuna X_{best}^t
  for (each tuna) **do**
    Update α_1, α_2, p by (5), (6), (3)
    if ($rand < z$) **then**
      Update X_i^{t+1} by (1)
    else if ($rand \geq z$) **then**
      if ($rand < 0.5$) **then**
        Update X_i^{t+1} by (4)
      else if ($rand \geq 0.5$) **then**
        Update X_i^{t+1} by (2)
t = t + 1
return the best fitness value f (X_{best}) and the best tuna X_{best}

In the iterative process of the TSO algorithm, each tuna will randomly choose to perform either the spiral foraging strategy or the parabolic foraging strategy. Tuna will also generate new individuals in the search range according to probability Z. Therefore, TSO will choose different strategies according to Z when generating new individual positions. During the execution of the TSO algorithm, all tuna individuals in the population are constantly updated until the number of iterations reaches a predetermined value. Finally, the TSO algorithm returns the optimal individual in the population and its optimal value.

The following advantages of TSO can be seen from Algorithm 1: (1) The TSO algorithm has fewer adjustable parameters, which is beneficial to the implementation of the algorithm. (2) This algorithm will save the position of the best tuna individual in each iteration; even if the quality of the candidate solution decreases, it will not affect the location of the optimal value. (3) The TSO algorithm can keep the balance between exploitation and exploration by selecting two foraging strategies.

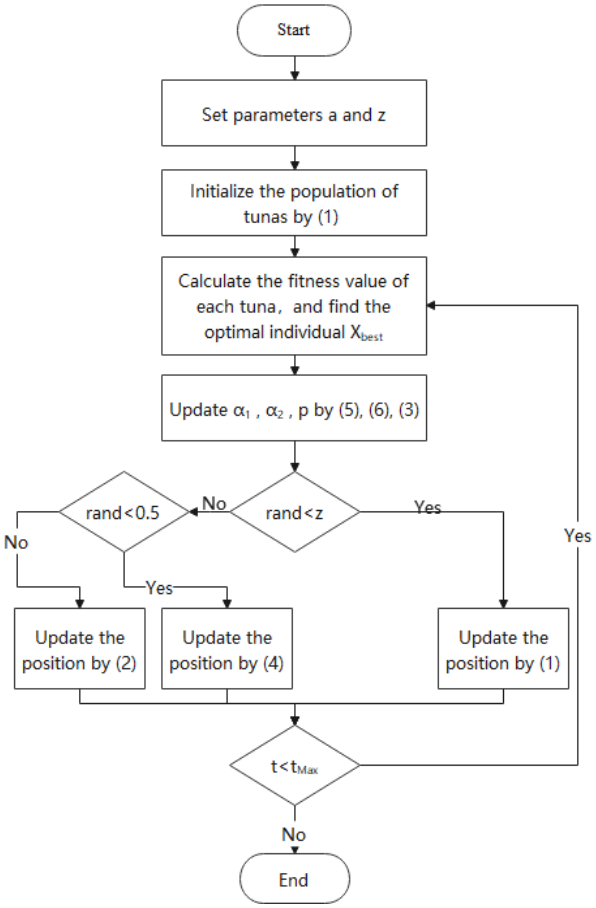

Figure 1. Flow chart of TSO.

3. The Improved Tuna Swarm Optimization Algorithm

This section introduces an improved nonlinear tuna swarm optimization algorithm, CLTSO, based on Circle chaotic map and Levy flight operator. Firstly, the population initialization using Circle chaotic map can increase the diversity of the swarm. The combination of TSO and Levy flight gives the algorithm an outstanding global exploration capability. Furthermore, a nonlinear adaptive weight operator is introduced to modify the weight coefficient of tuna following behavior in CLTSO. In CLTSO, the relationship between global exploration and local exploitation in the iterative process are well balanced.

3.1. Circle Chaotic Map

Many changes in nature are not random. They seem to conform to some special laws. Such a phenomenon is called chaos. Many movements in nature are chaotic [33]. Chaos is a random behavior, but it conforms to certain laws, which enables this operator to display more states in the search space of TSO [34].

Because the position of the tuna is randomly generated in the initialization phase of the tuna algorithm, it is easy to make the initial tuna gather at the same place. The initial tuna swarm does not fully cover the search space, resulting in a small difference between tuna individuals. This greatly reduces the global searching capability of the algorithm. The current popular chaotic mapping strategies are as follows: Tent [35], Logistic [36], Circle [37], Chebyshev [38], Sinusoidal [39], and Iterative chaotic map [40]. Studying the related literature on the above chaotic mapping strategies, we found that Circle chaotic map has a more stable chaotic value and has a higher coverage rate in the search space [41]. However, our experiments indicate that the distribution of Circle chaotic value is still not uniform. The chaotic values of the original Circle operator are clustered in the scope

of [0.2, 0.5]. To make the chaotic value distribution more uniform, we improved the mathematical model of the Circle chaotic mapping strategy.

The mathematical modeling of the original Circle chaotic map is as follows:

$$x_{i+1} = \mathrm{mod}(x_i + 0.2 - (0.5/2\pi)\sin(2\pi x_i), 1) \tag{9}$$

where x_i is the ith chaotic particle and x_{i+1} is the $(i + 1)$th chaotic particle. The scatter plot and frequency histogram of the initial candidate solution of the original Circle chaotic mapping operator are displayed in subgraphs (a) and (c) of Figure 2. In the Circle chaotic map experiment, the total number of particles is 2000. Chaotic particles denote the initial candidate solution of TSO.

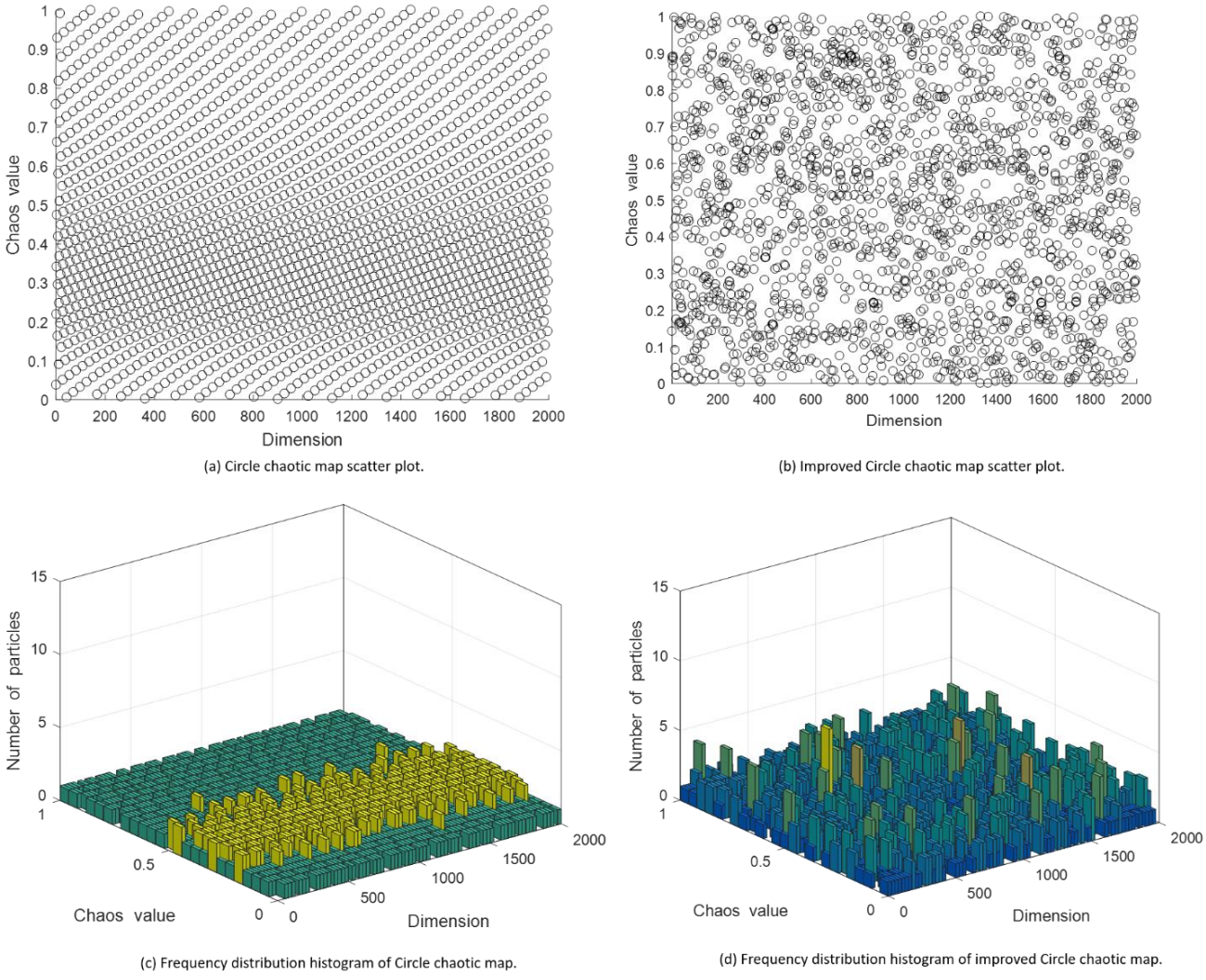

Figure 2. Frequency distribution histogram of improved Circle chaotic map.

As can be seen from subgraphs (a) and (c) of Figure 2, the chaotic particles are concentrated in the range of [0.2, 0.5] in the chaotic sequence initialized by Circle chaotic map. However, the initial candidate solutions are too concentrated, which will greatly reduce the population diversity of TSO. Therefore, the original Circle chaotic map is improved in this paper [42]. The mathematical modeling of the improved Circle chaotic map is as follows:

$$x_{i+1} = \mathrm{mod}(3.85x_i + 0.4 - (0.7/3.85\pi)\sin(3.85\pi x_i), 1) \tag{10}$$

where x_i is the ith chaotic particle and x_{i+1} is the $(i + 1)$th chaotic particle.

The scatter plot and frequency histogram of the initial candidate solution of the improved Circle chaotic map operator are displayed in subgraphs (b) and (d) of Figure 2.

From (b) and (d), we can clearly see that, compared to the original Circle chaotic map, the particle distribution of the improved Circle chaotic map is more uniform. Each candidate solution particle of the algorithm is explored in the search space. Therefore, using the improved Circle chaotic map operator to modify TSO can obtain more uniform candidate solutions. The initial tuna individuals uniformly distributed in the search space of the algorithm can significantly increase the population diversity of TSO.

3.2. Levy Flight

The movement and trajectory of many small animals and insects in life have the characteristics of Levy flight. These animals and insects include ants and flies. Many animals in nature use Levy flight strategy as an ideal way of foraging. By studying this phenomenon, French mathematician Paul Pierre Levy proposed the mathematical model of Levy flight [43]. Levy flight is an operator conforming to Levy distribution. The step size of Levy flight is random and mixed with long and short distances, which makes it easier to search over a large scale and with unknown scope compared to Brownian motion [44]. In the searching process, the Levy operator often uses short steps to walk and occasionally uses long steps to jump, which allows it to efficiently get rid of the effects of local attraction points. Therefore, in the random searching problem, many heuristic algorithms adopt this strategy to modify the iterative process, which efficiently helps the algorithm to get rid of the influence of local attraction points [45–47].

The Levy distribution can be expressed by the following mathematical model:

$$L(s) \sim |s|^{-1-\beta} \tag{11}$$

where β is in the range of (0, 2), s is the step size, and $L(s)$ is the probability density of a step size, s, according to Levy modeling. The mathematical modeling of Levy distribution is as follows:

$$L(s,\gamma,\mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \exp[-\frac{\gamma}{2(s-\mu)}] \frac{1}{(s-\mu)^{3/2}}, 0 < \mu < s < \infty \\ 0 \qquad\qquad\qquad\qquad\qquad ,otherwise \end{cases} \tag{12}$$

where μ represents the minimum step size and $\mu > 0$, γ represents size parameters. When $s \to \infty$, Equation (12) can be written in the following form:

$$L(s,\gamma,\mu) \approx \sqrt{\frac{\gamma}{2\pi}} \frac{1}{s^{3/2}} \tag{13}$$

Usually, scholars regard $L(s)$ approximation as the following mathematical formula:

$$L(s) \to \frac{\alpha\beta \cdot \Gamma(\beta) \sin(\pi\beta/2)}{\pi |s|^{1+\beta}}, s \to \infty \tag{14}$$

where Γ represents gamma function. Its mathematical model is as follows:

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \tag{15}$$

Due to the high complexity of Levy distribution, researchers often use the Mantegna [48] algorithm to simulate Levy flight step size, s, which is defined as follows:

$$s = \frac{\mu}{|\nu|^{1/\beta}} \tag{16}$$

where μ and v are defined as follows:

$$\mu \sim N\left(0, \sigma_\mu^2\right) \tag{17}$$

$$v \sim N\left(0, \sigma_v^2\right) \tag{18}$$

$$\sigma_\mu = \left\{ \frac{\Gamma(1+\beta)\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left[\frac{(1+\beta)}{2}\right] \cdot \beta \cdot 2^{\frac{(1+\beta)}{2}}} \right\}, \ \sigma_v = 1 \tag{19}$$

where the value of β is usually 1.5.

