*mathematics*

MDPI

*Article*

# An Enhanced Northern Goshawk Optimization Algorithm and Its Application in Practical Optimization Problems

**Yan Liang [1], Xianzhi Hu [2], Gang Hu [3,*] and Wanting Dou [1]**

[1] School of Technology, Xi'an Siyuan University, Xi'an 710038, China
[2] Division of Informationize Management, Xi'an University of Technology, Xi'an 710048, China
[3] Department of Applied Mathematics, Xi'an University of Technology, Xi'an 710054, China
* Correspondence: hugang@xaut.edu.cn

**Abstract:** As a kind of effective tool in solving complex optimization problems, intelligent optimization algorithms are paid more attention to their advantages of being easy to implement and their wide applicability. This paper proposes an enhanced northern goshawk optimization algorithm to further improve the ability to solve challenging tasks. Firstly, by applying the polynomial interpolation strategy to the whole population, the quality of the solutions can be enhanced to keep a fast convergence to the better individual. Then, to avoid falling into lots of local optimums, especially late in the whole search, different kinds of opposite learning methods are used to help the algorithm to search the space more fully, including opposite learning, quasi-opposite learning, and quasi-reflected learning, to keep the diversity of the population, which is noted as a multi-strategy opposite learning method in this paper. Following the construction of the enhanced algorithm, its performance is analyzed by solving the CEC2017 test suite, and five practical optimization problems. Results show that the enhanced algorithm ranks first on 23 test functions, accounting for 79.31% among 29 functions, and keeps a faster convergence speed and a better stability on most functions, compared with the original northern goshawk optimization algorithm and other popular algorithms. For practical problems, the enhanced algorithm is still effective. When the complexity of the TSP is increased, the performance of the improved algorithm is much better than others on all measure indexes. Thus, the enhanced algorithm can keep the balance between exploitation and exploration and obtain better solutions with a faster speed for problems of high complexity.

**Keywords:** northern goshawk optimization algorithm; polynomial interpolation; opposite learning method; engineering optimization problem; traveling salesman problem

**MSC:** 49K35; 68T20



**Citation:** Liang, Y.; Hu, X.; Hu, G.; Dou, W. An Enhanced Northern Goshawk Optimization Algorithm and Its Application in Practical Optimization Problems. *Mathematics* **2022**, *10*, 4383. https://doi.org/10.3390/math10224383

Academic Editor: Ioannis G. Tsoulos

Received: 8 October 2022
Accepted: 15 November 2022
Published: 21 November 2022



## 1. Introduction

Optimization theory is an important branch of mathematics, which studies the problem of how to find the best solution among lots of solutions [1]. It has been applied to solve the challenging optimization problems in agriculture, industry, national defense, transportation and other fields, especially for the optimization models with strong constraints, multivariable systems or multi-objectives [2]. The practice shows that under the same conditions, the optimization technology can improve the efficiency of the system, allocate resources rationally and reduce energy consumption [3]. This effect becomes more significant with the increase in the complexity of the optimization problem.

According to the characteristics of the approaches to solving optimization problems, massive optimization techniques can be classified into two categories: deterministic optimization techniques and intelligent optimization algorithms [4]. The former uses the analytic properties of the problem to generate a definite finite or infinite sequence of points to converge to the global optimal solution [5], including the gradient descent method [6],

the Newton method [7], the conjugate gradient method [8], and so on. The gradient descent method can search the global optimum and is easy to implement. The Newton method is to approximate the solution of the equations in the natural and complex domains with the characteristic of a fast convergence speed. The conjugate gradient method is between the gradient descent method and the Newton method. It overcomes the slow convergence of the gradient descent and the calculation of the Hessian matrix of the Newton method because it only uses the information of the first derivative [9].

These approaches are computationally efficient because they make full use of the analytic properties of the target problem. However, the deterministic optimization techniques may face challenges in the solution accuracy for problems with a high complexity, strong nonlinearity and many variables [10]. Thus, intelligent optimization algorithms (or meta-heuristic algorithms) were proposed.

Intelligent optimization algorithms are random search methods. Compared with traditional optimization methods, this kind of tool does not have any special requirements (differentiable or convex optimization) on the objective function, and is not limited to specific problems [11]. Therefore, its application scope is more extensive, and it has become an effective approach to solving the optimization problem.

There are several methods to classify intelligent algorithms [12]. Here, they are summarized into three types, including evolutionary-based algorithms, physics or mathematics-based algorithms and swarm intelligence algorithms [13]. Figure 1 shows the classification of intelligent optimization algorithms and some representative algorithms of these classifications.

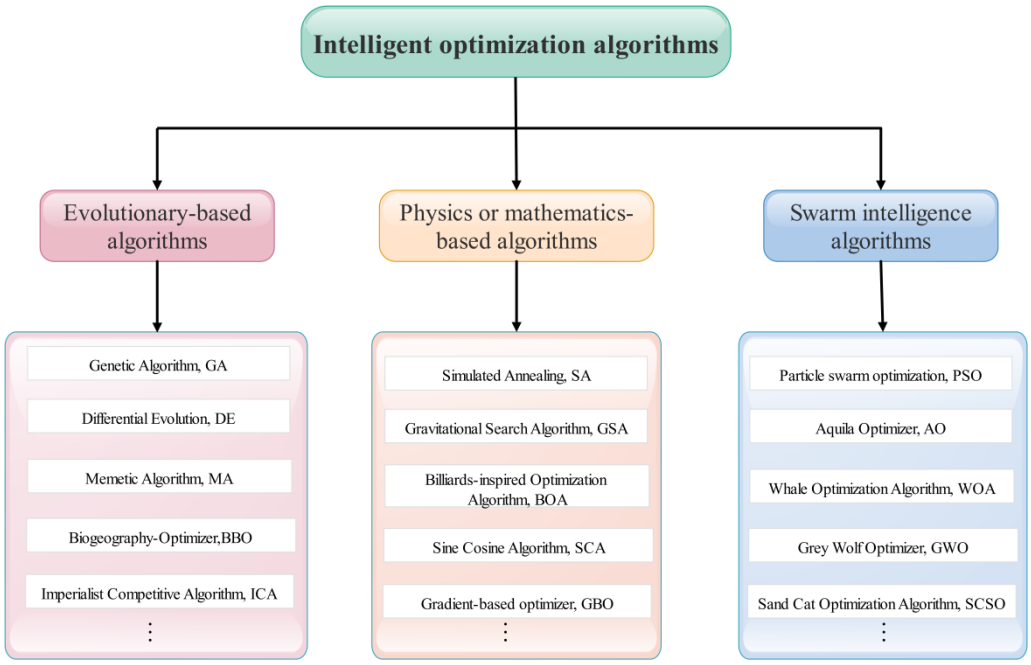


**Figure 1.** The classification of intelligent optimization algorithms.

The evolutionary approach takes natural law as the core idea. The most classical one of this kind is the genetic algorithm (GA), which simulates the evolutionary process of Darwin's genetic selection and natural elimination. It is an effective method to search for the optimal solution [14]. However, the GA involves the process of encoding and decoding, which makes it time-consuming. The differential evolution algorithm (DE) is also a popular algorithm, which consists of five stages, including the population initialization, progeny generation, progeny mutation, progeny selection and fitness evaluation. Because the encoding and decoding processes are abandoned, the search efficiency is improved [15]. The imperialist competitive algorithm (ICA) is an overall optimization algorithm, based on the evolutionary, inspired by the competition, occupation and annexation of colonial

countries among imperialist countries in the political and social colonial stage, in human history [16]. Other algorithms of this kind are the memetic algorithm (MA) [17], the bio-geography optimization algorithm (BBO) [18] and so on.

Physics or mathematics-based algorithms are inspired by the physical laws or mathematical theorems found in nature by scholars. For example, the simulated annealing algorithm (SA) was proposed, based on the similarity between the annealing process of solid matter in physics and the general combinatorial optimization problem. Starting from a high initial temperature, the global optimal solution of the objective function can be found in the solution space with the constant decrease of the temperature parameter. Meanwhile, it can jump out of the local optimum, based on the probability jump property [19]. According to the law of universal gravitation and Newton's second law, the gravitational search algorithm (GSA) was constructed. This algorithm searches for the optimal solution by moving the particle position of the population continuously in the search space by the universal gravitation between them. When the particle moves to the optimal position, the optimal solution is found [20]. The sine cosine algorithm (SCA) was proposed, based on the mathematical model of sine and cosine, to search the space region and find the optimal solution [21]. Other such algorithms are the billiards-inspired optimization algorithm (BOA) [22], the gradient-based optimizer (GBO) [23] and so on.

The swarm intelligence algorithms were proposed, based on the habits of social animals, such as birds, ants, wolves or bees. The particle swarm optimization algorithm (PSO) is one of the most popular algorithms, which mimics the behavior of birds. For each individual in the group, it changes the search pattern by learning from its own experience and the experience of other members [24]. Due to the simple construction and special learning method, the PSO algorithm performs well on the convergence speed and accuracy. Thus, it has been applied to solve some practical optimization problems. Moreover, lots of novel algorithms of this kind have emerged in recent years by simulating other animals. For example, the whale optimization algorithm (WOA) is proposed by mimicking humpback whale hunting behavior in 2016 [25]. The grey wolf optimizer (GWO) mimics the hierarchical system of the wolf population, which is unique in that a small number of elite gray wolves lead a group of gray wolves toward their prey [26]. In 2021, the Aquila optimizer (AO) was proposed. It was constructed in four stages using the hunting process of the Aquila in the wild. During different stages, the Aquila will fly in different methods to make sure their prey doesn't escape, such as a short glide attack, swooping, or high soaring [27]. In 2022, inspired by the features of the sand cat swarm of detecting low frequencies and digging for prey, the sand cat swarm optimization algorithm (SCSO) was proposed [28].

Though the above algorithms have been used to solve test functions or engineering optimization problems, there are still possibilities to further improve the capacity in the solution accuracy or convergence speed [29]. Thus, some improved strategies are introduced into the original algorithm. Here, these strategies are divided into two categories.

The first category is the strategies to enhance the exploitation ability of the algorithm. For example, in [30], a linearly dynamic random heuristic crossover strategy was added to the BBO algorithm, to obtain a stronger local search ability. Noticing the deficiency of the exploitation ability for the surface conversion model, Hu et al. introduced a golden sine learning strategy, based on the relationship between the sine function and the unit circle to help individuals to approach better solutions, which is an effective way to improve the quality of the whole population [31]. Similarly, to enhance the exploitation phase of the colliding bodies optimization algorithm, the quadratic interpolation of the utilized historically best solution was employed and showed outstanding performance in solving 24 mathematical optimization problems, including 30 design variables [32]. Aiming at the original algorithm's low capability in exploiting, Yousri et al. [33] adopted the fractional calculus, using the Caputo fractional differ-sum operator, to enhance the manta rays movement in the exploitation phase, via utilizing the history dependency of fractional

calculus to boost exploiting the optimal solutions, via sharing the past knowledge over the optimization process.

Moreover, some strategies are used to enhance the exploration capacity. In [34], the classical PSO algorithm was improved, based on a Lévy fly strategy, which is a random walk method. Following the application of this strategy to the original population, the individuals move by leaps and bounds in the search space, which is an effective approach to avoid the local optimums. Different mutation methods are also effective in improving the exploration capacity, such as the wavelet mutation and Cauchy mutation. In Jiang et al. [35], each individual has the possibility to experience the wavelet mutation by setting a mutation rate. Due to the volatility of wavelet function in the wavelet function, individuals have randomness in updating their positions to enhance the population diversity. Miao et al. [36] introduces the mutation ability of the Cauchy distribution, to help the original algorithm to explore a wider solution space and avoid falling into a local optimum prematurely. Compared with the state-of-the-art algorithms, the novel version of the improved algorithm performs best on the feature selection of sleep staging. The opposite learning (OL) strategy is also a common approach to assist the algorithm to explore the space quickly. In [37], the dynamic OL strategy is embedded into the population initialization stage and the generation jumping stage of the original dragonfly algorithm to solve the flexible job shop scheduling problem. The results obtained showed that the improved algorithm can efficiently achieve a better solution on 15 examples, compared to the algorithms.

Based on the above knowledge, this paper proposes a novel version of the northern goshawk optimization algorithm (ENGO), by enhancing the abilities of exploitation and exploration with the according strategies. The northern goshawk optimization (NGO) algorithm was proposed by Dehghani et al., by simulating the behavior of the northern goshawk during the process of hunting the prey, including two stages of prey identification and the tail and chase process [38]. Once it was tested on 68 different objective functions and four engineering design problems, we found that it can keep a balance between the exploration capacity and the exploitation capacity, and has an outstanding performance for the optimization problems, compared with other similar algorithms. To improve its capacity in solving problems with higher dimensions and stronger constraint conditions, this paper proposed an enhanced NGO algorithm, noted as ENGO, and verified it on the CEC2017 test suite and engineering optimization problems. The main contributions of this paper are as follows:

Firstly, an enhanced NGO algorithm is proposed by introducing the polynomial interpolation strategy and the multi-strategy opposite learning method.

- To improve the capacity of exploitation, this paper applies the quadratic interpolation function to each individual, to find a better solution.
- Different opposite learning methods are employed to help the algorithm search the space more fully, including opposite learning, quasi-opposite learning and quasi-reflected learning.

Secondly, the ENGO algorithm is used to solve the test functions of the CEC2017 test suite, four engineering optimization problems and the traveling salesman problem (TSP).

The structure of this paper is as follows: Section 2 introduces the original NGO algorithm and how to apply the two strategies to it to obtain the ENGO algorithm. Section 3 shows the results of the ENGO algorithm and other comparison algorithms in solving the CEC2017 test suite. Section 4 analyzes the performance of the ENGO algorithm in solving engineering problems and TSP. Section 5 is the conclusion of this paper.

## 2. The Enhanced Northern Goshawk Optimization Algorithm

### *2.1. The Original Northern Goshawk Optimization*

The search mechanism of the NGO algorithm stems from its efficient search and capture of prey. Thus, the algorithm consists of three stages, which are population initialization, prey identification and prey capture.

2.1.1. Initialization

Firstly, the initialization population of the northern goshawk can be presented by matrix $X$, shown as follows.

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,M} \\ x_{2,1} & x_{2,1} & \cdots & x_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,M} \end{bmatrix}, \tag{1}$$

where, $X_i$, $1 \le i \le N$ represents the $i$-th individual of the whole population. $N$ and $M$ are the size of the population and the dimension of the objective function, respectively. For a single objective optimization problem with upper bound $UB$ and lower bound $LB$, the elements of $X_i$ can be calculated by

$$x_{i,j} = LB + rand \cdot (UB - LB),\ 1 \le i \le N; 1 \le j \le M. \tag{2}$$

2.1.2. Prey Identification

In the first stage, the northern goshawk will select the prey and try to attack it. Because the prey is selected randomly, this behavior can reflect the global exploration ability of the algorithm in the feasible space. Suppose the $prey_i$, shown in Equation (3), is the target selected by the individual $X_i$, then Equation (4) mimics the process of the northern goshawk attacking its prey.

$$prey_i = X_p,\ i = 1, 2, \ldots, N; p = 1, 2, \ldots, i-1, i+1, \ldots, N. \tag{3}$$

$$\begin{cases} X_i^{new} = X_i + r(prey_i - IX_i), & Fit(prey_i) < Fit(X_i), \\ X_i^{new} = X_i + r(X_i - prey_i), & Fit(prey_i) \ge Fit(X_i), \end{cases} \tag{4}$$

where $r$ is a random vector with numbers in [0, 1], and $I$ is a vector consisting of 1 or 2. $r$ and $I$ are used to enhance the randomness of the algorithm to search the space more fully. Then, the individual $X_i$ will be updated by Equation (5).

$$\begin{cases} X_i = X_i^{new}, & Fit(X_i^{new}) < Fit(X_i), \\ X_i = X_i, & Fit(X_i^{new}) \ge Fit(X_i). \end{cases} \tag{5}$$

2.1.3. Prey Capture

When the northern goshawk locks on to its prey and starts attacking it, the prey will be spooked and start escaping. In this stage, the northern goshawk needs to keep chasing its prey. Due to its high pursuit speed, the northern goshawk can chase and eventually capture prey in almost any situation. Regarding a circle with radius $r$ as the range of the chasing behavior, this stage can be simulated by Equation (6).

$$X_i^{new} = X_i + R(2r - 1)X_i, \tag{6}$$

where $R = 0.02(1 - t/T)$. $T$ is the maximum number of iterations and $t$ is the current iteration. Then, the individual $X_i$ is updated by Equation (7).

$$\begin{cases} X_i = X_i^{new}, & Fit(X_i^{new}) < Fit(X_i), \\ X_i = X_i, & Fit(X_i^{new}) \ge Fit(X_i). \end{cases} \tag{7}$$

Figure 2 provides the flow chart of the original NGO algorithm.

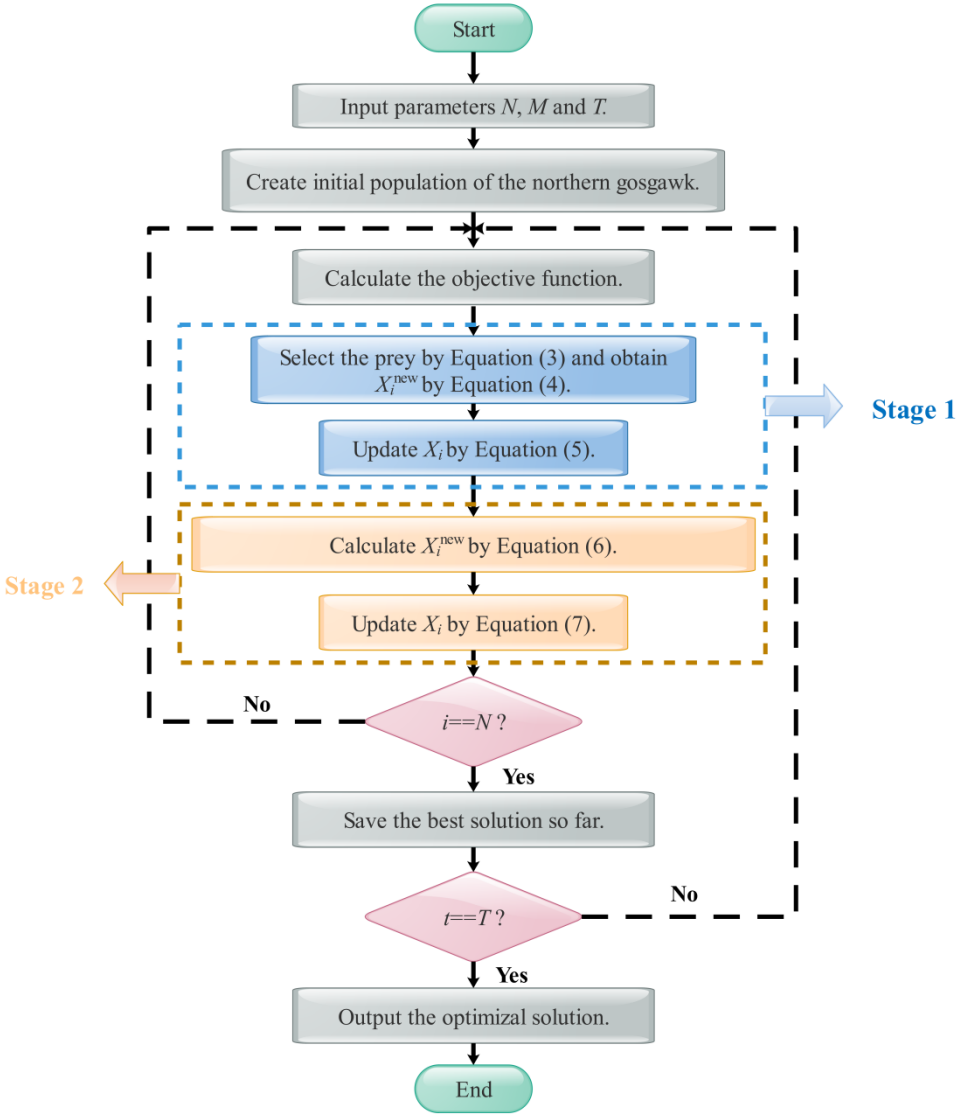


**Figure 2.** The flowchart of the NGO algorithm.

### *2.2. The Improved Northern Goshawk Optimization*

Although the NGO algorithm has the advantages of easy implementation, simple structure and high precision in solving test functions and some engineering optimization problems, there are still some opportunities to further enhance its capacities of exploration and exploitation. For example, the model of chasing prey in the original NGO algorithm is too simple, which will lead to the poor quality of the individuals and slow convergence speed to the optimal solution. Thus, this section proposes an enhanced northern goshawk optimization algorithm by introducing the polynomial interpolation strategy and the multi-strategy opposite learning models, noted as ENGO.

