

Article

Improved Dual-Center Particle Swarm Optimization Algorithm

Zhouxi Qin ¹ and Dazhi Pan ^{1,2,*}

¹ College of Mathematic and Information, China West Normal University, Nanchong 637009, China; zhouxiq@hotmail.com

² Sichuan Colleges and Universities Key Laboratory of Optimization Theory and Applications, Nanchong 637009, China

* Correspondence: pdzzj@cwnu.edu.cn

Abstract: This paper proposes an improved dual-center particle swarm optimization (IDCPSO) algorithm which can effectively improve some inherent defects of particle swarm optimization algorithms such as being prone to premature convergence and low optimization accuracy. Based on the in-depth analysis of the velocity updating formula, the most innovative feature is the vectorial decomposition of the velocity update formula of each particle to obtain three different flight directions. After combining these three directions, six different flight paths and eight intermediate positions can be obtained. This method allows the particles to search for the optimal solution in a wider space, and the individual extreme values are greatly improved. In addition, in order to improve the global extreme value, it is designed to construct the population virtual center and the optimal individual virtual center by using the optimal position and the current position searched by the particle. Combining the above strategies, an adaptive mutation factor that accumulates the coefficient of mutation according to the number of iterations is added to make the particle escape from the local optimum. By running the 12 typical test functions independently 50 times, the results show an average improvement of 97.9% for the minimum value and 97.7% for the average value. The IDCPSO algorithm in this paper is better than other improved particle swarm optimization algorithms in finding the optimum.

Keywords: particle swarm optimization; central particle; search path; mutation; extreme value

MSC: 68w50; 68T42; 90C59



Citation: Qin, Z.; Pan, D.

Improved Dual-Center Particle Swarm Optimization Algorithm.

Mathematics **2024**, *12*, 1698. <https://doi.org/10.3390/math12111698>

Academic Editor: Javier Sánchez

Received: 27 April 2024

Revised: 23 May 2024

Accepted: 26 May 2024

Published: 30 May 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Eberhart and Kennedy proposed a particle swarm optimization [1] (PSO) algorithm in 1995 based on the simulation of the foraging behavior of flocks of bird and fish. The particle swarm optimization technique developed has become one of the most effective methods for solving complex functional optimization problems. This algorithm relies on the population optimization mechanism, possesses excellent global search capability, can effectively deal with multi-peak optimization problems and has been widely used in the fields of science and engineering. Since it does not need to know the a priori knowledge of the objective function values, parameters, etc., it shows good adaptability and robustness in solving complex optimization problems.

However, PSO is easily trapped in the local optimal solution, which may lead to the stagnation of particle evolution. In addition, as the number of iterations increases, the convergence speed of the algorithm slows down. In order to enhance the performance of the PSO algorithm in optimization, many scholars have made a series of improvements. Some of these scholars have improved the inertia weights. Zhang et al. [2] proposed an adaptive method which determines the inertia weight of each particle in different dimensions according to the performance of each particle and the distance from its optimal position. Taherkhani et al. [3] proposed a new adaptive inertia weight adjusting approach