To show the global exploration capability of Levy flight more intuitively, this paper compares Levy flight with random walk strategy. The simulation steps of Levy flight and random walk are set to 300. The comparison results are presented in Figure 3.

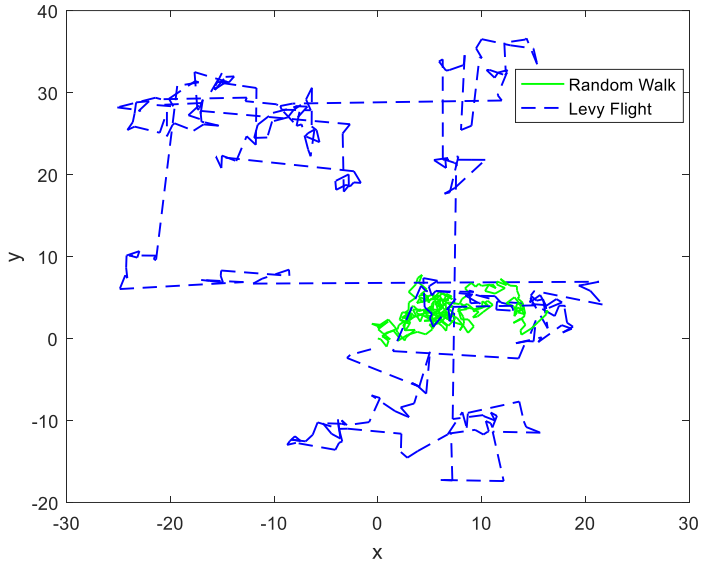

Figure 3. Simulation comparison experiment diagram of Levy flight and random walk.

Figure 3 shows that the Levy flight has a larger search range than random walk. The jump points of the random walk strategy are more concentrated, and the jump points of the Levy flight strategy are widely distributed. Figure 3 fully demonstrates the characteristics of Levy flight, which can make it better to explore in the whole searching space.

3.3. Nonlinear Adaptive Weight

How to balance the exploration capability and the exploration capability of the swarm intelligence optimization algorithm is very important. Weight parameters play an important role in the TSO algorithm. When the tuna chooses the spiral foraging strategy, in Equations (5) and (6), the weight parameters α_1 and α_2 determine the degree of how much tuna individuals follow the optimal individual to forage. This reflects the optimization process of the algorithm. Similarly, in the parabolic foraging strategy, the weight parameter p in Equation (2) determines the degree of how much ordinary individuals follow the optimal individual. When the weight parameter is large, the degree of tuna following the optimal individual is higher, which makes the whole tuna population better explore the whole space. When the weight parameter is small, ordinary tuna individuals do not follow the optimal individuals. They will swim around a small part of the space, which facilitates the ordinary tuna individual to develop the field around itself. To sum up, the exploration and the exploitation capabilities of TSO depend on the changes of weight parameters α_1, α_2, and p.

From Equations (5) and (6), it can be seen that the weight parameters α_1 and α_2 are linear changes. However, the optimization process of TSO is very complex, and the linear changes of weight parameters α_1 and α_2 cannot reflect the actual optimization process of the algorithm. Nowadays, in order to overcome the drawbacks caused by linear control weights, many scholars use nonlinear adaptive weights to improve the swarm intelligence optimization algorithms [49–51]. Repeated experiments indicate that the optimization effect of the nonlinear adaptive weight strategy is better than the linear weight strategy. Therefore, two improved nonlinear weight parameters α_{1i} and α_{2i} are introduced in this paper. Their mathematical models are as follows:

$$\alpha_{1i}(t) = \alpha_{1ini} - (\alpha_{1ini} - \alpha_{1fin}) \cdot \sin(\frac{t}{\mu \cdot T_{Max}} \cdot \pi) \tag{20}$$

$$\alpha_{2i}(t) = \alpha_{2ini} - (\alpha_{2ini} - \alpha_{2fin}) \cdot \sin(\frac{t}{\mu \cdot T_{Max}} \cdot \pi) \tag{21}$$

where $\mu = 2$, α_{1ini} denotes the initial value of α_1, α_{1fin} denotes the final value of α_1, α_{2ini} denotes the initial value of α_2, and α_{2fin} denotes the final value of α_2. We compared the improved weight parameters α_{1i} and α_{2i} with the original weight parameters α_1 and α_2. The results are displayed in Figure 4. In the experiment, $T_{\mathrm{Max}} = 500$.

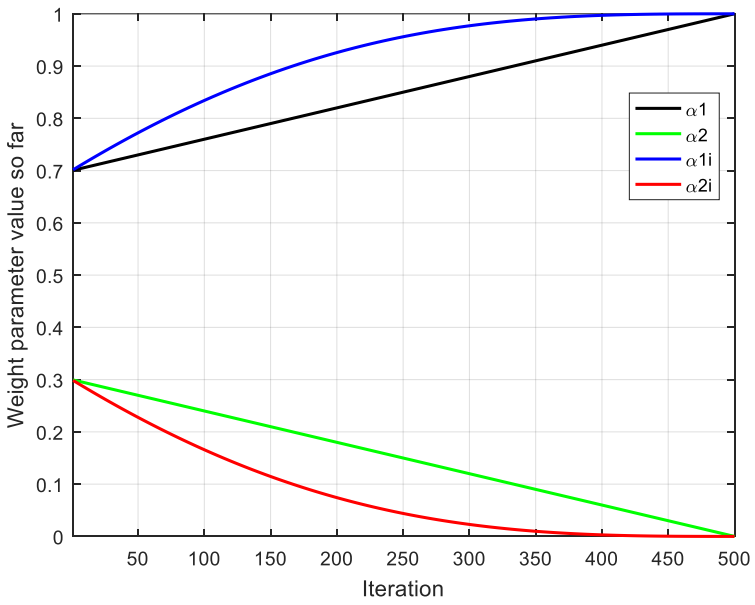

Figure 4. Comparison of weight coefficients α_1 and α_2 before and after improvement.

It can be clearly seen from Figure 4 that the improved weight parameters α_{1i} and α_{2i} change rapidly in the early stage, which makes ordinary tuna individuals more closely follow the optimal individual. It increases the global exploration capability of TSO. The weight parameters α_{1i} and α_{2i} change slowly in the late stage, which enables tuna individuals to explore their surrounding areas. It increases the local search capability of TSO.

In the spiral foraging strategy, a new nonlinear weight parameter p_i is proposed. Its mathematical model is as follows:

$$p_i(t) = p_{ini} - (p_{ini} - p_{fin}) \cdot \sin(\frac{t}{\mu \cdot T_{Max}} \cdot \pi) \tag{22}$$

where p_{ini} represents the initial value of p, and p_{fin} represents the final value of p. We compare the improved weight parameter ${p_i}^2$ with the original weight parameter p^2. The comparison curves are displayed in Figure 5. In the comparison curve, $T_{\mathrm{Max}} = 500$.

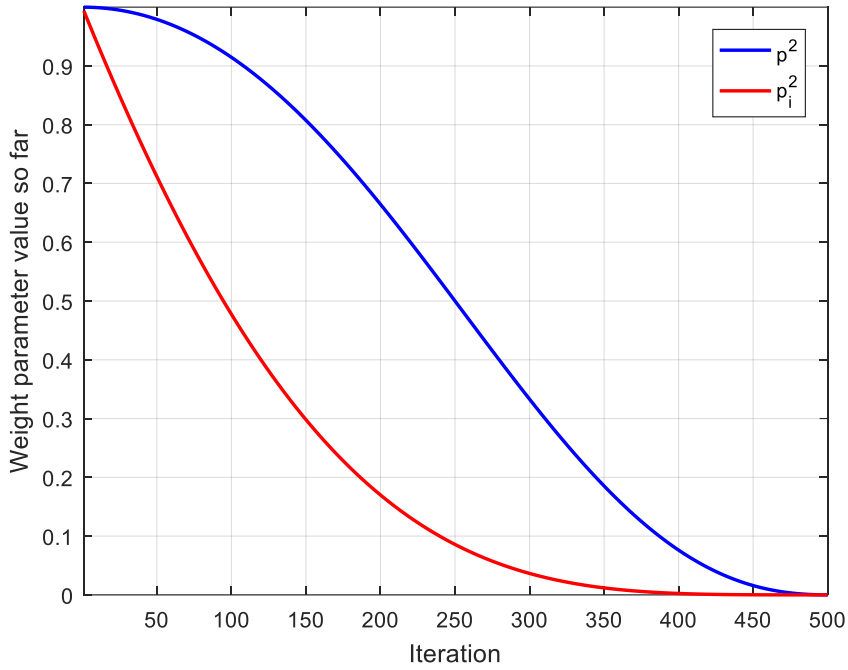

Figure 5. Comparison of the weight coefficient p before and after the improvement.

As can be seen from Figure 5, the improved weight parameter ${p_i}^2$ decreases rapidly in the early stage, so a tuna individual can follow its previous individual more closely. It increases the global exploration capability of TSO. The improved weight parameter ${p_i}^2$ decreases slowly in the late iteration, so tuna individuals can swim and explore in the surrounding space. It increases the local exploration capability of TSO.

3.4. Improved Nonlinear Tuna Swarm Optimization Algorithm Based on Circular Chaotic Map and Levy Flight Operator

The TSO algorithm usually uses random data to initialize population in solving function optimization problems, which may lead to the phenomenon that candidate solutions are clustered together. However, this phenomenon will lead to poor population diversity, which eventually leads to poor optimization results of the algorithm. Circle chaotic map has the advantages of randomness and ergodicity. In the optimization process of TSO, these advantages make it easier for the algorithm to escape the attraction of local extremum, and helps the algorithm to maintain the diversity of the swarm. Therefore, an improved Circle chaotic map strategy is introduced to initialize the tuna swarm. The swarm initialization mechanism is upgraded from Equations (1)–(10).

For the swarm intelligence optimization algorithm, how to get rid of the influence of local attraction points is a very important issue. The Levy flight strategy is an operator that can strengthen the global capability of TSO. This mechanism often uses short steps to walk and occasionally uses long steps to jump. The low-frequency use of long step length can ensure that TSO can extensively search the entire search area. The high-frequency use of short step length can ensure that TSO can locally search its nearest scope. Therefore, this paper introduces the Levy operator to modify the swarm update strategy of TSO. Considering that the jump of the Levy operator is too intense, and it may jump out of the main range in the process of operation, this paper adds step control parameters on the basis of the original Levy operator. The small step size control parameters can control the search of TSO in a small scope, which can enhance the local exploration ability of TSO without weakening the global exploration ability. The step size control parameters with large values can control the exploration of TSO in a large scope, which is conducive to solving the complex optimization problem.

The original TSO designed the parabolic foraging strategy and the spiral foraging strategy to balance the global exploration and the local exploitation capabilities of TSO. However, in the spiral foraging strategy, the linear changes of the weight parameters α_1 and α_2 cannot solve the actual complex problems well. In the parabolic foraging strategy, the change of the weight parameter p cannot effectively provide the solution to TSO for the global and the local exploration abilities. This paper uses nonlinear adaptive weight to modify the spiral foraging strategy and parabolic foraging strategy in TSO. The mathematical model of weight parameter p_i is upgraded from Equation (3) to Equation (22), and the mathematical models of α_{1i} and α_{2i} are upgraded from Equations (5) and (6) to Equations (20) and (21), respectively.

