sensors

Article

Multi-Group Gorilla Troops Optimizer with Multi-Strategies for 3D Node Localization of Wireless Sensor Networks

Qingwei Liang [1], Shu-Chuan Chu [1], Qingyong Yang [1], Anhui Liang [2] and Jeng-Shyang Pan [1,3,4,*]

[1] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; liangqw@sdust.edu.cn (Q.L.); scchu0803@sdust.edu.cn (S.-C.C.); yang_qy@sdust.edu.cn (Q.Y.)
[2] College of Electronic and Information Engineering, Shandong University of Science and Technology, Qingdao 266590, China; liangah8@sdust.edu.cn
[3] Department of Information Management, Chaoyang University of Technology, Taichung 413310, Taiwan
[4] Fujian Provincial Key Laboratory of Big Data Mining and Applications, Fujian University of Technology, Fuzhou 350118, China
* Correspondence: jspan@cc.kuas.edu.tw

Abstract: The localization problem of nodes in wireless sensor networks is often the focus of many researches. This paper proposes an opposition-based learning and parallel strategies Artificial Gorilla Troop Optimizer (OPGTO) for reducing the localization error. Opposition-based learning can expand the exploration space of the algorithm and significantly improve the global exploration ability of the algorithm. The parallel strategy divides the population into multiple groups for exploration, which effectively increases the diversity of the population. Based on this parallel strategy, we design communication strategies between groups for different types of optimization problems. To verify the optimized effect of the proposed OPGTO algorithm, it is tested on the CEC2013 benchmark function set and compared with Particle Swarm Optimization (PSO), Sine Cosine Algorithm (SCA), Whale Optimization Algorithm (WOA) and Artificial Gorilla Troops Optimizer (GTO). Experimental studies show that OPGTO has good optimization ability, especially on complex multimodal functions and combinatorial functions. Finally, we apply OPGTO algorithm to 3D localization of wireless sensor networks in the real terrain. Experimental results proved that OPGTO can effectively reduce the localization error based on Time Difference of Arrival (TDOA).

Keywords: artificial gorilla troops optimizer; opposition-based learning; parallel communication strategy; wireless sensor networks; 3D node localization

check for updates

Citation: Liang, Q.; Chu, S.-C.; Yang, Q.; Liang, A.; Pan, J.-S. Multi-Group Gorilla Troops Optimizer with Multi-Strategies for 3D Node Localization of Wireless Sensor Networks. *Sensors* **2022**, *22*, 4275. https://doi.org/10.3390/s22114275

Academic Editor: Jaime Lloret

Received: 8 May 2022
Accepted: 31 May 2022
Published: 3 June 2022

1. Introduction

Solving optimization problems by simulating the behavior of biological groups has become a new research hotspot, and a theoretical system with swarm intelligence as the core has been formed. Observations and studies of biological groups have found that swarm intelligence generated by complex behaviors such as cooperation and competition among individuals in biological groups can often provide efficient solutions to some specific problems. Research on swarm intelligence has made breakthroughs in some practical application fields, and more excellent optimization algorithms [1] have been proposed and applied in various fields such as wireless sensor networks, robotics, and electric power. For example,Particle Swarm Optimization (PSO) [2–4], Whale Optimization Algorithm (WOA) [5–7], Sine Cosine Algorithm (SCA) [8–10], Flower Pollination Algorithm (FPA) [11,12], Differential Evolution Algorithm (DE) [13–16], Cat Swarm Optimization (CSO) [17–20], Bat Algorithm (BA) [21–23], Artificial Bee Colony (ABC) [24,25], Pigeon-Inspired Optimization (PIO) [26,27], and Ant Colony Optimization (ACO) [28,29].

So far many optimization algorithms have been proposed and achieved excellent results. So why do we still need to research and come up with new optimization algorithms?

According to the No Free lunch (NFL) theorem proposed by Wolpert and Macready [30,31], there is no optimization algorithm that can solve a wide variety of problems. In 2021, Benyamin Abdollahzadeh and Farhad Soleimanian Gharehchopogh proposed a new optimization algorithm, which is called Artificial Gorilla Troops Optimizer (GTO) [32]. The new optimization algorithm is inspired by the social intelligence of gorilla troops in nature. The algorithm uses mathematical formulas to represent the collective life of gorillas and designs new mechanisms to exploration and exploitation. The experimental results show that the convergence speed and optimal solution of GTO significantly outperform some known optimization algorithms.

Although GTO has achieved excellent results in many research fields, many experiments also show that GTO is easy to obtain local optimal solutions when solving complex problems. Hamid R. Tizhoosh from the University of Waterloo proposed the opposition-based learning [33] strategy in 2005, which is used to generate the reverse solution of the particle, and construct the reverse swarm corresponding to the current swarm. This strategy helps to search for more effective areas to improve the diversity of groups and enhance the global exploration ability of the algorithm. Technologies related to evolutionary computing emerge in an endless stream, the parallel strategy has become a particularly important method to improve intelligent optimization algorithm. The parallel strategy means that the population is divided into multiple groups, and each group is iteratively updated independently. The significant advantage of this strategy is not only to strengthen the diversity of particle swarms, but also greatly enhances the global search ability of the algorithm. Especially when it comes to more complicated optimization problems, the parallel strategy is a particularly effective method to further upgrade the efficiency of the algorithm and find the optimal solution. Many studies prove that the parallel strategy has been successfully applied to a variety of optimization algorithms, such as parallel PSO [34] and parallel CSO [35]. Inspired by these strategies, we introduces opposition-based learning and parallel mechanism to enhance the performance of GTO, which is called OPGTO. OPGTO is obviously superior to GTO in terms of convergence speed and accuracy of solving the optimal solution, and has obvious advantages over some well-known optimization algorithms.

With the rapid development and maturity of communication technology, embedded computing technology and sensor technology, micro-sensors with sensing, computing and communication capabilities have begun to appear around the world. Wireless sensor networks composed of these tiny sensors have attracted great attention. WSN integrates sensor technology, embedded computing technology, distributed information processing technology and communication technology. It can collaboratively monitor, perceive and collect information of various environments or monitoring objects in the network distribution area in real time. Therefore, this network system can be widely used in various fields, such as forest fire prevention, ocean detection and predict nature disaster [36], etc.

In many scenarios, sensor nodes are randomly deployed in a certain area. Since nodes cannot know their location in advance, they need to obtain their location information through localization technology after deployment. It is known to all that the most common localization technology is GPS, which can locate nodes through satellites and achieve relatively high accuracy. However, due to the high energy consumption and cost of GPS receivers, it is impossible to equip each node with a GPS receiver. Therefore, WSN usually consist of anchor nodes with GPS and unknown nodes without GPS. Through some localization technologies, unknown nodes can determine their position based on anchor nodes. The location technology of WSN consists of ranging phase and evaluation phase. In the ranging phase, multiple sending nodes can transmit signals at the same time, and the receiving nodes can calculate the distance based on the time difference between the arrival of the signals and their propagation speed, which is called Time Difference of Arrival (TDOA) [37,38]. This paper applies OPGTO to solve the 3D localization problem in WSN and can achieve lower localization errors through the idea of TDOA.

The remainder of the paper is organized as follows. In Section 2, we will elaborate on the GTO algorithm in detail. Section 3 elaborates on the opposition-based learning and parallel mechanism in OPGTO, and how the new algorithm can be combined with TDOA and applied to 3D localization. Section 4 discusses in detail the simulation results of OPGTO on the CEC2013 benchmark functions and how to reduce the error of 3D localization in WSN. Section 5 discusses the innovations of our approach. Section 6 mainly summarizes and looks forward to this paper.

2. Related Work

In this section, will first introduce the design ideas and implementation steps of the GTO algorithm. Then, we will analyze how to use TDOA for node localization in WSN.

2.1. Artificial Gorilla Troops Optimizer

The GTO algorithm includes the exploration phase and the exploitation phase, Equations (1)–(13) [32] describe the main idea of the algorithm. The exploration phase is mainly used to perform a global search of the space. It uses three different mechanisms, includes migrating to an unknown position, migrating to a known position, and moving to the position of other gorillas. Equation (1) simulates the exploitation phase.

$$GX(t+1) = \begin{cases} (ub-lb) \times r_1 + lb, & r < p, \\ (r_2 - C) \times X_r(t) + L \times H, & r \geq 0.5, \\ X(i) - L \times (L \times (X(t) - GX_r(t)) + r_3 \times (X(t) - GX_r(t))), & r < 0.5. \end{cases} \quad (1)$$

In Equation (1), *X(t)* represents the gorilla current position and $GX(t+1)$ is the gorilla position in the $t+1$ iteration. p a parameter between 0 and 1 that determines which migration mechanism to choose. *lb* and *ub* are the lower and upper bounds of the variable, respectively. X_r is a randomly selected gorilla member from the population and GX_r is the randomly selected gorilla candidate position vector. r_1, r_2, r_3, and r are random values in the range 0 to 1 updated on each iteration. Moreover, C, L, and H are calculated from the following equations.

$$C = F \times \left(1 - \frac{It}{MaxIt}\right), \quad (2)$$

$$F = cos(2 \times r_4) + 1, \quad (3)$$

$$L = C \times l, \quad (4)$$

$$H = Z \times X(t), \quad (5)$$

$$Z = [-C, C]. \quad (6)$$

In Equation (2), *It* is the current number of iterations and *MaxIt* is the total number of iterations of the algorithm. In Equation (3) and (4), r_4 and l are random values between 0 to 1 updated on each iteration. In Equation (6), Z is a random value in the range $-C$ to C. At the end of the exploration phase, the algorithm calculated the fitness value of all GX solutions, and if the fitness value is $GX(t) < X(t)$, the *X(t)* solution is replaced by the *GX(t)* solution.

The exploitation process of the GTO algorithm uses two mechanisms, following silverback gorilla and competing for adult female gorillas. The mechanism is selected by comparing the C value calculated by the Equation (2) with the parameter W set in advance. If $C \geq W$, the GTO algorithm uses the following silverback gorilla mechanism, but if $C < W$, competing for adult female gorillas is selected. Equation (7) is used to simulate Follow the silverback gorilla.

$$GX(t+1) = L \times M \times (X(t) - X_{silverback}) + X(t), \quad (7)$$

$$M = \left(\left| \frac{1}{N} \sum_{i=1}^{N} GX_i(t) \right|^g \right)^{\frac{1}{g}}, \quad (8)$$

$$g = 2^L. \tag{9}$$

In Equation (7), $X_{silverback}$ is the silverback gorilla position. In Equation (8), $GX_i(t)$ is the position of each candidate gorilla in iteration t and N is the total number of gorillas. Furthermore, Equation (10) is used to simulate Compete with adult female gorillas.

$$GX(i) = X_{silverback} - (X_{silverback} \times Q - X(t) \times Q) \times A, \tag{10}$$

$$Q = 2 \times r_5 - 1, \tag{11}$$

$$A = \beta \times E, \tag{12}$$

$$E = \begin{cases} N_1, & r \geq 0.5, \\ N_2, & r < 0.5. \end{cases} \tag{13}$$

In Equation (11), r_5 is a random value between 0 to 1 updated on each iteration. In Equation (12), β is a parameter to be given value. In Equation (13), if $rand \geq 0.5$, E is a random value in the normal distribution and the dimensions of the problem, but if $rand < 0.5$, E is a random value chosen from a normal distribution. At the end of the exploitation phase, the algorithm calculated the fitness value of all GX solutions. If the fitness value is $GX(t) < X(t)$, the *X(t)* solution is replaced by the *GX(t)* solution and the optimal solution selected in the entire population is regarded as the silverback gorilla.