based on Bayesian techniques in PSO which is used to set up a sound tradeoff between the exploration and exploitation characteristics. Xinliang et al. [4] proposed a random walk autonomous group particle swarm optimization algorithm (RW-AGPSO) which introduced Levy flight and a dynamic adjustment weight strategy to balance exploration and exploitation. Yansong et al. [5] proposed a hybrid dynamic particle swarm optimization (HDPSO) algorithm by adaptively adjusting the inertia weight and introducing the coefficient of variation. A portion of scholars has grouped populations to learn. Hongwei et al. [6] proposed a cooperative hierarchical particle swarm optimization framework and designed contingency leadership, interactive cognition and self-directed utilization operators. Gou et al. [7] divided the whole population into three subgroups, quantified the individual differences by the competition coefficient of each particle and selected the specific evolution method according to it and the current fitness combined with a restart strategy to regenerate the corresponding particles and enhance the diversity of the population. Lai et al. [8] proposed a multi-population parallel PSO algorithm based on penetration. It can adaptively determine when, how many and from which subpopulation to which subpopulation particles travel. Xu et al. [9] proposed a two-swarm learning PSO (TSLPSO) algorithm based on different learning strategies. One sub-swarm constructs learning samples through DLS to guide the local search of particles, and the other constructs learning samples through comprehensive learning strategies to guide the global search. Chnoor [10] proposed to combine the PSO algorithm with a new meta-heuristic to accomplish the assignment of tasks by dividing the particles into a group leader and group members through their behaviors. Some other scholars have designed adaptive strategies in PSO algorithms. For instance, Aziz et al. [11] proposed a hybrid update sequence adaptive switching PSO (Switch-PSO) algorithm whose update strategy adaptively switches between two traditional iterative strategies according to the performance of the best individual of the particle swarm. Jiang et al. [12] used different parameter values to adjust the global and local search ability of the PSO algorithm. Tang et al. [13] proposed a PSO algorithm with adaptive update strategy which adaptively updates the three main control parameters of particles in SAPSO. Shifei et al. [14] proposed a dynamic quantum particle swarm optimization algorithm (DQPSO), designed the particle search ability factor and used it as feedback to realize the dynamic adjustment of the contraction–expansion (CE) coefficient. Yawen et al. [15] proposed a dual adaptive strategy to overcome the problem where the single learning factor and inertia weight cannot adjust the optimization process well when optimizing complex functions. There are also some improved particle swarm algorithms that incorporate strategies from other algorithms. Chen et al. [16] proposed a particle swarm optimization algorithm with crossover operation (PSOCO) which constructs an effective particle swarm guidance example by crossover operation on the best position of each particle's personal history. Tian et al. [17] proposed an improved particle swarm optimization and chaos-based initialization and robust update mechanism. Using logistic mapping and sigmoid inertia weight, they designed a maximum focusing distance while performing wavelet mutation to enhance the diversity of the population. Ren et al. [18] introduced the simplex algorithm (SA) in the fixed-point theory into the optimization of PSO so that the optimization problem of the objective function is transformed into the problem of solving a fixed-point equation set. Fuqiang et al. [19] introduced circle mapping and the sine cosine factor to better balance the global exploration ability and local development ability of the algorithm.

Although the PSO algorithm is a classic algorithm, experts and scholars have never stopped researching it. In 2023–2024, many scholars published papers on the improvement of the PSO algorithm. Bratislav et al. [20] proposed a modified particle swarm algorithm (MPSO) which does not consider the particle velocities but adopts a Quasi-Reverse Learning (QRL) method to update the optimal positions of the individuals. Sulaiman et al. [21] proposed a modified hybrid (PSO-SAO) algorithm based on PSO and Scent Agent Optimization (SAO) which incorporates the trailing mode of the SAO algorithm into the PSO framework to efficiently regulate the velocity update of the original PSO and continuously introduces intelligences to track the molecules of higher concentration, thus guiding the

particles of the PSO towards the optimal fitness position. Shiva Kumar Kannan and Urmila Diwekar [22] introduced an innovative PSO algorithm which combines Sobol and Halton random number sampling, resulting in improved convergence efficiency of the particles. Feng et al. [23] proposed particle swarm optimization based on an improved crowding distance (MOPSO-MCD) algorithm which is designed with an improved crowding distance (MCD) calculation method; it can more comprehensively evaluate the crowding relationship between individuals in the decision space and the goal space. In addition, an elite selection mechanism based on cosine similarity is combined with the offspring competition mechanism to further optimize the algorithm. Tian et al. [24] proposed a diversity-guided PSO algorithm with multi-level learning strategy (DPSO-MLS) which used chaotic opposition-based learning (OBL) combined with high-level and low-level learning mechanisms to make particles cover the entire search space and maintain its diversity. However, among the various improved PSO algorithms, there is still less literature on the in-depth analysis that can be performed from the bird's trajectory or further research literature on the dual-center particle swarm optimization (DCPSO) algorithm (Table 1).

Table 1. The available literature on improved PSO algorithms.

Publication	Algorithms
[2]	Bayesian PSO (BPSO) algorithm
[5]	Hybrid dynamic PSO (HDPSO) algorithm
[9]	Two-swarm learning PSO (TSLPSO) algorithm
[11]	Switching PSO (Switch-PSO) algorithm
[16]	PSO algorithm with crossover operation (PSOCO)
[25]	Dual-center PSO (DCPSO) algorithm

In order to solve the problems of the low optimization accuracy of the PSO algorithm and it being easy to fall into local optimal solutions, the contributions of this study are as follows:

1. The velocity update formula of the PSO algorithm is deeply analyzed, and its vector decomposition yields three different flight directions which are arranged and combined to obtain six flight routes that are different from each other and eight intermediate positions;
2. The optimal solution and the current solution searched by the particles are used to construct a weighted population virtual center and a weighted optimal individual virtual center. These two virtual centers are incorporated into the new way of updating the individual extremes and population extremes so that the two virtual centers follow the population to search for better positions;
3. Linearly decreasing inertia weights are used to further adjust the velocity update formula;
4. We determine whether the particle is caught in the local optimum and, if so, make the particle jump out of the local optimum based on the adaptive variance factor accrued from the number of iterations.