#### 2.2.1. Polynomial Interpolation Strategy

Polynomial interpolation is a kind of search method, which refers to seeking the minimal value of the interpolation polynomial $\varphi(t)$ constructed by some discrete data [39]. Firstly, the discrete points are selected to construct the interpolation polynomial $\varphi(t)$. If the constructed polynomial is quadratic, it is called the quadratic interpolation, and if it is cubic, it is cubic interpolation method. Then, the minimal value of the function $\varphi(t)$ can be solved by letting $\varphi'(t) = 0$.

Here, the northern goshawk population is regarded as the discrete data in the feasible space. Firstly, three individuals $X_i$, $X_{i+1}$ and $X_{i+2}$ of the northern goshawk population are selected to construct the quadratic interpolation function $\varphi(X)$, shown in Equation (8).

$$\varphi(X) = a_0 + a_1 X + a_2 X^2. \tag{8}$$

Submit the discrete individuals into Equation (8), we can obtain

$$\begin{cases} \varphi(X_i) = a_0 + a_1 X_i + a_2 X_i^2, \\ \varphi(X_{i+1}) = a_0 + a_1 X_{i+1} + a_2 X_{i+1}^2, \\ \varphi(X_{i+2}) = a_0 + a_1 X_{i+2} + a_2 X_{i+2}^2. \end{cases} \tag{9}$$

According to Equation (9), three coefficients $a_0$, $a_1$ and $a_2$ can be obtained by Equation (10).

$$\begin{cases} a_0 = -\frac{(X_{i+1}-X_{i+2})\varphi(X_i)+(X_{i+2}-X_i)\varphi(X_{i+1})+(X_i-X_{i+1})\varphi(X_{i+2})}{(X_i-X_{i+1})(X_{i+1}-X_{i+2})(X_{i+2}-X_i)}, \\ a_1 = \frac{(X_{i+1}^2-X_{i+2}^2)\varphi(X_i)+(X_{i+2}^2-X_i^2)\varphi(X_{i+1})+(X_i^2-X_{i+1}^2)\varphi(X_{i+2})}{(X_i-X_{i+1})(X_{i+1}-X_{i+2})(X_{i+2}-X_i)}, \\ a_2 = \frac{(X_{i+1}-X_{i+2})X_{i+1}X_{i+2}\varphi(X_i)+(X_{i+2}-X_i)X_{i+2}X_i\varphi(X_{i+1})+(X_i-X_{i+1})X_iX_{i+1}\varphi(X_{i+2})}{(X_i-X_{i+1})(X_{i+1}-X_{i+2})(X_{i+2}-X_i)}. \end{cases} \tag{10}$$

To obtain the minimal of quadratic curve $\varphi(X)$, let $\varphi'(X) = a_1 + 2a_2X = 0$. When $X^* = -\frac{a_1}{2a_2}$, the quadratic curve $\varphi(X)$ obtains the minimal value. Combined with Equation (10), the $X^*$ can be obtained as follows.

$$X^* = \frac{1}{2} \times \frac{(X_{i+1}^2 - X_{i+2}^2)Fit(X_i) + (X_{i+2}^2 - X_i^2)Fit(X_{i+1}) + (X_i^2 - X_{i+1}^2)Fit(X_{i+2})}{(X_{i+1} - X_{i+2})Fit(X_i) + (X_{i+2} - X_i)Fit(X_{i+1}) + (X_i - X_{i+1})Fit(X_{i+2})}. \tag{11}$$

Finally, comparing the obtained $X^*$ and the original $X_i$, the individual is updated, as Equation (12) shown.

$$\begin{cases} X_i = X^*, & Fit(X^*) < Fit(X_i), \\ X_i = X_i, & Fit(X^*) \geq Fit(X_i). \end{cases} \tag{12}$$

Following the performance of this operation on the whole population, the quality of the northern goshawk population will be improved, which is helpful to approach the optimal solution closer.

#### 2.2.2. Multi-Strategy Opposite Learning Method

Moreover, falling into local optimal solutions easily is also a dilemma for optimization algorithms, especially for some optimization problems with a large number of local optimums. Thus, the different kinds of opposite learning methods are introduced to help the NGO algorithm to enhance the capacity of the escaping local optimums. To make the learning modes of the individuals in the population more diverse, this paper adopts three kinds of learning mechanisms with their own characteristics, including opposite learning [40], quasi-opposite learning [41] and quasi-reflected learning [42].

For the $i$-th individual of the northern goshawk population, the newly generated individual, based on the different opposite learning mechanisms, can be calculated by the following equations.

$$\widetilde{X}_i = LB + UB - X_i. \tag{13}$$

$$\overline{X}_i = rand\left[\frac{LB + UB}{2}, LB + UB - X_i\right]. \tag{14}$$

$$\hat{X}_i = rand\left[\frac{LB + UB}{2}, X_i\right]. \tag{15}$$

Figure 3 illustrates the characteristics of the different opposite learning methods in the two-dimensional space. The red point is the $i$-th individual in the northern goshawk

population. Following the application of different learning methods to the $X_i$ in different dimensions, new individuals will be generated in the constructed areas A and B.

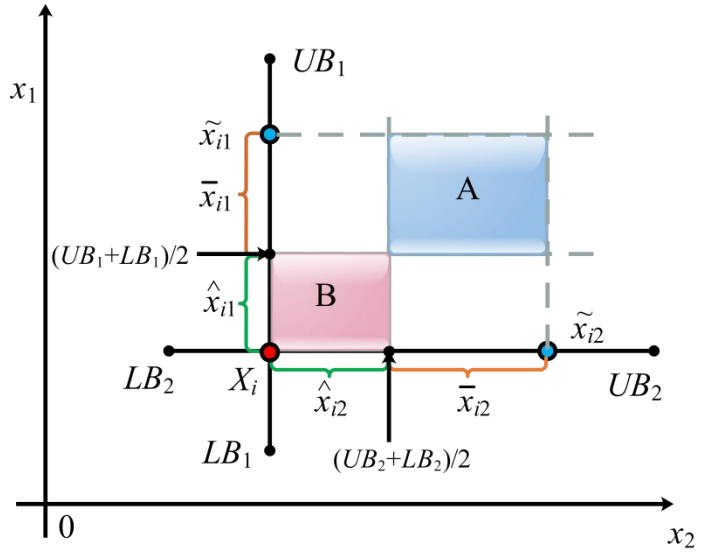


**Figure 3.** Schematic diagram of different learning methods in 2-D.

To widen the search range as much as possible, the above methods will be applied to the northern goshawk population. Suppose the size of the population $X$ is $N$, then the population will be divided into three groups randomly. Following the application of different methods to each group, a new population, based on opposite learning strategies, will be generated, noted as $X^{\text{oppo}}$. The two populations $X$ and $X^{\text{oppo}}$ are mixed and sorted, according to the fitness values, and $N$ individuals with better fitness values will be retained as the final population. Figure 4 shows the whole process of the learning methods for the population.

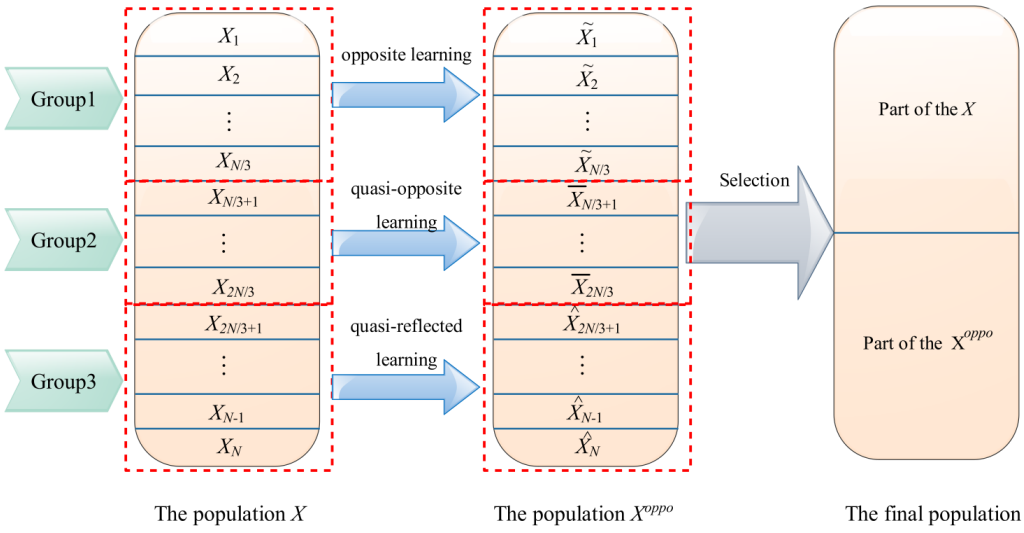


**Figure 4.** The process of opposite learning and selection.

Figure 5 shows the flowchart of the ENGO algorithm. The specific steps of the ENGO algorithm are as follows:

Step 1. Initialize the parameters, such as the size of the northern goshawk population $N$, the dimension of the problem $M$, and the maximum iteration time $T$.
Step 2. Create the initial northern goshawk population by Equation (2).

Step 3. when $t < T$, calculate the fitness value of each individual in the population. Following the selection of the according prey of the $i$-th individual $X_i$ by Equation (3), the updated solution can be obtained by Equations (4) and (5).
Step 4. Divide the population into three equal groups.
Step 5. Based on Equations (13)–(15), apply different opposite learning methods to the according group, and obtain new solutions.
Step 6. Mix the new solutions with the population, and $N$ better solutions are selected as the new population.
Step 7. By simulating the behavior of chasing the prey, calculate the new solution by Equation (6). Then, update the solution by Equation (7).
Step 8. Apply the polynomial interpolation strategy to each individual and update the individual by Equations (11) and (12).
Step 9. if $t < T$, return to Step 3. Otherwise, output the best individual and its fitness value.

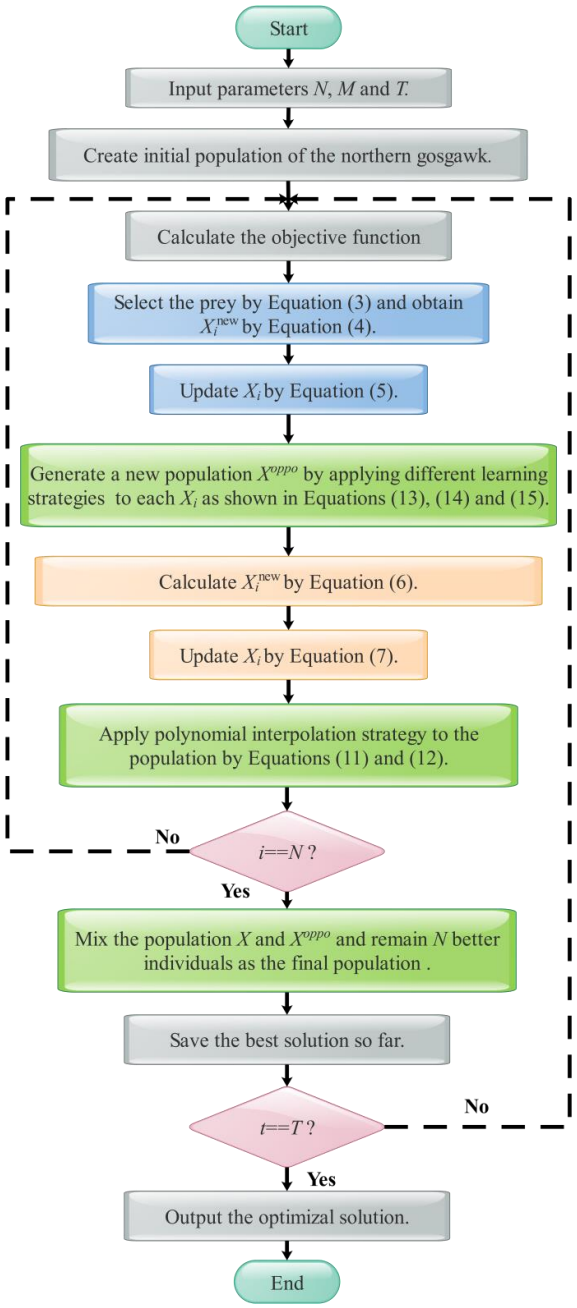


**Figure 5.** The flowchart of the ENGO algorithm.

## 3. Numerical Experiment on the Test Functions

### *3.1. Test Functions and Parameters Setting*

To test the performance of the improved algorithm, the CEC2017 test suite is used, which consists of 29 test functions (except for the F2), including uni-model functions, multi-modal functions, hybrid functions and composition functions [43]. These functions are constructed by rotating, combining or scaling benchmark functions, and have been used to test the ability of the optimization algorithms to search a complex space. Then, the original NGO algorithm and other six well-known algorithms are selected for comparison. They are the particle swarm optimization algorithm (PSO), the Archimedes optimization algorithm (AOA), the African vultures optimization algorithm (AVOA), the dandelion optimizer (DO), the sand cat optimization algorithm (SCSO) and the sparrow search algorithm (SSA). In Table 1, the time and parameters of these compared algorithms are shown. In the comparison experiments, the size of the population of all algorithms is set as 100, and the maximum iteration number is 500 times. The dimension of the test functions is set as 30. Meanwhile, to compare the different algorithms more fairly, all algorithms run 20 times, independently, to avoid random distractions. Once completed, the average value (Ave) and the standard deviation (Std) are recorded to show the accuracy and stability of the algorithm. All of the experimental series were implemented using Windows 10(64-bit), Intel(R) Core(TM) i7-1165G7 (Santa Clara, CA, USA), CPU @2.80GHz (Santa Clara, CA, USA), MATLAB 2016a (Natick, MA, USA) with 8 GB of ARM.

**Table 1.** Parameters of the algorithms.

| Algorithms | Proposed Time | Parameters |
|---|---|---|
| PSO [44] | 1988 | Learning factors $c_1 = 2$, $c_2 = 2.5$, the inertia factor $w = 2$. |
| AOA [45] | 2021 | Control parameter $\mu = 0.499$, Sensitive parameter $a = 5$. |
| AVOA [46] | 2021 | Hunger degrees $z$ is a random value in $[-1, 1]$. |
| DO [47] | 2022 | Parameters $\beta = 1.5$, $s = 0.01$. |
| SCSO [28] | 2022 | The value of hearing characteristics $S_M = 2$. |
| SSA [48] | 2017 | The update probability of the leader $c_3 = 0.5$. |

### *3.2. Results Analysis and Discussion*

The results of the different algorithms in solving the functions of the CEC2017 test suite are displayed in Table 2. The data in bold represents the best results among all algorithms. The results show that the enhanced NGO algorithm ranks first on 23 test functions, accounting for 79.31% of all functions. However, the original NGO algorithm only ranks first on four functions, accounting for 13.79% of all functions. Obviously, after introducing the polynomial interpolation strategy and the multi-strategy opposite learning method into the original NGO algorithm, the solution accuracy has been greatly improved. When compared with other algorithms, the average rank of the ENGO algorithm is 1.3448, which is less than the others. According to the final rank, the performance of these algorithms is ranked as ENGO > NGO > DO > SSA > SCSO > PSO > AOA.

**Table 2.** Results of the different algorithms on the CEC2017 functions.

| Function | Index | PSO | AOA | AVOA | DO | SCSO | SSA | NGO | ENGO |
|---|---|---|---|---|---|---|---|---|---|
| F1 | Ave | 5.1802E + 09 | 4.0462E + 10 | 1.1221E + 05 | 1.8277E + 05 | 3.3426E + 09 | 5.5010E + 03 | 2.0690E + 05 | 2.3466E + 03 |
| | Std | 1.3161E + 09 | 3.5061E + 09 | 4.8240E + 05 | 1.3250E + 05 | 2.2407E + 09 | 3.6044E + 03 | 1.3089E + 05 | 1.8986E + 03 |
| | Rank | 7 | 8 | 3 | 4 | 6 | 2 | 5 | 1 |
| F3 | Ave | 7.1287E + 04 | 3.9949E + 04 | 3.0323E + 04 | 2.1871E + 03 | 4.1388E + 04 | 4.7198E + 04 | 5.8724E + 04 | 4.5869E + 04 |
| | Std | 1.3573E + 04 | 7.5639E + 03 | 9.2550E + 03 | 1.5687E + 03 | 8.6106E + 03 | 5.6481E + 03 | 5.3424E + 03 | 5.4097E + 03 |
| | Rank | 8 | 3 | 2 | 1 | 4 | 6 | 7 | 5 |
| F4 | Ave | 9.1671E + 02 | 1.0676E + 04 | 5.2458E + 02 | 5.0657E + 02 | 6.6922E + 02 | 5.0147E + 02 | 5.0824E + 02 | 4.9688E + 02 |
| | Std | 1.3168E + 02 | 1.2735E + 03 | 3.0283E + 01 | 2.7533E + 01 | 1.0356E + 02 | 3.2163E + 01 | 1.6062E + 01 | 2.8701E + 01 |
| | Rank | 7 | 8 | 5 | 3 | 6 | 2 | 4 | 1 |
| F5 | Ave | 7.3096E + 02 | 8.3088E + 02 | 6.9320E + 02 | 6.7696E + 02 | 7.3262E + 02 | 7.4290E + 02 | 6.5192E + 02 | 6.1849E + 02 |
| | Std | 2.3716E + 01 | 2.0645E + 01 | 3.8796E + 01 | 3.0970E + 01 | 4.2318E + 01 | 5.1374E + 01 | 2.6417E + 01 | 2.1285E + 01 |
| | Rank | 5 | 8 | 4 | 3 | 6 | 7 | 2 | 1 |