2.2. 3D Localization of Wireless Sensor Network

In this paper, OPGTO is applied to the 3D node localization in WSN. The new algorithm uses the TDOA method to locate the nodes. The TDOA localization method is to determine the location of the unknown node by measuring the propagation time difference of the signal from the unknown node to the two anchor nodes. According to the distance difference corresponding to this propagation time difference, a hyperbola with the corresponding two anchor nodes as the focus can be obtained. Therefore, if we want to get the estimated position value of the unknown node, we only need to measure the difference between the two sets of TDOA. Two hyperbolas can be obtained from these two sets of TDOA data values, and the intersection of these two hyperbolas is the estimated position of the node to be measured. The schematic diagram of this method is shown in Figure 1.

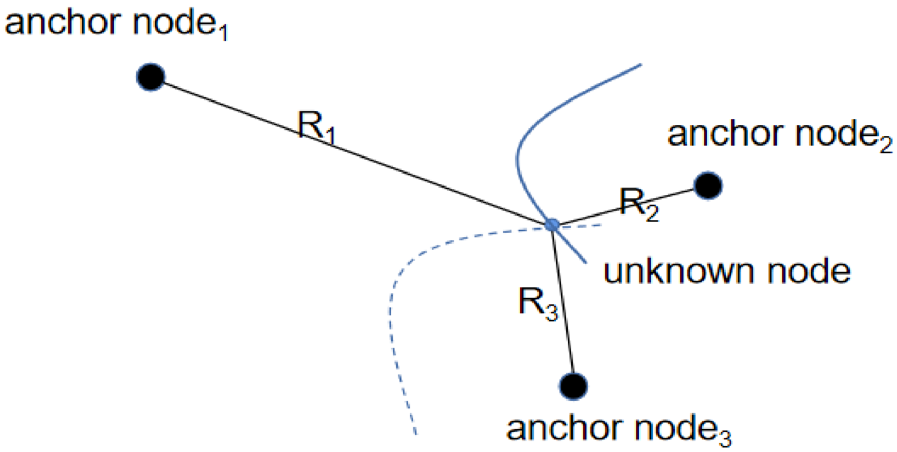


Figure 1. The principle of TDOA.

3. Artificial Gorilla Troops Optimizer with Opposition-Based Learning and Parallel Strategy (OPGTO) for 3D Localization

In this section, we propose the opposition-based learning strategy for expanding the search space of the algorithm. On this basis, we propose two communication strategies based on the idea of parallel to increase the diversity of the population and the exploration ability of the algorithm. Finally, we apply the new algorithm to 3D localization.

3.1. Opposition-Based Learning

In many cases, population generates individuals in a random fashion. However, if the algorithm starts its search from a location far from the optimal solution, especially in the worst case where the initial solution is located opposite the optimal solution, the search will take a considerable amount of time. Therefore, the best strategy is to search in all directions, or more specifically, in the opposite direction. Following this idea, this paper uses opposition-based learning in the two stages of population initialization and algorithm stagnation. Opposition-based learning mainly completes the following three parts.

- Initial position of individual gorillas is $X_{i,j}$, $i = 1, 2, \cdots, N$, $j = 1, 2, \cdots, D$. N is the population size, and D is the dimension of the search space. The population at this time is called P_1.
- The population is generated according to Equation (14) and called P_2.

$$X'_{i,j} = lb(j) + ub(j) - rand \times X_{i,j}, \tag{14}$$

  On the basis of Hamid R. Tizhoosh's method [33], we used *rand* to broaden the search space. In Equation (14), *lb(j)* is the lower bound of the *j*th dimension and *ub(j)* is the upper bound of the *j*th dimension. *rand* is a random value in the range 0 to 1. $X'_{i,j}$ is the opposite gorilla.
- In the population P_1 and P_2, the algorithm selects the number N of individuals with the best fitness to form an initialization population.

In the population initialization phase, the algorithm first generates an initial population, and then performs opposition-based learning.

According to the flow of the algorithm, OPGTO will gradually approach the initial solution to the optimal solution. However, due to the limitation of various factors, the algorithm is prone to premature phenomenon and falls into the local optimal solution. This leads to the evolutionary stagnation. Therefore, when the algorithm stagnates, OPGTO will perform opposition-based learning to expand the scope of exploration and escape the current local optimum solution as much as possible.

First, OPGTO needs to perform stagnant detection. In order to design a method to effectively detect the stagnation of the algorithm, this paper introduces the sliding window strategy. The size of the window is set to 4. When the optimal values calculated by the algorithm are all equal in the adjacent 4 iterations, it means that the algorithm is stagnant. At this time, the original population is updated by opposition-based learning. Figure 2 describes the specific process.

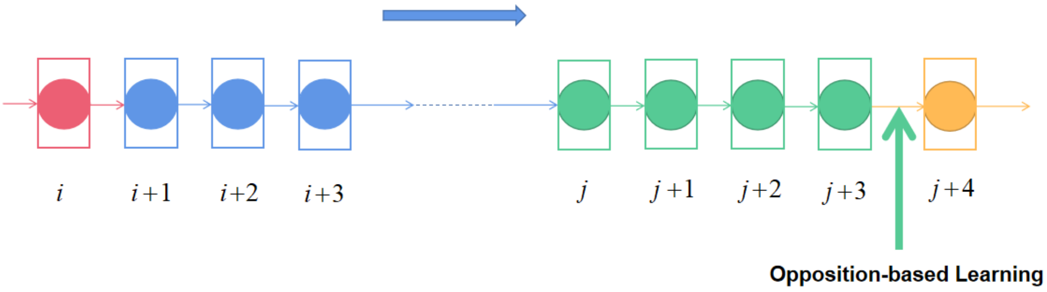


Figure 2. Opposition-based learning with sliding window strategy.

3.2. Parallel Strategy

This section mainly introduces the parallel strategy of OPGTO. In this paper, the parallel strategy is mainly expressed as a variety of intergroup communication strategies, including multi-group merge communication strategy and multi-group competition communication strategy. The communication strategies can effectively update information between each group, so using these can improve the performance of the algorithm.

3.2.1. Multi-Group Merge Communication Strategy

Some communication methods in parallel strategies generally attempt to change the way information is exchanged between groups rather than changing the number of groups. Intelligent optimization algorithms are usually inspired by the evolutionary process of biological populations. In the real development process, there are often mergers and fusions between populations, which also leads to the continuous development of populations. In the process, although weaker populations are swallowed up, they join stronger groups that are better adapted to the development of nature.

Inspired by this, the communication strategy divides the initial population into some groups. Firstly, Each group explores independently and exploits its own optimal individual. After a certain degree of evolution, every two neighbor groups complete the communication strategy and merge into a new group. For example, the first group and the second group will be merged into the new first group after the communication is completed. With this idea, the new group not only retains the effective information of the two groups after merging, but also can conduct new explorations. At the same time, the new group needs to update the global best individual, which is selected from all previous groups. Figure 3 helps us better understand the merge process.

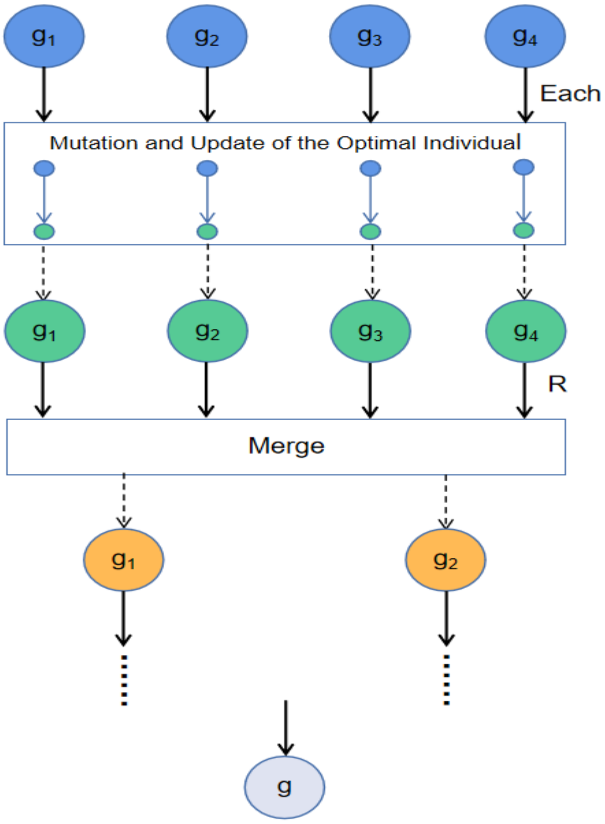


Figure 3. The running process of the population merge communication strategy.

In Figure 3, each group is individually updated iteratively, and when the number of iterations is multiple of *R*, all groups perform merge operations. In addition, the algorithm performs mutation and update operations after each iteration. We set the value of R in Section 4.1. Mutation refers to selecting the optimal individuals from each group, and then subjecting them to Gaussian mutation and Cauchy mutation to expand the searchable range of optimal values. Finally, the individual with the best fitness is selected from the original optimal individual and the mutant individual as the optimal individual of each group.

Li C [39] used Cauchy and Gaussian mutation to update the particles, and we've made some improvements based on that. Equations (15)–(17) describe the mutation operation.

$$MX_{best}^{t} = X_{best}^{t} + X_{best}^{t} \times (\lambda_1 Cauchy(0,1) + \lambda_2 Gauss(0,1)), \tag{15}$$

$$\lambda_1 = \frac{T-t}{T}, \tag{16}$$

$$\lambda_2 = \frac{t}{T}. \tag{17}$$

In Equation (15), X_{best}^{t} is the original optimal individual and MX_{best}^{t} represents the mutant individual, $Cauchy(0,1)$ is a random value from the standard Cauchy distribution and $Gauss(0,1)$ is a random value from the standard Gaussian distribution. In addition, λ_1 and λ_2 are calculated from Equations (16) and (17). Among them, T is the maximum number of iterations and t is the current number of iterations. In the iterative process of the algorithm, λ_1 gradually decreases as the number of iterations increases, but λ_2 gradually increases. This approach can make the algorithm escape the current stagnation position and enhance the ability of global exploration.

3.2.2. Multi-Group Competition Communication Strategy

In the process of biological development and evolution, competition among populations is an important way to promote individuals to reform of nature. Nature uses competition to select individuals, which means that individuals with good fitness will have more resources, while individuals with poor fitness will be gradually eliminated. In addition, the genes of individuals with good fitness will be spread, because more and more individuals will learn their behavior and evolve gradually. Therefore, we designs a communication strategy that simulates this phenomenon and Figure 4 shows the detailed idea of it.