The mathematical model of the improved spiral foraging strategy based on the Levy operator and nonlinear adaptive weight strategy is as follows:

$$X_i^{t+1} = \begin{cases} \alpha_{1i} \cdot (X_{rand}^t + L\tau \cdot \left|X_{rand}^t - X_i^t\right| + \alpha_{2i} \cdot X_i^t), \\ \qquad i = 1 \\ \alpha_{1i} \cdot (X_{rand}^t + L\tau \cdot \left|X_{rand}^t - X_i^t\right| + \alpha_{2i} \cdot X_{i-1}^t), \quad ,if \text{ rand} < \frac{t}{t_{\max}} \\ \qquad i = 2, 3, \ldots, NP \\ \alpha_{1i} \cdot (X_{best}^t + L\tau \cdot \left|X_{best}^t - X_i^t\right| + \alpha_{2i} \cdot X_i^t), \\ \qquad i = 1 \qquad\qquad\qquad\qquad ,if \text{ rand} < \frac{t}{t_{\max}} \\ \alpha_{1i} \cdot (X_{best}^t + L\tau \cdot \left|X_{best}^t - X_i^t\right| + \alpha_{2i} \cdot X_{i-1}^t), \\ \qquad i = 2, 3, \ldots, NP \end{cases} \tag{23}$$

where $L\tau$ is an improved distance control parameter combined with the Levy operator. Its mathematical model is as follows:

$$L\tau = e^{\alpha \cdot Levy(s) \cdot l} \cdot \cos(2\pi \cdot Levy(s) \cdot \alpha) \tag{24}$$

where $Levy(s)$ is the step size of the Lévy operator, and α is the step size control coefficient. In this article, α = 0.01. The mathematical model of improved parabolic foraging strategy based on the Levy operator and nonlinear adaptive weight strategy is as follows:

$$X_i^{t+1} = \begin{cases} X_{best}^t + \alpha \cdot Levy(s) \cdot (X_{best}^t - X_i^t) + TF \cdot p^2 \cdot (X_{best}^t - X_i^t), & \text{if rand } < 0.5 \\ TF \cdot {p_i}^2 \cdot X_i^t, & \text{if rand } \geq 0.5 \end{cases} \tag{25}$$

Based on the above improvement strategies, an improved TSO is proposed, called CLTSO. The pseudocode of CLTSO is shown in Algorithm 2, and the process diagram of CLTSO is shown in Figure 6.

Algorithm 2 Pseudocode of CLTSO Algorithm

Initialization: Set parameters *NP*, *Dim*, *a*, *z* and T_{Max}. Initialize **the position of** tuna X_i *(i = 1, 2, . . . , NP)* by (10)
Counter t = 0
while $T < T_{\text{Max}}$ **do**
  Calculate the fitness value of all tuna
  Update the position and value of the best tuna X_{best}^t
  for (each tuna) **do**
    Update α_{1i}, α_{2i}, p_i by (20), (21), (22)
    if (*rand* < *z*) **then**
      Update X_i^{t+1} by (10)
    else if (*rand* $\geq$ *z*) **then**
      if (*rand* < *0.5*) **then**
        Update X_i^{t+1} by (23)
      else if (*rand* $\geq$ 0.5) **then**
        Update X_i^{t+1} by (25)
$t = t + 1$
return the best fitness value f (X_{best}) and the best tuna X_{best}

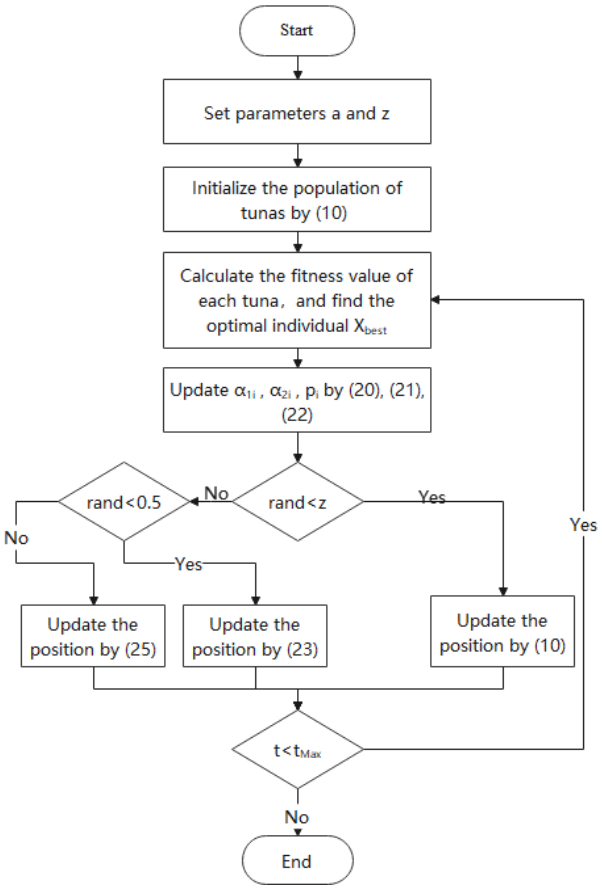

Figure 6. Flow chart of CLTSO.

Comparing Algorithms 1 and 2, it is clear that the overall structures are similar. The update strategies have been changed. Therefore, the improved operator proposed in this paper does not destroy the structural simplicity of the original TSO algorithm.

3.5. Time Complexity Analysis

Time complexity is an important measurement tool for evaluating the efficiency of an algorithm. In much of the research literature, it is represented by the symbol O. The time complexity is closely related to the number of instruction operations of the algorithm. The time complexity of TSO is closely related to iteration times, location update mechanism, and the evaluation times of fitness value function. The time complexity of CLTSO is closely related to the number of iterations, the number of fitness function evaluations, and the improvement operator. To compare the time cost differences between TSO and CLTSO, the time complexity of TSO and CLTSO is evaluated as follows. The time complexity of each operation instruction in the TSO is discussed below.

1. Initialize N individuals in the TSO, each with a dimension of D, so $N \cdot D$ calculations are required.
2. Calculate the fitness value of each individual in the tuna population and select the optimal individual in the current population. Therefore, it needs to calculate $[N \cdot (N-1)]/2$ times.
3. Update the values of parameters α_1, α_2, and p, which are computed 3 times.
4. Update all tuna individuals in the search space, which are computed $N \cdot D$ times.
5. Return the best individual, X_{best}, in the tuna population, which requires this code to be executed 1 time.

The instructions in steps 2 to 4 need to be iteratively run T_{Max} times. Combining the above analysis process, the time complexity of TSO can be expressed as $O(TSO) = T_{Max} \cdot [(N^2 - N)/2 + N \cdot D + 3]$.

The time complexity of each operation instruction in CLTSO is analyzed as follows.

1. Initialize N individuals in the CLTSO, each with a dimension of D, so $N \cdot D$ calculations are required.
2. Calculate the fitness value of each individual in the tuna population and select the optimal individual in the current population. Therefore, it needs to be calculated $[N \cdot (N-1)]/2$ times.
3. Update the values of parameters α_{1i}, α_{2i}, and p_i, which needs to be calculated 3 times.
4. Update all tuna individuals in the search space, which needs to be calculated $N \cdot D$ times.
5. When each individual in the tuna population is updated, the Levy operator needs to be calculated 1 time. Therefore, it needs to be run N times in total.
6. Return the best individual, X_{best}, in the tuna population, which requires this code to be executed 1 time.

Steps 2 to 5 require a total of T_{Max} iterations. Therefore, the time complexity of CLTSO can be expressed as $O(CLTSO) = T_{Max} \cdot [(N^2 - N)/2 + N \cdot D + 3 + N]$.

Compared with the tuna swarm optimization algorithm, the three operators proposed in this paper slightly increase the time cost. CLTSO and TSO have very close time complexity.

4. Simulation Experiments and Results Analysis

To verify the effectiveness of the proposed CLTSO in solving different optimization problems, in this section, 22 benchmark functions are applied to design a series of experiments to compare CLTSO with other famous meta-heuristic algorithms. In addition, to illustrate the outstanding performance of CLTSO, we compared it against the tuna swarm optimization algorithm (TSO), the improved TSO based on the Levy flight operator (LTSO), the improved TSO based on the Circle chaotic map, and nonlinear adaptive weights (CTSO). Finally, this section provides a detailed analysis of the experimental results.

4.1. Benchmark Function

Twenty-two different types of benchmark functions are selected to evaluate the capability of CLTSO, which cover unimodal, multimodal, fixed-dimension multimodal, and combined functions in the CEC2014 [52]. Through a survey of relevant literature, we find that CEC2014 is a classic test function, so it can be used as a benchmark to evaluate the performance of the proposed algorithm. Its mathematical model is given in Table 2. F_1~F_7 are unimodal functions, which are used to evaluate the convergence rate of the algorithm. F_8~F_{14} are multimodal functions, which are applied to verify whether the algorithm has good global exploration capability. F_{15}~F_{22} are the CEC2014 functions, which are applied to test the comprehensive capability of these algorithms.

4.2. Comparison Algorithm and Parameter Setting

Based on these 22 benchmark functions, a series of comparative experiments are designed to test the selected algorithms, which include Accelerated Particle Swarm Optimization (APSO) [53], WOA, the Fitness-Distance Balance based adaptive guided differential evolution (FDB-AGDE) algorithm [54], Covariance Matrix Adaptation Evolutionary Strategies (CMA-ES) [55], TSO, and CLTSO. The parameter values of the algorithms involved in these experiments are shown in Table 3. The symbol ' ~ ' indicates that the algorithm does not set parameter values. Functions F_1~F_{13} are tested in 30 and 100 dimensions, respectively, and F_{14} is tested in its suitable dimension. Eight CEC2014 benchmark functions are tested in 50 dimensions. The maximum number of evaluations of F_1~F_{14} are 1000. Because CEC benchmark functions are complex, the number of evaluations of 8 CEC2014 functions are simplified to 5000 without losing representativeness. The swarm size of each algorithm is

30. To avoid accidental interference, we run each algorithm 30 times independently in each experiment.

Table 2. Benchmark functions.

Function	Dim	Range	f_{min}
$F_1(x) = \sum_{i=1}^{D} x_i^2$	30,100	$[-100, 100]$	0
$F_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	30,100	$[-10, 10]$	0
$F_3(x) = \sum_{j=1}^{D} \left(\sum_{i=1}^{j} x_i \right)^2$	30,100	$[-100, 100]$	0
$F_4(x) = \max_i \{ \lvert x_i \rvert, 1 \le i \le D \}$	30,100	$[-100, 100]$	0
$F_5(x) = \sum_{i=1}^{D} 100(x_{i+1}^2 - x_i^2)^2 + (x_i - 1)^2$	30,100	$[-30, 30]$	0
$F_6(x) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	30,100	$[-100, 100]$	0
$F_7(x) = \sum_{i=1}^{D} i x_i^4 + \text{random}[0, 1)$	30,100	$[-1.28, 1.28]$	0
$F_8(x) = \sum_{i=1}^{D} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30,100	$[-500, 500]$	$-418.9829 \times D$
$F_9(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$	30,100	$[-5.12, 5.12]$	0
$F_{10}(x) = -20\exp\left(-0.2\sqrt{(1/D)\sum_{i=1}^{D} x_i^2}\right) - \exp\left((1/D)\sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + \exp(1)$	30,100	$[-32, 32]$	8.8818×10^{-16}
$F_{11}(x) = (1/4000)\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(x_i/\sqrt{i}\right) + 1$	30,100	$[-600, 600]$	0
$F_{12}(x) = \pi/D\left\{10\sin^2(\pi y_i) + \sum_{i=1}^{D} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)\right\}$ $+\sum_{i=1}^{D} u(x_i, 10, 100, 4)$ $y_i = 1 + [(x_i + 1)/4]$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a < x_i < a \\ k(-x_i - a)^m, x_i < a \end{cases}$	30,100	$[-50, 50]$	0
$F_{13}(x) = 0.1\left\{\sin^2(3\pi x_i) + \sum_{i=1}^{D} (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_D - 1)^2 [1 + \sin^2(2\pi x_D)]\right\}$ $+\sum_{i=1}^{D} u(x_i, 5, 100, 4)\Big\}$	30,100	$[-50, 50]$	0
$F_{14}(x) = ((1/500) + \sum_{j=1}^{25} (1/(j + \sum_{i=1}^{2} (x_i - a_{ij})^6)))^{-1}$	2	$[-65.53, 65.53]$	0.998004
$F_{15}(x)(CEC2014\ 1 : \text{Rotated High Conditioned Elliptic Function})$	50	$[-100, 100]$	100
$F_{16}(x)(CEC2014\ 2 : \text{Rotated Bent Cigar Function})$	50	$[-100, 100]$	200
$F_{17}(x)(CEC2014\ 3 : \text{Rotated Discus Function})$	50	$[-100, 100]$	300
$F_{18}(x)(CEC2014\ 5 : \text{Shifted and Rotated Rosenbrock})$	50	$[-100, 100]$	500
$F_{19}(x)(CEC2014\ 18 : \text{Shifted and Rotated Expanded Scaffer's F6 Function})$	50	$[-100, 100]$	1800
$F_{20}(x)(CEC2014\ 20 : \text{Hybrid Function 4 (N = 4)})$	50	$[-100, 100]$	2000
$F_{21}(x)(CEC2014\ 21 : \text{Hybrid Function 5 (N = 5)})$	50	$[-100, 100]$	2100
$F_{22}(x)(CEC2014\ 30 : \text{Composition Function 8 (N = 3)})$	50	$[-100, 100]$	3000

Table 3. Parameter values of the algorithms.