The rest of the paper is organized in the following manner. Section 2 introduces the basic concepts and formal definitions of the modified particle swarm optimization algorithm. Section 3 introduces the conventional dual-center particle swarm algorithm and the improvement strategies made in this paper. Section 4 shows that the algorithm of this paper has strong optimality finding ability by comparing it with the dual-center particle swarm optimization (DCPSO) algorithm, particle swarm optimization (PSO) algorithm, linear decreasing weight particle swarm optimization (LDWPSO) algorithm and adaptive particle swarm optimization (APSO) algorithm.

2. Particle Swarm Optimization Algorithm

Particle swarm optimization (PSO) [26,27] is to consider each individual in the population as a particle in space, ignoring mass and volume, so that these particles fly in the

search space at a certain speed. The flight speed can be dynamically adjusted according to the different locations of the individual or the population reached.

In 1998, Yuhui Shi and Russell Eberhart introduced inertia weight [28] into the basic particle swarm optimization algorithm and proposed to dynamically adjust the inertia weight to balance the global nature of convergence and convergence speed. M particles fly in a D -dimensional search space, and the position of each particle is a potential feasible solution; f denotes the value of the objective function; t denotes the number of current iterations; t_{max} denotes the maximum number of iterations; $X_i^{(t)} = (x_{i,1}^{(t)}, x_{i,2}^{(t)}, \dots, x_{i,D}^{(t)})$ denotes the position of the i -th particle in the t -th iteration; and $V_i^{(t)} = (v_{i,1}^{(t)}, v_{i,2}^{(t)}, \dots, v_{i,D}^{(t)})$ denotes the velocity of the i -th particle in the t -th iteration. Its velocity and position update equations are as follows:

$$v_{i,j}^{(t+1)} = w^{(t)} \times v_{i,j}^{(t)} + c_1 \times r_1 \times (p_{i,j} - x_{i,j}^{(t)}) + c_2 \times r_2 \times (p_{g,j} - x_{i,j}^{(t)}) \quad (1)$$

$$x_{i,j}^{(t+1)} = x_{i,j}^{(t)} + v_{i,j}^{(t+1)} \quad (2)$$

where c_1 and c_2 are the learning factors; r_1 and r_2 are pseudo-random numbers that obey the $[0, 1]$ uniform distribution and are independent of each other; $p_{i,j}$ is the optimal position currently searched by the i -th particle; and $p_{g,j}$ is the optimal position currently searched by the population. In order to accelerate the convergence of the algorithm, the PSO algorithm adopts a linearly decreasing weight w so that the weight decreases linearly from the maximum value of w_{max} to w_{min} , and its change is shown in Equation (3).

$$w^{(t)} = w_{max} - \frac{t \times (w_{max} - w_{min})}{t_{max}} \quad (3)$$

3. Improvement of Dual-Center Particle Swarm Optimization Algorithm

3.1. Dual-Center Particle Swarm Optimization Algorithm

In this paper, we adopt the velocity decomposition method from Reference [25] to increase the position searched by the particle. The modified particle swarm velocity update method is decomposed into three parts: $w^{(t)}v_{i,j}^{(t)}$, $c_1r_1(pbest_{i,j} - x_{i,j}^{(t)})$ and $c_2r_2(gbest_j - x_{i,j}^{(t)})$. As in Figure 1, when the particle i flies from the position of the t moment to the position of the $(t + 1)$ moment, from the point of view of fit vector decomposition, it embodies the particle flight route.

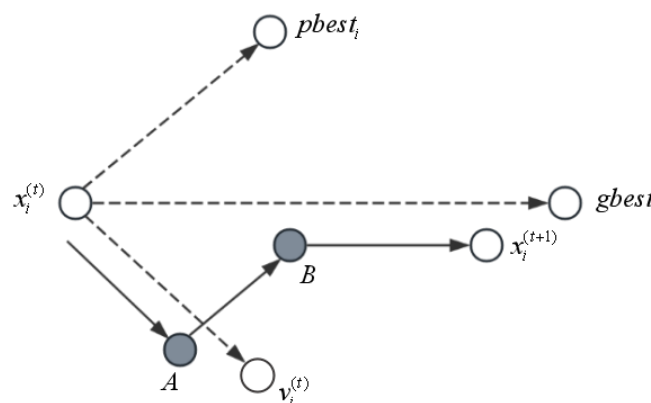


Figure 1. Vector decomposition diagram of velocity update formula.