**Table 2.** *Cont.*

| Function | Index | PSO | AOA | AVOA | DO | SCSO | SSA | NGO | ENGO |
|---|---|---|---|---|---|---|---|---|---|
| F6 | Ave | 6.2781E + 02 | 6.7466E + 02 | 6.4592E + 02 | 6.4002E + 02 | 6.5735E + 02 | 6.4691E + 02 | 6.0692E + 02 | 6.0030E + 02 |
| | Std | 4.0829E + 00 | 4.6522E + 00 | 5.7921E + 00 | 1.2329E + 01 | 1.0318E + 01 | 8.0761E + 00 | 5.0220E + 00 | 4.5294E − 01 |
| | Rank | 3 | 8 | 5 | 4 | 7 | 6 | 2 | 1 |
| F7 | Ave | 1.1906E + 03 | 1.2735E + 03 | 1.1822E + 03 | 1.0234E + 03 | 1.1217E + 03 | 1.2322E + 03 | 9.1927E + 02 | 8.9333E + 02 |
| | Std | 7.9530E + 01 | 4.7253E + 01 | 8.5067E + 01 | 7.0898E + 01 | 9.2206E + 01 | 7.6023E + 01 | 3.4636E + 01 | 1.9863E + 01 |
| | Rank | 6 | 8 | 5 | 3 | 4 | 7 | 2 | 1 |
| F8 | Ave | 1.0399E + 03 | 1.0744E + 03 | 9.4314E + 02 | 9.3114E + 02 | 9.9586E + 02 | 9.7208E + 02 | 9.3873E + 02 | 9.1502E + 02 |
| | Std | 2.0781E + 01 | 1.3098E + 01 | 2.3945E + 01 | 3.4521E + 01 | 3.5091E + 01 | 3.1781E + 01 | 1.4437E + 01 | 1.9785E + 01 |
| | Rank | 7 | 8 | 4 | 2 | 6 | 5 | 3 | 1 |
| F9 | Ave | 3.9823E + 03 | 7.4560E + 03 | 4.8178E + 03 | 4.9962E + 03 | 5.7009E + 03 | 5.3918E + 03 | 2.9437E + 03 | 1.9618E + 03 |
| | Std | 8.1851E + 02 | 7.3024E + 02 | 8.2016E + 02 | 1.9850E + 03 | 9.5008E + 02 | 2.2118E + 02 | 4.5211E + 02 | 5.7069E + 02 |
| | Rank | 3 | 8 | 4 | 5 | 7 | 6 | 2 | 1 |
| F10 | Ave | 8.2352E + 03 | 8.0569E + 03 | 5.7230E + 03 | 4.9592E + 03 | 5.6354E + 03 | 5.3943E + 03 | 5.4102E + 03 | 5.3437E + 03 |
| | Std | 3.2222E + 02 | 4.0120E + 02 | 8.9928E + 02 | 4.9182E + 02 | 8.5290E + 02 | 4.9178E + 02 | 2.5354E + 02 | 3.8350E + 02 |
| | Rank | 8 | 7 | 6 | 1 | 5 | 3 | 4 | 2 |
| F11 | Ave | 2.0359E + 03 | 4.0694E + 03 | 1.2502E + 03 | 1.2517E + 03 | 1.8404E + 03 | 1.2747E + 03 | 1.2135E + 03 | 1.1957E + 03 |
| | Std | 2.8771E + 02 | 8.8072E + 02 | 5.1866E + 01 | 4.8974E + 01 | 6.1253E + 02 | 4.3691E + 01 | 3.4790E + 01 | 2.8866E + 01 |
| | Rank | 7 | 8 | 3 | 4 | 6 | 5 | 2 | 1 |
| F12 | Ave | 2.5414E + 08 | 9.4131E + 09 | 6.7289E + 06 | 6.6982E + 06 | 3.2798E + 07 | 1.5966E + 06 | 7.3235E + 05 | 6.3577E + 05 |
| | Std | 3.5963E + 07 | 9.5712E + 08 | 6.2126E + 06 | 2.1447E + 06 | 2.4691E + 07 | 1.1237E + 06 | 2.1315E + 05 | 2.3050E + 05 |
| | Rank | 7 | 8 | 5 | 4 | 6 | 3 | 2 | 1 |
| F13 | Ave | 3.1682E + 07 | 7.2400E + 09 | 1.2733E + 05 | 1.2456E + 05 | 1.3647E + 05 | 2.1858E + 04 | 1.0714E + 04 | 2.5724E + 04 |
| | Std | 1.9843E + 07 | 3.5318E + 09 | 5.4956E + 04 | 1.0581E + 05 | 1.9868E + 04 | 1.2436E + 04 | 8.5012E + 03 | 1.6919E + 04 |
| | Rank | 7 | 8 | 5 | 4 | 6 | 2 | 1 | 3 |
| F14 | Ave | 1.1780E + 05 | 6.2017E + 05 | 1.8606E + 05 | 5.7101E + 04 | 3.0132E + 05 | 5.1864E + 04 | 8.2371E + 03 | 4.6760E + 03 |
| | Std | 8.0871E + 04 | 4.6962E + 05 | 2.2180E + 05 | 2.6573E + 04 | 4.7135E + 05 | 4.0322E + 04 | 4.6052E + 03 | 2.5158E + 03 |
| | Rank | 5 | 8 | 6 | 4 | 7 | 3 | 2 | 1 |
| F15 | Ave | 1.8054E + 06 | 8.5030E + 07 | 3.4814E + 04 | 3.3985E + 04 | 1.7336E + 05 | 1.1045E + 04 | 4.6995E + 03 | 2.4524E + 03 |
| | Std | 2.3121E + 06 | 6.6009E + 07 | 1.8818E + 04 | 2.2745E + 04 | 4.3157E + 05 | 1.2212E + 04 | 2.5192E + 03 | 6.9164E + 02 |
| | Rank | 7 | 8 | 5 | 4 | 6 | 3 | 2 | 1 |
| F16 | Ave | 3.3186E + 03 | 4.9326E + 03 | 3.1043E + 03 | 2.6516E + 03 | 3.1768E + 03 | 2.8377E + 03 | 2.5559E + 03 | 2.4806E + 03 |
| | Std | 2.7582E + 02 | 4.0993E + 02 | 3.0041E + 02 | 2.5849E + 02 | 3.3518E + 02 | 4.1478E + 02 | 1.9722E + 02 | 1.5468E + 02 |
| | Rank | 7 | 8 | 5 | 3 | 6 | 4 | 2 | 1 |
| F17 | Ave | 2.3003E + 03 | 3.1673E + 03 | 2.5071E + 03 | 2.2951E + 03 | 2.3644E + 03 | 2.4993E + 03 | 1.9907E + 03 | 1.9748E + 03 |
| | Std | 2.1638E + 02 | 1.8004E + 02 | 2.0256E + 02 | 2.0889E + 02 | 1.9855E + 02 | 2.1525E + 02 | 7.6050E + 01 | 6.6039E + 01 |
| | Rank | 4 | 8 | 7 | 3 | 5 | 6 | 2 | 1 |
| F18 | Ave | 2.6966E + 06 | 5.6400E + 06 | 8.7903E + 05 | 5.9533E + 05 | 1.7970E + 06 | 5.9365E + 05 | 9.6472E + 04 | 9.4033E + 04 |
| | Std | 1.6107E + 06 | 6.7615E + 06 | 9.5620E + 05 | 3.3879E + 05 | 2.1560E + 06 | 5.1722E + 05 | 4.6634E + 04 | 5.4539E + 04 |
| | Rank | 7 | 8 | 5 | 4 | 6 | 3 | 2 | 1 |
| F19 | Ave | 2.1838E + 06 | 1.6601E + 08 | 3.3801E + 04 | 5.9278E + 04 | 2.0900E + 06 | 1.0631E + 04 | 5.3849E + 03 | 5.0245E + 03 |
| | Std | 2.2968E + 06 | 1.1869E + 08 | 1.9333E + 04 | 4.8092E + 04 | 2.5655E + 06 | 9.8222E + 03 | 1.3140E + 03 | 1.4903E + 03 |
| | Rank | 7 | 8 | 4 | 5 | 6 | 3 | 2 | 1 |
| F20 | Ave | 2.6074E + 03 | 2.7081E + 03 | 2.6796E + 03 | 2.5232E + 03 | 2.6236E + 03 | 2.7266E + 03 | 2.3560E + 03 | 2.3617E + 03 |
| | Std | 1.9460E + 02 | 1.0941E + 02 | 1.9055E + 02 | 1.9280E + 02 | 2.1442E + 02 | 2.1721E + 02 | 7.4857E + 01 | 4.2260E + 01 |
| | Rank | 4 | 7 | 6 | 3 | 5 | 8 | 1 | 2 |
| F21 | Ave | 2.5259E + 03 | 2.6337E + 03 | 2.4945E + 03 | 2.4552E + 03 | 2.4992E + 03 | 2.4968E + 03 | 2.4270E + 03 | 2.4084E + 03 |
| | Std | 1.8736E + 01 | 2.0020E + 01 | 5.4731E + 01 | 3.3110E + 01 | 4.3535E + 01 | 5.5648E + 01 | 1.2745E + 01 | 1.3815E + 01 |
| | Rank | 7 | 8 | 4 | 3 | 6 | 5 | 2 | 1 |
| F22 | Ave | 6.4159E + 03 | 6.4679E + 03 | 4.8378E + 03 | 5.5543E + 03 | 3.4848E + 03 | 5.7974E + 03 | 2.3034E + 03 | 2.3005E + 03 |
| | Std | 3.2348E + 03 | 7.8528E + 02 | 2.3835E + 03 | 2.0029E + 03 | 1.4360E + 03 | 2.1210E + 03 | 1.9491E + 00 | 1.2498E + 00 |
| | Rank | 7 | 8 | 4 | 5 | 3 | 6 | 2 | 1 |
| F23 | Ave | 2.8792E + 03 | 3.4845E + 03 | 2.9717E + 03 | 2.8672E + 03 | 2.9288E + 03 | 2.9679E + 03 | 2.7722E + 03 | 2.7413E + 03 |
| | Std | 2.1797E + 01 | 7.5564E + 01 | 8.0333E + 01 | 4.7990E + 01 | 5.9829E + 01 | 7.6021E + 01 | 2.1701E + 01 | 1.6470E + 01 |
| | Rank | 4 | 8 | 7 | 3 | 5 | 6 | 2 | 1 |
| F24 | Ave | 3.0411E + 03 | 3.8398E + 03 | 3.1300E + 03 | 3.0604E + 03 | 3.0968E + 03 | 3.0891E + 03 | 2.9197E + 03 | 2.8797E + 03 |
| | Std | 9.3852E + 00 | 9.8943E + 01 | 8.3199E + 01 | 6.9980E + 01 | 7.1139E + 01 | 6.9832E + 01 | 1.4223E + 01 | 1.2927E + 01 |
| | Rank | 3 | 8 | 7 | 4 | 6 | 5 | 2 | 1 |
| F25 | Ave | 3.2031E + 03 | 4.3262E + 03 | 2.9137E + 03 | 2.8939E + 03 | 3.0406E + 03 | 2.8956E + 03 | 2.9170E + 03 | 2.8886E + 03 |
| | Std | 6.4161E + 01 | 2.5324E + 02 | 2.4682E + 01 | 1.3634E + 01 | 5.1759E + 01 | 1.3847E + 01 | 1.7329E + 01 | 6.2293E + 00 |
| | Rank | 7 | 8 | 4 | 2 | 6 | 3 | 5 | 1 |
| F26 | Ave | 6.1568E + 03 | 9.9657E + 03 | 6.6772E + 03 | 5.7903E + 03 | 6.1648E + 03 | 6.3669E + 03 | 3.2578E + 03 | 3.5280E + 03 |
| | Std | 4.9151E + 02 | 3.4324E + 02 | 1.0058E + 03 | 4.4628E + 02 | 1.0908E + 03 | 1.5443E + 03 | 7.6200E + 02 | 1.2912E + 03 |
| | Rank | 4 | 8 | 7 | 3 | 5 | 6 | 1 | 2 |
| F27 | Ave | 3.3027E + 03 | 4.4396E + 03 | 3.2770E + 03 | 3.2687E + 03 | 3.3635E + 03 | 3.2875E + 03 | 3.2230E + 03 | 3.2184E + 03 |
| | Std | 1.7512E + 01 | 2.0391E + 02 | 3.0678E + 01 | 4.2722E + 01 | 8.8901E + 01 | 4.2029E + 01 | 9.6877E + 00 | 6.4926E + 00 |
| | Rank | 6 | 8 | 4 | 3 | 7 | 5 | 2 | 1 |
| F28 | Ave | 3.5240E + 03 | 6.3214E + 03 | 3.2690E + 03 | 3.2365E + 03 | 3.4941E + 03 | 3.2461E + 03 | 3.2798E + 03 | 3.2054E + 03 |
| | Std | 7.3227E + 01 | 3.5836E + 02 | 2.2702E + 01 | 2.6406E + 01 | 1.1313E + 02 | 2.2172E + 01 | 1.4775E + 01 | 6.6437E + 00 |
| | Rank | 7 | 8 | 4 | 2 | 6 | 3 | 5 | 1 |
| F29 | Ave | 4.3235E + 03 | 6.2964E + 03 | 4.2739E + 03 | 3.9694E + 03 | 4.4828E + 03 | 4.2342E + 03 | 3.8593E + 03 | 3.8632E + 03 |
| | Std | 1.9360E + 02 | 5.6231E + 02 | 2.8515E + 02 | 1.9885E + 02 | 3.4699E + 02 | 3.4541E + 02 | 9.4104E + 01 | 1.5308E + 02 |
| | Rank | 6 | 8 | 5 | 3 | 7 | 4 | 1 | 2 |
| F30 | Ave | 7.0571E + 06 | 8.7344E + 08 | 5.0128E + 05 | 6.2123E + 05 | 7.0001E + 06 | 2.0995E + 04 | 1.7041E + 04 | 6.8796E + 03 |
| | Std | 5.5024E + 06 | 2.6904E + 08 | 3.0621E + 05 | 4.2992E + 05 | 4.9031E + 06 | 7.5318E + 03 | 1.4933E + 04 | 9.3090E + 02 |
| | Rank | 7 | 8 | 4 | 5 | 6 | 3 | 2 | 1 |
| Average rank | | 6.0000 | 7.7586 | 4.7931 | 3.3448 | 5.7586 | 4.4828 | 2.5172 | 1.3448 |

The average convergence curves of the different algorithms after it was run 20 times, are shown in Figure 6. The accuracy and the convergence speed of the algorithm can be measured by the descending rate and the final position of its curve. Compared with the original NGO algorithm, the ENGO algorithm can converge to the optimum with a faster speed, especially in the early stages of the whole search. It is because the polynomial interpolation strategy improves the quality of the northern goshawk population by selecting better solutions around the three selected individuals randomly. Meanwhile, among all algorithms, the ENGO algorithm can obtain solutions with a better accuracy when the search ends.

Figure 7 displays the boxplots of the different algorithms, which are used to measure the stability and robustness of the algorithm. Comparing the length of the boxes, the ENGO is smaller than the original NGO algorithm and others on F6, F7, F11, F15, F17, F20, F22, F23, F24, F25, F27, F28 and F30, which illustrates that the distribution of the results of the ENGO algorithm is more centralized after it was run 20 times on these functions. For the remaining functions, the median of the data also needs to be considered. On functions F1, F5, F8, F9, F12, F14, F16, F18, F19, F21 and F26, the median of the ENGO is smaller than others. Thus, considering the length and median of the box, the improved algorithm has an advantage in stability and robustness, except for F3, F4, F10, F13 and F29. It is because the introduction of the two improvement strategies that keeps a better balance between the exploitation capacity and the exploration capacity.