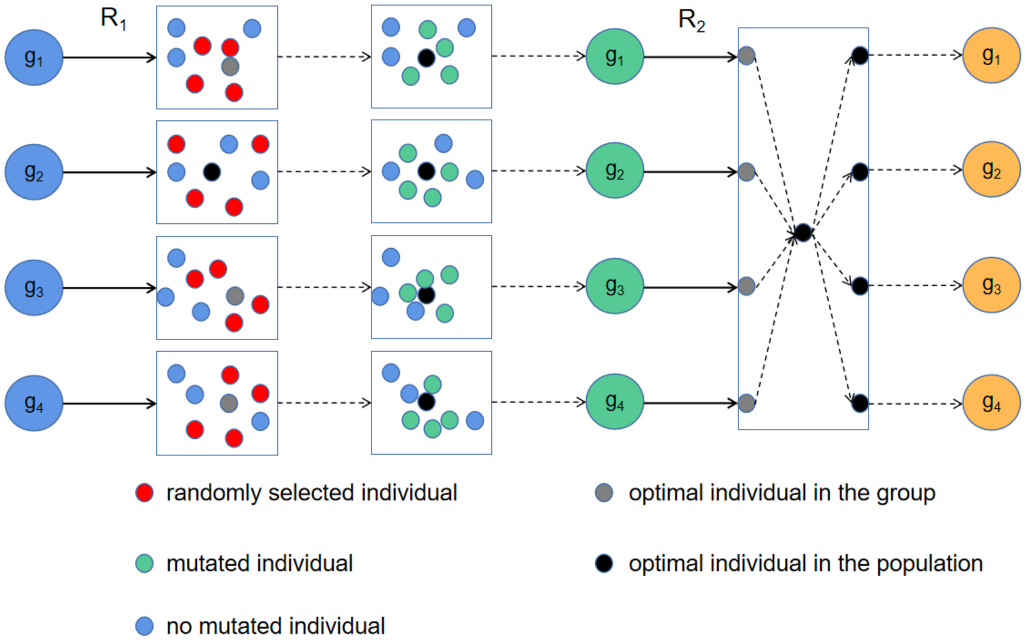

R1
R2
g1
g2
g3
g4
g1
g2
g3
g4
g1
g2
g3
g4
randomly selected individual
optimal individual in the group
mutated individual
optimal individual in the population
no mutated individual

Figure 4. The running process of the population competition communication strategy.

In this strategy, the entire population is firstly divided into multiple groups and the iterations of the GTO algorithm are performed separately. The update method of PWOA [40] gave us a lot of inspiration. When the number of iterations is a multiple of R_1, the following operations are performed:

(1) Select the optimal individual in each group and compare them to get the optimal individual in the entire population.

(2) Randomly select some individuals in each group and mutate them according to Equation (18):

$$X_{i,d}^{t+1} = X_{best}{}_{d}^{t} \times (0.2 \cdot rand + 0.8). \tag{18}$$

where the $X_{i,d}^{t+1}$ indicates the value of a random individual in the dth dimension at the $(t+1)$th iteration, i is a random number between 0 and the group size. The $X_{best}{}_{d}^{t}$ represents the value of the optimal individual of the population in the dth dimension at the tth iteration and $rand \in [0,1]$. This propagation strategy is to randomly select the same number of individuals from each group and learn the optimal individual in the population in different dimensions. In Figure 3, the mutated individual will be closer to the optimal individual in the whole population than before.

(3) Each group's the optimal individual is updated according to the following equation. $X(g)_{best}^{t+1}$ indicates the optimal individual in the gth group at the *(t+1)*th iteration and X_{best}^{t} is the optimal individual in the population at the tth iteration.

$$X(g)_{best}^{t+1} = X_{best}^{t}. \tag{19}$$

Then, when the number of iterations of is a multiple of R_2, X_{best}^{t+1} and $X(g)_{best}^{t+1}$ are updated by the following equation. In Equation (20), *fitness(X)* means calculating the value of X in the fitness function.

$$X_{best}^{t+1} = \begin{cases} X(g)_{best}^{t+1}, & fitness(X(g)_{best}^{t+1}) < fitness(X_{best}^{t}), \\ X_{best}{}^{t}, & fitness(X(g)_{best}^{t+1}) \geq fitness(X_{best}^{t}). \end{cases} \tag{20}$$

$$X(g)_{best}^{t+1} = X_{best}^{t+1}. \tag{21}$$

3.3. Apply the OPGTO in 3D Localization

In this paper, OPGTO is used to reduce TDOA-based localization errors in WSN. Inspired by PCSCA [9], the error values is calculated from Equations (22)–(24). In Equations (22) and (23), R_i represents the distance from the unknown node to the anchor node i that it can visit, and D_i represents the distance from the algorithm estimated node to the anchor node i that the unknown node can visit. Since TDOA uses the method of signal arrival time difference for positioning, the signal will inevitably be interfered during the transmission process. Therefore, we set random numbers $Noise_1$ and $Noise_2$ to simulate the disturbance of the signal. In order to facilitate the calculation, we take the first anchor node that the unknown node can visit as the benchmark, and use other anchor nodes that the unknown node can visit to calculate the difference with this benchmark. $error_{1,i}$ represents the error between the unknown node and node i, and $error_{2,i}$ represents the error between the node estimated by the algorithm and node i.

$$error_{1,i} = R_{i+1} - R_1 + Noise_1, \tag{22}$$

$$error_{2,i} = D_{i+1} - D_1 + Noise_2, \tag{23}$$

$$error = \sum_{i=1}^{N-1} (\sqrt{(error_{1,i} - error_{2,i})^2}). \tag{24}$$

R_i and D_i are calculated according to Equations (25) and (26). (x_i, y_i, z_i) is the location of anchor node i. (X_1, Y_1, Z_1) represents the location of the unknown node, and (X_2, Y_2, Z_2) represents the location of the node estimated by the algorithm.

$$R_i = \sqrt{(X_1 - x_i)^2 + (Y_1 - y_i)^2 + (Z_1 - z_i)^2}, \tag{25}$$

$$D_i = \sqrt{(X_2 - x_i)^2 + (Y_2 - y_i)^2 + (Z_2 - z_i)^2}, \tag{26}$$

Positioning in 3D space is often more complex than in 2D space, so the first thing we have to consider is the obstacle problem. For example, in disaster rescue, the signal may be interfered by obstacles when a node in the wireless sensor network communicates with other nodes, which will cause the node to fail to connect with other nodes normally. In other words, the node cannot send its own location information, so that the localization cannot be completed. Therefore, it is very important to check whether there are obstacles between nodes. For obstacle judgment, people often use the line-of-sight (LOS) [41] algorithm for computation.

The principle of the LOS method is shown in Figure 5. There are two nodes A and B that need to communicate. If a message is sent from node A to node B, the most intuitive way is to treat the two nodes as coordinate points and draw a straight line L to connect point A and point B. If any terrain point C between two points is above the straight line L, then terrain point C is a communication obstacle between point A and point B. Conversely, if there is no terrain point C is above the straight line L, then we can consider that there is no obstacle between point A and point B. In Figure 5, the communication between node A and node B_1 is affected because there is an obstacle C, but node A and node B_2 can communicate normally.

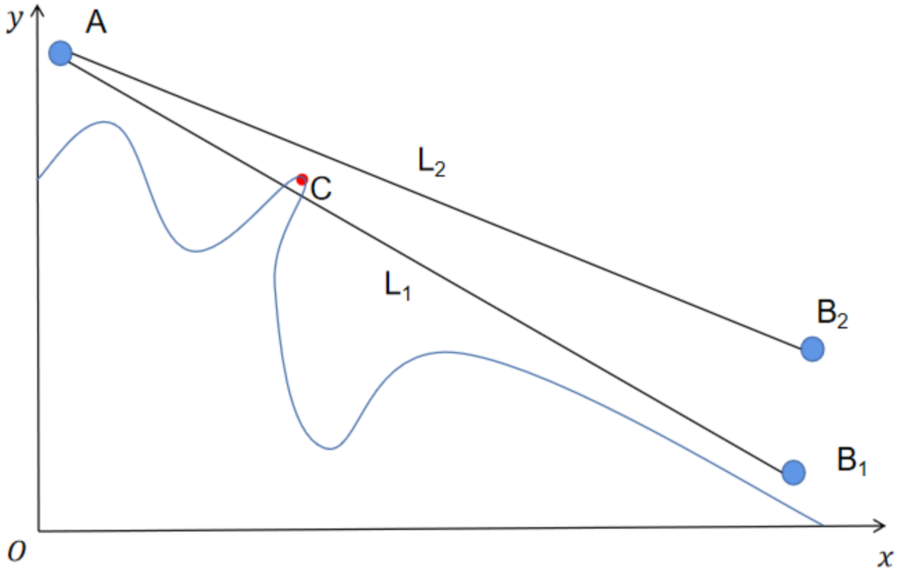


Figure 5. Judging obstacles based on LOS.

4. Results

This section shows the simulation results of OPGTO under the CEC2013 [42] test function set and the practical application effect of TDOA in 3D localization.

4.1. Results of Benchmark Functions

This paper uses 28 classical mathematical test functions from CEC2013 to verify the performance of the new algorithm. CEC2013 has a total of 28 test functions. F1–F5 are ordinary unimodal functions. F6–F20 are some basic multimodal functions that can effectively detect whether the algorithm can escape from the local optimal solution. Moreover, F21–F28 are some complex functions, which can effectively test the global search ability of the algorithm. All experiments have been tested and executed using the Matlab R2018a, laptop computer running Windows 10 64-bit with an Intel Core i7-8750H 2.20 GHz processor and 8.00 GB RAM.

To evaluate the performance of the new algorithm in detail, we compare OPGTO with the traditional optimization algorithms PSO, WOA, SCA and the original GTO algorithm.

Table 1 describes the initial parameters [32,40] for these comparison algorithms. In order to ensure the fairness of the test, each algorithm is independently run 20 times under the same hardware environment to obtain the average value, which represents the performance of the algorithm.

The OPGTO algorithm proposed in this paper uses opposition-based learning for expanding the search range and two communication strategies for communication. The first communication strategy of OPGTO is multi-group merge communication strategy, which reduces the number of groups from 4 to 1. Since the number of iterations is 2000, set R to 700. The second communication strategy of OPGTO is multi-group competition communication strategy (OPGTOS2), which requires multiple individuals to gradually approach the optimal individual. So R_1 is set to 100 and R_2 is set to 250. In this way, inter-group communication is more frequent and the algorithm is more likely to escape from local optimal solutions. Algorithm 1 introduces the execution process of OPGTOS1 and Algorithm 2 introduces the execution process of OPGTOS2.

Algorithm 1 OPGTOS1

Input: The maximum number of iterations T, the population size N, the lower boundary lb, the upper boundary ub, the dimension dim, the objective function *fobj*;

```
Initialization: Generate the individual $X_i (i = 1, 2, \cdots, N)$.
%Opposition-based learning
Calculate the opposition-based individual $X_i^{'} (i = 1, 2, \cdots, N)$ using Equation (14). Select $N$ individuals with the best fitness from $X_i$ and $X_i^{'}$;
Divide the population into $g$ groups, every group is $Group(j) (j = 1, 2, \cdots, g)$;
The best gorilla in each group is Group(j).silverback and its fitness value is Group(j).bestfit;
t = 1;
while t < T do
    %Multi-group merge communication strategy
    if t == R then
        j = 0
        for i = 1:2:g do
            j = j + 1
            Merge groups
            %Update the optimal individual for each group
            Group(j).silverback = BestGorilla
            Group(j).bestfit = BestFitness
        end for
        g = g/2
    end if
    for i = 1:g do
        Update Group(i) using GTO
        Mutate Group(i).silverback' using Equation (15)
        Select the best fit individual from Group(i).silverback' and Group(i).silverback
        %Update the optimal individual for the entire population
        if Group(i).BestFitness<BestGorilla then
            BestGorilla=Group(i).silverback
            BestFitness=Group(i).bestfit
        end if
        if t > 100 then
            Opposition-based learning with sliding window strategy
        end if
    end for
end while
```

Output: *BestGorilla* and *BestFitness*.