The method changes the original flight path $x_i^{(t)} \rightarrow x_i^{(t+1)}$ to the zigzag motion path $x_i^{(t)} \rightarrow A \rightarrow B \rightarrow x_i^{(t+1)}$ preferred by birds, which means the search space can be covered

to a greater extent, and the individual extreme value and the population extreme value can be improved to a certain extent.

3.2. Diversified Design of Particle Motion Routes

In the conventional DCPSO algorithm, the update of particle position is based on the ordering of the particle swarm velocity update formula. In detail, it means that the particles fly according to Equation (1). The particle i firstly travels along the direction of the velocity of the t -th iteration, flying at a distance of $wv_i^{(t)}$, and arrives at A ; then travels along the direction of $pbest_i - x_i^{(t)}$, flying at a distance of $c_1r_1(pbest_i - x_i^{(t)})$, and arrives at B ; then finally travels along the direction of $gbest - x_i^{(t)}$, flying at a distance of $c_2r_2(gbest - x_i^{(t)})$, and arrives at the $(t + 1)$ moment position. This decomposition method has its inherent limitations. The particle swarm simulates the behavior of a flock of birds searching for food, and the flock should have a high degree of freedom on the foraging path. Therefore, the particles do not have to strictly follow the direction of flying first along the direction parallel to $v_i^{(t)}$, then along the direction parallel to $pbest - x_i^{(t)}$ and finally along the direction parallel to $gbest - x_i^{(t)}$, but, rather, they choose an arbitrary sequence of flight among the three different flight directions of $w^{(t)} - v_{ij}^{(t)}$, $c_1r_1(pbest_{i,j} - x_{ij}^{(t)})$ and $c_2r_2(gbest_j - x_{ij}^{(t)})$.

In this paper, the three different directions of particle movement are arranged and combined, and the results show that there are six different particle movement routes, as shown in Table 2. Therefore, in the conventional DCPSO algorithm, the flight route of particles (see Figure 1) is only one of many different routes, and the particles can have five other different movement routes. Figure 2 shows the six different routes of the i -th particle in the t moment of flight. In this way, during the flight of particles per unit of time, from only two (A , B) intermediate positions, to remove the repetition of the point, there are eight (A , B , C , D , E , F , G) intermediate positions, a great increase in the coverage of the search space, as far as possible, to avoid the phenomenon of early maturity of the swarm of particles.

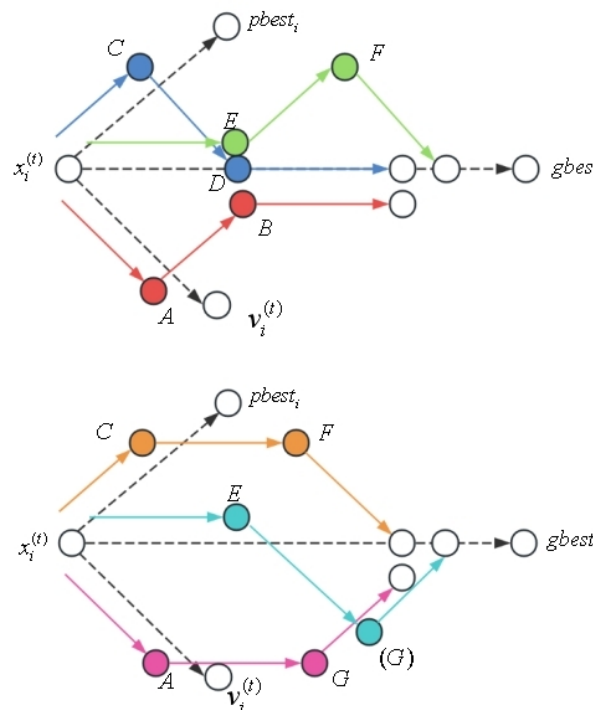


Figure 2. Six different update routes of particles.

Table 2. The update routes of particles.

Route	Direction 1	Direction 2	Direction 3	Mid-Position
1	$v_i^{(t)}$	$pbest_i - x_i^{(t)}$	$gbest - x_i^{(t)}$	A, B
2	$v_i^{(t)}$	$gbest - x_i^{(t)}$	$pbest_i - x_i^{(t)}$	A, G
3	$pbest_i - x_i^{(t)}$	$v_i^{(t)}$	$gbest - x_i^{(t)}$	C, D
4	$pbest_i - x_i^{(t)}$	$gbest - x_i^{(t)}$	$v_i^{(t)}$	C, F
5	$gbest - x_i^{(t)}$	$v_i^{(t)}$	$pbest_i - x_i^{(t)}$	E, G
6	$gbest - x_i^{(t)}$	$pbest_i - x_i^{(t)}$	$v_i^{(t)}$	E, F