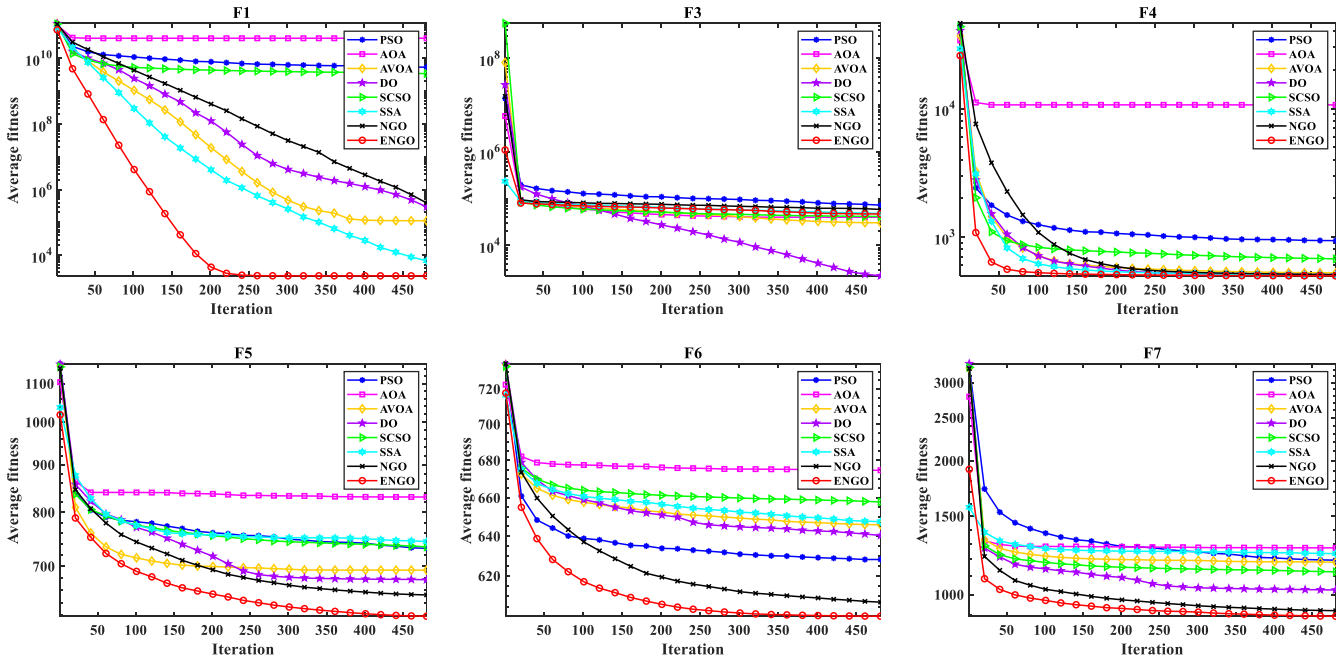


**Figure 6.** *Cont.*

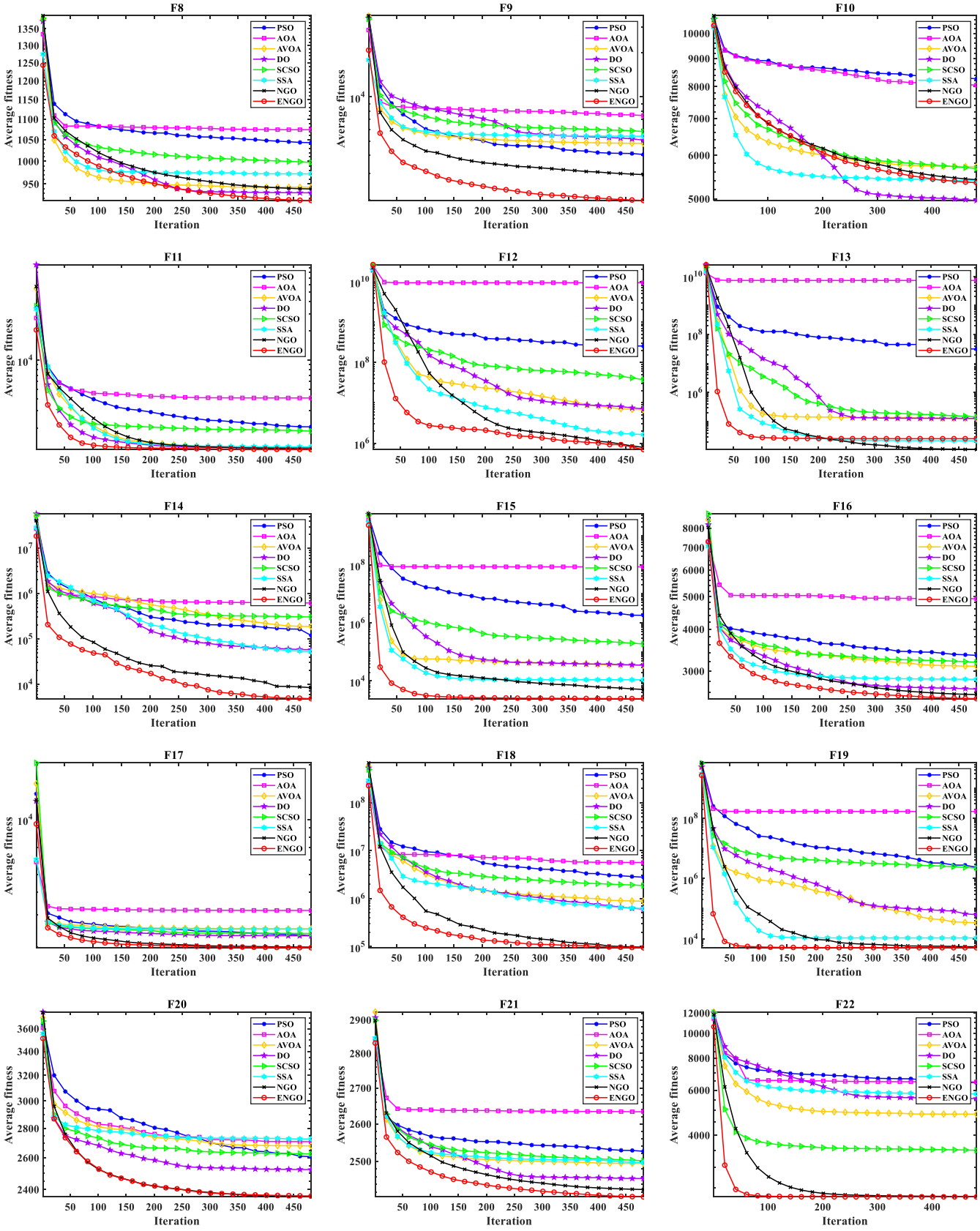


**Figure 6.** *Cont.*

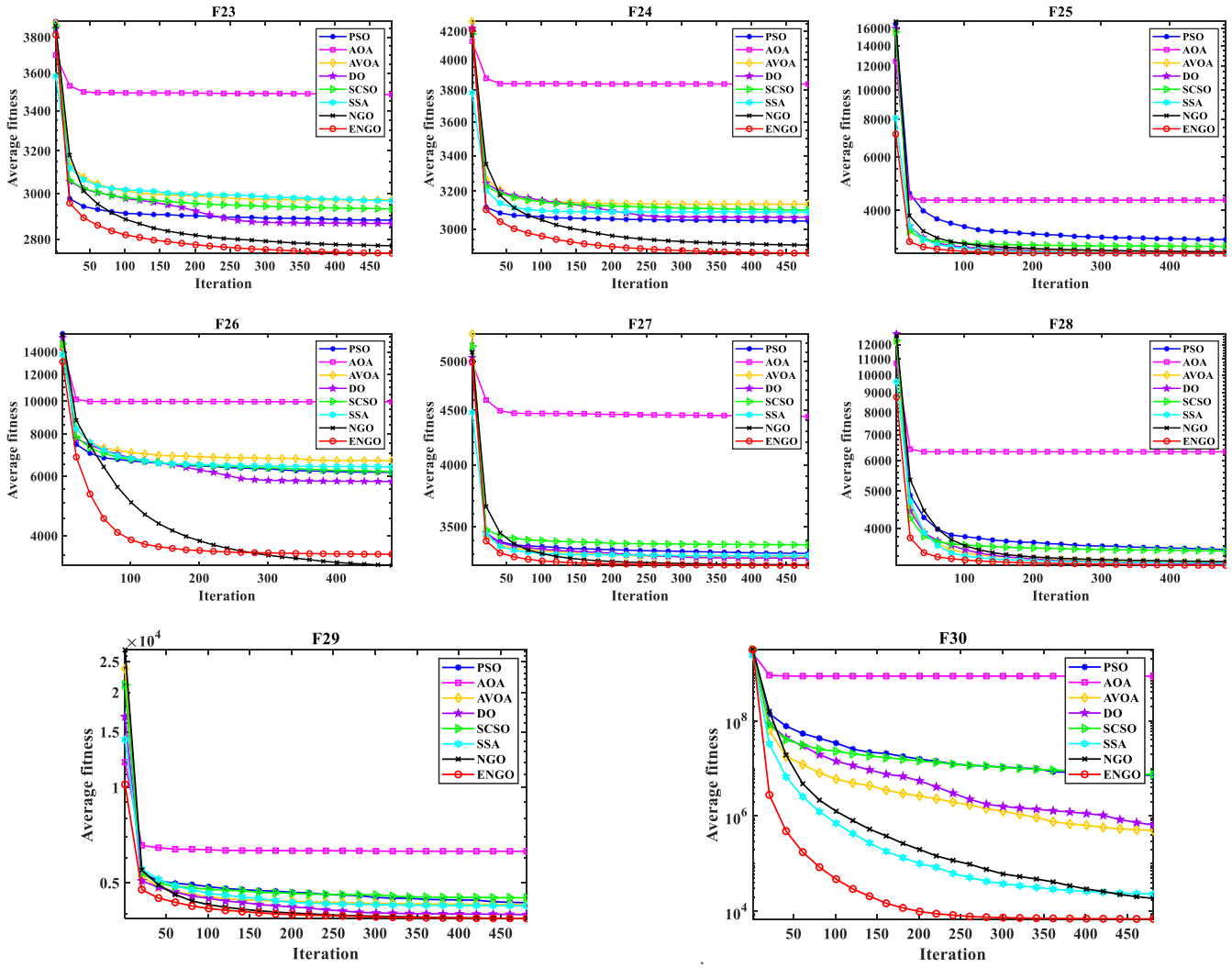


**Figure 6.** The average convergence curves of different algorithms on CEC2017.

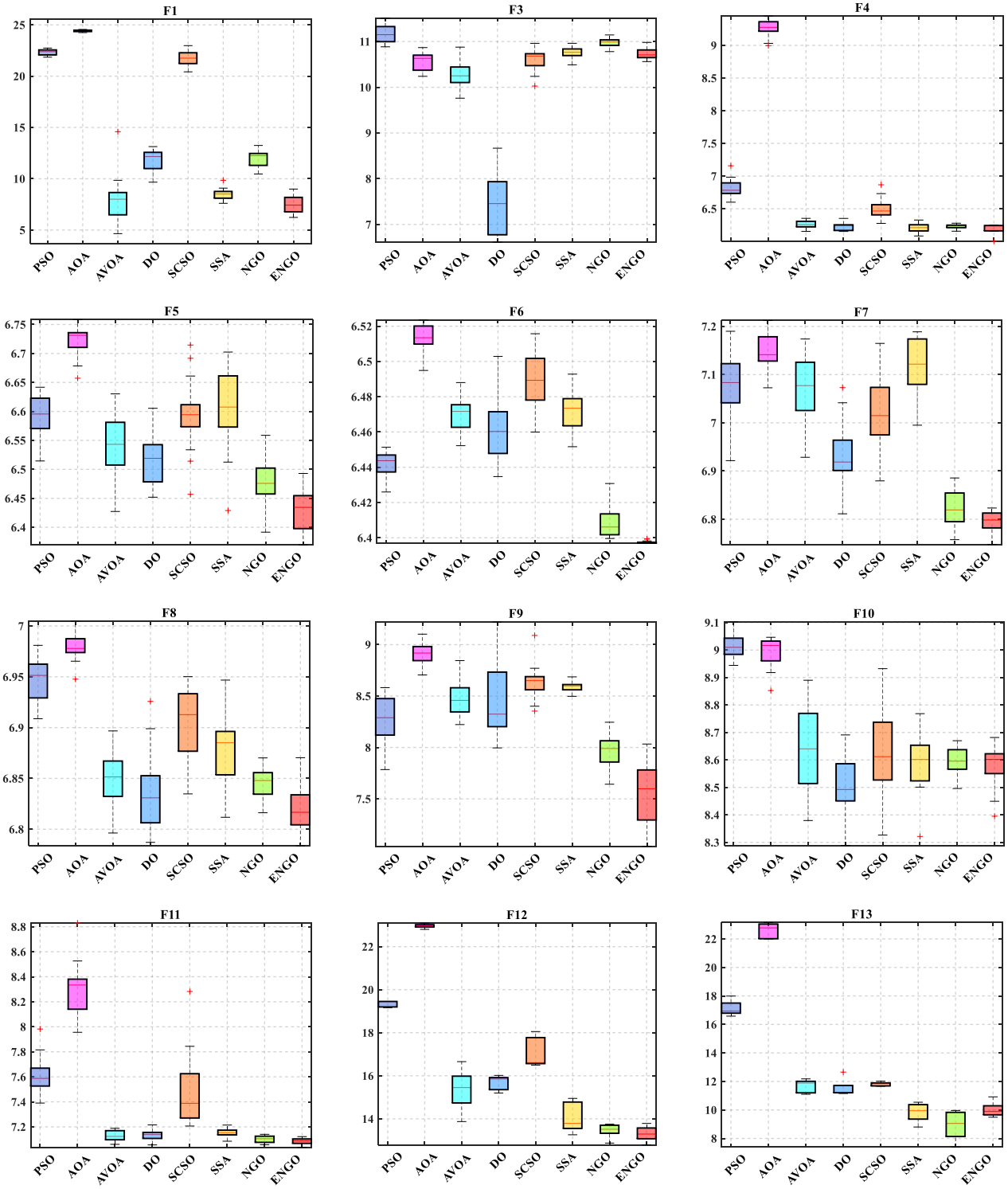


**Figure 7.** *Cont.*

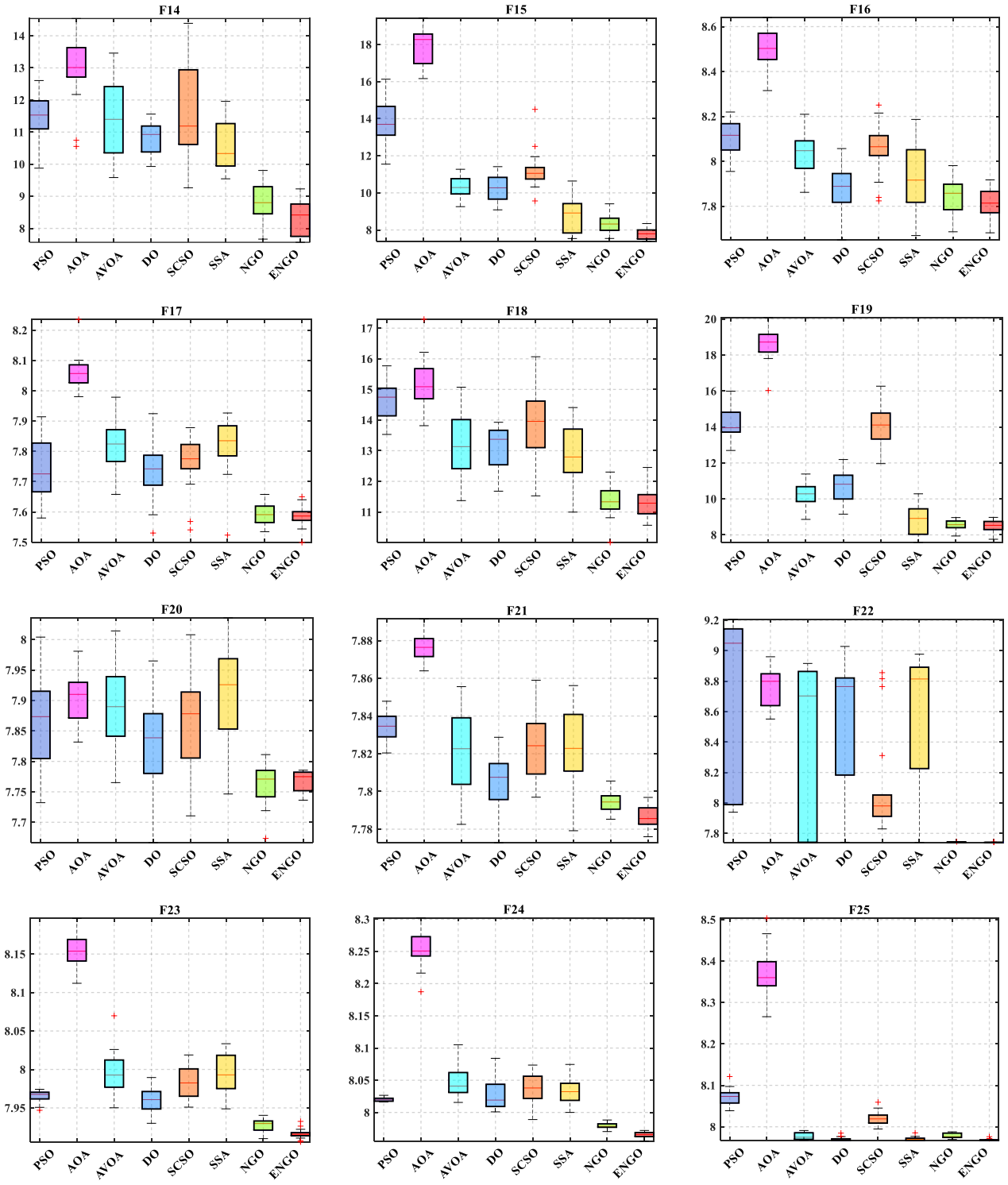


**Figure 7.** *Cont.*

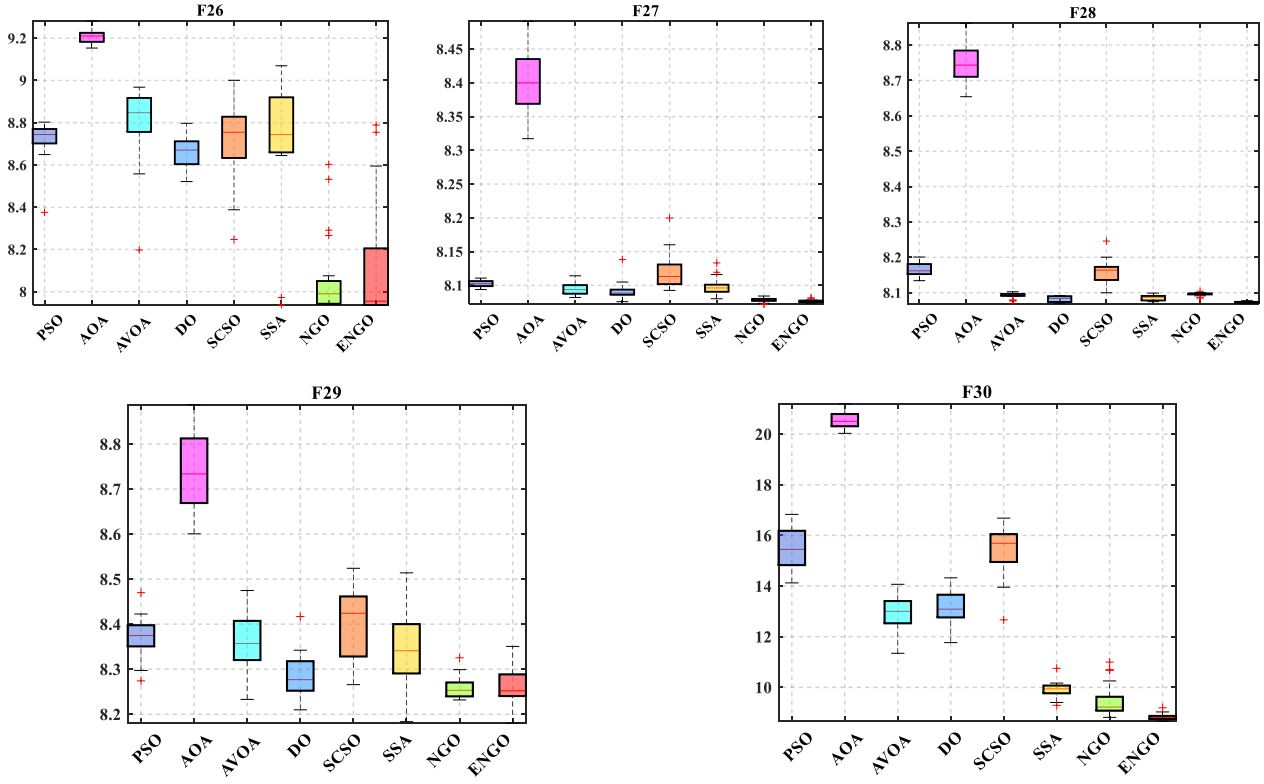


**Figure 7.** The boxplots of the different algorithms on CEC2017.

Figure 8 includes the radar maps of eight algorithms in solving the CEC2017 test suite. They are constructed by the ranks of different algorithms on each test function, which is used to reflect the comprehensive performance of the algorithm more intuitively. The radar graph of the ENGO algorithm has the minimum area, which also shows its outstanding performance on most functions. The area of the NGO algorithm is obviously bigger than the ENGO algorithm. The AOA algorithm has the biggest area, indicating that there are still opportunities to be improved. Table 3 is the *p*-value of the Wilcoxon rank sum test between the ENGO and other algorithms. Usually, when the *p*-value is smaller than 0.05, there is an obvious difference between the two groups of test data. Thus, combined with the data in Table 2, '+' represents the ENGO better than the comparison algorithm, obviously, while '−' represents the ENGO worse than the comparison algorithm, obviously. The '=' illustrates that there is no obvious difference between the two algorithms. From Table 3, the improved algorithm outperforms the original algorithm in more than half of the test functions with the support of statistics.