Algorithm 2 OPGTOS2

```
Input: The maximum number of iterations $T$, the population size $N$, the lower boundary $lb$, the upper boundary $ub$, the dimension $dim$, the objective function $fobj$;
Initialization: Generate the individual $X_i (i = 1, 2, \cdots, N)$.
%Opposition-based learning
Calculate the opposition-based individual $X_i^{'} (i = 1, 2, \cdots, N)$ using Equation (14). Select $N$ individuals with the best fitness from $X_i$ and $X_i^{'}$;
Divide the population into $g$ groups, every group is $Group(j) (j = 1, 2, \cdots, g)$;
The best gorilla in each group is Group(j).silverback and its fitness value is Group(j).bestfit;
t = 1;
while t < T do
    %Multi-group competition communication strategy
    for i = 1:g do
        if t == $R_1$ then
            Randomly select some individuals in Group(i)
            Mutate the selected individuals using Equation (18)
            Update Group(i).silverback using Equation (19)
        end if
        if t == $R_2$ then
            Update BestGorilla using Equation (20)
            Update Group(i).silverback using Equation (21)
        end if
    end for
    for i = 1:g do
        Update Group(i) using GTO
        %Update the optimal individual for the entire population
        if Group(i).BestFitness<BestGorilla then
            BestGorilla=Group(i).silverback
            BestFitness=Group(i).bestfit
        end if
        if t > 100 then
            Opposition-based learning with sliding window strategy
        end if
    end for
end while
Output: BestGorilla and BestFitness.
```

Table 1. Parameter settings for each algorithm.

Algorithm	Parameter
PSO	c = 2.0, w = 0.9, pop_size = 40, iterations = 2000
WOA	a = 2.0, b = 1, pop_size = 40, iterations = 2000
SCA	a = 2.0, pop_size = 40, iterations = 2000
GTO	p = 0.03, β = 3, W = 0.8, pop_size = 40, iterations = 2000
OPGTOS1	p = 0.03, β = 3, W = 0.8, groups = 4, pop_size = 40, iterations = 2000
OPGTOS2	p = 0.03, β = 3, W = 0.8, groups = 4, pop_size = 40, iterations = 2000

In the experiment, in order to test the optimization ability of the new algorithm in different dimensions, the new algorithm and the comparison algorithm were tested in 10, 30 and 50 dimensions respectively. In Table 2, all algorithms are tested in 10 dimensions. Compared to other algorithms, OPGTOS1 achieves 16 better solutions and 2 identical solutions, while OPGTOS2 achieves 7 better solutions and 2 identical solutions. Compared with the GTO algorithm, OPGTOS1 achieved 24 better solutions and 2 identical solutions, and OPGTOS2 achieved 22 better solutions and 2 identical solutions, which shows that the improved strategy proposed in this paper is effective.

Table 2. Simulation results of CEC2013 functions in 10 dimensions.

Functions	PSO	WOA	SCA	GTO	OPGTOS1	OPGTOS2
F1	$-\mathbf{1.4000 \times 10^{3}}$	$-\mathbf{1.4000 \times 10^{3}}$	-8.3476×10^{2}	$-\mathbf{1.4000 \times 10^{3}}$	$-\mathbf{1.4000 \times 10^{3}}$	$-\mathbf{1.4000 \times 10^{3}}$
F2	4.8719×10^{4}	2.8500×10^{6}	3.4431×10^{6}	4.9012×10^{3}	$\mathbf{4.2691 \times 10^{3}}$	1.8311×10^{4}
F3	3.9529×10^{7}	1.4331×10^{9}	7.1982×10^{8}	$\mathbf{2.4489 \times 10^{7}}$	2.5205×10^{7}	7.6459×10^{7}
F4	-1.0871×10^{3}	2.5367×10^{4}	3.9081×10^{3}	-1.0911×10^{3}	$-\mathbf{1.0989 \times 10^{3}}$	-1.0031×10^{3}
F5	$-\mathbf{1.0000 \times 10^{3}}$	-9.7888×10^{2}	-8.6577×10^{2}	$-\mathbf{1.0000 \times 10^{3}}$	$-\mathbf{1.0000 \times 10^{3}}$	$-\mathbf{1.0000 \times 10^{3}}$
F6	-8.9274×10^{2}	-8.6231×10^{2}	-8.4421×10^{2}	-8.9150×10^{2}	$-\mathbf{8.9772 \times 10^{2}}$	-8.9166×10^{2}
F7	-7.5654×10^{2}	-7.1223×10^{2}	-7.5606×10^{2}	-7.5235×10^{2}	-7.6200×10^{2}	$-\mathbf{7.6313 \times 10^{2}}$
F8	-6.7968×10^{2}	-6.7967×10^{2}	-6.7964×10^{2}	-6.7965×10^{2}	$-\mathbf{6.7968 \times 10^{2}}$	-6.7967×10^{2}
F9	-5.9409×10^{2}	-5.9271×10^{2}	-5.9211×10^{2}	-5.9407×10^{2}	$-\mathbf{5.9508 \times 10^{2}}$	-5.9489×10^{2}
F10	-4.9850×10^{2}	-4.8844×10^{2}	-4.1692×10^{2}	-4.9822×10^{2}	$-\mathbf{4.9932 \times 10^{2}}$	-4.9882×10^{2}
F11	-3.7289×10^{2}	-3.2896×10^{2}	-3.4374×10^{2}	-3.7712×10^{2}	-3.7751×10^{2}	$-\mathbf{3.8312 \times 10^{2}}$
F12	-2.6204×10^{2}	-2.1853×10^{2}	-2.4195×10^{2}	-2.6748×10^{2}	-2.6781×10^{2}	$-\mathbf{2.6841 \times 10^{2}}$
F13	-1.5345×10^{2}	-1.3115×10^{2}	-1.4399×10^{2}	-1.5926×10^{2}	-1.6091×10^{2}	$-\mathbf{1.6431 \times 10^{2}}$
F14	6.4271×10^{2}	9.1978×10^{2}	1.1730×10^{3}	4.9112×10^{2}	$\mathbf{3.7030 \times 10^{2}}$	4.3141×10^{2}
F15	8.1545×10^{2}	1.2740×10^{3}	1.5704×10^{3}	8.7427×10^{2}	$\mathbf{6.3787 \times 10^{2}}$	7.0925×10^{2}
F16	2.0057×10^{2}	2.0081×10^{2}	2.0121×10^{2}	2.0046×10^{2}	$\mathbf{2.0041 \times 10^{2}}$	2.0043×10^{2}
F17	$\mathbf{3.2301 \times 10^{2}}$	3.7425×10^{2}	3.6092×10^{2}	3.3686×10^{2}	3.3341×10^{2}	3.3634×10^{2}
F18	$\mathbf{4.2450 \times 10^{2}}$	4.7709×10^{2}	4.6342×10^{2}	4.4208×10^{2}	4.4060×10^{2}	4.3527×10^{2}
F19	$\mathbf{5.0095 \times 10^{2}}$	5.0670×10^{2}	5.1151×10^{2}	5.0180×10^{2}	5.0190×10^{2}	$\mathbf{5.0159 \times 10^{2}}$
F20	6.0319×10^{2}	6.0396×10^{2}	6.0341×10^{2}	6.0332×10^{2}	$\mathbf{6.0318 \times 10^{2}}$	6.0319×10^{2}
F21	1.1002×10^{3}	1.0606×10^{3}	1.1131×10^{3}	1.1002×10^{3}	$\mathbf{1.0902 \times 10^{3}}$	1.1002×10^{3}
F22	1.9130×10^{3}	2.2000×10^{3}	2.3646×10^{3}	1.5850×10^{3}	$\mathbf{1.3574 \times 10^{3}}$	1.5500×10^{3}
F23	2.1078×10^{3}	2.5235×10^{3}	2.6434×10^{3}	2.2005×10^{3}	$\mathbf{1.8884 \times 10^{3}}$	1.9143×10^{3}
F24	1.2204×10^{3}	1.2212×10^{3}	1.2259×10^{3}	1.1974×10^{3}	$\mathbf{1.1723 \times 10^{3}}$	1.1754×10^{3}
F25	1.3194×10^{3}	1.3247×10^{3}	1.3192×10^{3}	1.2868×10^{3}	1.2800×10^{3}	$\mathbf{1.2702 \times 10^{3}}$
F26	1.4499×10^{3}	1.3972×10^{3}	1.3988×10^{3}	1.3964×10^{3}	1.3644×10^{3}	$\mathbf{1.3616 \times 10^{3}}$
F27	1.7437×10^{3}	1.8893×10^{3}	1.9051×10^{3}	1.8341×10^{3}	$\mathbf{1.7000 \times 10^{3}}$	1.7078×10^{3}
F28	2.0522×10^{3}	2.1678×10^{3}	2.0683×10^{3}	1.9700×10^{3}	$\mathbf{1.9667 \times 10^{3}}$	2.0163×10^{3}

Bold indicates the optimal solution to the test function.

As shown in Table 3, all algorithms are tested in 30 dimensions. Among the 28 tested functions, OPGTOS1 achieves 10 better solutions and 2 identical solutions, while OPGTOS2 achieves 8 better solutions and 1 identical solution compared to other algorithms. Compared with the GTO algorithm, OPGTOS1 has achieved 20 better solutions and 2 identical solutions, and OPGTOS2 has achieved 21 better solutions and 1 identical solution. In order to analyze the iterative process of different algorithms more clearly, the experiment records the data of the optimal solution every 100 iterations in detail in 30 dimensions and draws it as Figures 6–8. In these figures, the horizontal axis is the number of iterations of the algorithm and the vertical axis is the optimal solution for the corresponding number of iterations.

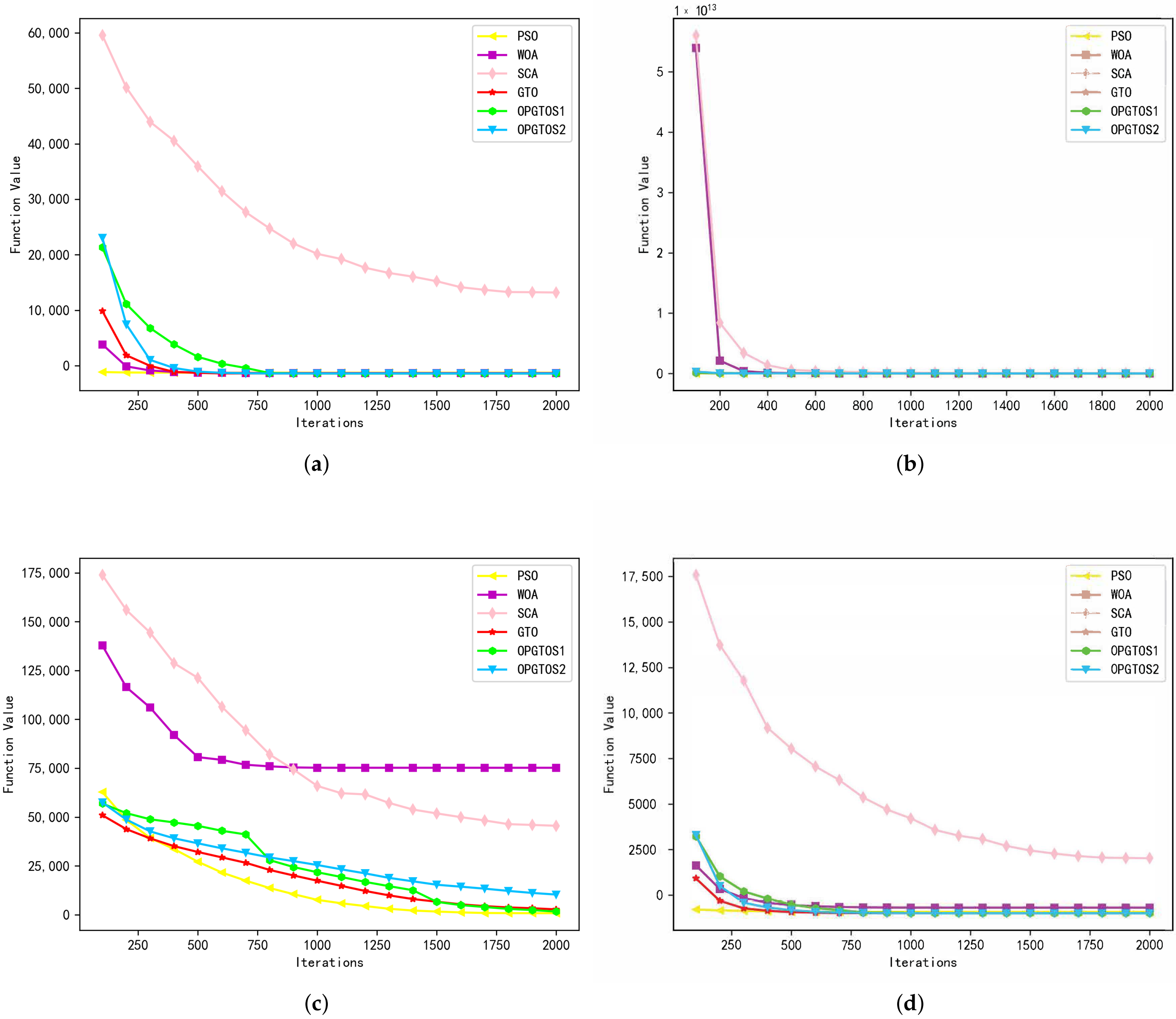
According to Table 3, and Figure 6, the experimental results show that multi-group merge communication strategy is the best choice for finding the optimal value of a unimodal function. The test results of functions F1, F3, F4 and F5 show that the optimal value searched by OPGTOS1 is much smaller than the traditional optimization algorithm and the original GTO algorithm. Especially in function F4, the convergence speed of OPGTOS1 increases significantly around 700 generations and 1400 iterations, because the algorithm performs group merging every 700 generations. Experiments show that the merged group not only inherits the advantages of the original group, but also achieves effective mutation.