3.3. Center Particle Design Improvement

However, the use of average values to construct the generalized center particle (GCP) and the special center particle (SCP) in Reference [25] may make them susceptible to extreme values and, when the population size is small, and if the two center particles do not fly, may reduce the search speed of the particle swarm. Therefore, in this paper, the weight of the i -th particle in the center construction is calculated by Equation (6), and then the positions of the individual center and the population center are adaptively controlled by Equations (4) and (5). Figure 3 shows a combination diagram of the particle position change and optimal solution evolution process, and the trigonometric function $f(x) = x \sin x \cos 2x - 2x \sin 3x + 3x \sin 4x$ is selected for testing. Where the hollow circle clearly records the position that the particle flies to in each iteration, the solid circle is the center point constructed in the algorithm flying to the position. As can be seen from Figure 3, since the center particle designed using the average value in Reference [25] does not have the ability to search and is susceptible to extreme values, it is more evenly dispersed over the solution interval, and it is more difficult to reach the global optimal position. On the other hand, the center particle of the algorithm in this paper is not only closer to the current optimal position but also has the ability to search and can easily search the optimal position in the later stage of the optimization search. The rate of convergence is also improved.

$$GCP^{(t)} = \sum_{i=1}^M R_i^{(t)} \times X_i^{(t)} \quad (4)$$

$$SCP^{(t)} = \sum_{i=1}^M R_i^{(t)} \times p_i \quad (5)$$

$$R_i^{(t)} = \frac{1}{M-1} \left[1 - \frac{f(X_i^{(t)})}{\sum_{m=1}^M f(X_m^{(t)})} \right] \quad (6)$$

In Equations (4)–(6), $GCP^{(t)}$ denotes the population virtual center constructed at the t time iteration; $SCP^{(t)}$ denotes the optimal individual virtual center constructed iteratively at the t time; $X_i^{(t)}$ denotes the position of the i -th particle in the t time iteration; p_i denotes the current optimal position of the i -th particle; and $R_i^{(t)}$ denotes the weight occupied by the i -th particle in the virtual center construction.

When the objective function f is smaller and the $R_i^{(t)}$ is larger, then the constructed population center is closer to the better position; similarly, the optimal individual center is closer to the better individual optimal position. The two centers constructed in this paper are virtual centers; when $f(GCP^{(t)}) < f(X_i^{(t)})$, they replace the individual optimal position with $GCP^{(t)}$, and, when $f(SCP^{(t)}) < f(p_i)$, they replace the individual optimal position with $SCP^{(t)}$.

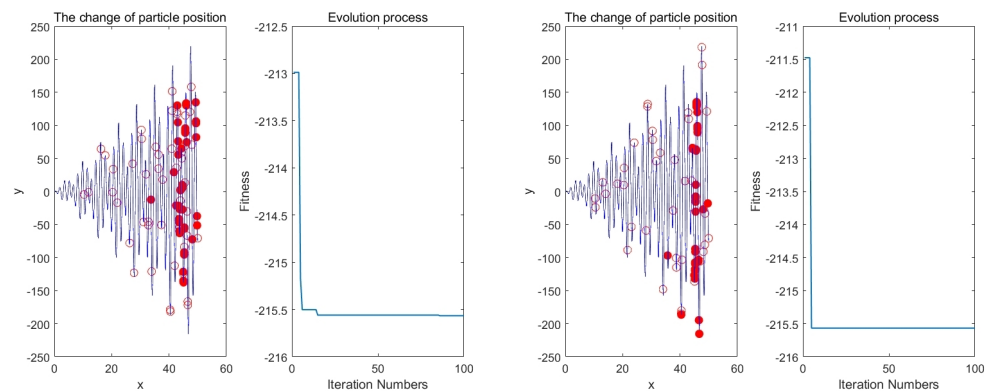


Figure 3. Particle distribution map.

3.4. Mutation Strategy

In the modified PSO algorithm, particles very easily fall into the local optimal trap and cannot jump out. Based on this problem, in recent years, many scholars have introduced the mutation strategy of genetic algorithms [29,30] into PSO algorithms. Wei et al. [31] and others used a local update strategy of neighborhood difference mutation (NDM) to increase the diversity of the algorithm; Duan et al. [32] proposed an exact mutation strategy associated with two clustering coefficients which distinguishes the degree of mutation required by the particle population at different times and coordinates the performance of exploration and exploitation; Quanbin et al. [33] proposed a strategy that takes into account the suboptimal solutions that are chosen to discard the depth information of the mutation strategy.