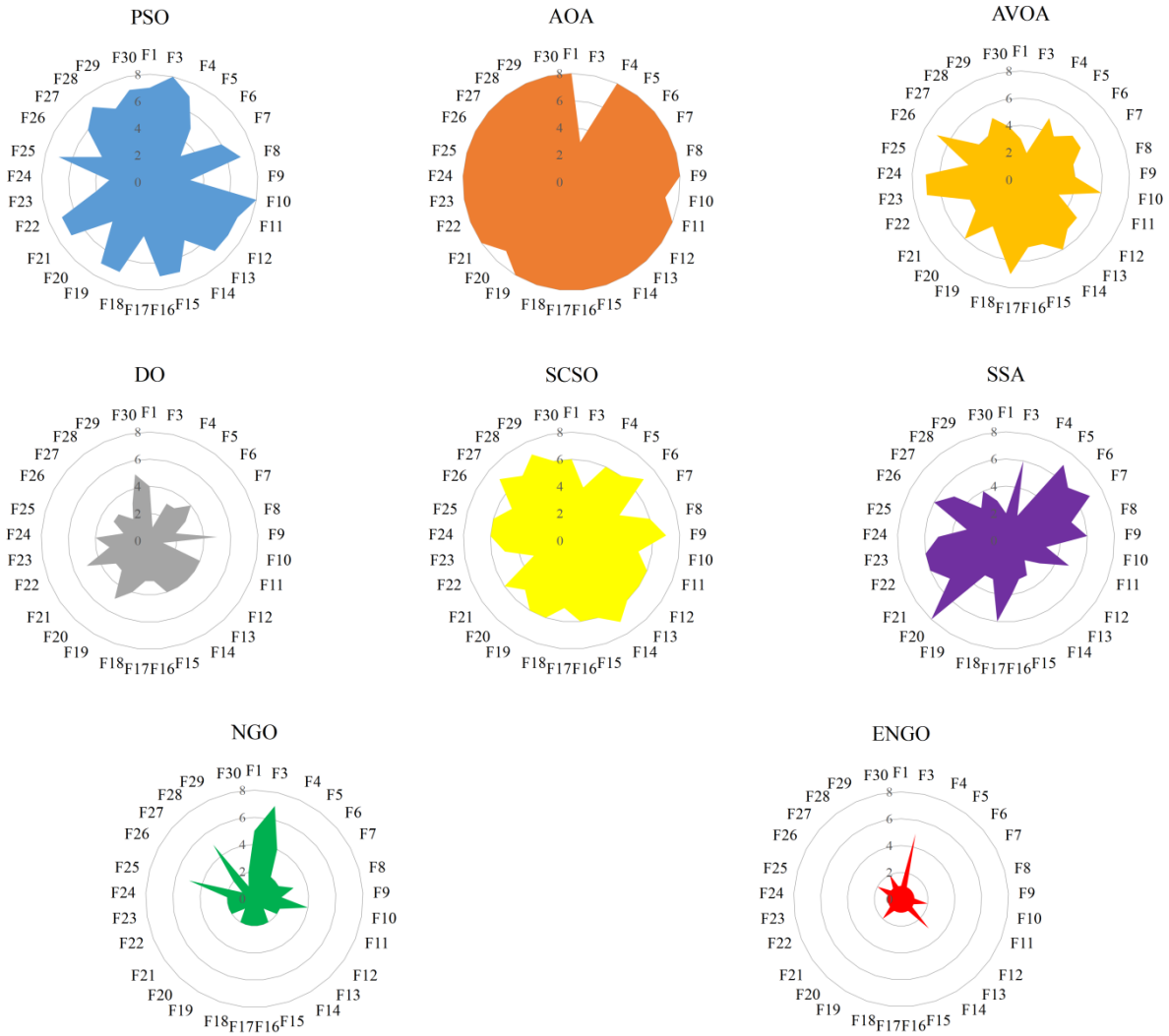


**Figure 8.** The radar maps of the different algorithms on CEC2017.

**Table 3.** The *p*-value of Wilcoxon rank sum test between the ENGO and other algorithms.

| | PSO | AOA | AVOA | DO | SCSO | SSA | NGO |
|---|---|---|---|---|---|---|---|
| F1 | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $5.255 \times 10^{-5}/+$ | $2.960 \times 10^{-7}/+$ | $6.796 \times 10^{-8}/+$ | $4.540 \times 10^{-6}/+$ | $6.796 \times 10^{-8}/+$ |
| F3 | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/-$ | $6.796 \times 10^{-8}/-$ | $5.979 \times 10^{-1}/-$ | $6.796 \times 10^{-8}/-$ | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ |
| F4 | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $5.979 \times 10^{-1}/=$ | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ |
| F5 | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $1.481 \times 10^{-3}/+$ | $2.184 \times 10^{-1}/=$ | $6.796 \times 10^{-8}/+$ | $8.817 \times 10^{-1}/=$ | $7.712 \times 10^{-3}/+$ |
| F6 | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $1.657 \times 10^{-7}/+$ | $1.431 \times 10^{-7}/+$ | $7.898 \times 10^{-8}/+$ | $1.235 \times 10^{-7}/+$ | $1.803 \times 10^{-6}/+$ |
| F7 | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $8.585 \times 10^{-2}/=$ |
| F8 | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $9.173 \times 10^{-8}/+$ |
| F9 | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $5.091 \times 10^{-4}/+$ | $2.073 \times 10^{-2}/+$ | $6.015 \times 10^{-7}/+$ | $3.987 \times 10^{-6}/+$ | $4.155 \times 10^{-4}/+$ |
| F10 | $1.953 \times 10^{-3}/+$ | $7.898 \times 10^{-8}/+$ | $2.041 \times 10^{-5}/+$ | $3.382 \times 10^{-4}/-$ | $1.376 \times 10^{-6}/+$ | $1.201 \times 10^{-6}/+$ | $3.942 \times 10^{-1}/=$ |
| F11 | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $8.597 \times 10^{-6}/+$ | $1.116 \times 10^{-3}/+$ | $9.748 \times 10^{-6}/+$ | $3.069 \times 10^{-6}/+$ | $1.047 \times 10^{-6}/+$ |
| F12 | $7.937 \times 10^{-8}/+$ | $7.937 \times 10^{-3}/+$ | $6.905 \times 10^{-1}/=$ | $8.413 \times 10^{-1}/=$ | $7.937 \times 10^{-3}/+$ | $1.000 \times 10^{-0}/=$ | $5.476 \times 10^{-1}/=$ |
| F13 | $7.937 \times 10^{-8}/+$ | $7.937 \times 10^{-3}/+$ | $2.222 \times 10^{-1}/=$ | $3.175 \times 10^{-2}/+$ | $7.937 \times 10^{-3}/+$ | $3.095 \times 10^{-1}/=$ | $9.524 \times 10^{-2}/-$ |
| F14 | $7.937 \times 10^{-8}/+$ | $7.937 \times 10^{-3}/+$ | $7.937 \times 10^{-3}/+$ | $7.937 \times 10^{-3}/+$ | $7.937 \times 10^{-3}/+$ | $8.413 \times 10^{-1}/=$ | $4.206 \times 10^{-1}/=$ |
| F15 | $4.903 \times 10^{-1}/=$ | $5.166 \times 10^{-6}/+$ | $8.392 \times 10^{-1}/=$ | $1.058 \times 10^{-2}/+$ | $9.246 \times 10^{-1}/=$ | $4.703 \times 10^{-3}/+$ | $1.065 \times 10^{-7}/+$ |
| F16 | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $1.143 \times 10^{-2}/+$ | $6.389 \times 10^{-2}/=$ | $2.062 \times 10^{-6}/+$ | $7.113 \times 10^{-3}/+$ | $4.540 \times 10^{-6}/+$ |
| F17 | $6.917 \times 10^{-7}/+$ | $6.796 \times 10^{-8}/+$ | $1.600 \times 10^{-5}/+$ | $2.616 \times 10^{-1}/=$ | $3.705 \times 10^{-5}/+$ | $5.310 \times 10^{-2}/=$ | $7.353 \times 10^{-1}/=$ |
| F18 | $6.168 \times 10^{-1}/=$ | $6.796 \times 10^{-8}/+$ | $6.220 \times 10^{-4}/+$ | $5.428 \times 10^{-1}/=$ | $7.205 \times 10^{-2}/=$ | $6.868 \times 10^{-4}/+$ | $1.014 \times 10^{-3}/+$ |
| F19 | $6.220 \times 10^{-4}/+$ | $3.069 \times 10^{-6}/+$ | $1.719 \times 10^{-1}/=$ | $5.652 \times 10^{-2}/=$ | $5.428 \times 10^{-1}/=$ | $2.073 \times 10^{-2}/+$ | $7.898 \times 10^{-8}/+$ |

**Table 3.** *Cont.*

| | PSO | AOA | AVOA | DO | SCSO | SSA | NGO |
|---|---|---|---|---|---|---|---|
| F20 | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $4.735 \times 10^{-1}/=$ | $1.929 \times 10^{-2}/+$ | $6.796 \times 10^{-8}/+$ | $2.341 \times 10^{-3}/+$ | $1.294 \times 10^{-4}/-$ |
| F21 | $6.040 \times 10^{-3}/+$ | $1.415 \times 10^{-5}/+$ | $4.155 \times 10^{-4}/+$ | $1.333 \times 10^{-1}/=$ | $8.355 \times 10^{-3}/+$ | $1.610 \times 10^{-4}/+$ | $2.393 \times 10^{-1}/=$ |
| F22 | $3.416 \times 10^{-7}/+$ | $6.796 \times 10^{-8}/+$ | $1.159 \times 10^{-4}/+$ | $7.712 \times 10^{-3}/+$ | $1.807 \times 10^{-5}/+$ | $1.794 \times 10^{-4}/+$ | $2.977 \times 10^{-1}/=$ |
| F23 | $5.979 \times 10^{-1}/=$ | $3.152 \times 10^{-2}/+$ | $8.604 \times 10^{-1}/=$ | $2.503 \times 10^{-1}/=$ | $6.610 \times 10^{-5}/+$ | $6.389 \times 10^{-2}/=$ | $6.796 \times 10^{-8}/+$ |
| F24 | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $1.657 \times 10^{-7}/+$ | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $7.205 \times 10^{-2}/=$ |
| F25 | $9.127 \times 10^{-7}/+$ | $6.796 \times 10^{-8}/+$ | $1.235 \times 10^{-7}/+$ | $4.539 \times 10^{-7}/+$ | $3.939 \times 10^{-7}/+$ | $1.918 \times 10^{-7}/+$ | $1.436 \times 10^{-2}/+$ |
| F26 | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $5.255 \times 10^{-5}/+$ | $1.556 \times 10^{-1}/=$ | $6.796 \times 10^{-8}/+$ | $6.787 \times 10^{-2}/=$ | $6.796 \times 10^{-8}/-$ |
| F27 | $1.047 \times 10^{-6}/+$ | $6.796 \times 10^{-8}/+$ | $2.356 \times 10^{-6}/+$ | $6.917 \times 10^{-7}/+$ | $1.997 \times 10^{-4}/+$ | $1.600 \times 10^{-5}/+$ | $8.597 \times 10^{-6}/+$ |
| F28 | $1.065 \times 10^{-7}/+$ | $6.796 \times 10^{-8}/+$ | $2.302 \times 10^{-5}/+$ | $5.091 \times 10^{-4}/+$ | $1.431 \times 10^{-7}/+$ | $6.674 \times 10^{-6}/+$ | $4.249 \times 10^{-1}/=$ |
| F29 | $6.796 \times 10^{-8}/+$ | $6.796 \times 10^{-8}/+$ | $9.620 \times 10^{-2}/=$ | $2.748 \times 10^{-2}/+$ | $1.065 \times 10^{-7}/+$ | $1.333 \times 10^{-1}/=$ | $2.561 \times 10^{-3}/-$ |
| F30 | $1.047 \times 10^{-6}/+$ | $6.796 \times 10^{-8}/+$ | $2.041 \times 10^{-5}/+$ | $2.748 \times 10^{-2}/+$ | $9.127 \times 10^{-7}/+$ | $1.794 \times 10^{-4}/+$ | $4.249 \times 10^{-1}/=$ |
| +/=/− | 26/3/0 | 28/0/1 | 21/7/1 | 16/11/2 | 25/3/1 | 21/8/0 | 15/10/4 |

## 4. Practical Optimization Problems

Except for the test functions, the practical optimization problems with high dimensions and strong constraints can also reflect the performance of the algorithm. In this section, the ENGO algorithm, the original NGO algorithm and the other eight algorithms need to solve five problems to show their performance. These algorithms are AOA, AO [23], the Harris hawks optimization (HHO) [49], the WOA, PSO, GWO [50], the manta ray foraging optimization (MRFO) [51] and the multi-verse optimizer (MVO) [52]. All algorithms run 20 times, independently. The size of the population is 30, the maximum iteration is 100. The results and discussion of these examples are as follows.

### *4.1. Gear Train Design Problem*

The gear train design problem is used to obtain the most suitable number of teeth to minimize the cost of the gear ratio. This problem includes four variables, denoted as $T_A$, $T_B$, $T_C$ and $T_D$, respectively [53]. The structure of the four gears is shown in Figure 9. Equation (16) lists its mathematical model.

$$\begin{aligned} \vec{x} &= [x_1, x_2, x_3, x_4] = [T_A, T_B, T_C, T_D], \\ f(\vec{x}) &= \left(\frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4}\right)^2, \\ 12 &\leq x_1, x_2, x_3, x_4 \leq 60. \end{aligned} \tag{16}$$

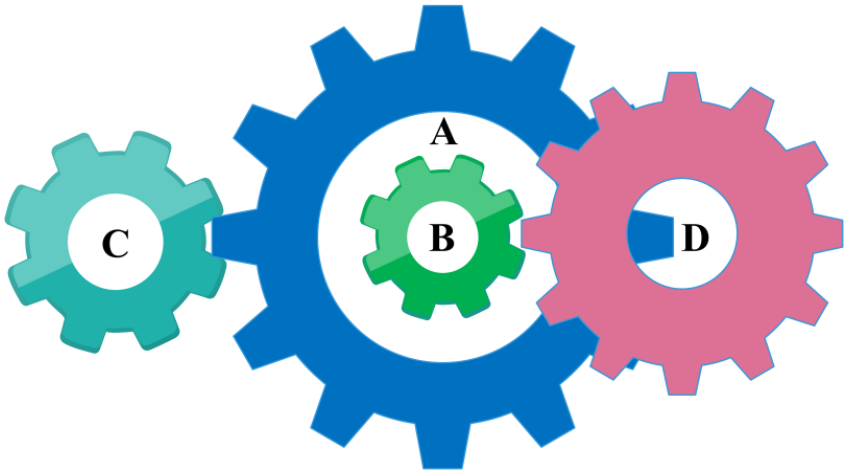


**Figure 9.** The construction of the gear train design. Here, A B C D represent the serial number of the gear.

Table 4 provides the final results of the ENGO algorithm and other algorithms in solving the gear train design problem. Among these algorithms, the ENGO algorithm has the best average value after it was run 20 times. That is, the ENGO algorithm is effective in solving this problem. Meanwhile, the smaller standard deviation illustrates that the improved algorithm is more stable, while the original NGO algorithm is largely fluctuating. Table 5 includes the best variables of the gear train obtained by each algorithm.

**Table 4.** Results of the different algorithms in solving the gear train design problem.

| Algorithms | The Best | The Average | The Worst | The Std | Rank |
|---|---|---|---|---|---|
| NGO | 6059.99 | 6074.13 | 6127.80 | 30.01 | 2 |
| ENGO | 6059.73 | 6066.12 | 6090.62 | 13.70 | 1 |
| AOA | 6071.98 | 6545.79 | 7222.18 | 444.38 | 8 |
| AO | 6113.00 | 6199.99 | 6410.86 | 120.66 | 4 |
| HHO | 6118.61 | 6795.99 | 7306.63 | 449.83 | 9 |
| WOA | 6570.10 | 7922.69 | 10,623.26 | 1569.47 | 10 |
| PSO | 6059.72 | 6276.06 | 6439.73 | 197.57 | 5 |
| GWO | 6069.89 | 6076.51 | 6091.00 | 9.24 | 3 |
| MRFO | 6060.57 | 6276.27 | 6410.09 | 183.54 | 6 |
| MVO | 6327.37 | 6518.43 | 6821.12 | 213.74 | 7 |

**Table 5.** The best solution of the gear train design problem obtained by the different algorithms.

| Algorithms | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| NGO | 0.7782 | 0.3847 | 40.3197 | 200.0000 |
| ENGO | 0.7782 | 0.3847 | 40.3201 | 200.0000 |
| AOA | 0.7788 | 0.3851 | 40.3452 | 199.6439 |
| AO | 0.7910 | 0.4106 | 40.8942 | 193.8638 |
| HHO | 0.8351 | 0.4132 | 43.2690 | 162.6520 |
| WOA | 0.8270 | 0.4523 | 42.2240 | 175.0869 |
| PSO | 0.7988 | 0.3948 | 41.3871 | 185.6578 |
| GWO | 0.7785 | 0.3850 | 40.3279 | 199.9139 |
| MRFO | 0.7799 | 0.3855 | 40.4039 | 198.8352 |
| MVO | 0.7850 | 0.3884 | 40.6637 | 195.3838 |

### *4.2. Pressure Vessel Design Problem*

The pressure vessel design problem aims to obtain the optimal total cost of the cylindrical vessel, by adjusting the parameter values of the material, its formation, and welding. The schematic diagram of the pressure vessel is shown in Figure 10. There are four parameter values that need to be optimized. $T_s$ represents the thickness of the shell, $T_h$ is the thickness of the head, and the inner radius and length of the cylindrical section are noted as $R$ and $L$ (without considering the head), respectively. The mathematical model of this problem is shown as follows [54,55]:

$$\vec{x} = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L],$$

$$\begin{aligned} f(\vec{x}) &= 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3, \\ g_1(\vec{x}) &= -x_1 + 0.0193x_3 \le 0, \\ g_2(\vec{x}) &= -x_3 + 0.00954x_3 \le 0, \\ g_3(\vec{x}) &= -\pi x_3^2 x_4 - \tfrac{4}{3}\pi x_3^3 + 1296000 \le 0, \\ g_4(\vec{x}) &= x_4 - 240 \le 0, \\ 0 \le x_1 \le 99,&\ \ 0 \le x_2 \le 99,\ \ 0 \le x_3 \le 200,\ \ 0 \le x_4 \le 200. \end{aligned} \tag{17}$$

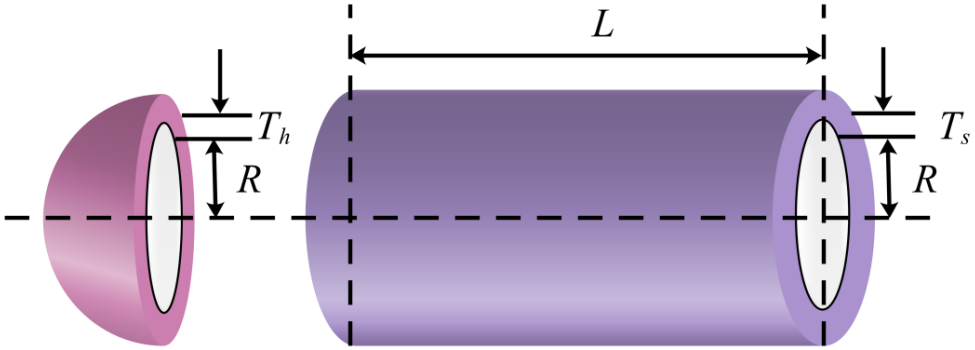


**Figure 10.** The construction of the pressure vessel design problem.

Table 6 shows the results of solving the pressure vessel design problem. The ENGO algorithm also has the best performance of accuracy and stability. The original NGO algorithm ranks second. Though the GWO algorithm also shows a great performance on the measure index of the best value, there are still defects in the stability. In Table 7, the best parameters of the pressure vessel design problem are listed.

**Table 6.** Results of the different algorithms in solving the pressure vessel design problem.

| Algorithms | The Best | The Average | The Worst | The Std | Rank |
|---|---|---|---|---|---|
| NGO | 5885.40 | 5891.23 | 5912.01 | 7.63 | 2 |
| ENGO | 5885.53 | 5889.23 | 5903.66 | 4.43 | 1 |
| AOA | 5888.21 | 6177.71 | 8486.53 | 579.23 | 4 |
| AO | 6015.94 | 6827.33 | 7996.55 | 649.54 | 9 |
| HHO | 5991.42 | 6529.04 | 7347.30 | 389.76 | 8 |
| WOA | 6191.17 | 8298.09 | 13,900.72 | 1763.29 | 10 |
| PSO | 5921.55 | 6192.62 | 6694.52 | 248.26 | 5 |
| GWO | 5885.32 | 6415.77 | 7318.93 | 543.48 | 7 |
| MRFO | 5889.04 | 5899.86 | 5927.37 | 11.37 | 3 |
| MVO | 5902.31 | 6376.45 | 7181.97 | 368.37 | 6 |

**Table 7.** The best solution of the pressure vessel design problem obtained by the different algorithms.

| Algorithms | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| NGO | 13.1101 | 6.7661 | 42.1004 | 176.6150 |
| ENGO | 13.3703 | 7.0200 | 42.0984 | 176.6376 |
| AOA | 12.0123 | 5.9525 | 40.3278 | 199.9302 |
| AO | 12.2204 | 6.1872 | 40.5586 | 196.7161 |
| HHO | 12.7969 | 5.5343 | 41.3533 | 186.0939 |
| WOA | 13.8206 | 6.6719 | 41.5674 | 183.3307 |
| PSO | 13.3587 | 6.7702 | 42.0985 | 176.6364 |
| GWO | 11.7017 | 6.0078 | 40.3203 | 200.0000 |
| MRFO | 13.1352 | 6.8732 | 42.0915 | 176.7231 |
| MVO | 14.9302 | 7.0317 | 48.6240 | 109.7093 |

### *4.3. Four-Stage Gearbox Design Problem*

Figure 11 is the schematic diagram of the four-stage gearbox design problem, which is proposed to reduce the weight of the gearbox. There are twenty-two parameters to be determined, which can be divided into four categories, including the positions of the gears, the positions of the pinions, the thickness of the blanks, and the number of teeth. Moreover, there are eighty-six constraints that need to be satisfied. They cover the contact ratio, pitch, strength of the gears, assembly of the gears, kinematics, and the size of the gears. Thus, the four-stage gearbox design problem is challenging for most algorithms to search for the optimum in a small search area with lots of local solutions. The model can be represented by the following [56].