Algorithm	Parameter Value
APSO	$\alpha = 1,\ \beta = 0.5,\ \gamma = 0.95$
WOA	$l \in (-1, 1)$
FDB-AGDE	$\mu_{CR} = 0.5$
CMA-ES	$\mu = 2$
TSO	$a = 0.7,\ z = 0.05$
CLTSO	$a = 0.7,\ z = 0.05$
CTSO	$a = 0.7,\ z = 0.05$
LTSO	$a = 0.7,\ z = 0.05$

4.3. Results and Analysis

Table 4 shows the experimental results of CLTSO and other algorithms in low dimensional benchmark functions (dimension = 30), where Std is standard deviation and Mean is mean value. Mean represents the solution accuracy of these algorithms. Std reflects the stability of these algorithms in the solution process. F_{14} is tested in its own dimension. Table 5 displays the experimental results of CLTSO and other algorithms in high dimensional benchmark functions (dimension = 100). The experimental results of eight composite functions in CEC2014 are displayed in Table 6.

Table 4. Experimental results in 30 dimensions.

Function	Performance	APSO	WOA	FDB-AGDE	CMA-ES	TSO	CLTSO
F_1	Mean	5.09×10^{-39}	4.82×10^{-150}	1.23×10^{-8}	5.99×10^{-15}	0	0
	Std	1.05×10^{-39}	2.59×10^{-149}	1.34×10^{1}	3.94×10^{-15}	0	0
F_2	Mean	4.38×10^{-1}	8.02×10^{-103}	4.49×10^{-6}	1.83×10^{-7}	1.71×10^{-252}	0
	Std	5.79×10^{-1}	3.63×10^{-102}	5.61×10^{1}	4.53×10^{-8}	0	0
F_3	Mean	1.32×10^{1}	2.01×10^{4}	1.08×10^{-96}	6.14×10^{-6}	0	0
	Std	4.84×10^{0}	9.41×10^{3}	6.35×10^{4}	1.16×10^{-5}	0	0
F_4	Mean	4.96×10^{-1}	3.61×10^{1}	1.64×10^{0}	8.37×10^{-6}	6.42×10^{-249}	0
	Std	1.75×10^{-1}	2.69×10^{1}	1.45×10^{1}	2.29×10^{-6}	0	0
F_5	Mean	5.02×10^{1}	2.72×10^{1}	1.50×10^{2}	6.64×10^{1}	2.94×10^{-4}	2.12×10^{-4}
	Std	4.56×10^{1}	4.96×10^{-1}	1.14×10^{2}	1.52×10^{2}	7.65×10^{-1}	3.82×10^{-5}
F_6	Mean	4.01×10^{-32}	8.72×10^{-2}	1.09×10^{-8}	6.59×10^{-15}	1.37×10^{-9}	2.04×10^{-10}
	Std	1.60×10^{-32}	9.95×10^{-2}	1.33×10^{1}	3.44×10^{-15}	8.89×10^{-6}	2.40×10^{-10}
F_7	Mean	1.54×10^{-1}	1.53×10^{-3}	2.36×10^{-2}	2.44×10^{-2}	2.16×10^{-5}	1.81×10^{-5}
	Std	2.03×10^{-2}	2.05×10^{-3}	9.40×10^{0}	6.71×10^{-3}	2.19×10^{-4}	6.32×10^{-5}
F_8	Mean	-1.09×10^{2}	-1.16×10^{4}	-1.26×10^{4}	-4.41×10^{11}	-8.38×10^{2}	-1.26×10^{4}
	Std	3.25×10^{0}	1.50×10^{3}	3.32×10^{-1}	2.34×10^{12}	1.17×10^{4}	6.00×10^{-8}
F_9	Mean	7.42×10^{1}	0	3.11×10^{1}	5.60×10^{1}	0	0
	Std	5.96×10^{0}	0	1.62×10^{1}	6.32×10^{1}	0	0
F_{10}	Mean	5.36×10^{-1}	4.56×10^{-15}	6.78×10^{-7}	7.01×10^{-1}	8.88×10^{-16}	8.88×10^{-16}
	Std	5.28×10^{-1}	2.15×10^{-15}	3.73×10^{-1}	3.78×10^{0}	8.29×10^{-16}	0
F_{11}	Mean	8.49×10^{-3}	1.61×10^{-3}	2.85×10^{-7}	3.29×10^{-4}	0	0
	Std	1.67×10^{-2}	8.69×10^{-3}	1.64×10^{1}	1.77×10^{-3}	0	0
F_{12}	Mean	1.08×10^{-1}	6.17×10^{-3}	1.74×10^{-25}	2.01×10^{-15}	2.65×10^{-10}	6.75×10^{-14}
	Std	1.21×10^{-1}	6.67×10^{-3}	2.01×10^{1}	1.14×10^{-15}	1.14×10^{-7}	3.87×10^{-11}
F_{13}	Mean	2.38×10^{-3}	3.11×10^{-1}	1.87×10^{-19}	3.77×10^{-14}	5.12×10^{-8}	1.24×10^{-9}
	Std	4.42×10^{-3}	2.72×10^{-1}	1.54×10^{1}	3.15×10^{-14}	2.84×10^{-3}	3.77×10^{-9}
F_{14}	Mean	1.27×10^{1}	2.27×10^{0}	9.98×10^{-1}	7.65×10^{0}	9.98×10^{-1}	9.98×10^{-1}
	Std	1.12×10^{-13}	2.91×10^{0}	2.71×10^{0}	3.59×10^{0}	9.31×10^{-1}	2.69×10^{-16}

Table 5. Experimental results in 100 dimensions.

Function	Performance	APSO	WOA	FDB-AGDE	CMA-ES	TSO	CLTSO
F_1	Mean	1.84×10^{1}	1.03×10^{-149}	7.27×10^{1}	1.83×10^{-3}	0	0
	Std	1.40×10^{0}	4.01×10^{-149}	1.98×10^{1}	4.15×10^{-4}	0	0
F_2	Mean	4.13×10^{1}	6.59×10^{-102}	7.95×10^{0}	2.99×10^{-1}	8.66×10^{-235}	0
	Std	2.74×10^{0}	2.42×10^{-101}	7.89×10^{0}	1.09×10^{-1}	0	0
F_3	Mean	2.23×10^{2}	8.92×10^{5}	7.05×10^{-86}	4.80×10^{5}	0	0
	Std	1.40×10^{1}	2.09×10^{5}	1.05×10^{1}	1.31×10^{5}	0	0
F_4	Mean	2.29×10^{0}	7.06×10^{1}	5.91×10^{1}	1.73×10^{0}	5.52×10^{-229}	0
	Std	8.35×10^{-2}	2.77×10^{1}	6.11×10^{0}	3.03×10^{-1}	0	0
F_5	Mean	6.22×10^{3}	9.77×10^{1}	1.30×10^{5}	3.59×10^{2}	1.08×10^{-1}	1.89×10^{-3}
	Std	2.28×10^{3}	4.05×10^{-1}	9.66×10^{5}	1.49×10^{3}	1.99×10^{-1}	4.41×10^{-3}
F_6	Mean	2.66×10^{1}	1.76×10^{0}	5.27×10^{-5}	1.71×10^{-3}	5.10×10^{-5}	4.65×10^{-5}
	Std	7.14×10^{0}	6.30×10^{-1}	1.37×10^{1}	3.09×10^{-4}	2.37×10^{-2}	7.68×10^{-5}
F_7	Mean	1.83×10^{3}	1.67×10^{-3}	2.81×10^{-1}	1.39×10^{-1}	2.76×10^{-4}	1.04×10^{-4}
	Std	6.20×10^{2}	1.19×10^{-3}	1.37×10^{1}	1.83×10^{-2}	3.08×10^{-4}	1.13×10^{-4}
F_8	Mean	-2.35×10^{2}	-3.73×10^{4}	-3.45×10^{4}	-1.81×10^{5}	-2.79×10^{3}	-4.19×10^{4}
	Std	9.35×10^{0}	5.59×10^{3}	4.00×10^{3}	3.12×10^{4}	3.91×10^{4}	2.95×10^{-3}
F_9	Mean	4.33×10^{2}	0	2.25×10^{2}	6.69×10^{2}	0	0
	Std	2.60×10^{1}	0	1.18×10^{2}	1.64×10^{2}	0	0
F_{10}	Mean	3.58×10^{0}	4.20×10^{-15}	5.86×10^{0}	9.41×10^{-3}	8.88×10^{-16}	8.88×10^{-16}
	Std	3.04×10^{-1}	2.23×10^{-15}	4.11×10^{0}	1.04×10^{1}	8.29×10^{-16}	0
F_{11}	Mean	4.39×10^{-1}	0	1.71×10^{0}	3.49×10^{-2}	0	0
	Std	6.34×10^{-2}	0	1.12×10^{1}	6.78×10^{-3}	0	0
F_{12}	Mean	5.46×10^{-1}	1.80×10^{-2}	6.02×10^{-3}	2.12×10^{-4}	2.49×10^{-8}	2.78×10^{-9}
	Std	1.09×10^{-1}	7.22×10^{-3}	9.60×10^{0}	5.70×10^{-5}	5.86×10^{-5}	2.19×10^{-7}
F_{13}	Mean	8.73×10^{0}	1.65×10^{0}	5.93×10^{-1}	3.76×10^{-3}	2.02×10^{-4}	6.80×10^{-6}
	Std	2.13×10^{0}	7.21×10^{-1}	1.53×10^{1}	2.83×10^{-3}	4.38×10^{-3}	7.14×10^{-6}

Table 6. Simulation results of CEC2014 functions.

Function	Performance	APSO	WOA	FDB-AGDE	CMA-ES	TSO	CLTSO
F_{15}	Mean	1.19×10^{10}	8.85×10^{8}	3.36×10^{5}	1.84×10^{7}	2.22×10^{6}	4.35×10^{5}
	Std	3.12×10^{7}	3.34×10^{8}	7.01×10^{7}	3.94×10^{6}	1.21×10^{6}	4.35×10^{5}
F_{16}	Mean	1.65×10^{11}	7.71×10^{10}	3.36×10^{2}	2.02×10^{4}	1.00×10^{4}	2.75×10^{2}
	Std	3.25×10^{8}	7.86×10^{9}	1.20×10^{1}	5.08×10^{4}	4.94×10^{9}	1.15×10^{4}
F_{17}	Mean	1.99×10^{8}	9.77×10^{4}	7.36×10^{2}	8.33×10^{5}	8.38×10^{3}	3.59×10^{2}
	Std	3.10×10^{3}	9.92×10^{3}	1.09×10^{1}	1.18×10^{5}	7.60×10^{3}	3.29×10^{1}
F_{18}	Mean	5.20×10^{2}	5.21×10^{2}	5.21×10^{2}	5.21×10^{2}	5.21×10^{2}	5.20×10^{2}
	Std	9.69×10^{2}	8.62×10^{-2}	1.14×10^{1}	4.43×10^{-2}	3.13×10^{2}	1.06×10^{-1}
F_{19}	Mean	1.39×10^{9}	4.83×10^{5}	2.96×10^{3}	5.07×10^{4}	2.27×10^{3}	1.98×10^{3}
	Std	1.98×10^{4}	4.22×10^{5}	2.00×10^{3}	2.89×10^{4}	2.91×10^{3}	1.56×10^{3}
F_{20}	Mean	3.17×10^{3}	3.04×10^{5}	3.13×10^{3}	8.77×10^{5}	4.66×10^{3}	2.67×10^{3}
	Std	5.98×10^{2}	2.31×10^{5}	1.56×10^{1}	3.70×10^{5}	4.86×10^{3}	2.25×10^{2}
F_{21}	Mean	9.39×10^{8}	1.12×10^{7}	3.56×10^{4}	5.20×10^{6}	4.10×10^{4}	6.27×10^{3}
	Std	1.14×10^{6}	5.43×10^{6}	1.05×10^{5}	2.42×10^{6}	2.33×10^{5}	6.06×10^{4}
F_{22}	Mean	3.20×10^{3}	3.79×10^{5}	1.26×10^{4}	4.00×10^{3}	3.20×10^{3}	3.20×10^{3}
	Std	7.83×10^{-4}	2.41×10^{5}	9.53×10^{0}	2.93×10^{2}	1.92×10^{3}	0

As can be seen from Table 3, in low-dimensional functions, the optimization accuracy of CLTSO is only slightly weaker than its competitors in F_6, F_8, F_{12}, and F_{13}. Among the remaining 10 benchmark functions, CLTSO not only has significantly better solution accuracy than its competitors, but also has better robustness. This shows that the Circle chaotic map operator can help CLTSO obtain more diverse candidate solutions, and each candidate solution can continuously update and finally select the optimal solution during the iteration.