In this paper, we introduce the mutation strategy into the genetic algorithm; in a genetic algorithm, it can produce gene templates that are not found in the original population, and, in the PSO algorithm, it will greatly enrich the diversity of the population, which helps to make the individual escape from the local optimal solution. First of all, each particle is assigned a small initial mutation probability value $pm = 0.0005$, then, through the calculation of Equation (7) to determine whether the particle is caught in the local optimum, if Q is greater than 0.9 five consecutive times, it is judged that the particle may be caught in the local optimum. At this time, it begins to accrue the coefficient of mutation in increments of 0.001 each time. When, in the interval $(0, 1)$, it produces a random number less than pm , the particle is carried out according to the Equation (8) mutation. The pseudo-code for the mutation strategy in this paper is shown in Algorithm 1.

$$Q = \frac{f_i^{(t-1)}}{f_i^{(t)}} \quad (7)$$

$$x_{i,j} = \zeta x_{i,j} + (1 - \zeta) pBest_{k,j} \quad (8)$$

ζ in Equation (8) is the proportion coefficient, which is specified in Reference [5], in the range $(0.2, 0.8)$, so, in this paper, we take a random value between 0.2 and 0.8; $x_{i,j}$ is the value of the j -dimension of the particle that needs to be varied, and $pBest_{k,j}$ is the value of the j -dimension of the optimal position of the randomly selected k -th individual.

Algorithm 1: Mutation Operation.

```

1 Input:  $pm, fitness(i)^{t-1}, fitness(i)^t$ 
2 Output:  $x_{i,j}$ 
3 for  $j = 1, 2, \dots, D$  do
4   if  $rand < pm(i)$  then
5     //  $pm$  is the coefficient of mutation for each particle
6     use Equation (7) to get  $x_{i,j}$ ;
7   end if
8 end for
9   calculate the  $fitness(i)$ ;
10   $Q \leftarrow fitness(i)^t / fitness(i)^{t-1}$ ;
11  if  $R > 0.9$  occurs five consecutive times then
12     $pm(i) \leftarrow pm(i) + pm$ ;
13    //  $pm$  is the increment of variable coefficient
14  end if
15  if five mutations were performed then
16    initialize  $pm(i)$ ;
17  end if

```

3.5. Optimization Process Steps

The flowchart of the IDCPSO algorithm proposed in this paper is shown in Figure 4.

Begin:

Step 1. Initialize the particle swarm. Given the population size M , randomly generate the position X_i and velocity V_i of each particle.

Step 2. Evaluate the fitness of each particle according to the test function and find $pbest_i$ and $gbest$.

Step 3. Compare the fitness of $pbest_i$ in Step 2 with the fitness of eight intermediate positions to update the individual extreme values.

Step 4. Determine the positions of GCP and SCP according to Equations (4) and (5) and compare the fitness of $gbest$ with the fitness of GCP and SCP to update the global extreme value.

Step 5. Determine whether the particle needs to be mutated according to Equation (7). If it needs to be mutated, then carry out the mutation operation according to Equation (8).

Step 6. If the maximum number of iterations is reached, terminate the execution of the algorithm; otherwise, turn to Step 2.

End.

3.6. Analysis of Time Complexity

Let the total number of particle swarms be M , the particle dimension be D and the maximum number of iterations be I . The time complexity of the modified particle swarm optimization algorithm is $O(MDI)$. Firstly, the time added to the IDCPSO algorithm within an iteration is calculated. When calculating the six different flight routes for each particle in Section 3.2, eight intermediate positions need to be calculated, and the added time is $O(8MD)$. In Section 3.3, calculating the weights that each particle occupies in the construction of the virtual centers takes time $O(2M)$, and the time spent in constructing the GCP and SCP is $O(2MD)$. In Section 3.4, accumulating the coefficients of mutation and judging whether the particles need to be mutated takes time $O(2)$, and, when the number of particles needing mutation is e , the added mutation time is $O(2eD)$. Therefore, the time complexity of IDCPSO in this paper is $O(I(2 + 10MD + 2eD + 2M))$ in I iterations, and the time complexity of IDCPSO after omitting the low-order terms is $O(MDI)$. The magnitude of the algorithmic time complexity does not change, indicating that, compared to the PSO algorithm, the improved algorithm in this paper increases the time spent, but the increase is not significant.