$$\vec{x} = [x_1, x_2, x_3, \ldots, x_{22}] = [N_{p1}, \ldots, N_{p4}, N_{g1}, \ldots, N_{g4}, b_1 \ldots, b_4, x_{p1}, x_{g1}, \ldots, x_{g4}, y_{p1}, y_{g1}, \ldots, y_{g4}],$$

$$f(\vec{x}) = \left(\frac{\pi}{1000}\right)\sum_{i=1}^{4} \frac{b_i c_i^2 (N_{pi}^2 + N_{gi}^2)}{(N_{pi} + N_{gi})^2}, \tag{18}$$

$$g_1(x) = \left(\frac{366000}{\pi w_1} + \frac{2c_1 N_{p1}}{N_{p1} + N_{g1}}\right)\left(\frac{(N_{p1} + N_{g1})^2}{4b_1 c_1^2 N_{p1}}\right) - \frac{\sigma_N J_R}{0.0167 W K_o K_m} \leq 0,$$

$$g_2(x) = \left(\frac{366000 N_{g1}}{\pi w_1 N_{p1}} + \frac{2c_2 N_{p2}}{N_{p2} + N_{g2}}\right)\left(\frac{(N_{p2} + N_{g2})^2}{4b_2 c_2^2 N_{p2}}\right) - \frac{\sigma_N J_R}{0.0167 W K_o K_m} \leq 0,$$

$$g_3(x) = \left(\frac{366000 N_{g1} N_{g2}}{\pi w_1 N_{p1} N_{p2}} + \frac{2c_3 N_{p3}}{N_{p3} + N_{g3}}\right)\left(\frac{(N_{p3} + N_{g3})^2}{4b_3 c_3^2 N_{p3}}\right) - \frac{\sigma_N J_R}{0.0167 W K_o K_m} \leq 0,$$

$$g_4(x) = \left(\frac{366000 N_{g1} N_{g2} N_{g3}}{\pi w_1 N_{p1} N_{p2} N_{p3}} + \frac{2c_4 N_{p4}}{N_{p4} + N_{g4}}\right)\left(\frac{(N_{p4} + N_{g4})^2}{4b_4 c_4^2 N_{p4}}\right) - \frac{\sigma_N J_R}{0.0167 W K_o K_m} \leq 0,$$

$$g_5(x) = \left(\frac{366000}{\pi w_1} + \frac{2c_1 N_{p1}}{N_{p1} + N_{g1}}\right)\left(\frac{(N_{p1} + N_{g1})^3}{4b_1 c_1^2 N_{g1} N_{p1}^2}\right) - \left(\frac{\sigma_H}{C_p}\right)\left(\frac{\sin(\varphi)\cos(\varphi)}{0.0334 W K_o K_m}\right) \leq 0,$$

$$g_6(x) = \left(\frac{366000 N_{g1}}{\pi w_1 N_{p1}} + \frac{2c_2 N_{p2}}{N_{p2} + N_{g2}}\right)\left(\frac{(N_{p2} + N_{g2})^3}{4b_2 c_2^2 N_{g2} N_{p2}^2}\right) - \left(\frac{\sigma_H}{C_p}\right)\left(\frac{\sin(\varphi)\cos(\varphi)}{0.0334 W K_o K_m}\right) \leq 0,$$

$$g_7(x) = \left(\frac{366000 N_{g1} N_{g2}}{\pi w_1 N_{p1} N_{p2}} + \frac{2c_3 N_{p3}}{N_{p3} + N_{g3}}\right)\left(\frac{(N_{p3} + N_{g3})^3}{4b_3 c_3^2 N_{g3} N_{p3}^2}\right) - \left(\frac{\sigma_H}{C_p}\right)\left(\frac{\sin(\varphi)\cos(\varphi)}{0.0334 W K_o K_m}\right) \leq 0,$$

$$g_8(x) = \left(\frac{366000 N_{g1} N_{g2} N_{g3}}{\pi w_1 N_{p1} N_{p2} N_{p3}} + \frac{2c_4 N_{p4}}{N_{p4} + N_{g4}}\right)\left(\frac{(N_{p4} + N_{g4})^3}{4b_4 c_4^2 N_{g4} N_{p4}^2}\right) - \left(\frac{\sigma_H}{C_p}\right)\left(\frac{\sin(\varphi)\cos(\varphi)}{0.0334 W K_o K_m}\right) \leq 0,$$

$$g_9(x) - g_{12}(x) = -N_{pi}\sqrt{\frac{\sin^2(\varphi)}{4} - \frac{1}{N_{pi}} + \left(\frac{1}{N_{pi}}\right)^2} + N_{pi}\sqrt{\frac{\sin^2(\varphi)}{4} - \frac{1}{N_{pi}}\left(\frac{1}{N_{pi}}\right)^2} + \frac{\sin(\varphi)\left(N_{pi} + N_{gi}\right)}{2} + CR_{\min}\pi\cos(\varphi) \leq 0,$$

$$g_{13}(x) - g_{16}(x) = d_{\min} - \frac{2c_i N_{pi}}{N_{pi} + N_{gi}} \leq 0, g_{17}(x) - g_{20}(x) = d_{\min} - \frac{2c_i N_{gi}}{N_{pi} + N_{gi}} \leq 0,$$

$$g_{21}(x) = x_{p1} + \left(\frac{(N_{p1} + 2)c_1}{N_{p1} + N_{g1}}\right) - L_{\max} \leq 0,$$

$$g_{22}(x) - g_{24}(x) = -L_{\max}\left(\frac{(N_{pi} + 2)c_i}{N_{pi} + N_{gi}}\right)_{i=2,3,4} + x_{g(i-1)} \leq 0,$$

$$g_{25}(x) = -x_{p1} + \left(\frac{(N_{p1} + 2)c_1}{N_{p1} + N_{g1}}\right) \leq 0, g_{26\text{–}28}(x) = \left(\frac{(N_{pi} + 2)c_i}{N_{pi} + N_{gi}} - x_{g(i-1)}\right) \leq 0,$$

$$g_{29}(x) = y_{p1} + \left(\frac{(N_{p1} + 2)c_1}{N_{p1} + N_{g1}}\right) - L_{\max} \leq 0,$$

$$g_{30\text{–}32}(x) = -L_{\max} + \left(\frac{(N_{pi} + 2)c_i}{N_{pi} + N_{gi}} - y_{g(i-1)}\right)_{i=2,3,4} \leq 0,$$

$$g_{33}(x) = \frac{(N_{p1}+2)c_1}{N_{p1}+N_{g1}} - y_{p1} \le 0, g_{34\text{–}36}(x) = \left(\frac{(N_{pi}+2)c_i}{N_{pi}+N_{gi}} - y_{g(i-1)}\right)_{i=2,3,4} \le 0,$$

$$g_{37\text{–}40}(x) = -L_{\max} + \frac{(N_{pi}+2)c_i}{N_{pi}+N_{gi}} + x_{gi} \le 0, g_{41\text{–}44}(x) = -x_{gi} + \left(\frac{(N_{pi}+2)c_i}{N_{pi}+N_{gi}}\right) \le 0,$$

$$g_{45\text{–}48}(x) = y_{gi} + \frac{(N_{pi}+2)c_i}{N_{pi}+N_{gi}} - L_{\max} \le 0, g_{49\text{–}52}(x) = -y_{gi} + \left(\frac{(N_{pi}+2)c_i}{N_{pi}+N_{gi}}\right) \le 0,$$

$$g_{53\text{–}56}(x) = (b_i - 8.255)(b_i - 5.715)(b_i - 12.70)(-N_{pi} + 0.945c_i - N_{gi})(-1) \le 0,$$

$$g_{57\text{–}60}(x) = (b_i - 8.255)(b_i - 3.175)(b_i - 12.70)(-N_{pi} + 0.646c_i - N_{gi}) \le 0,$$

$$g_{61\text{–}64}(x) = (b_i - 5.715)(b_i - 3.175)(b_i - 12.70)(-N_{pi} + 0.504c_i - N_{gi}) \le 0,$$

$$g_{65\text{–}68}(x) = (b_i - 5.715)(b_i - 3.175)(b_i - 8.255)(-N_{pi} - N_{gi}) \le 0,$$

$$g_{69\text{–}72}(x) = (b_i - 8.255)(b_i - 5.715)(b_i - 12.70)(N_{pi} + N_{gi} - 1.812c_i)(-1) \le 0,$$

$$g_{73\text{–}76}(x) = (b_i - 8.255)(b_i - 3.175)(b_i - 12.70)(N_{pi} + N_{gi} - 0.945c_i) \le 0,$$

$$g_{77\text{–}80}(x) = (b_i - 5.715)(b_i - 3.175)(b_i - 12.70)(N_{pi} + N_{gi} - 0.646c_i)(-1) \le 0,$$

$$g_{81\text{–}84}(x) = (b_i - 5.715)(b_i - 3.175)(b_i - 8.255)(N_{pi} + N_{gi} - 0.504c_i) \le 0,$$

$$g_{85}(x) = w_{\min} - \frac{w_1(N_{p1}N_{p2}N_{p3}N_{p4})}{N_{g1}N_{g2}N_{g3}N_{g4}} \le 0, g_{86}(x) = -w_{\max}\frac{w_1(N_{p1}N_{p2}N_{p3}N_{p4})}{N_{g1}N_{g2}N_{g3}N_{g4}} \le 0,$$

$$c_i = \sqrt{(y_{gi} - y_{p1})^2 + (x_{gi} - x_{p1})^2}, K_o = 1.5, d_{\min} = 25, J_R = 0.2,$$

$$\varphi = 120, W = 55.9, K_m = 1.6, CR_{\min} = 1.4, L_{\max} = 127, C_p = 464,$$

$$\sigma_H = 3290, w_{\max} = 255, w_1 = 5000, \sigma_N = 2090, w_{\min} = 245,$$

$$b_i \in \{3.175, 12.7, 8.255, 5.715\}, 7 \le N_{gi}, N_{pi} \le 76,$$

$$y_{pi}, x_{pi}, y_{gi}, x_{gi} \in \{12.7, 38.1, 25.4, 50.8, 76.2, 63.5, 88.9, 114.3, 101.6\}.$$

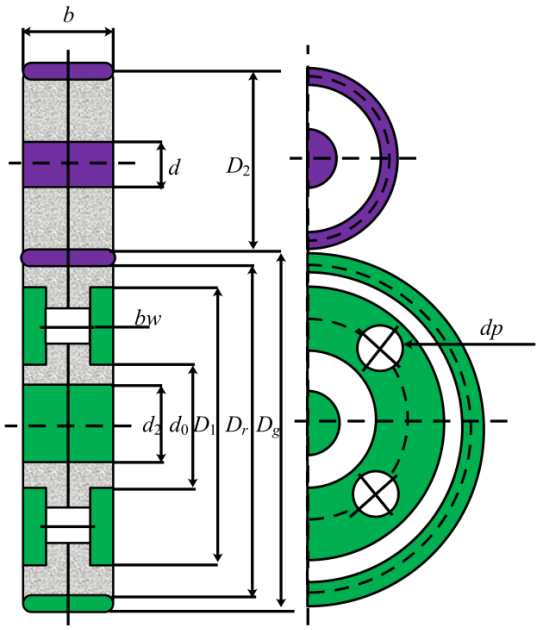


**Figure 11.** The construction of the four-stage gearbox design problem.

Tables 8 and 9 are the according results of all algorithms in solving the four stage gearbox design problem. In Table 8, the ENGO algorithm and MRFO algorithm both

perform well in this problem. Though the MRFO has a smaller average than the ENGO, the ENGO is more stable than the MRFO during many runs, which is also significant in engineering applications. For the original NGO algorithm, it ranks sixth among all algorithms. Thus, by introducing the polynomial interpolation strategy and the multi-strategy opposite learning models into the original NGO algorithm, its comprehensive performance has been significantly improved.

**Table 8.** Results of the different algorithms in solving the four stage gearbox design problem.

| Algorithms | The Best | The Average | The Worst | The Std | Rank |
|---|---|---|---|---|---|
| NGO | 18,398.72 | 166,899.45 | 355,937.63 | 98,874.36 | 6 |
| ENGO | 40.29 | 31,057.83 | 118,898.03 | 38,591.38 | 2 |
| AOA | 236,593.12 | 593,321.31 | 1,367,729.52 | 318,308.48 | 9 |
| AO | 75,789.09 | 334,881.90 | 1,053,952.90 | 240,302.34 | 7 |
| HHO | 16,877.30 | 335,825.38 | 826,083.75 | 241,230.91 | 8 |
| WOA | 148,853.31 | 639,888.43 | 2,175,565.64 | 412,516.64 | 10 |
| PSO | 72.27 | 110,875.33 | 293,548.53 | 91,901.27 | 4 |
| GWO | 43.52 | 147,549.85 | 489,288.82 | 133,316.04 | 5 |
| MRFO | 40.09 | 29,320.84 | 126,516.25 | 42,610.57 | 1 |
| MVO | 61.07 | 103,131.19 | 363,070.87 | 128,562.16 | 3 |

**Table 9.** The best solution of the four stage gearbox design problem obtained by the different algorithms.

| Algorithms | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ |
|---|---|---|---|---|---|---|---|---|
| NGO | 16.3597 | 43.4036 | 15.6042 | 20.7061 | 15.9103 | 45.3901 | 18.1319 | 37.3685 |
| ENGO | 16.3498 | 36.2778 | 18.0688 | 32.4697 | 13.9053 | 36.2515 | 15.7787 | 31.2109 |
| AOA | 7.3482 | 16.5814 | 9.7969 | 18.9440 | 8.9957 | 11.7763 | 6.7875 | 23.3493 |
| AO | 6.7509 | 26.2480 | 15.9342 | 28.5684 | 18.4118 | 42.2120 | 11.9579 | 14.9953 |
| HHO | 19.5464 | 53.4613 | 24.5021 | 44.3106 | 20.7301 | 44.8527 | 26.8726 | 52.5285 |
| WOA | 11.5902 | 27.4520 | 12.1870 | 32.3820 | 11.1318 | 20.5502 | 13.5990 | 23.5141 |
| PSO | 23.4903 | 54.2967 | 24.7237 | 51.7276 | 32.4498 | 54.2332 | 18.4690 | 43.6656 |
| GWO | 21.7829 | 59.0988 | 22.9404 | 33.5723 | 20.0176 | 47.8837 | 20.2588 | 42.3957 |
| MRFO | 22.1205 | 43.6337 | 17.6126 | 38.1229 | 20.9525 | 41.6324 | 21.0797 | 48.9074 |
| MVO | 32.4530 | 57.1252 | 22.4880 | 47.3220 | 22.7186 | 67.3639 | 13.1444 | 22.6758 |
| Algorithms | $x_9$ | $x_{10}$ | $x_{11}$ | $x_{12}$ | $x_{13}$ | $x_{14}$ | $x_{15}$ | $x_{16}$ |
| NGO | 0.7977 | 1.9562 | 1.2978 | 0.7257 | 1.5711 | 3.5749 | 5.1603 | 4.5826 |
| ENGO | 1.1626 | 1.0218 | 1.1244 | 1.2402 | 1.8459 | 3.5264 | 4.2182 | 4.8197 |
| AOA | 2.6529 | 1.2972 | 1.6242 | 1.9436 | 1.6410 | 2.7802 | 4.2547 | 3.5011 |
| AO | 2.4254 | 0.5351 | 1.0247 | 1.6666 | 1.7472 | 3.5758 | 2.1305 | 3.9710 |
| HHO | 0.7225 | 0.9587 | 1.2449 | 1.1576 | 6.2971 | 2.8094 | 4.5773 | 3.2258 |
| WOA | 0.8123 | 1.1016 | 1.4739 | 1.4853 | 0.7152 | 3.7964 | 3.5428 | 3.1702 |
| PSO | 1.2110 | 1.4161 | 1.1434 | 1.1996 | 1.6512 | 6.2798 | 2.9257 | 5.6438 |
| GWO | 0.6744 | 0.5895 | 0.6186 | 0.8372 | 8.0862 | 4.0926 | 5.5300 | 5.1202 |
| MRFO | 1.1778 | 1.1063 | 1.0636 | 1.0945 | 2.3573 | 5.2736 | 5.1814 | 4.8497 |
| MVO | 1.4973 | 1.0831 | 0.7566 | 1.9664 | 7.2887 | 5.9129 | 5.7797 | 5.5464 |
| Algorithms | $x_{17}$ | $x_{18}$ | $x_{19}$ | $x_{20}$ | $x_{21}$ | $x_{22}$ | | |
| NGO | 2.9657 | 1.1195 | 4.3970 | 2.4098 | 3.5683 | 3.6964 | | |
| ENGO | 4.8350 | 1.8115 | 5.3561 | 3.8104 | 3.7590 | 2.7212 | | |
| AOA | 4.5457 | 2.9304 | 5.6055 | 2.3556 | 1.6346 | 1.6271 | | |
| AO | 2.7660 | 6.2850 | 3.4033 | 2.7208 | 3.2026 | 2.8172 | | |
| HHO | 2.9383 | 4.7721 | 2.9564 | 2.4774 | 3.4950 | 2.9967 | | |
| WOA | 2.5494 | 1.0331 | 1.7917 | 2.8701 | 1.7390 | 2.7638 | | |
| PSO | 6.0429 | 7.3123 | 5.0504 | 3.1684 | 6.1872 | 3.9215 | | |

**Table 9.** *Cont.*

| Algorithms | $x_{17}$ | $x_{18}$ | $x_{19}$ | $x_{20}$ | $x_{21}$ | $x_{22}$ |
|---|---|---|---|---|---|---|
| GWO | 5.1050 | 6.3408 | 6.4331 | 3.6878 | 4.2577 | 5.0673 |
| MRFO | 6.2644 | 4.2654 | 4.1077 | 2.8940 | 3.6149 | 4.3536 |
| MVO | 3.6218 | 1.9314 | 5.9677 | 5.2745 | 6.1850 | 3.9999 |

### *4.4. 72-Bar Truss Design Problem*

Figure 12 is the construction of the 72-bar truss design problem, including its node and element numbering schemes [57]. As Table 10 shown, the structural members of the 72-bar space truss are organized into 16 groups, symmetrically. The unit weight of the material is 0.1 lb/in.$^3$, and the modulus of elasticity is 107 psi. The structure has the following constraints: a maximum displacement of $\pm$0.25 in. at the uppermost joints in the $x$, $y$, or $z$ directions, and a maximum allowable stress of $\pm$25 ksi in any element and the range of acceptable cross sectional areas varies from 0.1 in.$^2$ to 3.0 in.$^2$ [58].

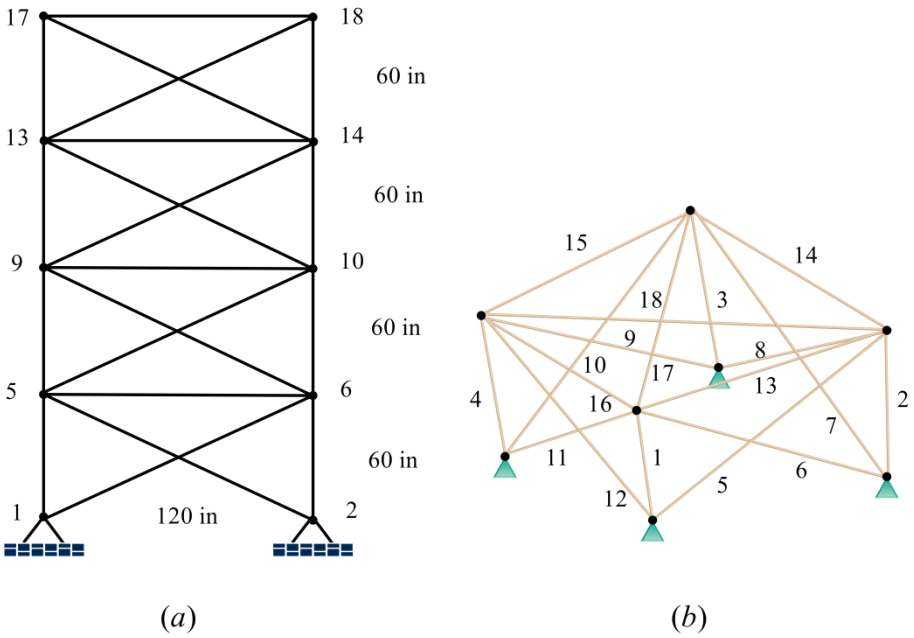


**Figure 12.** The construction of the 72-bar truss design problem. (**a**) The front view; (**b**) Structure diagram in 3-dimension.