Table 3. Simulation results of CEC2013 functions in 30 dimensions.

Functions	PSO	WOA	SCA	GTO	OPGTOS1	OPGTOS2
F1	-1.2584×10^{3}	-1.3351×10^{3}	1.3195×10^{4}	$-\mathbf{1.4000 \times 10^{3}}$	$-\mathbf{1.4000 \times 10^{3}}$	$-\mathbf{1.4000 \times 10^{3}}$
F2	2.2255×10^{6}	7.3459×10^{7}	1.8283×10^{8}	$\mathbf{1.7558 \times 10^{6}}$	1.9812×10^{6}	4.0057×10^{6}
F3	7.3140×10^{9}	3.1777×10^{10}	5.1810×10^{10}	2.1691×10^{9}	$\mathbf{1.3189 \times 10^{9}}$	1.9336×10^{9}
F4	7.6646×10^{2}	7.5215×10^{4}	4.5524×10^{4}	2.7143×10^{3}	$\mathbf{1.5924 \times 10^{3}}$	1.0304×10^{4}
F5	-9.3156×10^{2}	-6.9280×10^{2}	2.0185×10^{3}	$-\mathbf{1.0000 \times 10^{3}}$	$-\mathbf{1.0000 \times 10^{3}}$	-9.9999×10^{2}
F6	-8.3101×10^{2}	-7.2677×10^{2}	1.2517×10^{2}	-8.3725×10^{2}	$-\mathbf{8.3792 \times 10^{2}}$	-8.3341×10^{2}
F7	-6.6617×10^{2}	-3.6581×10^{2}	-5.8299×10^{2}	-6.5157×10^{2}	-6.7202×10^{2}	$-\mathbf{6.7829 \times 10^{2}}$
F8	$-\mathbf{6.7905 \times 10^{2}}$	-6.7902×10^{2}	-6.7900×10^{2}	-6.7903×10^{2}	-6.7904×10^{2}	-6.7901×10^{2}
F9	$-\mathbf{5.6993 \times 10^{2}}$	-5.6310×10^{2}	-5.5912×10^{2}	-5.6529×10^{2}	-5.6846×10^{2}	-5.6716×10^{2}
F10	-4.5624×10^{2}	-2.6043×10^{2}	1.4309×10^{3}	$-\mathbf{4.9968 \times 10^{2}}$	-4.9937×10^{2}	-4.9778×10^{2}
F11	-8.5753×10	1.0837×10^{2}	-2.3930×10	-1.3550×10^{2}	-1.4663×10^{2}	$-\mathbf{2.0006 \times 10^{2}}$
F12	1.1691×10	2.4957×10^{2}	1.0852×10^{2}	-1.2259×10	1.3963×10	$-\mathbf{5.0813 \times 10}$
F13	1.5989×10^{2}	3.3666×10^{2}	2.0759×10^{2}	1.4414×10^{2}	1.0630×10^{2}	$\mathbf{9.5097 \times 10}$
F14	3.7839×10^{3}	4.9830×10^{3}	7.2659×10^{3}	3.7392×10^{3}	3.0732×10^{3}	$\mathbf{3.0626 \times 10^{3}}$
F15	4.3491×10^{3}	6.3085×10^{3}	7.7583×10^{3}	4.7744×10^{3}	$\mathbf{3.9834 \times 10^{3}}$	4.7250×10^{3}
F16	2.0137×10^{2}	2.0225×10^{2}	2.0274×10^{2}	2.0140×10^{2}	$\mathbf{2.0117 \times 10^{2}}$	2.0129×10^{2}
F17	$\mathbf{5.0260 \times 10^{2}}$	9.6147×10^{2}	8.5302×10^{2}	6.8331×10^{2}	6.3125×10^{2}	6.1835×10^{2}
F18	$\mathbf{5.8747 \times 10^{2}}$	1.0157×10^{3}	9.2840×10^{2}	7.5694×10^{2}	7.9862×10^{2}	7.0279×10^{2}
F19	$\mathbf{5.0917 \times 10^{2}}$	5.8332×10^{2}	8.8438×10^{3}	5.5840×10^{2}	5.2631×10^{2}	5.2416×10^{2}
F20	6.1458×10^{2}	6.1490×10^{2}	6.1419×10^{2}	$\mathbf{6.1333 \times 10^{2}}$	6.1337×10^{2}	6.1334×10^{2}
F21	1.0419×10^{3}	1.2307×10^{3}	2.7498×10^{3}	1.0518×10^{3}	1.1077×10^{3}	$\mathbf{1.0087 \times 10^{3}}$
F22	5.8150×10^{3}	7.5899×10^{3}	8.6594×10^{3}	5.0295×10^{3}	$\mathbf{4.3238 \times 10^{3}}$	4.7720×10^{3}
F23	6.5847×10^{3}	7.4593×10^{3}	9.0820×10^{3}	6.1455×10^{3}	$\mathbf{5.8594 \times 10^{3}}$	6.3281×10^{3}
F24	1.3008×10^{3}	1.3146×10^{3}	1.3204×10^{3}	1.3073×10^{3}	$\mathbf{1.2980 \times 10^{3}}$	1.3010×10^{3}
F25	1.4240×10^{3}	1.4270×10^{3}	1.4331×10^{3}	1.4167×10^{3}	$\mathbf{1.4055 \times 10^{3}}$	1.4117×10^{3}
F26	1.5502×10^{3}	1.5580×10^{3}	1.4167×10^{3}	1.4669×10^{3}	$\mathbf{1.4001 \times 10^{3}}$	1.4097×10^{3}
F27	2.4895×10^{3}	2.6709×10^{3}	2.6966×10^{3}	2.5753×10^{3}	2.4954×10^{3}	$\mathbf{2.4638 \times 10^{3}}$
F28	4.1396×10^{3}	6.0367×10^{3}	4.2822×10^{3}	4.3647×10^{3}	4.0969×10^{3}	$\mathbf{3.3420 \times 10^{3}}$

Bold indicates the optimal solution to the test function.

OPGTOS1 and OPGTO2 perform extremely well when solving multimodal functions. For functions F11 and F12, OPGTOS2 not only has the fastest convergence speed and convergence stability, but also achieves the optimal value far smaller than other algorithms, which indicates that the algorithm has strong global exploration ability. In the mutation and update phase of strategy 2, many individuals will learn from the optimal individual, which is beneficial for the algorithm to quickly approach the optimal solution. In the test functions F13 and F14, compared with other algorithms, both OPGTOS1 and OPGTO2 achieve better optimal solutions. For functions F15 and F16, OPGTOS1 has the fastest convergence speed and the most suitable solution. It can be seen from Figure 7 that strategy 1 converges quickly when merging groups, which indicates that the merging operation brings more possibilities for the algorithm to find the optimal solution.



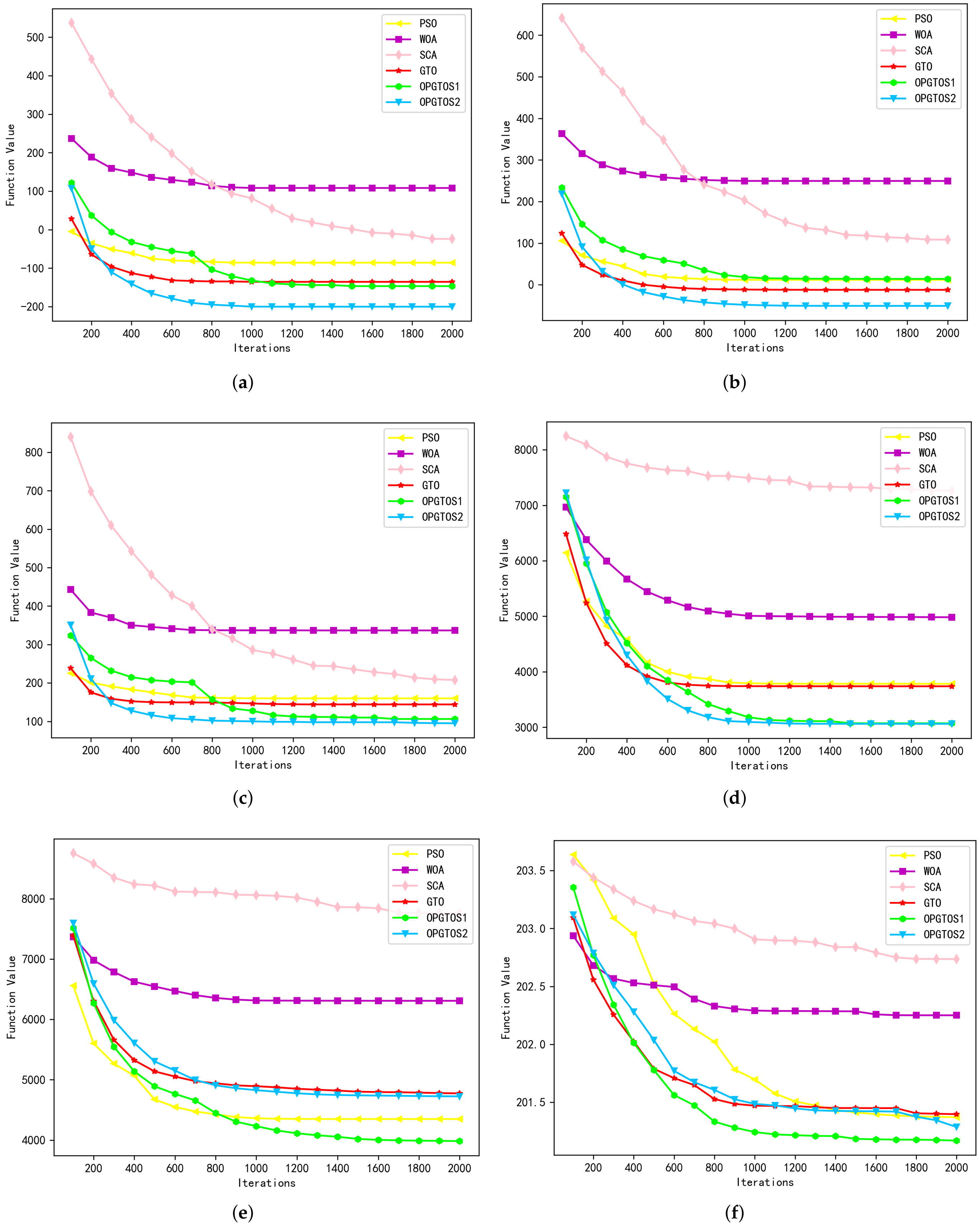
(**a**) (**b**)

(**c**) (**d**)

Figure 6. Simulation results of unimodal functions in 30 dimensions. (**a**) F1. (**b**) F3. (**c**) F4. (**d**) F5.