When the dimension of the benchmark function is 100, CLTSO has better optimization performance in dealing with higher dimensional and more complex problems. Only in the F_8 test function is the optimization accuracy of CLTSO slightly worse than that of CMA-ES. In the remaining 12 functions, CLTSO has the best optimization accuracy, and CLTSO can find the theoretical optimal value in F_1, F_2, F_3, F_4, F_9, F_{10}, and F_{11}. From the robustness of the algorithm, CLTSO obtains the minimum Std value in all benchmark functions, which indicates that CLTSO has more stable exploration ability than other competitors. This is due to the fact that the Circle chaotic map strategy helps CLTSO to obtain a richer population diversity, which allows the initial tuna to be evenly distributed in the search space. In addition, during the execution of CLTSO, the Levy flight operator strengthens the exploration capability of the algorithm, and the nonlinear adaptive weight operator can well balance the exploration and exploitation capability of CLTSO.

The experimental results of the CEC2014 function indicate that all algorithms do not obtain the theoretical optimal value, but CLTSO can still achieve more excellent optimization accuracy than other competitors in F_{16}~F_{22}. This effectively proves that the improved nonlinear tuna swarm optimization algorithm based on the Circle chaotic map strategy and the Levy flight operator can adapt to more complex and challenging optimization problems.

To more intuitively observe the convergence ability of CLTSO and the competitors, Figure 7 draw their operating curves. The images of F_1~F_{13} are drawn in 100 dimensions, the image of F_{14} is drawn in its suitable dimension, and the images of F_{15}~F_{22} are drawn in 50 dimensions.

The convergence curves of these algorithms indicate that CLTSO has a better convergence performance than the competitors. For simple optimization problems, CLTSO can obtain theoretical optimal values within 500~600 iterations. For complex and challenging problems, CLTSO can also maintain a faster convergence rate and get rid of the influence of local attraction points, and ultimately achieve higher optimization accuracy.

In order to further show whether CLTSO has obvious advantage over other algorithms, this paper uses the Wilcoxon [56] statistical method and the Friedman method to analyze the experimental results of these algorithms in 100-dimensional benchmark functions. The results of F_{14} is based on its suitable dimensions. The experimental data of eight CEC2014 benchmark functions are measured in 50 dimensions. The results of the Friedman test and the p-value of the Wilcoxon test are listed in Tables 7 and 8, respectively.

Table 7. Results of Friedman test.

Algorithm	Rank Mean
CLTSO	1.39
TSO	2.48
FDB-AGDE	3.68
CMA-ES	4.09
WOA	4.14
APSO	5.23

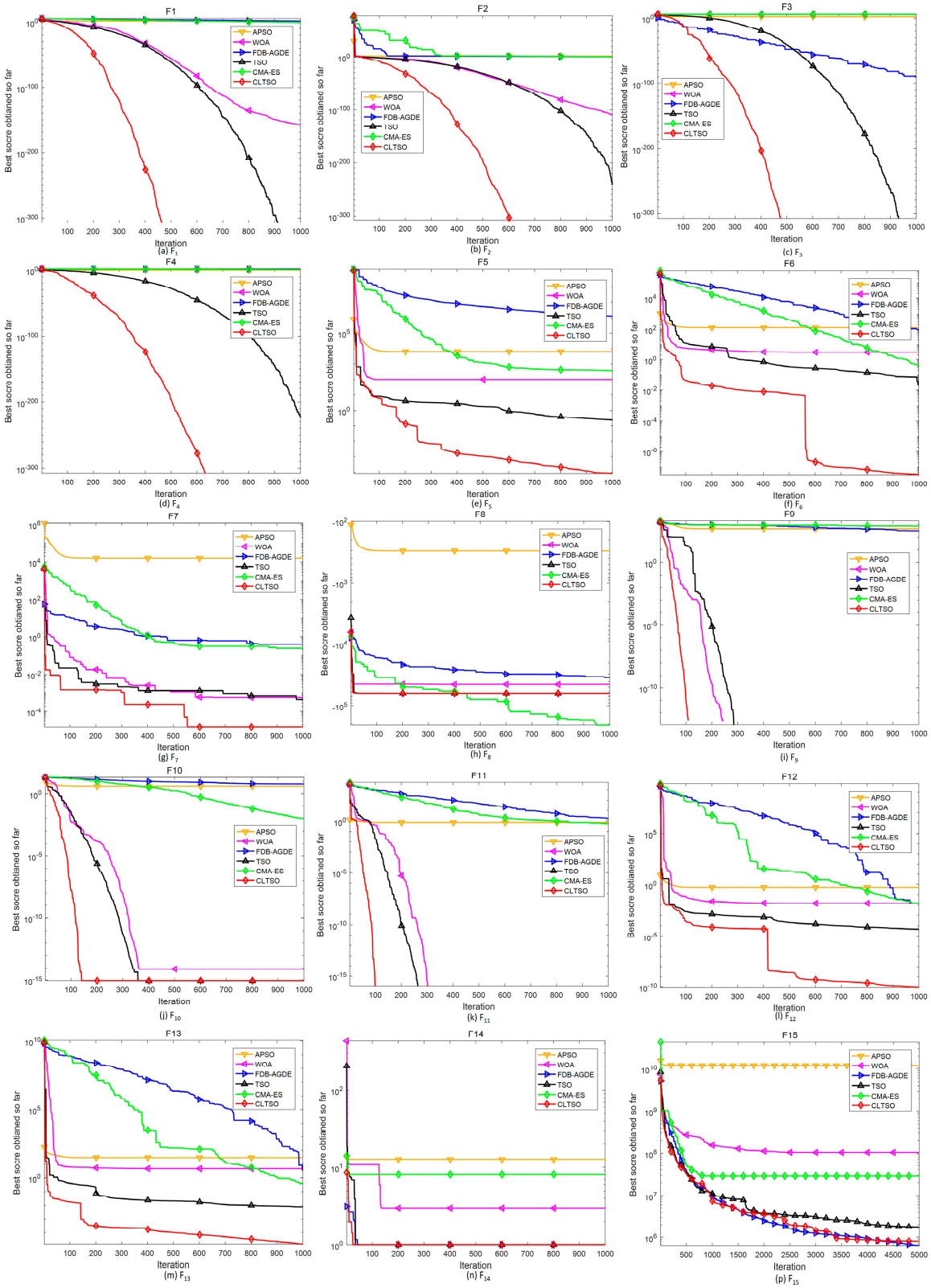

Figure 7. *Cont.*

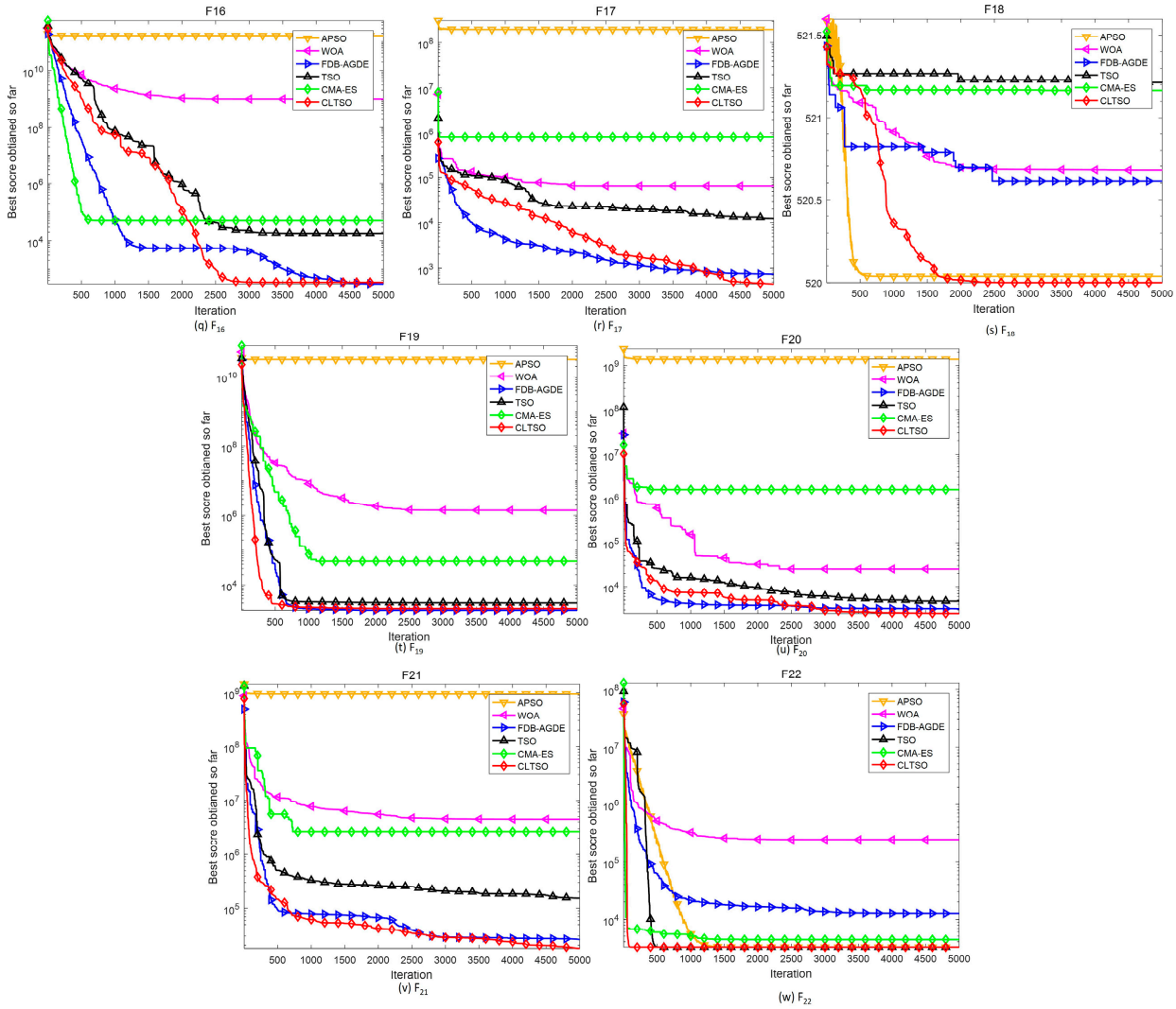

Figure 7. Convergence curve of each algorithm (F1~F22).

The Friedman test is a nonparametric statistical analysis method, which uses rank mean to test whether there are significant differences in multiple population distributions. Because the problem in this paper is to find the minimum value, a smaller rank mean value in the Friedman test results indicates better performance of the algorithm. As can be seen from Table 6, CLTSO has the smallest rank mean and TSO ranks the second, followed by CMA-ES, WOA, DE, and APSO.

In the Wilcoxon statistical test results, if the *p*-value is less than 0.05 and close to 0, this indicates that the experimental results of the two algorithms are significantly different. If the *p*-value exceeds 0.05, this indicates that the experimental results of the two algorithms are not significantly different. If the *p*-value is equal to NaN, this means that the experimental results of the two algorithms are not different. As can be seen from Table 7, except for the last column, the *p*-values of CLTSO are basically less than 0.05 and close to 0, which indicates that CLTSO has significant advantages compared with other algorithms. It is not difficult to find that half of the *p*-values for Wilcoxon analysis of CLTSO vs. TSO are greater than 0.05. This is because both CLTSO and TSO can find the theoretical optimal value in these functions, or the optimal value found by TSO is not much different from that found by CLTSO. From the optimization curves of TSO and CLTSO, we can see that although the calculation results of these two algorithms are not very different in those functions with *p*-values greater than 0.05, the speed of CLTSO is generally much faster than that of TSO.

Table 8. Results of Wilcoxon test.