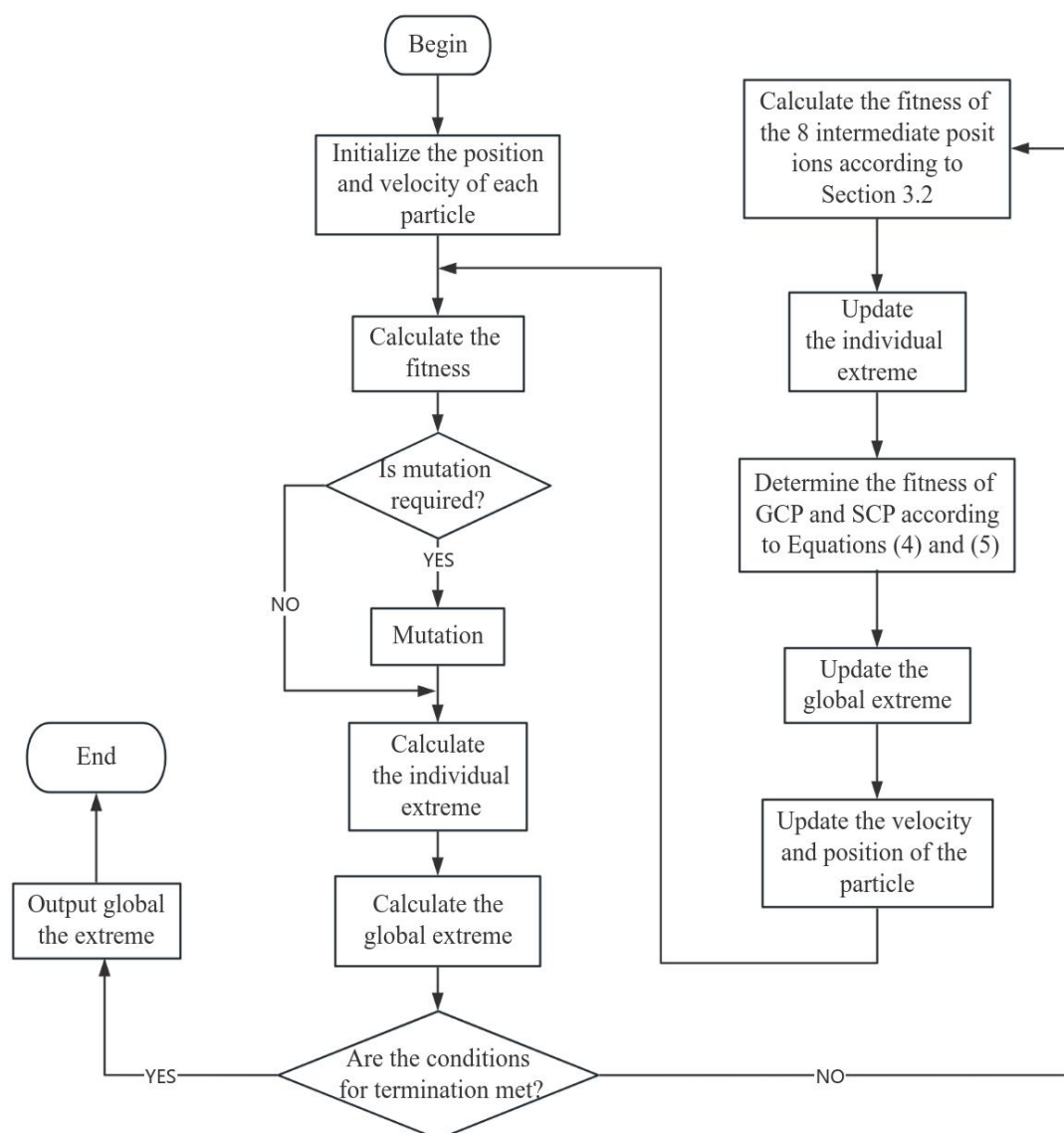


Figure 4. Flowchart of the IDCPSO algorithm.

4. Simulation Experiment

In this paper, the 12 typical functions shown in Table 3 are used to test the performance of the improved dual-center particle swarm optimization algorithm (IDCPSO), and the comparison algorithms chosen are the dual-center particle swarm optimization algorithm [25] (DCPSO), particle swarm optimization algorithm [34] (PSO), linear decreasing particle swarm optimization algorithm [35] (LDWPSO) and adaptive particle swarm optimization algorithm [36] (APSO). The objective of the test function is to minimize the function within a specified range using the MatlabR2023a compilation algorithm, 2.60 GHz, 4 GB, on 64-bit operating system on a computer. In order to enhance the comparability, the population size $M = 20$; the dimension of the solution set space $D = 40$; the maximum number of iterations $t_{\max} = 1000$ (except for $t_{\max} = 3000$ in f_2 and $t_{\max} = 10,000$ in f_{11}); and the other parameter settings in each algorithm are shown in Table 4. In order to reduce the impact of the randomness of the algorithms on the test results, five different algorithms were run independently 50 times for the 12 classical test functions, and the results of this test are shown in Table 5, including the minimum, mean and variance.