F21–F28 belong to more complex combinatorial functions, and in the experimental results of these functions, OPGTOS1 and OPGTOS2 have achieved all the leading positions. In functions F22–F26, the solution found by OPGTOS1 is much smaller than other algorithms. It can be seen from the Figure 8 that OPGTOS1 will have some rapid declines on the convergence curves of these functions, which indicates that the Gauss-Cauchy mutation strategy has achieved good results, and the algorithm has found a better solution. Among the functions F27 and F28, OPGTOS2 has the fastest convergence speed, and the obtained optimal solution is also the smallest.

In Table 4, all algorithms are tested in 50 dimensions. Compared to other algorithms, OPGTOS1 achieves 12 better solutions and 1 identical solution, and OPGTOS2 achieves 7 better solutions. Compared with the GTO algorithm, OPGTOS1 achieved 20 better solutions and 1 identical solution, and OPGTOS2 achieved 18 better solutions. OPGTOS1 shows far better performance than other optimization algorithms on single-modal functions and complex combination functions, and OPGTOS2 has better performance on multi-modal functions.



(**a**)

(**b**)

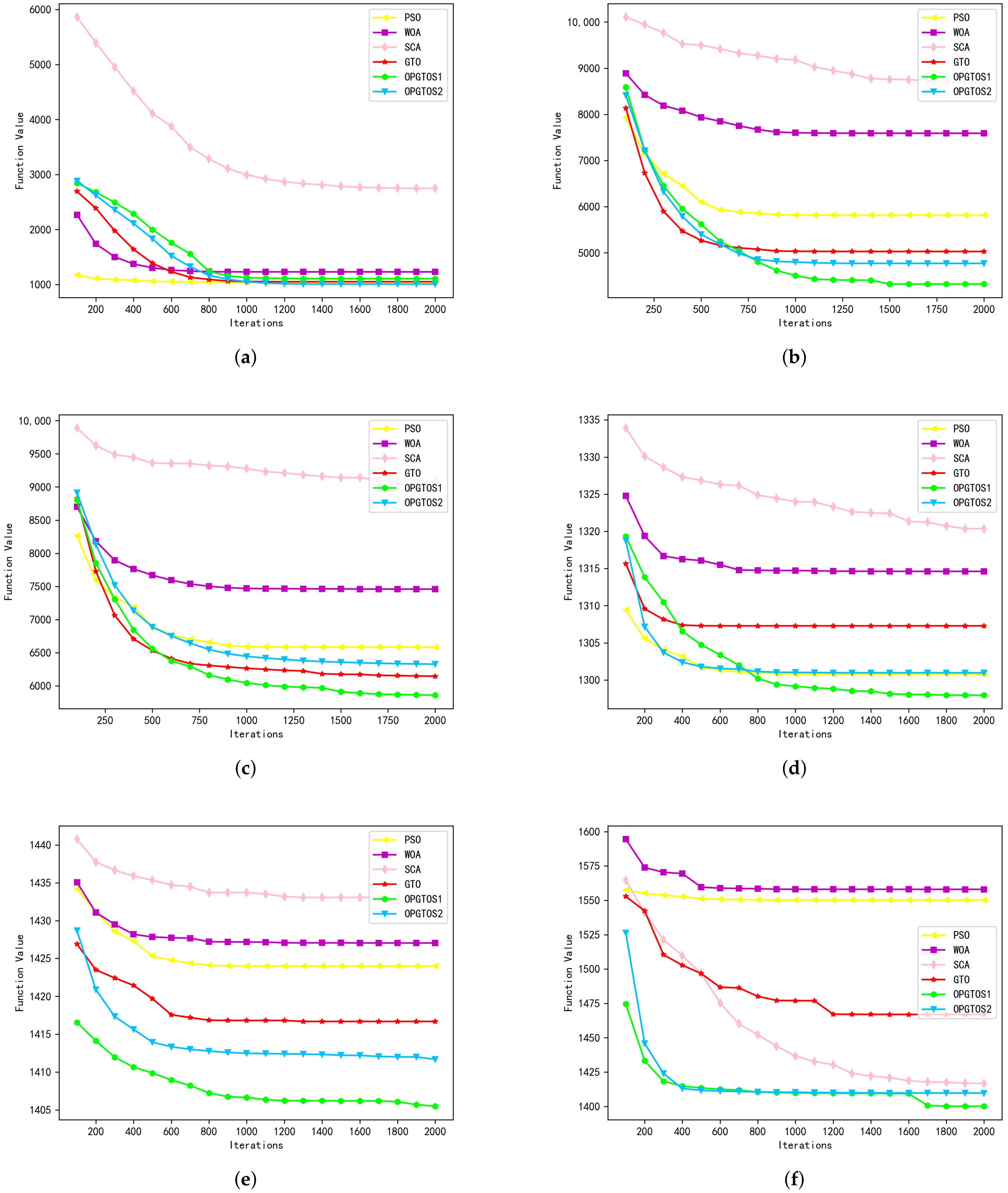
(**c**)

(**d**)

(**e**)

(**f**)

Figure 7. Simulation results of basic multimodal functions in 30 dimensions. (**a**) F11. (**b**) F12. (**c**) F13. (**d**) F14. (**e**) F15. (**f**) F16.



(**a**)

(**b**)

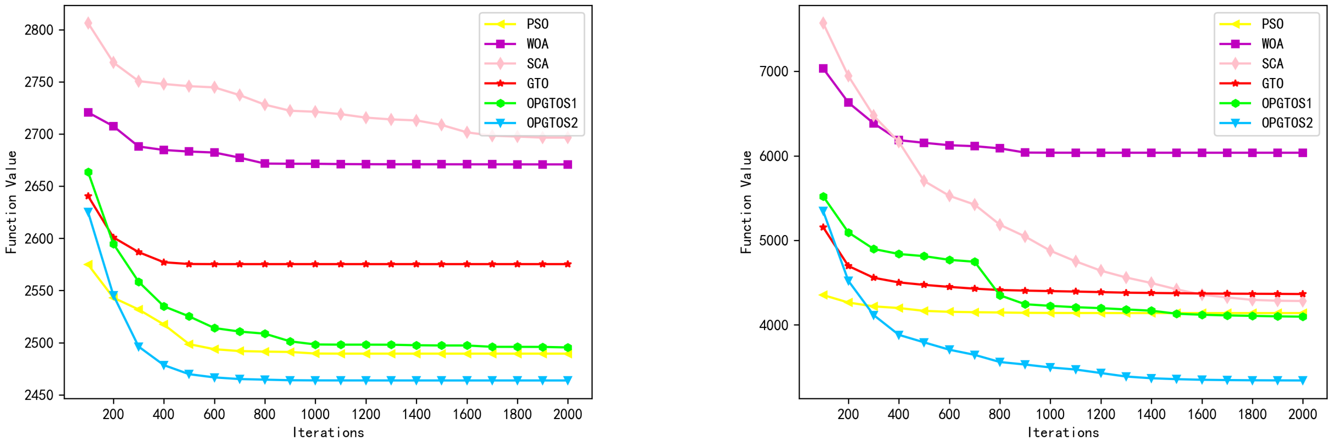
(**c**)

(**d**)

(**e**)

(**f**)

Figure 8. *Cont.*



(**g**)

(**h**)

Figure 8. Simulation results of composition functions in 30 dimensions. (**a**) F21. (**b**) F22. (**c**) F23. (**d**) F24. (**e**) F25. (**f**) F26. (**g**) F27. (**h**) F28.

Table 4. Simulation results of CEC2013 functions in 50 dimensions.

Functions	PSO	WOA	SCA	GTO	OPGTOS1	OPGTOS2
F1	-1.2309×10^{3}	-7.9109×10^{2}	3.2256×10^{4}	$-\mathbf{1.4000 \times 10^{3}}$	$-\mathbf{1.4000 \times 10^{3}}$	-1.4000×10^{3}
F2	8.7200×10^{6}	1.2286×10^{8}	6.0050×10^{8}	6.1892×10^{6}	$\mathbf{5.6450 \times 10^{6}}$	1.3470×10^{7}
F3	8.1706×10^{9}	5.5928×10^{10}	1.3181×10^{11}	6.7839×10^{9}	$\mathbf{4.8846 \times 10^{9}}$	7.3252×10^{9}
F4	6.9780×10^{3}	7.6695×10^{4}	7.6340×10^{4}	1.1636×10^{4}	$\mathbf{1.0673 \times 10^{4}}$	2.1880×10^{4}
F5	-9.5832×10^{2}	-3.1282×10^{2}	3.6436×10^{3}	$-\mathbf{1.0000 \times 10^{3}}$	-1.0000×10^{3}	-9.9999×10^{2}
F6	-7.8109×10^{2}	-5.4184×10^{2}	1.7422×10^{3}	$-\mathbf{8.0498 \times 10^{2}}$	-8.0446×10^{2}	-7.9179×10^{2}
F7	-6.7300×10^{2}	4.9938×10^{2}	-5.4663×10^{2}	-6.2500×10^{2}	-6.7626×10^{2}	$-\mathbf{6.7978 \times 10^{2}}$
F8	$-\mathbf{6.7885 \times 10^{2}}$	-6.7883×10^{2}	-6.7880×10^{2}	-6.7883×10^{2}	-6.7883×10^{2}	-6.7884×10^{2}
F9	$-\mathbf{5.4053 \times 10^{2}}$	-5.2915×10^{2}	-5.2499×10^{2}	-5.3580×10^{2}	-5.3903×10^{2}	-5.3883×10^{2}
F10	-3.9532×10^{2}	1.2906×10^{2}	3.9252×10^{3}	$-\mathbf{4.9399 \times 10^{2}}$	-4.9172×10^{2}	-4.6296×10^{2}
F11	1.0823×10^{2}	4.6861×10^{2}	3.7045×10^{2}	1.0867×10^{2}	4.3230×10	**8.6476**
F12	2.7200×10^{2}	6.6069×10^{2}	5.1083×10^{2}	2.7110×10^{2}	3.0672×10^{2}	$\mathbf{1.9867 \times 10^{2}}$
F13	4.6241×10^{2}	8.5380×10^{2}	5.9303×10^{2}	5.6585×10^{2}	4.3875×10^{2}	$\mathbf{4.0373 \times 10^{2}}$
F14	7.2073×10^{3}	1.0615×10^{4}	1.3649×10^{4}	6.5409×10^{3}	5.7605×10^{3}	$\mathbf{5.6061 \times 10^{3}}$
F15	9.0707×10^{3}	1.1938×10^{4}	1.4869×10^{4}	9.5541×10^{3}	$\mathbf{8.1922 \times 10^{3}}$	9.6221×10^{3}
F16	2.0224×10^{2}	2.0272×10^{2}	2.0378×10^{2}	2.0206×10^{2}	2.0206×10^{2}	$\mathbf{2.0203 \times 10^{2}}$
F17	$\mathbf{7.9120 \times 10^{2}}$	1.4946×10^{3}	1.3869×10^{3}	1.0208×10^{3}	1.0331×10^{3}	9.9202×10^{2}
F18	$\mathbf{8.2142 \times 10^{2}}$	1.5835×10^{3}	1.4754×10^{3}	1.1978×10^{3}	1.1768×10^{3}	1.0739×10^{3}
F19	$\mathbf{5.3004 \times 10^{2}}$	8.0833×10^{2}	6.4222×10^{4}	6.5196×10^{2}	5.7426×10^{2}	5.9644×10^{2}
F20	6.2389×10^{2}	6.2472×10^{2}	6.2420×10^{2}	6.2327×10^{2}	$\mathbf{6.2303 \times 10^{2}}$	6.2392×10^{2}
F21	1.6093×10^{3}	2.5666×10^{3}	4.7081×10^{3}	1.6079×10^{3}	$\mathbf{1.5936 \times 10^{3}}$	1.6222×10^{3}
F22	1.1971×10^{4}	1.3664×10^{4}	1.5815×10^{4}	9.3579×10^{3}	$\mathbf{7.6946 \times 10^{3}}$	7.9538×10^{3}
F23	1.2290×10^{4}	1.4687×10^{4}	1.6388×10^{4}	1.2314×10^{4}	$\mathbf{1.0373 \times 10^{4}}$	1.1825×10^{4}
F24	1.3894×10^{3}	1.4260×10^{3}	1.4317×10^{3}	1.3966×10^{3}	$\mathbf{1.3824 \times 10^{3}}$	1.3883×10^{3}
F25	1.5269×10^{3}	1.5327×10^{3}	1.5530×10^{3}	1.5279×10^{3}	$\mathbf{1.4976 \times 10^{3}}$	1.5109×10^{3}
F26	1.6460×10^{3}	1.6469×10^{3}	1.6219×10^{3}	1.5892×10^{3}	$\mathbf{1.5046 \times 10^{3}}$	1.5335×10^{3}
F27	3.3618×10^{3}	3.6134×10^{3}	3.6892×10^{3}	3.3915×10^{3}	$\mathbf{3.3555 \times 10^{3}}$	3.3588×10^{3}
F28	4.2321×10^{3}	9.9957×10^{3}	7.0825×10^{3}	4.8026×10^{3}	4.7862×10^{3}	$\mathbf{3.7541 \times 10^{3}}$