**Table 10.** Multiple loading conditions for the 72-bar truss.

| Case | *Node* | *Px (Kips)* | *PY (Kips)* | *Pz (Kips)* |
|---|---|---|---|---|
| 1 | 17 | 0.0 | 0.0 | −5.0 |
| | 18 | 0.0 | 0.0 | −5.0 |
| | 19 | 0.0 | 0.0 | −5.0 |
| | 20 | 0.0 | 0.0 | −5.0 |
| 2 | 17 | 5.0 | 5.0 | −5.0 |

Once the algorithm is run 20 times, all results are summarized in Tables 11 and 12. As shown in Table 11, the ENGO algorithm keeps outstanding performance on all four measure indexes, while the original NGO algorithm only ranks fourth. Thus, for engineering optimization problems of high dimension, the difference between the enhanced algorithm and the original algorithm is more clear. Finally, the optimal solutions obtained by the different algorithms are listed in Table 12, for reference.

**Table 11.** Results of the different algorithms in solving the 72-bar truss design problem.

| Algorithms | The Best | The Average | The Worst | The Std | Rank |
|---|---|---|---|---|---|
| NGO | 392.18 | 411.71 | 444.66 | 15.70 | 4 |
| ENGO | 364.94 | 369.08 | 374.79 | 2.37 | 1 |
| AOA | 521.87 | 633.25 | 797.57 | 79.64 | 8 |
| AO | 450.14 | 519.94 | 643.44 | 49.42 | 5 |
| HHO | 469.52 | 535.59 | 653.30 | 46.82 | 6 |
| WOA | 657.74 | 1276.02 | 2551.19 | 500.03 | 10 |
| PSO | 561.18 | 656.26 | 817.32 | 58.92 | 9 |
| GWO | 366.77 | 385.89 | 544.00 | 41.89 | 2 |
| MRFO | 374.05 | 387.49 | 398.93 | 6.14 | 3 |
| MVO | 403.42 | 631.44 | 818.15 | 130.68 | 7 |

**Table 12.** The best solutions of the 72-bar truss design problem obtained by the different algorithms.

| Variables | Elements | NGO | ENGO | AOA | AO | HHO | WOA | PSO | GWO | MRFO | MVO |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A1 ($cm^2$) | 1–4 | 2.0920 | 1.8590 | 1.5676 | 1.1585 | 1.8381 | 1.1766 | 1.7266 | 1.9659 | 2.1464 | 2.0780 |
| A2 ($cm^2$) | 5–12 | 0.4767 | 0.5651 | 0.5686 | 0.9007 | 0.5715 | 1.3878 | 0.5236 | 0.5256 | 0.4721 | 0.6499 |
| A3 ($cm^2$) | 13–16 | 0.0947 | 0.0010 | 0.1243 | 0.0010 | 0.5927 | 1.0805 | 1.0676 | 0.0076 | 0.0282 | 0.4172 |
| A4 ($cm^2$) | 17–18 | 0.1207 | 0.0036 | 0.5637 | 0.7380 | 0.0010 | 0.5005 | 0.4696 | 0.0116 | 0.0133 | 0.0010 |
| A5 ($cm^2$) | 19–22 | 0.9872 | 1.1815 | 2.6426 | 0.8930 | 0.8664 | 1.0783 | 0.9505 | 1.3216 | 1.1074 | 1.3433 |
| A6 ($cm^2$) | 23–30 | 0.5752 | 0.5100 | 0.7967 | 0.5979 | 0.3892 | 0.4719 | 1.0107 | 0.4859 | 0.5052 | 0.5199 |
| A7 ($cm^2$) | 31–34 | 0.0220 | 0.0087 | 0.0677 | 0.0371 | 0.0010 | 0.3239 | 0.3293 | 0.0239 | 0.0035 | 0.0010 |
| A8 ($cm^2$) | 35–36 | 0.0156 | 0.0027 | 0.6353 | 0.0010 | 0.9025 | 1.0858 | 0.2507 | 0.0553 | 0.0259 | 0.0066 |
| A9 ($cm^2$) | 37–40 | 0.6838 | 0.5605 | 0.9351 | 0.5663 | 0.7672 | 0.3525 | 1.1277 | 0.5764 | 0.6552 | 0.8319 |
| A10 ($cm^2$) | 41–48 | 0.5559 | 0.5261 | 0.3389 | 0.4346 | 0.6098 | 0.8691 | 0.6233 | 0.5201 | 0.5437 | 0.4748 |
| A11 ($cm^2$) | 49–52 | 0.0664 | 0.0010 | 0.3367 | 0.0010 | 0.5841 | 0.2747 | 0.1287 | 0.0254 | 0.0472 | 0.0010 |
| A12 ($cm^2$) | 53–54 | 0.0944 | 0.1566 | 0.6734 | 0.4884 | 0.0737 | 0.9883 | 0.4048 | 0.0536 | 0.0857 | 0.3795 |
| A13 ($cm^2$) | 55–58 | 0.2149 | 0.1639 | 0.9447 | 0.1893 | 0.1440 | 0.5191 | 0.3632 | 0.1694 | 0.2025 | 0.1548 |
| A14 ($cm^2$) | 59–66 | 0.5023 | 0.5753 | 0.5190 | 0.7363 | 0.5593 | 0.3495 | 0.3299 | 0.5391 | 0.5462 | 0.5078 |
| A15 ($cm^2$) | 67–70 | 0.4853 | 0.3682 | 0.6180 | 0.5549 | 0.9750 | 1.0178 | 0.7666 | 0.4751 | 0.4264 | 0.2240 |
| A16 ($cm^2$) | 71–72 | 0.8930 | 0.5086 | 0.5435 | 0.7684 | 0.5203 | 1.0450 | 1.3540 | 0.5289 | 0.7368 | 0.6657 |

### *4.5. Traveling Salesman Problem*

The traveling salesman problem (TSP) is a classical combinatorial optimization problem [59]. In the TSP, there is a commodity salesman who needs to sell his goods in several designated cities. During this process, the salesman has to go through all of the cities before he returns to the original city. Therefore, how to determine the most suitable route to minimize the total travel, is the question to be considered. Since the feasible solution to this problem is the full permutation of all vertices, the challenge of this problem will increase dramatically with the increase in the number of cities, which can be regarded as a NP-hard problem [60]. Here, the ENGO algorithm is employed to solve the TSP. Then, two examples are given to analyze the performance of the ENGO algorithm and other comparison algorithms.

#### 4.5.1. Two Traveling Salesmen and 80 Cities

The first example includes 80 randomly generated cities and two traveling salesmen. The results are summarized in Table 13. Once the algorithm is run 20 times, the average travel obtained by the ENGO algorithm is the smallest. Then, the MRFO algorithm performs better on the best value than the ENGO algorithm, which illustrates that there are still opportunities to improve the ENGO by borrowing the structure of the MRFO algorithm. Figure 13 includes the best routes for the traveling salesman, the convergence curves and the box plots of the different algorithms. From Figure 13a–h, the routes obtained by the AOA or GWO are more complicated, while the route obtained by the ENGO algorithm is reasonable. Furthermore, in Figure 13k, the red line of the ENGO algorithm has the

fastest convergence speed during the whole search process, which illustrates that the improvement strategies are helpful to enhance the quality of the population to make the algorithm approach the optimum faster.

**Table 13.** The results of the different algorithms for the TSP with 80 cities and two2 traveling salesmen.

| Algorithms | The Best | The Average | The Worst | The Std | Rank |
|---|---|---|---|---|---|
| NGO | 7.4210 | 7.5834 | 7.7711 | 0.1558 | 6 |
| ENGO | 7.3777 | 7.4481 | 7.5576 | 0.0727 | 1 |
| AOA | 7.3134 | 7.5703 | 7.8840 | 0.2039 | 5 |
| AO | 7.3397 | 7.4836 | 7.6206 | 0.1105 | 3 |
| HHO | 7.4713 | 7.9101 | 8.3391 | 0.3088 | 10 |
| WOA | 7.4717 | 7.5239 | 7.5539 | 0.0346 | 4 |
| PSO | 7.6181 | 7.8428 | 8.0443 | 0.1876 | 9 |
| GWO | 7.6058 | 7.7255 | 7.8716 | 0.1072 | 7 |
| MRFO | 7.2231 | 7.4833 | 7.7176 | 0.1832 | 2 |
| MVO | 7.5609 | 7.7854 | 7.9852 | 0.1874 | 8 |

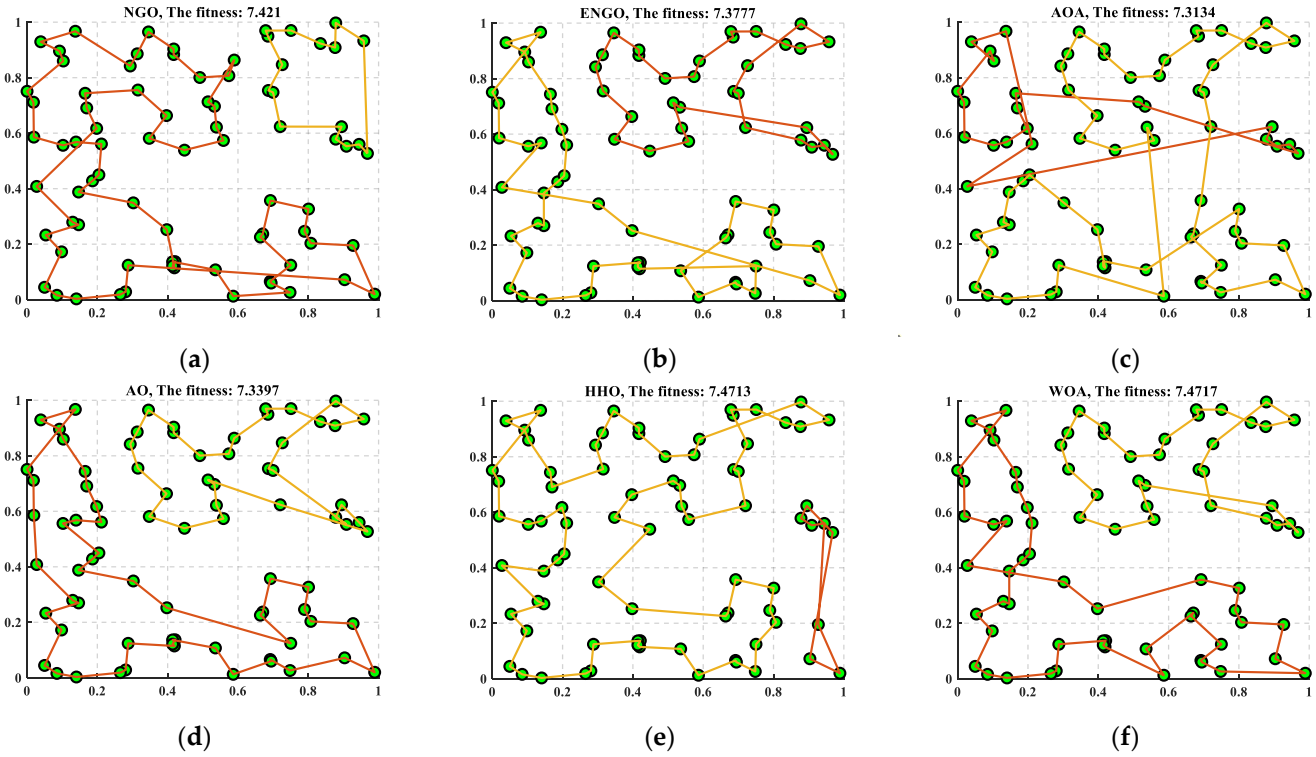


**Figure 13.** *Cont.*

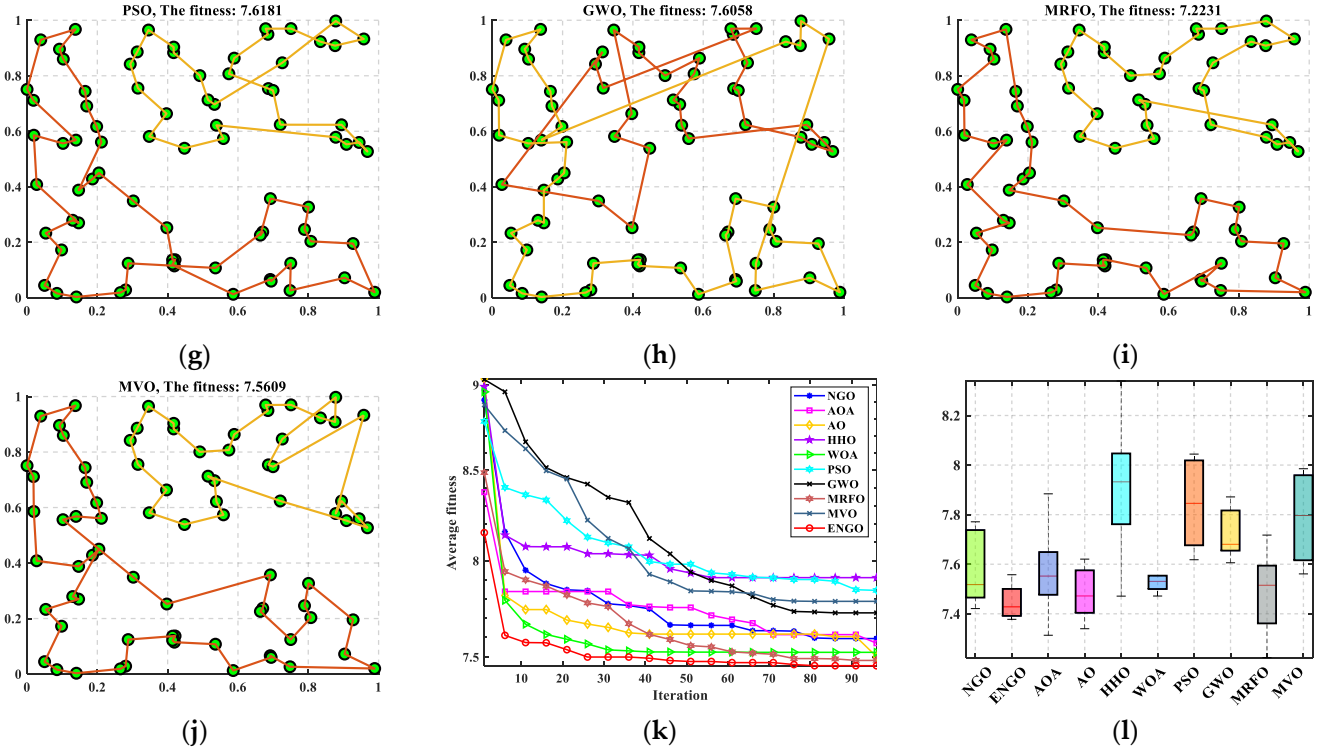


**Figure 13.** The best results, the convergence and box plot of all algorithms for the TSP with 80 cities and two traveling salesmen. (**a**) NGO; (**b**) ENGO; (**c**) AOA; (**d**) AO; (**e**) HHO; (**f**) WOA; (**g**) PSO; (**h**) GWO; (**i**) MRFO; (**j**) MVO; (**k**) The convergence curve; (**l**) The box plot.

#### 4.5.2. Three Traveling Salesmen and 80 Cities

Then, to increase the difficulty of the TSP, the number of traveling salesmen is set as three. Table 14 are the results obtained by all algorithms. Obviously, the ENGO algorithm has the best performance on four measure indexes. Meanwhile, from Figure 14k,l, the ENGO algorithm also has the fastest convergence speed and is the most stable one. Therefore, when the difficulty of the TSP is increased, the ENGO algorithm will show a stronger capacity to obtain the optimal route.

**Table 14.** The results of the different algorithms for the TSP with 80 cities and three travelling salesmen.

| Algorithms | The Best | The Average | The Worst | The Std | Rank |
|---|---|---|---|---|---|
| NGO | 7.4785 | 7.6461 | 7.8232 | 0.1497 | 4 |
| ENGO | 7.4153 | 7.5262 | 7.6154 | 0.0719 | 1 |
| AOA | 7.5364 | 7.7051 | 7.8416 | 0.1225 | 5 |
| AO | 7.5042 | 7.6262 | 7.6752 | 0.0695 | 3 |
| HHO | 7.6903 | 7.9476 | 8.1682 | 0.1786 | 7 |
| WOA | 7.6291 | 7.7467 | 7.8408 | 0.0973 | 6 |
| PSO | 7.8003 | 7.9549 | 8.1152 | 0.1396 | 8 |
| GWO | 7.7822 | 8.0106 | 8.3583 | 0.2167 | 9 |
| MRFO | 7.4747 | 7.6107 | 7.8013 | 0.1459 | 2 |
| MVO | 7.7944 | 8.0646 | 8.6528 | 0.3388 | 10 |

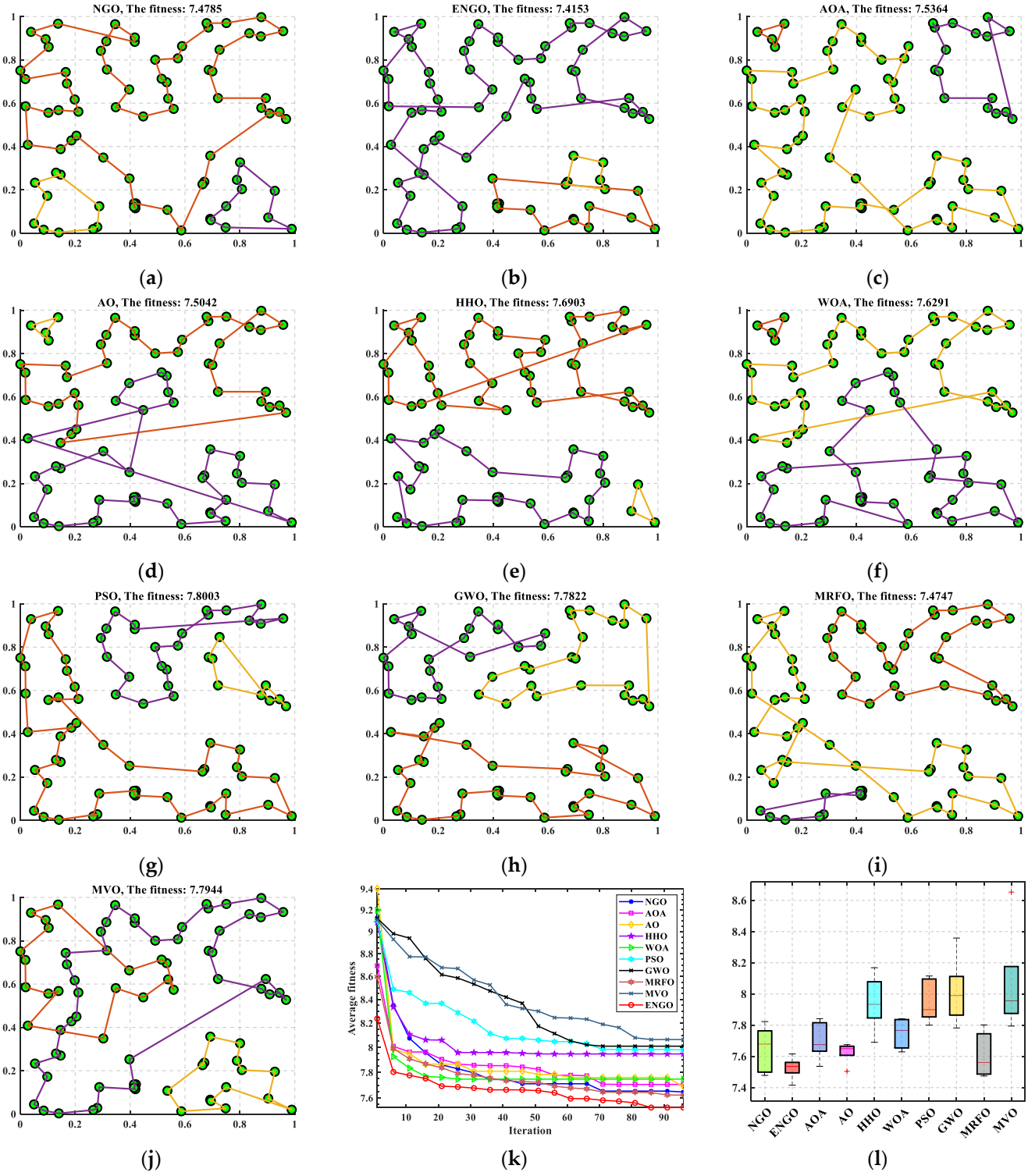


**Figure 14.** The best results, convergence and box plot of all algorithms for the TSP with 80 cities and three traveling salesmen. (**a**) NGO; (**b**) ENGO; (**c**) AOA; (**d**) AO; (**e**) HHO; (**f**) WOA; (**g**) PSO; (**h**) GWO; (**i**) MRFO; (**j**) MVO; (**k**) The convergence curve; (**l**) The box plot.