Function	CLTSO vs. WOA	CLTSO vs. APSO	CLTSO vs. FDB-AGDE	CLTSO vs. CMAES	CLTSO vs. TSO
F_1	1.21×10^{-12}	1.21×10^{-12}	7.94×10^{-3}	1.21×10^{-12}	NaN
F_2	1.21×10^{-12}	1.21×10^{-12}	7.94×10^{-3}	1.21×10^{-12}	1.21×10^{-12}
F_3	1.21×10^{-12}	1.21×10^{-12}	7.94×10^{-3}	1.21×10^{-12}	NaN
F_4	1.21×10^{-12}	1.21×10^{-12}	7.94×10^{-3}	1.21×10^{-12}	1.21×10^{-12}
F_5	3.02×10^{-11}	3.02×10^{-11}	7.94×10^{-3}	3.02×10^{-11}	4.18×10^{-9}
F_6	3.02×10^{-11}	3.02×10^{-11}	7.94×10^{-3}	3.02×10^{-11}	2.78×10^{-7}
F_7	3.82×10^{-10}	3.02×10^{-11}	7.94×10^{-3}	3.02×10^{-11}	6.20×10^{-4}
F_8	3.02×10^{-11}	3.02×10^{-11}	7.94×10^{-3}	3.02×10^{-11}	3.65×10^{-8}
F_9	NaN	1.21×10^{-12}	7.94×10^{-3}	1.21×10^{-12}	NaN
F_{10}	3.06×10^{-9}	1.21×10^{-12}	7.94×10^{-3}	1.21×10^{-12}	NaN
F_{11}	NaN	1.21×10^{-12}	7.94×10^{-3}	1.21×10^{-12}	NaN
F_{12}	3.02×10^{-11}	3.02×10^{-11}	7.94×10^{-3}	3.02×10^{-11}	6.53×10^{-8}
F_{13}	3.02×10^{-11}	3.02×10^{-11}	7.94×10^{-3}	3.02×10^{-11}	1.69×10^{-9}
F_{14}	1.57×10^{-11}	1.39×10^{-4}	NaN	1.57×10^{-11}	1.22×10^{-1}
F_{15}	7.94×10^{-3}	7.94×10^{-3}	1.59×10^{-2}	7.94×10^{-3}	1.51×10^{-1}
F_{16}	7.94×10^{-3}	7.94×10^{-3}	8.41×10^{-1}	4.21×10^{-1}	6.90×10^{-1}
F_{17}	7.94×10^{-3}	7.94×10^{-3}	7.94×10^{-3}	7.94×10^{-3}	7.94×10^{-3}
F_{18}	7.94×10^{-3}	7.94×10^{-3}	7.94×10^{-3}	7.94×10^{-3}	7.94×10^{-3}
F_{19}	7.94×10^{-3}	7.94×10^{-3}	8.41×10^{-1}	7.94×10^{-3}	8.41×10^{-1}
F_{20}	7.94×10^{-3}	7.94×10^{-3}	7.94×10^{-3}	7.94×10^{-3}	7.94×10^{-3}
F_{21}	7.94×10^{-3}	7.94×10^{-3}	4.21×10^{-1}	7.94×10^{-3}	1.51×10^{-1}
F_{22}	7.94×10^{-3}	7.94×10^{-3}	7.94×10^{-3}	7.94×10^{-3}	1.00×10^{0}

Finally, this paper quantitatively analyzes all the algorithms in the experiment. The quantitative analysis of these algorithms is based on the mean absolute error (MAE) of 22 benchmark functions. In mathematics, MAE is a measure of the error between paired observations expressing the same phenomenon. The mathematical model of MAE is as follows:

$$MAE = \frac{\sum_{i=1}^{N} |m_i - o_i|}{N} \tag{26}$$

where N is the total amount of benchmark functions used for testing, m_i is the average of the optimal results calculated by the algorithm, and o_i is the theoretical optimal value of the ith benchmark function.

Table 9 shows the MAE ranking results of these algorithms. The MAE value of CLTSO ranks the first among all competitors, and FDB-AGDE ranks the second. The above data intuitively illustrate the advantage of CLTSO.

Table 9. MAE ranking results of each algorithm.

Algorithm	MAE
CLTSO	2.06×10^{4}
FDB-AGDE	2.70×10^{4}
CMA-ES	1.21×10^{6}
TSO	3.82×10^{6}
WOA	3.55×10^{9}
APSO	9.60×10^{9}

The time consumed by these algorithms in functions F_1~F_{22} are shown in Table 10. The numerical unit is second. The analysis of the time they consumed indicates that the time complexity of CLTSO is slightly higher than that of TSO, but the increase is trivial. The improved operator proposed in this paper only increases the time complexity a little but greatly enhances the optimization performance of the CLTSO algorithm.

Table 10. The execution time of each algorithm.

Function	APSO	WOA	FDB-AGDE	CMA-ES	TSO	CLTSO
F_1	0.5014	0.2206	8.2466	2.6347	0.2277	0.2603
F_2	0.4465	0.3333	8.7963	2.8552	0.2389	0.27260
F_3	0.5782	1.3826	8.4424	4.0751	1.2743	1.2819
F_4	4.3284	0.2083	8.8472	2.4857	0.1960	0.1942
F_5	0.7193	0.2458	3.586	2.7163	0.2440	0.3442
F_6	0.8572	0.1921	8.8967	2.4867	0.1879	0.1946
F_7	0.7557	0.4226	13.6329	2.7073	0.3906	0.3860
F_8	1.0016	0.2599	23.6696	2.6709	0.2689	0.2698
F_9	0.5113	0.2074	26.1973	2.5678	0.2126	0.2246
F_{10}	0.5619	0.2370	21.3863	3.9514	0.3425	0.3350
F_{11}	0.5321	0.4213	20.6873	2.9784	0.3491	0.4298
F_{12}	1.9510	0.9889	19.5429	3.5528	0.8823	0.8381
F_{13}	1.8874	0.8468	11.6196	6.1577	2.2185	1.5804
F_{14}	2.0775	3.0364	4.9752	4.7462	2.2598	2.2449
F_{15}	1.9327	2.3167	15.5858	21.6152	2.4425	2.6542
F_{16}	1.4227	1.9362	14.8224	21.4082	1.9444	2.0605
F_{17}	1.4958	2.0395	14.6579	22.1484	2.0020	1.9974
F_{18}	1.2775	2.2311	15.6855	21.5255	2.1373	2.2258
F_{19}	1.7849	2.1867	15.3276	22.2718	2.0847	2.2293
F_{20}	2.5258	2.2063	29.3849	20.9746	2.1806	2.2839
F_{21}	3.0010	2.4824	30.1273	23.3055	2.4337	2.5681
F_{22}	6.1130	6.2426	49.8559	27.5203	5.9850	6.0176

4.4. Effectiveness Analysis of Improved Operators

This paper makes three improvements to the original tuna swarm optimization algorithm. Firstly, the improved Circle chaotic mapping strategy is introduced in the initialization phase, which expands the swarm diversity. Secondly, the Levy operator is introduced in the position update phase, which strengthens the global swimming ability of tuna. Finally, the nonlinear adaptive weight strategy is introduced in the TSO iteration stage, which can effectively balance the exploration and the exploitation capabilities of the tuna swarm. Section 3 of this chapter proves that the proposed operator significantly improves the optimization performance of TSO. In addition, to verify the effectiveness of the improvements proposed in this paper, we selected the tuna swarm optimization algorithm (TSO), the improved TSO based on the Levy flight operator (LTSO), the improved TSO based on the Circle chaotic map and nonlinear adaptive weights (CTSO), and CLTSO to conduct a set of comparative experiments. Functions F_1~F_{22} are used to test these algorithms in this section, and each algorithm runs 30 times independently. F_1~F_{13} are experiments in 100 dimensions, F_{15}~F_{22} are experiments in 50 dimensions.

The experimental results of various versions of the improved tuna swarm optimization algorithm are displayed in Table 11. Their convergence curves are displayed in Figure 8.

Table 11. Experimental results of various versions of the improved TSO.

Function	Performance	TSO	LTSO	CTSO	CLTSO
F_1	Mean	0	0	0	0
	Std	0	0	0	0
F_2	Mean	9.66×10^{-237}	0	0	0
	Std	0	0	0	0
F_3	Mean	0	0	0	0
	Std	0	0	0	0
F_4	Mean	1.26×10^{-233}	0	0	0
	Std	0	0	0	0

Table 11. *Cont.*

Function	Performance	TSO	LTSO	CTSO	CLTSO
F_5	Mean	2.52×10^{-3}	4.84×10^{-4}	2.35×10^{-3}	1.27×10^{-5}
	Std	3.37×10^{-4}	2.47×10^{-2}	6.09×10^{-3}	5.71×10^{-5}
F_6	Mean	9.33×10^{-4}	6.07×10^{-5}	2.92×10^{-4}	3.07×10^{-5}
	Std	2.93×10^{-3}	8.82×10^{-5}	1.76×10^{-2}	2.78×10^{-5}
F_7	Mean	1.34×10^{-4}	4.20×10^{-5}	8.46×10^{-5}	4.07×10^{-5}
	Std	1.62×10^{-4}	6.46×10^{-5}	1.41×10^{-4}	3.07×10^{-5}
F_8	Mean	-4.19×10^{4}	-4.19×10^{4}	-4.19×10^{4}	-4.19×10^{4}
	Std	2.51×10^{4}	2.51×10^{4}	7.54×10^{3}	5.30×10^{-8}
F_9	Mean	0	0	0	0
	Std	0	0	0	0
F_{10}	Mean	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}
	Std	0	0	0	0
F_{11}	Mean	0	0	0	0
	Std	0	0	0	0
F_{12}	Mean	5.26×10^{-5}	1.41×10^{-6}	7.65×10^{-7}	5.14×10^{-10}
	Std	2.72×10^{-5}	2.04×10^{-7}	8.98×10^{-5}	1.52×10^{-7}
F_{13}	Mean	7.86×10^{-4}	7.47×10^{-6}	9.84×10^{-6}	3.60×10^{-7}
	Std	7.73×10^{-4}	5.16×10^{-3}	1.58×10^{-3}	2.08×10^{-5}
F_{14}	Mean	9.98×10^{-1}	9.98×10^{-1}	9.98×10^{-1}	9.98×10^{-1}
	Std	5.99×10^{-1}	5.99×10^{-1}	5.99×10^{-1}	1.72×10^{-16}
F_{15}	Mean	1.15×10^{6}	1.69×10^{6}	2.32×10^{6}	9.51×10^{5}
	Std	1.45×10^{6}	6.65×10^{5}	2.06×10^{6}	3.35×10^{5}
F_{16}	Mean	1.38×10^{4}	1.12×10^{4}	1.61×10^{3}	3.40×10^{2}
	Std	7.08×10^{3}	1.93×10^{3}	8.20×10^{3}	2.05×10^{3}
F_{17}	Mean	3.16×10^{3}	4.72×10^{2}	4.57×10^{3}	3.71×10^{2}
	Std	6.55×10^{3}	2.28×10^{2}	1.62×10^{4}	4.36×10^{1}
F_{18}	Mean	5.21×10^{2}	5.20×10^{2}	5.21×10^{2}	5.20×10^{2}
	Std	3.13×10^{2}	3.12×10^{2}	3.13×10^{2}	4.47×10^{-2}
F_{19}	Mean	5.54×10^{3}	8.74×10^{3}	4.22×10^{3}	3.78×10^{3}
	Std	2.93×10^{3}	3.42×10^{3}	2.35×10^{3}	1.34×10^{3}
F_{20}	Mean	6.52×10^{3}	3.09×10^{3}	5.49×10^{3}	2.62×10^{3}
	Std	3.39×10^{3}	1.59×10^{3}	4.34×10^{3}	1.85×10^{2}
F_{21}	Mean	2.15×10^{4}	7.56×10^{4}	1.95×10^{4}	1.90×10^{4}
	Std	1.30×10^{5}	3.01×10^{4}	6.81×10^{4}	3.08×10^{4}
F_{22}	Mean	3.20×10^{3}	3.20×10^{3}	3.20×10^{3}	3.20×10^{3}
	Std	0	0	0	0

As can be seen from Table 10 and Figure 8, CLTSO has higher optimization accuracy than the competitors. In benchmark functions F_1, F_2, F_3, F_4, F_9, F_{10}, F_{11}, and F_{14}, CLTSO, CTSO, and LTSO can calculate the theoretical optimal values, but CLTSO converges much faster than CTSO and LTSO. The above data indicate that the optimization performance of CTSO and LTSO is more enhanced than the original tuna swarm optimization algorithm, which further confirms the validity of the three modified operators in CLTSO. To demonstrate that the optimization capability of CLTSO is greatly enhanced compared to CTSO and LTSO, Friedman statistical analysis and MAE ranking are conducted based on the data in Table 11. The analysis and ranking results are listed in Tables 12 and 13.

Table 12. Results of Friedman statistical analysis.

Algorithm	Rank Mean
CLTSO	1.80
LTSO	2.25
CTSO	2.84
TSO	3.11

Table 13. MAE ranking results.

Algorithm	MAE
CLTSO	4.42×10^4
LTSO	8.06×10^4
CTSO	1.08×10^5
TSO	1.18×10^5

According to the above two tables, it is clear that CLTSO has the smallest rank mean in the Friedman analysis test, LTSO ranks the second, and CTSO ranks the third, followed by TSO. According to the MAE value of each algorithm, CLTSO ranks the first. The above ranking shows that CLTSO can better approximate the theoretical optimal value when dealing with optimization problems. CLTSO has shown much better performance than the competitors. Therefore, the above data and analysis results confirm that the three improved operators proposed in this paper are effective.