Bold indicates the optimal solution to the test function.

We take the average solution time of each algorithm for all functions of CEC2013 as the resource consumption of the algorithm. In Table 5, PSO, WOA, and SCA have simple exploration and development mechanisms, so the average solution time of the algorithm is shorter, but the optimal solution is poor. GTO has more complex operations, which leads to longer solution times. Compared with GTO, our proposed OPGTOS1 and OPGTOS2 have more mutation and update mechanisms. Although the resource consumption of the algorithm increases, a more suitable optimal solution is obtained.

Table 5. Average solution time on CEC2013 in 30 dimensions.

Functions	PSO	WOA	SCA	GTO	OPGTOS1	OPGTOS2
Runtime(s)	44.015153	39.547344	53.934907	112.934504	163.106233	171.35796

4.2. Simulation Results of OPGTO in 3D Localization

In this section, OPGTO is applied to optimize node localization in a piece of actual 3D terrain. The terrain chosen for the experiment is Bijia Mountain in Qingdao. The terrain of this mountain is complex and rugged with peaks, ravines and valleys. It is very suitable for simulating node localization in real terrain and can also test various performances of the algorithm more efficiently. The 3D terrain used in this paper is first extracted using Google Maps and then drawn using Matlab, as shown in Figure 7. When a node in WSN communicates with other nodes in the area, the node may not be able to communicate properly. The reason may be that the communication distance exceeds the transmission distance of the node or the signal will be blocked by objects on the mountain. Therefore, we need to first detect whether the nodes can communicate normally. In three-dimensional space, the communication range of sensor nodes resembles a sphere. If the communication distance between two nodes is greater than the transmission distance of the nodes, it indicates that the communication ranges of the two nodes do not cover each other. As a result, the nodes cannot communicate properly. Then using the LOS method, we can effectively determine whether there are obstacles between nodes. Figure 9 shows the result of this process, x, y and z represent the three-dimensional space coordinates of a point in the terrain.

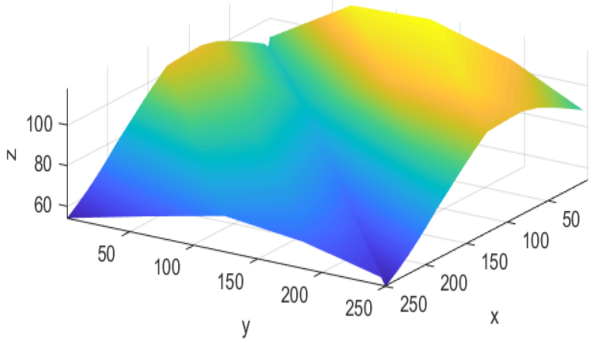


Figure 9. 3D terrain of Bijia Mountain.

In Figure 10, 25 anchor nodes and a randomly generated unknown node are included. The green square represents an unknown node, the black marker represents the anchor node that the unknown node can communicate with normally, and the red marker represents the anchor node that the unknown node cannot reach. In order to represent the localization error, the fitness function calculation formula is proposed in this paper as follows:

$$f(x,y) = min(\sqrt{\frac{error}{N-1}}) \tag{27}$$

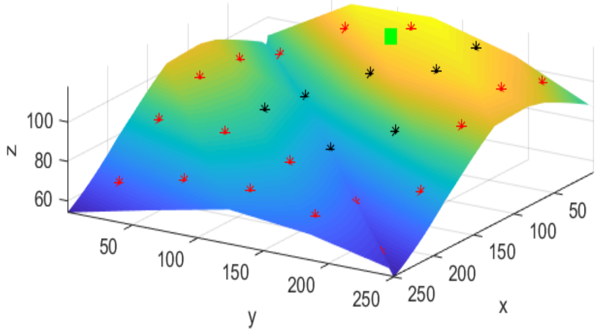


Figure 10. Communication status between nodes.

In Equation (27), *error* is calculated by Equation (24) and N represents the number of anchor nodes that the unknown node can communicate normally. The experiment sets 25 anchor nodes, which are uniformly distributed in a 3D terrain of size 100 × 250 × 250. To comprehensively test the performance of OPGTO in TDOA, we randomly sprinkle unknown nodes into the terrain and locate them. At the same time, we change the signal transmission distance of the node and detect the change of localization error under different transmission distances. Signal transmission distance is a crucial factor for normal communication between sensor nodes. If the transmission distance of the signal is too long, the node will consume more energy to transmit information to anchor nodes. In addition, for anchor nodes that have established communication connections. Too long transmission distance will inevitably increase the time error of signal arrival, which will have an impact on the final node location accuracy. If the transmission distance is too short, the node cannot communicate with any anchor node and send its own location information.

In Figure 11, the x-axis represents the detection radius of the node and the y-axis represents the corresponding localization error. Figure 11 shows the errors of TDOA, PSO, WOA, SCA, GTO, OPGTOS1 and OPGTOS2 under different detection radius, and we can see that the error rate of the optimization algorithms is much smaller than TDOA. Figure 12 shows the positioning errors of PSO, GTO, OPGTOS1 and OPGTOS2. When the detection radius is between 60 m and 120 m, the localization error rate of OPGTOS1 is smaller than other algorithms. In addition, its variation range is smaller than other algorithms, which indicates that the stability of the algorithm is better.

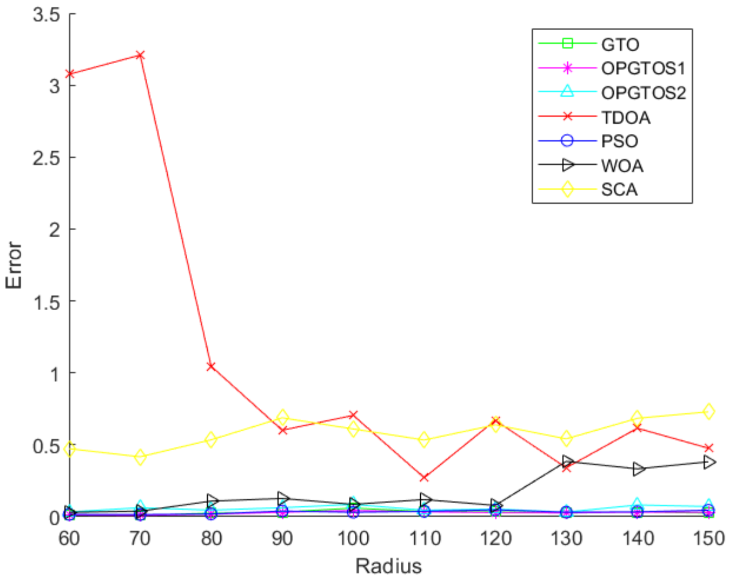


Figure 11. The localization error of TDOA and the optimization algorithms.

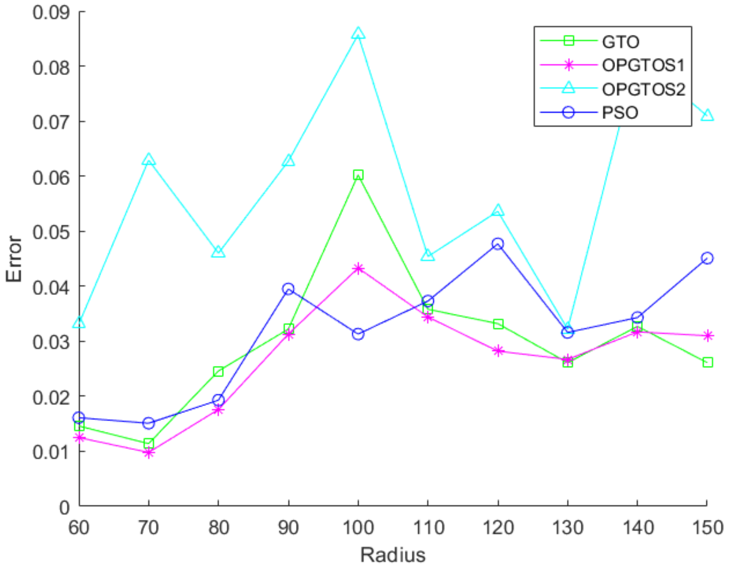


Figure 12. The localization error of PSO, GTO, OPGTOS1 and OPGTOS2.

5. Discussion

The main innovations of the OPGTO algorithm proposed in this paper are opposition-based learning and multi-group communication strategies. From Tables 2–4, it can be found that OPGTO leads GTO in solving multimodal and combinatorial functions. This illustrates that our proposed strategy improves the global exploration ability of the algorithm. Compared with PPSO [34], PWOA [40], PCSCA [9], we innovatively combine the strategy of opposition-based learning, which has been widely used to extend reinforcement learning and neural networks. It helps us to achieve better optimal solution and faster convergence curve in CEC2013, which further reduces the error in 3D localization of WSN. However, the running speed of OPGTO is limited to a certain extent due to more complicated operations and strategies.

There are many other improvement strategies for GTO as well. For example, two major improvements to MIGTO [43] are Explorative Gorilla with Adaptive Mutation Mechanism (EGAMM) and Gorilla Memory Saving Technology. Different from MIGTO, we propose two communication strategies between groups based on the idea of parallel. Communication strategies play a huge role in the exploration and development of particle populations, especially when algorithm faces complex optimization problems.