## 5. Conclusions and Future Work

This paper proposed a novel version of the NGO algorithm by introducing the polynomial interpolation strategy and the multi-strategy opposite learning method, called the ENGO algorithm. To test the ability of the improved algorithm, the ENGO algorithm is employed to solve 29 benchmark functions of the CEC2017 test suite, four engineering

examples and the TSP and, compared it with other famous optimization algorithms. On the one hand, the results show that the ENGO algorithm ranks first on 23 test functions, accounting for 79.31% of the 29 functions, while the original NGO algorithm ranks first on only four functions, accounting for 13.79%. Combined with the box plots, the fluctuation of the results obtained by ENGO algorithm is significantly smaller. These gaps suggest that the introduction of the improvement strategies keeps a better balance between the exploration and exploitation, which helps the algorithm to enhance the search ability, and to obtain results with higher precision. The average convergence curves can also reflect the effect of the different strategies. Compared with other algorithms, the curve of the ENGO decreases rapidly. It is because the quality of the whole population is enhanced by the polynomial interpolation strategy, which leads the individuals to determine the appropriate search direction and move toward the optimal solution quickly. At the end of the search process, from the convergence curves, it can be observed that the ENGO algorithm can avoid the search stagnation phenomenon, which reflects that the multi-strategy opposite learning method helps the algorithm to expand the search range, to escape the local optimums. On the other hand, the results of the four engineering examples and the TSP show that the ENGO algorithm is still competitive for the optimization problems with high dimensions and strong constraints. When facing a more complex search space, the ENGO algorithm can keep an outstanding performance on all measure indexes, such as the 72 truss design problem. For the two TSP examples with high dimensions, the differences between the ENGO algorithm and the original NGO algorithm become more obvious, which illustrates that the two improvement strategies can maintain the efficient search ability of the algorithm as the complexity of the problem increases.

For the future, we can introduce the polynomial interpolation strategy and the multi-strategy opposite learning method into other intelligent optimization algorithms and discuss whether these strategies are still effective, and obtain an outstanding performance. Moreover, the ENGO algorithm can be employed to solve other changing problems, such as the parameters optimization of some complex models, feature selection, degree reduction, shape optimization, production scheduling problems, flight optimization problems, and so on.

**Author Contributions:** Conceptualization, G.H.; Data curation, Y.L., X.H., G.H. and W.D.; Formal analysis, Y.L. and W.D.; Funding acquisition, X.H. and G.H.; Investigation, Y.L., X.H. and W.D.; Methodology, Y.L., X.H., G.H. and W.D.; Project administration, X.H. and G.H.; Resources, Y.L. and G.H.; Software, Y.L. and W.D.; Supervision, X.H. and G.H.; Validation, Y.L. and G.H.; Writing—original draft, Y.L., X.H., G.H. and W.D.; Writing—review & editing, Y.L., X.H., G.H. and W.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** Foundation items: Joint Fund of National Natural Science Foundation of China (U21A20485).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** All data generated or analyzed during this study were included in this published article.

**Acknowledgments:** The authors are very grateful to the reviewers for their insightful suggestions and comments, which helped us to improve the presentation and content of the paper.

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

## Nomenclature

| Notations | Explanation |
|---|---|
| NGO | Northern goshawk optimization algorithm |
| ENGO | Enhanced goshawk optimization algorithm proposed in this paper |
| $N$ | The size of the northern goshawk population |
| $M$ | The dimension of the optimization problem |

| | |
|---|---|
| $t$ | The number of current iterations of the algorithm |
| $T$ | The maximum iterations |
| $UB$ | The upper bound of the optimization problem |
| $LB$ | The lower bound of the optimization problem |
| $X_i$ | The $i$-th individual in the population |
| $X_{i,j}$ | The $j$-th element of the $i$-th individual |
| $prey_i$ | The selected prey in the population |
| $r$ | A random number in the goshawk optimization algorithm |
| $I$ | A vector consisted of 1 or 2 |
| $a_0, a_1, a_2$ | Three coefficients in the quadratic polynomial function |
| $\widetilde{X}_i$ | New solution obtained by opposite learning |
| $\overline{X}_i$ | New solution obtained by quasi-opposite learning |
| $\hat{X}_i$ | New solution obtained by quasi-reflected learning |
| $X^{\text{oppo}}$ | New population including the solutions after different opposite learning methods |
| $T_A, T_B, T_C, T_D$ | Number of teeth in gears $T_A, T_B, T_C, T_D$ |
| $T_s$ | The thickness of the shell section |
| $T_h$ | The thickness of the head |
| $R$ | The inner radius |
| $L$ | The length of the cylindrical |
| $N_{p1}, \ldots, N_{p4}$ | The positions of the pinions |
| $N_{g1}, \ldots, N_{g4}$ | The positions of the gears |
| $b_1 \ldots, b_4$ | The thickness of the blanks |
| $x_{p1}, x_{g1}, \ldots, x_{g4}, y_{p1}, y_{g1}, \ldots, y_{g4}$ | The number of teeth of the different gears |
| TSP | Traveling salesman problem |
| GA | Genetic algorithm |
| DE | Differential evolution algorithm |
| ICA | Imperialist competitive algorithm |
| MA | Memetic algorithm |
| BBO | Bio-geography optimization algorithm |
| SA | Simulated annealing algorithm |
| GSA | Gravitational search algorithm |
| SCA | Sine cosine algorithm |
| BOA | Billiards-inspired optimization algorithm |
| GBO | Gradient-based optimizer |
| PSO | Particle swarm optimization algorithm |
| WOA | Whale optimization algorithm |
| GWO | Grey wolf optimizer |
| AO | Aquila optimizer |
| SCSO | Sand cat swarm optimization algorithm |
| AOA | Archimedes optimization algorithm |
| AVOA | African vultures optimization algorithm |
| DO | Dandelion optimizer |
| SSA | Sparrow search algorithm |
| HHO | Harris hawks optimization algorithm |
| MRFO | Manta ray foraging optimization algorithm |

## References


1. Hu, G.; Yang, R.; Qin, X.Q.; Wei, G. MCSA: Multi-strategy boosted chameleon-inspired optimization algorithm for engineering applications. *Comput. Methods Appl. Mech. Eng.* **2023**, *403*, 115676. [CrossRef]
2. Sotelo, D.; Favela-Contreras, A.; Avila, A.; Pinto, A.; Beltran-Carbajal, F.; Sotelo, C. A New Software-Based Optimization Technique for Embedded Latency Improvement of a Constrained MIMO MPC. *Mathematics* **2022**, *10*, 2571. [CrossRef]
3. Hu, G.; Du, B.; Wang, X.; Wei, G. An enhanced black widow optimization algorithm for feature selection. *Knowl.-Based Syst.* **2022**, *235*, 107638. [CrossRef]
4. Kvasov, D.E.; Mukhametzhanov, M.S. Metaheuristic vs. deterministic global optimization algorithms: The univariate case. *Appl. Math. Comput.* **2018**, *318*, 245–259. [CrossRef]
5. Sotelo, D.; Favela-Contreras, A.; Kalashnikov, V.V.; Sotelo, C. Model Predictive Control with a Relaxed Cost Function for Constrained Linear Systems. *Math. Probl. Eng.* **2020**, *2020*, 7485865. [CrossRef]

6. Yan, L.; Zou, X. Gradient-free Stein variational gradient descent with kernel approximation. *Appl. Math. Lett.* **2021**, *121*, 107465. [CrossRef]
7. Rubio, J.d.J.; Islas, M.A.; Ochoa, G.; Cruz, D.R.; Garcia, E.; Pacheco, J. Convergent newton method and neural network for the electric energy usage prediction. *Inf. Sci.* **2022**, *585*, 89–112. [CrossRef]
8. Gonçalves, M.L.N.; Lima, F.S.; Prudente, L.F. A study of Liu-Storey conjugate gradient methods for vector optimization. *Appl. Math. Comput.* **2022**, *425*, 127099. [CrossRef]
9. Guan, G.; Yang, Q.; Gu, W.; Jiang, W.; Lin, Y. A new method for parametric design and optimization of ship inner shell based on the improved particle swarm optimization algorithm. *Ocean Eng.* **2018**, *169*, 551–566. [CrossRef]
10. Rahman, I.; Mohamad-Saleh, J. Hybrid bio-Inspired computational intelligence techniques for solving power system optimization problems: A comprehensive survey. *Appl. Soft. Comput.* **2018**, *69*, 72–130. [CrossRef]
11. Kiran, M.S.; Hakli, H. A tree–seed algorithm based on intelligent search mechanisms for continuous optimization. *Appl. Soft. Comput.* **2021**, *98*, 106938. [CrossRef]
12. Montoya, O.D.; Molina-Cabrera, A.; Gil-Gonzalez, W. A Possible Classification for Metaheuristic Optimization Algorithms in Engineering and Science. *Ingeniería* **2022**, *27*, e19815. [CrossRef]
13. Hu, G.; Zhong, J.; Du, B.; Wei, G. An enhanced hybrid arithmetic optimization algorithm for engineering applications. *Comput. Methods Appl. Mech. Eng.* **2022**, *394*, 114901. [CrossRef]
14. Zhou, J.; Hua, Z. A correlation guided genetic algorithm and its application to feature selection. *Appl. Soft. Comput.* **2022**, *123*, 108964. [CrossRef]
15. Gonçalves, E.N.; Belo, M.A.R.; Batista, A.P. Self-adaptive multi-objective differential evolution algorithm with first front elitism for optimizing network usage in networked control systems. *Appl. Soft. Comput.* **2022**, *114*, 108112. [CrossRef]
16. Atashpaz-Gargari, E.; Lucas, C. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 25–28.
17. Seo, W.; Park, M.; Kim, D.-W.; Lee, J. Effective memetic algorithm for multilabel feature selection using hybridization-based communication. *Expert Syst. Appl.* **2022**, *201*, 117064. [CrossRef]
18. Chen, X.; Tianfield, H.; Du, W.; Liu, G. Biogeography-based optimization with covariance matrix based migration. *Appl. Soft. Comput.* **2016**, *45*, 71–85. [CrossRef]
19. Lee, J.; Perkins, D. A simulated annealing algorithm with a dual perturbation method for clustering. *Pattern Recogn.* **2021**, *112*, 107713. [CrossRef]
20. Wang, Y.; Gao, S.; Yu, Y.; Cai, Z.; Wang, Z. A gravitational search algorithm with hierarchy and distributed framework. *Knowl.-Based Syst.* **2021**, *218*, 106877. [CrossRef]
21. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
22. Kaveh, A.; Khanzadi, M.; Rastegar Moghaddam, M. Billiards-inspired optimization algorithm; a new meta-heuristic method. *Structures* **2020**, *27*, 1722–1739. [CrossRef]
23. Ahmadianfar, I.; Bozorg-Haddad, O.; Chu, X. Gradient-based optimizer: A new metaheuristic optimization algorithm. *Inf. Sci.* **2020**, *540*, 131–159. [CrossRef]
24. Zhang, Y.; Hu, X.; Cao, X.; Wu, C. An efficient hybrid integer and categorical particle swarm optimization algorithm for the multi-mode multi-project inverse scheduling problem in turbine assembly workshop. *Comput. Ind. Eng.* **2022**, *169*, 108148. [CrossRef]
25. Sun, Y.; Yang, T.; Liu, Z. A whale optimization algorithm based on quadratic interpolation for high-dimensional global optimization problems. *Appl. Soft. Comput.* **2019**, *85*, 105744. [CrossRef]
26. Yu, X.; Wu, X. Ensemble grey wolf Optimizer and its application for image segmentation. *Expert Syst. Appl.* **2022**, *209*, 118267. [CrossRef]
27. Abualigah, L.; Yousri, D.; Abd Elaziz, M.; Ewees, A.A.; Al-qaness, M.A.A.; Gandomi, A.H. Aquila Optimizer: A novel meta-heuristic optimization algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [CrossRef]
28. Seyyedabbasi, A.; Kiani, F. Sand Cat swarm optimization: A nature-inspired algorithm to solve global optimization problems. *Eng. Comput.* **2022**, 1–25. [CrossRef]
29. Zhang, H.; Zhang, K.; Zhou, Y.; Ma, L.; Zhang, Z. An immune algorithm for solving the optimization problem of locating the battery swapping stations. *Knowl.-Based Syst.* **2022**, *248*, 108883. [CrossRef]
30. Reihanian, A.; Feizi-Derakhshi, M.-R.; Aghdasi, H.S. NBBO: A new variant of biogeography-based optimization with a novel framework and a two-phase migration operator. *Inf. Sci.* **2019**, *504*, 178–201. [CrossRef]
31. Hu, G.; Du, B.; Wang, X. An improved black widow optimization algorithm for surfaces conversion. *Appl. Intell.* **2022**, 1–42. [CrossRef]
32. Kaveh, A.; Ghazaan, M.I.; Saadatmand, F. Colliding bodies optimization with Morlet wavelet mutation and quadratic interpolation for global optimization problems. *Eng. Comput.* **2022**, *38*, 2743–2767. [CrossRef]
33. Yousri, D.; AbdelAty, A.M.; Al-qaness, M.A.A.; Ewees, A.A.; Radwan, A.G.; Abd Elaziz, M. Discrete fractional-order Caputo method to overcome trapping in local optima: Manta Ray Foraging Optimizer as a case study. *Expert Syst. Appl.* **2022**, *192*, 116355. [CrossRef]

34. Lu, X.-l.; He, G. QPSO algorithm based on Lévy flight and its application in fuzzy portfolio. *Appl. Soft. Comput.* **2021**, *99*, 106894. [CrossRef]
35. Jiang, F.; Xia, H.; Anh Tran, Q.; Minh Ha, Q.; Quang Tran, N.; Hu, J. A new binary hybrid particle swarm optimization with wavelet mutation. *Knowl.-Based Syst.* **2017**, *130*, 90–101. [CrossRef]
36. Miao, F.; Yao, L.; Zhao, X. Symbiotic organisms search algorithm using random walk and adaptive Cauchy mutation on the feature selection of sleep staging. *Expert Syst. Appl.* **2021**, *176*, 114887. [CrossRef]
37. Yang, D.; Wu, M.; Li, D.; Xu, Y.; Zhou, X.; Yang, Z. Dynamic opposite learning enhanced dragonfly algorithm for solving large-scale flexible job shop scheduling problem. *Knowl.-Based Syst.* **2022**, *238*, 107815. [CrossRef]
38. Dehghani, M.; Hubálovský, H.; Trojovský, A. Northern Goshawk Optimization: A New Swarm-Based Algorithm for Solving Optimization Problems. *IEEE Access* **2021**, *9*, 162059–162080. [CrossRef]
39. Hu, G.; Du, B.; Li, H.; Wang, X. Quadratic interpolation boosted black widow spider-inspired optimization algorithm with wavelet mutation. *Math. Comput. Simul.* **2022**, *200*, 428–467. [CrossRef]
40. Yang, Y.; Gao, Y.; Tan, S.; Zhao, S.; Wu, J.; Gao, S.; Wang, Y.-G. An opposition learning and spiral modelling based arithmetic optimization algorithm for global continuous optimization problems. *Eng. Appl. Artif. Intell.* **2022**, *113*, 104981. [CrossRef]
41. Li, B.; Li, Z.; Yang, P.; Xu, J.; Wang, H. Modeling and optimization of the thermal-hydraulic performance of direct contact heat exchanger using quasi-opposite Jaya algorithm. *Int. J. Therm. Sci.* **2022**, *173*, 107421. [CrossRef]
42. Guo, W.; Xu, P.; Dai, F.; Zhao, F.; Wu, M. Improved Harris hawks optimization algorithm based on random unscented sigma point mutation strategy. *Appl. Soft. Comput.* **2021**, *113*, 108012. [CrossRef]
43. Hu, G.; Zhu, X.; Wang, X.; Wei, G. Multi-strategy boosted marine predators algorithm for optimizing approximate developable surface. *Knowl.-Based Syst.* **2022**, *254*, 109615. [CrossRef]
44. Houssein, E.H.; Gad, A.G.; Hussain, K.; Suganthan, P.N. Major Advances in Particle Swarm Optimization: Theory, Analysis, and Application. *Swarm Evol. Comput.* **2021**, *63*, 100868. [CrossRef]
45. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The Arithmetic Optimization Algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
46. Abdollahzadeh, B.; Gharehchopogh, F.S.; Mirjalili, S. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Comput. Ind. Eng.* **2021**, *158*, 107408. [CrossRef]
47. Zhao, S.; Zhang, T.; Ma, S.; Chen, M. Dandelion Optimizer: A nature-inspired metaheuristic algorithm for engineering applications. *Eng. Appl. Artif. Intell.* **2022**, *114*, 105075. [CrossRef]
48. 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]
49. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]
50. Zheng, J.; Hu, G.; Ji, X.; Qin, X. Quintic generalized Hermite interpolation curves: Construction and shape optimization using an improved GWO algorithm. *Comput. Appl. Math.* **2022**, *41*, 115. [CrossRef]
51. Zhao, W.; Zhang, Z.; Wang, L. Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. *Eng. Appl. Artif. Intel.* **2020**, *87*, 103300. [CrossRef]
52. Mirjalili, S.; Mirjalili, S.M.; Hatamlou, A. Multi-Verse Optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **2016**, *27*, 495–513. [CrossRef]
53. Hu, G.; Li, M.; Wang, X.; Wei, G.; Chang, C.-T. An enhanced manta ray foraging optimization algorithm for shape optimization of complex CCG-Ball curves. *Knowl.-Based Syst.* **2022**, *240*, 108071. [CrossRef]
54. Naruei, I.; Keynia, F. A new optimization method based on COOT bird natural life model. *Expert Syst. Appl.* **2021**, *183*, 115352. [CrossRef]
55. Hu, G.; Zhu, X.; Wei, G.; Chang, C.-T. An improved marine predators algorithm for shape optimization of developable Ball surfaces. *Eng. Appl. Artif. Intell.* **2021**, *105*, 104417. [CrossRef]
56. Gurrola-Ramos, J.; Hernàndez-Aguirre, A.; Dalmau-Cedeño, O. COLSHADE for Real-World Single-Objective Constrained optimization Problems. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–8.
57. Camp, C.V.; Farshchin, M. Design of space trusses using modified teaching-learning based optimization. *Eng. Struct.* **2014**, *62–63*, 87–97. [CrossRef]
58. Hu, G.; Dou, W.; Wang, X.; Abbas, M. An enhanced chimp optimization algorithm for optimal degree reduction of Said-Ball curves. *Math. Comput. Simul.* **2022**, *197*, 207–252. [CrossRef]
59. Uğur, A.; Aydin, D. An interactive simulation and analysis software for solving TSP using Ant Colony Optimization algorithms. *Adv. Eng. Softw.* **2009**, *40*, 341–349. [CrossRef]
60. Skinderowicz, R. Improving Ant Colony Optimization efficiency for solving large TSP instances. *Appl. Soft. Comput.* **2022**, *120*, 108653. [CrossRef]