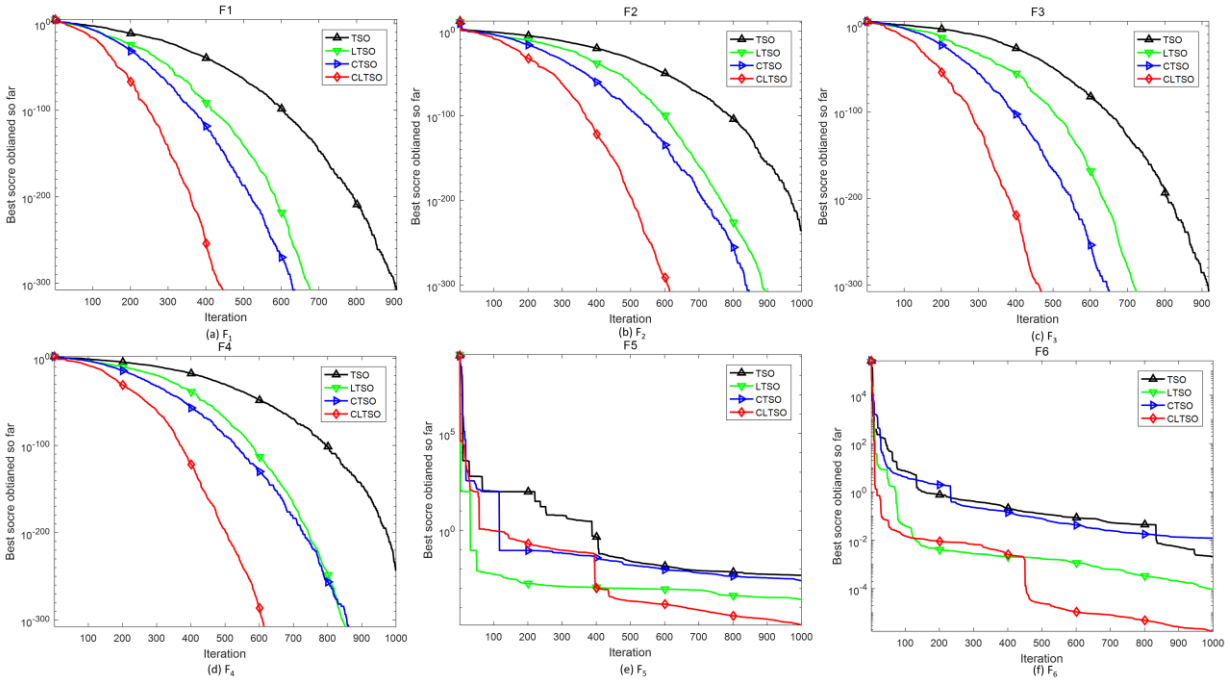

Figure 8. *Cont.*

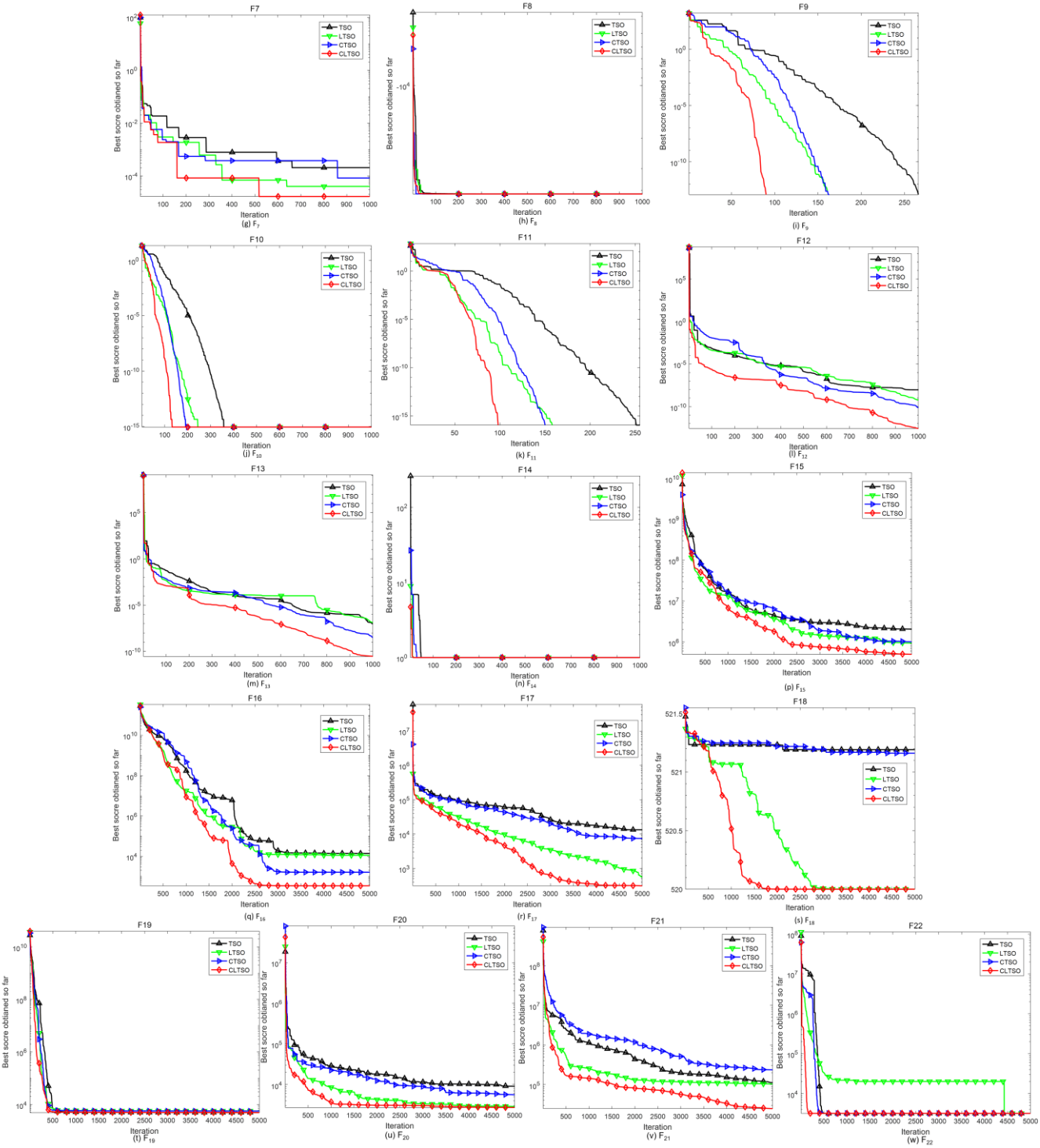

Figure 8. Convergence curves of each version of improved TSO.

5. Optimization Engineering Example Using CLTSO

The original intention of meta-heuristic algorithms is to optimize the engineering problems encountered. How to improve the precision of engineering practice is the concern of researchers. To verify the effectiveness of CLTSO for real engineering problems, CLTSO is applied to the modification design of a BP neural network. The BP neural network is a model proposed by McCulloch to train the network based on error back propagation. It is one of the most mature and widely used artificial neural network modules. The BP neural network is widely used in pattern recognition, classification and prediction, nonlinear

modeling, etc. Figure 9 shows a BP neural network topology with d input neurons, l output neurons, and q hidden layer neurons.

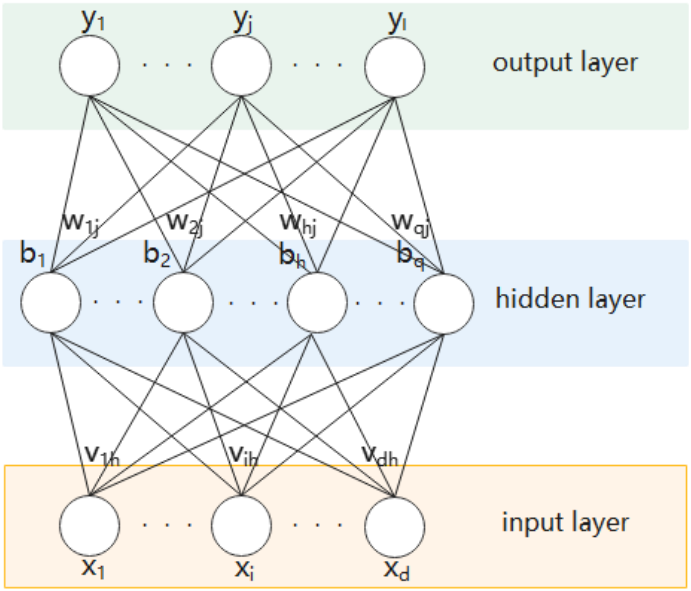

Figure 9. BP neural network topology.

v_{ih} is the weight between the ith node in the input layer and the hth node in the hidden layer. w_{hj} is the weight between the hth node in the input layer and the jth node in the hidden layer. The threshold of the jth node in the output layer is expressed by θ_j. Therefore, the input value received by the hidden layer hth neurons in the network model is as follows:

$$\alpha_h = \sum_{i=1}^{d} v_{ih} x_i \tag{27}$$

The value received by the jth node in the output layer is as follows:

$$\beta_j = \sum_{h=1}^{q} w_{ih} b_h \tag{28}$$

where b_h is the output value of the hth neuron in the hidden layer. Taking training case (x_k, y_k) as an example, we assume that the output of the network model is as follows:

$$\hat{y}_k = (\hat{y}_1^k, \hat{y}_2^k, \cdots, \hat{y}_l^k) \tag{29}$$

$$\hat{y}_j^k = f(\beta_j - \theta_j) \tag{30}$$

Therefore, the mean square error of the network on example (x_k, y_k) is as follows:

$$E_k = 1/2 \sum_{j=1}^{l} (\hat{y}_j^k - y_j^k)^2 \tag{31}$$

where n is the total amount of training samples, m is the total amount of input nodes, x_i^k is the output value of the network model, and d_i^k is the real value of training samples.

In the training process of the model, the error will be transmitted back to the hidden nodes. The model will adjust the weights and thresholds between each layer of nodes based on the error, and finally make the error achieve satisfactory accuracy. At present, the training methods of the BP neural network are mostly gradient descent. The training accuracy of the network model is extremely sensitive to the initial weight value and the learning rate. Therefore, when the objective function has multiple extreme values, the neural network is easily attracted by local extreme values. This will lead to a serious degradation in the performance of the algorithm. In order to optimize the performance of the BP network model and verify the optimization ability of CLTSO, a CLTSO-BP neural

network model is proposed. The basic idea of the model is to use the weights and thresholds of each node in the BP model as the tuna individual in the CLTSO algorithm and use MSE as the fitness function in the CLTSO algorithm. CLTSO optimizes the MSE of the model to obtain the optimized initial value weight and the threshold.

To compare the capability of the CLTSO-BP neural network with the original BP model, three popular datasets from the UCI machine learning and intelligent system center, Iris, Wine, and Wine Quality, are selected to design a comparative experiment. This experiment compares the classification accuracy of the CLTSO-BP neural network model and the BP model on the above three datasets.

In the experiment, the total amount of tuna is 30, the CLTSO algorithm is executed 30 times in total, and the neural network is executed 500 times in total. Table 14 shows the comparison results of the CLTSO-BP neural network model and the original BP model.

Table 14. The comparison results of the two models.

Dataset	Model	Classification Accuracy
Iris	CLTSO-BP neural network	100%
	BP neural network	95.2%
Wine	CLTSO-BP neural network	100%
	BP neural network	94.4%
Wine Quality	CLTSO-BP neural network	65.6%
	BP neural network	45.2%

By comparing the result of the CLTSO-BP neural network and the original BP model on three datasets, it is found that the new model can obtain more ideal classification results. It also indicates that CLTSO can show excellent performance in multi-layer perceptron training difficulties.

6. Conclusions

The tuna swarm optimization algorithm is widely recognized by scholars because of its simple structure and low number of parameters. The tuna swarm optimization algorithm has excellent optimization performance, but it can still be further improved. When dealing with simple problems, the solving speed of TSO can still be further improved. When facing complex problems, it is difficult for TSO to escape the attraction of local optimal value. Therefore, this article proposes a modified nonlinear tuna swarm optimization algorithm based on Circle chaotic map and Levy flight operator. The optimization performance of CLTSO has been fully verified in 22 benchmark functions. The results show that CLTSO outperforms the comparable algorithms. Comparation data based on 22 benchmark functions were analyzed using Wilcoxon's test, Friedman's test, and MAE. The analysis conclusion indicates that the rank mean and MAE value of CLTSO are superior to other advanced algorithms such as CMA-ES. Finally, this paper optimizes the BP neural network based on CLTSO. The CLTSO-BP neural network model is tested using three popular datasets from the UCI Machine Learning and Intelligent System Center. Compared with the original BP model, the new model optimizes the classification accuracy. However, for the optimization problem of more complex datasets, the classification ability of the CLTSO-BP neural network still needs to be improved. Possible directions include increasing the swarm size of the algorithm and the total number of CLTSO operations to obtain a higher quality solution, which is also the target of continuous research in the future. In addition, the advantages of CLTSO in solving some complex multimodal functions can still be further improved, which is also one of the key research directions in the future. CLTSO has the advantages of fast convergence and high convergence accuracy, which can be applied in practical projects such as workshop scheduling and distribution network reconstruction.