6. Conclusions and Feature Works

In this paper, we propose a GTO algorithm based on opposition-based learning and parallel strategy. OPGTO has demonstrated extraordinary ability in terms of convergence speed and global exploration. We conduct extensive experiments on the CEC2013 benchmark functions in 10, 30, and 50 dimensions, and the simulation results demonstrate OPGTO's particularly robust exploration capabilities. Compared with the existing traditional optimization algorithms PSO, WOA, SCA and the original GTO algorithm, OPGTO has more appropriate global optimal solutions and faster convergence. Especially in solving more complex multimodal functions and combinatorial functions, OPGTO has shown far superior performance than other algorithms. Finally, we apply the OPGTO algorithm to the node localization of WSN in 3D real terrain and obtain smaller localization errors than the TDOA method, which further verifies the ability of the new algorithm to solve optimization problems. However, OPGTO has more operations in the way of complex mutations and updates. This results in the algorithm requiring more computing power consumption and resource occupation. In the future, we will consider using compact [9,27,44] strategies to reduce the time complexity of the algorithm. The compact method is a population-free method, which is essentially a probabilistic model based on the Estimated Distribution Algorithm (EDA). During the execution of the algorithm, the distribution model will replace the population to perform related operations, and the continuous search of the exploration space is realized by updating the distribution model. Furthermore, we will combine some promising technologies [44–47] to make more improvements on the existing algorithms and expand more application fields [48–50].

Author Contributions: Conceptualization, Q.L., S.-C.C. and J.-S.P.; Methodology, Q.L., S.-C.C., A.L., J.-S.P. and Q.Y.; Validation, S.-C.C., A.L. and Q.Y.; Writing—original draft, Q.L.; Writing—review & editing, Q.L., S.-C.C., A.L., J.-S.P. and Q.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Zhou, A.; Qu, B.Y.; Li, H.; Zhao, S.Z.; Suganthan, P.N.; Zhang, Q. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm Evol. Comput.* **2011**, *1*, 32–49. [CrossRef]
2. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
3. Sun, C.; Zeng, J.; Pan, J.; Xue, S.; Jin, Y. A new fitness estimation strategy for particle swarm optimization. *Inf. Sci.* **2013**, *221*, 355–370. [CrossRef]
4. Wang, H.; Sun, H.; Li, C.; Rahnamayan, S.; Pan, J.S. Diversity enhanced particle swarm optimization with neighborhood search. *Inf. Sci.* **2013**, *223*, 119–135. [CrossRef]
5. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
6. Mafarja, M.M.; Mirjalili, S. Hybrid whale optimization algorithm with simulated annealing for feature selection. *Neurocomputing* **2017**, *260*, 302–312. [CrossRef]
7. Kaur, G.; Arora, S. Chaotic whale optimization algorithm. *J. Comput. Des. Eng.* **2018**, *5*, 275–284. [CrossRef]
8. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
9. Zhang, S.; Fan, F.; Li, W.; Chu, S.C.; Pan, J.S. A parallel compact sine cosine algorithm for TDOA localization of wireless sensor network. *Telecommun. Syst.* **2021**, *78*, 213–223. [CrossRef]
10. Abd Elaziz, M.; Oliva, D.; Xiong, S. An improved opposition-based sine cosine algorithm for global optimization. *Expert Syst. Appl.* **2017**, *90*, 484–500. [CrossRef]
11. Yang, X.S. Flower pollination algorithm for global optimization. In Proceedings of the International conference on Unconventional Computing and Natural Computation, Orléan, France, 3–7 September 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 240–249.
12. Nabil, E. A modified flower pollination algorithm for global optimization. *Expert Syst. Appl.* **2016**, *57*, 192–203. [CrossRef]
13. Storn, R.; Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
14. Mininno, E.; Neri, F.; Cupertino, F.; Naso, D. Compact differential evolution. *IEEE Trans. Evol. Comput.* **2010**, *15*, 32–54. [CrossRef]
15. Meng, Z.; Pan, J.S.; Tseng, K.K. PaDE: An enhanced Differential Evolution algorithm with novel control parameter adaptation schemes for numerical optimization. *Knowl.-Based Syst.* **2019**, *168*, 80–99. [CrossRef]
16. Meng, Z.; Yang, C. Hip-DE: Historical population based mutation strategy in differential evolution with parameter adaptive mechanism. *Inf. Sci.* **2021**, *562*, 44–77. [CrossRef]
17. Chu, S.C.; Tsai, P.W.; Pan, J.S. Cat swarm optimization. In *Pacific Rim International Conference on Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 854–858.
18. Santosa, B.; Ningrum, M.K. Cat swarm optimization for clustering. In Proceedings of the 2009 International Conference of Soft Computing and Pattern Recognition, Malacca, Malaysia, 4–7 December 2009; pp. 54–59.
19. Pradhan, P.M.; Panda, G. Solving multiobjective problems using cat swarm optimization. *Expert Syst. Appl.* **2012**, *39*, 2956–2964. [CrossRef]
20. Guo, L.; Meng, Z.; Sun, Y.; Wang, L. Parameter identification and sensitivity analysis of solar cell models with cat swarm optimization algorithm. *Energy Convers. Manag.* **2016**, *108*, 520–528. [CrossRef]
21. Yang, X.S.; Gomi, A.H. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **2012**, *29*, 464–483. [CrossRef]
22. Mirjalili, S.; Mirjalili, S.M.; Yang, X.S. Binary bat algorithm. *Neural Comput. Appl.* **2014**, *25*, 663–681. [CrossRef]

23. Fister, I.; Yang, X.S.; Fong, S.; Zhuang, Y. Bat algorithm: Recent advances. In Proceedings of the 2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, 19–21 November 2014; pp. 163–167.
24. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization*; Technical Report-tr06; Computer Engineering Department, Engineering Faculty, Erciyes University: Kayseri, Turkey, 2005.
25. Chong, C.S.; Low, M.Y.; Sivakumar, A.I.; Gay, K.L. A bee colony optimization algorithm to job shop scheduling. In Proceedings of the 2006 Winter Simulation Conference, Monterey, CA, USA, 3–6 December 2006; pp. 1954–1961.
26. Duan, H.; Qiao, P. Pigeon-Inspired optimization: A new swarm intelligence optimizer for air robot path planning. *Int. J. Intell. Comput. Cybern.* **2014**, *7*, 24–37. [CrossRef]
27. Alazzam, H.; Sharieh, A.; Sabri, K.E. A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer. *Expert Syst. Appl.* **2020**, *148*, 113249. [CrossRef]
28. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [CrossRef]
29. Zhao, D.; Liu, L.; Yu, F.; Heidari, A.A.; Wang, M.; Liang, G.; Muhammad, K.; Chen, H. Chaotic random spare ant colony optimization for multi-threshold image segmentation of 2D Kapur entropy. *Knowl.-Based Syst.* **2021**, *216*, 106510. [CrossRef]
30. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
31. Joyce, T.; Herrmann, J.M. A review of no free lunch theorems, and their implications for metaheuristic optimisation. *Nat.-Inspired Algorithms Appl. Optim.* **2018**, *744*, 27–51.
32. Abdollahzadeh, B.; Gharehchopogh, F.S.; Mirjalili, S. Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. *Int. J. Intell. Syst.* **2021**, *36*, 5887–5958. [CrossRef]
33. Tizhoosh, H.R. Opposition-Based learning: A new scheme for machine intelligence. In Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), Vienna, Austria, 28–30 November 2005; Volume 1, pp. 695–701.
34. Chu, S.C.; Roddick, J.F.; Pan, J.S. A parallel particle swarm optimization algorithm with communication strategies. *J. Inf. Sci. Eng.* **2005**, *21*, 809–818.
35. Tsai, P.W.; Pan, J.S.; Chen, S.M.; Liao, B.Y. Enhanced parallel cat swarm optimization based on the Taguchi method. *Expert Syst. Appl.* **2012**, *39*, 6309–6319. [CrossRef]
36. Wu, C.I.; Kung, H.Y.; Chen, C.H.; Kuo, L.C. An intelligent slope disaster prediction and monitoring system based on 14 and ANP. *Expert Syst. Appl.* **2014**, *41*, 4554–4562. [CrossRef]
37. Boukerche, A.; Oliveira, H.A.; Nakamura, E.F.; Loureiro, A.A. Localization systems for wireless sensor networks. *IEEE Wirel. Commun.* **2007**, *14*, 6–12. [CrossRef]
38. Gustafsson, F.; Gunnarsson, F. Positioning using time-difference of arrival measurements. In Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'03), Hong Kong, China, 6–10 April 2003; Volume 6, p. VI-553.
39. Li, C.; Zhang, N.; Lai, X.; Zhou, J.; Xu, Y. Design of a fractional-order PID controller for a pumped storage unit using a gravitational search algorithm based on the Cauchy and Gaussian mutation. *Inf. Sci.* **2017**, *396*, 162–181. [CrossRef]
40. Chai, Q.W.; Chu, S.C.; Pan, J.S.; Hu, P.; Zheng, W.M. A parallel WOA with two communication strategies applied in DV-Hop localization method. *EURASIP J. Wirel. Commun. Netw.* **2020**, *2020*, 50. [CrossRef]
41. Benedetto, F.; Giunta, G.; Toscano, A.; Vegni, L. Dynamic LOS/NLOS statistical discrimination of wireless mobile channels. In Proceedings of the 2007 IEEE 65th Vehicular Technology Conference-VTC2007-Spring, Dublin, Ireland, 22–25 April 2007; pp. 3071–3075.
42. Liang, J.J.; Qu, B.Y.; Suganthan, P.N.; Hernandez-Diaz, A.G. *Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization*; Technical Report; Computational Intelligence Laboratory, Zhengzhou University: Zhengzhou, China; Nanyang Technological University: Singapore, 2013; Volume 201212, pp. 281–295.
43. Abdel-Basset, M.; El-Shahat, D.; Sallam, K.M.; Munasinghe, K. Parameter extraction of photovoltaic models using a memory-based improved gorilla troops optimizer. *Energy Convers. Manag.* **2022**, *252*, 115134. [CrossRef]
44. Neri, F.; Mininno, E.; Iacca, G. Compact particle swarm optimization. *Inf. Sci.* **2013**, *239*, 96–121. [CrossRef]
45. Sun, C.; Jin, Y.; Cheng, R.; Ding, J.; Zeng, J. Surrogate-Assisted cooperative swarm optimization of high-dimensional expensive problems. *IEEE Trans. Evol. Comput.* **2017**, *21*, 644–660. [CrossRef]
46. Chai, Q.W.; Chu, S.C.; Pan, J.S.; Zheng, W.M. Applying Adaptive and Self Assessment Fish Migration Optimization on Localization of Wireless Sensor Network on 3-D Te rrain. *J. Inf. Hiding Multim. Signal Process.* **2020**, *11*, 90–102.
47. Pérez-Solano, J.J.; Felici-Castell, S.; Soriano-Asensi, A.; Segura-Garcia, J. Time synchronization enhancements in wireless networks with ultra wide band communications. *Comput. Commun.* **2022**, *186*, 80–89. [CrossRef]
48. Deng, W.; Shang, S.; Cai, X.; Zhao, H.; Zhou, Y.; Chen, H.; Deng, W. Quantum differential evolution with cooperative coevolution framework and hybrid mutation strategy for large scale optimization. *Knowl.-Based Syst.* **2021**, *224*, 107080. [CrossRef]
49. Ibrahim, R.A.; Ewees, A.A.; Oliva, D.; Abd Elaziz, M.; Lu, S. Improved salp swarm algorithm based on particle swarm optimization for feature selection. *J. Ambient Intell. Humaniz. Comput.* **2019**, *10*, 3155–3169. [CrossRef]
50. Wang, G.G.; Cai, X.; Cui, Z.; Min, G.; Chen, J. High performance computing for cyber physical social systems by using evolutionary multi-objective optimization algorithm. *IEEE Trans. Emerg. Top. Comput.* **2017**, *8*, 20–30. [CrossRef]