Author Contributions: Conceptualization, W.W. and J.T.; methodology, W.W.; software, W.W.; validation, W.W.; formal analysis, W.W.; investigation, W.W.; resources, W.W.; data curation, W.W.; writing—original draft preparation, W.W.; writing—review and editing, W.W. and J.T.; visualization, W.W.; supervision, J.T.; project administration, J.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Narendra, P.M.; Fukunaga, K. A branch and bound algorithm for feature subsets election. *IEEE Trans Comput.* **1977**, *26*, 917–922. [CrossRef]
2. Wu, G.; Pedrycz, W.; Suganthan, P.N.; Mallipeddi, R. A variable reduction strategy for evolutionary algorithms handling equality constraints. *Appl. Soft Comput. J.* **2015**, *37*, 774–786. [CrossRef]
3. Zhang, H.; Cui, L.; Zhang, X.; Luo, Y. Data-driven robust approximate optimal tracking control for unknown general non-linear systems using adaptive dynamic programming method. *IEEE Trans. Neural Netw.* **2011**, *22*, 2226–2236. [CrossRef]
4. Slowik, A.; Halina, K. Nature inspired methods and their industry applications—Swarm intelligence algorithms. *IEEE Trans. Ind. Inform.* **2017**, *14*, 1004–1015. [CrossRef]
5. Chakraborty, A.; Kar, A.K. Swarm intelligence: A review of algorithms. *Nat.-Inspir. Comput. Optim.* **2017**, *10*, 475–494.
6. Liu, W.; Dridi, M.; Fei, H.; el Hassani, A.H. Hybrid metaheuristics for solving a home health care routing and scheduling problem with time windows, synchronized visits and lunch breaks. *Expert Syst. Appl.* **2021**, *183*, 115307. [CrossRef]
7. Wang, X.; Zhao, H.; Han, T.; Zhou, H.; Li, C. A grey wolf optimizer using Gaussian estimation of distribution and its application in the multi-UAV multi-target urban tracking problem. *Appl. Soft Comput.* **2019**, *78*, 240–260. [CrossRef]
8. Wang, Y.-G. A maximum-likelihood method for estimating natural mortality and catchability coefficient from catch-and-effort data. *Mar. Freshw. Res.* **1999**, *50*, 307–311. [CrossRef]
9. Wu, J.; Ding, Z. Improved grey model by dragonfly algorithm for Chinese tourism demand forecasting. In Proceedings of the 33rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Kitakyushu, Japan, 22–25 September 2020.
10. Wu, J.; Cui, Z.; Chen, Y.; Kong, D.; Wang, Y.-G. A new hybrid model to predict the electrical load in five states of Australia. *Energy* **2019**, *166*, 598–609. [CrossRef]
11. Webb, B. Swarm Intelligence: From Natural to Artificial Systems. *Connect. Sci.* **2002**, *14*, 163–164. [CrossRef]
12. Kennedy, J. Swarm Intelligence. In *Handbook of Nature-Inspired and Innovative Computing*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 187–219.
13. Ashlock, D. *Evolutionary Computation for Modeling and Optimization*; Springer: New York, NY, USA, 2006; Volume 51, p. 743.
14. Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput.-Aided Des.* **2011**, *43*, 303–315. [CrossRef]
15. Mirjalili, S.; Mirjalili, S.M.; Hatamlou, A. Multi-Verse Optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **2015**, *27*, 495–513. [CrossRef]
16. Chopra, N.; Ansari, M.M. Golden jackal optimization: A novel nature-inspired optimizer for engineering applications. *Expert Syst. Appl.* **2022**, *198*, 116924. [CrossRef]
17. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
18. Chen, D.; Ge, Y.; Wan, Y.; Deng, Y.; Chen, Y.; Zou, F. Poplar optimization algorithm: A new meta-heuristic optimization technique for numerical optimization and image segmentation. *Expert Syst. Appl.* **2022**, *200*, 1–17. [CrossRef]
19. Man, K.F.; Tang, K.S.; Kwong, S. Genetic Algorithms. *Perspect. Neural Comput.* **1989**, *83*, 55–80.
20. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
21. Du, Z.; Li, S.; Sun, Y.; Li, N. Adaptive particle swarm optimization algorithm based on levy flights mechanism. In Proceedings of the 2017 Chinese Automation Congress (CAC), Jinan, China, 20–22 October 2017.
22. Yu, H.; Yu, Y.; Liu, Y.; Wang, Y.; Gao, S. Chaotic grey wolf optimization. In Proceedings of the 2016 International Conference on Progress in Informatics and Computing (PIC), Beijing, China, 23–26 December 2016; pp. 103–113.
23. Yuan, X.; Yang, D.; Liu, H. MPPT of PV system under partial shading condition based on adaptive inertia weight particle swarm optimization algorithm. In Proceedings of the 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), Shenyang, China, 8–12 June 2015; pp. 729–733.
24. Park, S.; Kim, Y.; Kim, J.; Lee, J. Speeded-up cuckoo search using opposition-based learning. In Proceedings of the 2014 14th International Conference on Control, Automation and Systems (ICCAS 2014), Seoul, Korea, 22–25 October 2014; pp. 535–539.

25. Hatamlou, A. Black hole: A new heuristic optimization approach for data clustering. *Inf. Sci.* **2013**, *222*, 175–184. [CrossRef]
26. Xie, W.; Wang, J.S.; Tao, Y. Improved Black Hole Algorithm Based on Golden Sine Operator and Levy Flight Operator. *IEEE Access* **2019**, *7*, 161459–161486. [CrossRef]
27. Xie, L.; Han, T.; Zhou, H.; Zhang, Z.-R.; Han, B.; Tang, A. Tuna Swarm Optimization: A Novel Swarm-Based Metaheuristic Algorithm for Global Optimization. *Comput. Intell. Neurosci.* **2021**, *2021*, 9210050. [CrossRef]
28. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
29. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
30. Hu, D.; Yang, S. Improved Tuna Algorithm to Optimize ELM Model for PV Power Prediction. *J. Wuhan Univ. Technol.* **2022**, *44*, 97–104.
31. Kumar, C.; Mary, D.M. A novel chaotic-driven Tuna Swarm Optimizer with Newton-Raphson method for parameter identification of three-diode equivalent circuit model of solar photovoltaic cells/modules. *Optik* **2022**, *264*, 169379. [CrossRef]
32. Arora, S.; Anand, P. Chaotic grasshopper optimization algorithm for global optimization. *Neural Comput. Appl.* **2019**, *31*, 4385–4405. [CrossRef]
33. Guangyuan, P.; Junfei, Q.; Honggui, H. A new strategy of chaotic PSO and its application in optimization design for pipe network. In Proceedings of the 32nd Chinese Control Conference, Xi'an, China, 26–28 July 2013; pp. 8022–8027.
34. Pluhacek, M.; Senkerik, R.; Zelinka, I.; Davendra, D. Designing PID Controllers by Means of PSO Algorithm Enhanced by Various Chaotic Maps. In Proceedings of the 2013 8th EUROSIM Congress on Modelling and Simulation, Cardiff, UK, 10–13 September 2013; pp. 19–23.
35. Zhao, J. Chaotic particle swarm optimization algorithm based on tent mapping for dynamic origin-destination matrix estimation. In Proceedings of the 2011 International Conference on Electric Information and Control Engineering, Wuhan, China, 15–17 April 2011; pp. 221–224.
36. Zhang, J.; Zhu, Y.; Zhu, H.; Cheng, J. Some improvements to logistic map for chaotic signal generator. In Proceedings of the 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, 13–16 December 2017; pp. 1090–1093.
37. Li, M.; Sun, X.; Li, W.; Wang, Y. Improved Chaotic Particle Swarm Optimization using circle map for training SVM. In Proceedings of the 2009 Fourth International on Conference on Bio-Inspired Computing, Beijing, China, 16–19 October 2009; pp. 1–7.
38. Vasuyta, K.; Zakharchenko, I. Modified discrete chaotic map bas-ed on Chebyshev polynomial. In Proceedings of the 2016 Third International Scientific-Practical Conference Problems of Info communications Science and Technology (PIC S&T), Kharkov, Ukraine, 4–6 October 2016; pp. 217–219.
39. Jiteurtragool, N.; Ketthong, P.; Wannaboon, C.; San-Um, W. A topologically simple keyed hash function based on circular chaotic sinusoidal map network. In Proceedings of the 2013 15th International Conference on Advanced Communications Technology (ICACT), Seoul, Korea, 27–30 January 2013; pp. 1089–1094.
40. Petavratzis, E.; Moysis, L.; Volos, C.; Nistazakis, H.; Muñoz-Pacheco, J.M.; Stouboulos, I. Motion Control of a Mobile Robot Based on a Chaotic Iterative Map. In Proceedings of the 2020 9th International Conference on Modern Circuits and Systems Technologies (MOCAST), Pradesh, India, 7–9 September 2020; pp. 1–4.
41. Zhang, D.M.; Xu, H.; Wang, Y.R.; Song, T.; Wang. Whale optimization algorithm for embedded circle mapping and one-dimensional oppositional learning based small hole imaging. *Control. Decis.* **2021**, *36*, 1173–1180.
42. Song, L.; Chen, W.; Chen, W.; Lin, Y.; Sun, X. Improvement and application of sparrow search algorithm based on hybrid strategy. *J. Beijing Univ. Aeronaut. Astronaut.* **2021**, *1*, 1–16.
43. Viswanathan, G.M.; Afanasyev, V.; Buldyrev, S.V.; Havlin, S.; Da Luz, M.G.E.; Raposo, E.P.; Stanley, H.E. Levy fights search patterns of biological organisms. *Phys. A Stat. Mech. Its Appl.* **2001**, *295*, 85–88. [CrossRef]
44. Biagini, F.; Hu, Y.; Øksendal, B.; Zhang, T. Stochastic Optimal Control and Applications. In *Stochastic Calculus for Fractional Brownian Motion and Applications*; Springer: London, UK, 2008.
45. Yan, X.F.; Ye, D.Y. An improved flora foraging algorithm based on Levy flight. *Comput. Syst. Appl.* **2015**, *24*, 124–132.
46. Haklı, H.; Uğuz, H. A novel particle swarm optimization algorithm with Levy flight. *Appl. Soft Comput.* **2014**, *23*, 333–345. [CrossRef]
47. Liu, C.; Ye, C. Bat algorithm with Levy flight characteristics. *Chin. J. Intell. Syst.* **2013**, *8*, 240–246.
48. Mantegna, R.N. Fast accurate algorithm for numerical simulation of Lévy stable stochastic processes. *Phys. Rev.* **1994**, *49*, 4677–4689. [CrossRef] [PubMed]
49. Zhang, J.; Wang, J.S. Improved Whale Optimization Algorithm Based on Nonlinear Adaptive Weight and Golden Sine Operator. *IEEE Access.* **2020**, *8*, 77013–77048. [CrossRef]
50. Zhang, J.; Wang, J.S. Improved Salp Swarm Algorithm Based on Levy Flight and Sine Cosine Operator. *IEEE Access.* **2020**, *8*, 99740–99771. [CrossRef]
51. Aloui, M.; Hamidi, F. A Chaotic Krill Herd Optimization Algorithm for Global Numerical Estimation of the Attraction. *Domain Nonlinear Syst.* **2021**, *9*, 1743.
52. Liang, J.J.; Qu, B.Y.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*; Technical Report; Computational Intelligence Laboratory, Zhengzhou University: Zhengzhou, China, 2013; p. 201311.

53. Reddy, R.B.; Uttara, K.M. Performance Analysis of Mimo Radar Waveform Using Accelerated Particle Swarm Optimization Algorithm. *Signal Image Process.* **2012**, *3*, 4. [CrossRef]
54. Guvenc, U.; Duman, S.; Kahraman, H.T.; Aras, S.; Katı, M. Fitness-Distance Balance based adaptive guided differential evolution algorithm for security-constrained optimal power flow problem incorporating renewable energy sources. *Appl. Soft Comput.* **2021**, *108*, 107421. [CrossRef]
55. Hansen, N.; Müller, S.D.; Koumoutsakos, P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol. Comput.* **2003**, *11*, 1–18. [CrossRef]
56. García, S.; Molina, D.; Lozano, M.; Herrera, F. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization. *J. Heuristics* **2008**, *15*, 617. [CrossRef]