Table 3. Description of test functions.

Functions	Function Expressions	Variable Range	Minimum
Griewank	$f_1(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^{40}$	0
Rastrigrin	$f_2(x) = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^{40}$	0
Sphere	$f_3(x) = \sum_{i=1}^n x_i^2$	$[-5.12, 5.12]^{40}$	0
Ackley	$f_4(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)\right) + 20 + e$	$[-30, 30]^{40}$	0
Rosenbrock	$f_5(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$[-30, 30]^{40}$	0
Alpine	$f_6(x) = \sum_{i=1}^n x_i \sin(x_i) + 0.1x_i $	$[-10, 10]^{40}$	0
De Jong's (noise)	$f_7(x) = \sum_{i=1}^n ix_i^4$	$[-1.28, 1.28]^{40}$	0
Schwefel's 2.21	$f_8(x) = \max\{ x_i , 1 \leq i \leq 40\}$	$[-100, 100]^{40}$	0
Schwefel's 2.22	$f_9(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	$[-10, 10]^{40}$	0
Sum of Different Power	$f_{10}(x) = \sum_{i=1}^n x_i ^{(i+1)}$	$[-1, 1]^{40}$	0
Zakharov	$f_{11}(x) = \sum_{i=1}^n x_i^2 + \left(\sum_{i=1}^n 0.5ix_i\right)^2 + \left(\sum_{i=1}^n 0.5ix_i\right)^4$	$[-5, 10]^{40}$	0
Step	$f_{12}(x) = \sum_{i=1}^n (x_i + 0.5)^2$	$[-10, 10]^{40}$	0

Table 4. The setting of simulation parameters.

Algorithms	Parameter Setting
IDCPSO	$c_1 = c_2 = 2, w \in [0.4, 0.9]$
DCPSO	$c_1 = c_2 = 2, w \in [0.4, 0.9]$
PSO	$c_1 = c_2 = 2, w = 0.6$
LDWPSO	$c_1 = c_2 = 2, w \in [0.4, 0.9]$
APSO	$c_1 = 2.5 - (1.5t/t_{\max}), c_2 = 1 + 1.5t/t_{\max}, w \in [0.4, 0.9]$

Simulation Results

Analyzing the data in Table 5, it can be seen that the algorithms in this paper are optimal in calculating the minimum value, followed by the DCPSO algorithm, and the accuracy of these two algorithms compared to the other algorithms in finding the optimal result is improved by at least two orders of magnitude. The PSO algorithm is the worst among the five test functions. In terms of variance, for the algorithms in this paper, except for the function f_2 and DCPSO algorithm, the improvement is not obvious; in the other test functions, there are more obvious improvements. The improvement is most noticeable in function f_{11} . It is worth mentioning that the IDCPSO algorithm is able to obtain the optimal solution quickly and accurately on the function f_5 . In general, this paper's algorithm is not only far better than other functions in single-peak function optimization, but also performs well in multi-peak function [37] optimization. In order to more intuitively reflect the superiority of this paper's algorithm, Figure 5 gives the optimal solution evolution diagrams of different functions with the above algorithms, from which it can be seen that this paper's algorithm is the best in both convergence speed and minimization optimization, which shows that the algorithm has a better optimization effect.

Table 5. The simulation results of different algorithms.

Functions	Algorithms	Minimum	Mean	Variance
f_1	IDCPSO	0	4.59×10^{-3}	1.07×10^{-4}
	DCPSO	1.88×10^{-4}	9.52×10^{-2}	8.04×10^{-2}
	PSO	3.71	5.66	4.77×10^{-1}
	LDWPSO	1.03	9.89×10^{-1}	2.90×10^{-3}
	APSO	1.71	2.06	8.60×10^{-2}
f_2	IDCPSO	1.57×10^1	3.48×10^1	2.80×10^2
	DCPSO	3.48×10^1	2.08×10^2	5.58×10^2
	PSO	1.31×10^2	1.67×10^2	1.09×10^3
	LDWPSO	5.57×10^1	7.76×10^1	2.36×10^2
	APSO	1.28×10^2	1.72×10^2	5.39×10^2
f_3	IDCPSO	0	1.96×10^{-33}	2.94×10^{-66}
	DCPSO	7.95×10^{-8}	3.03×10^{-7}	4.57×10^{-14}
	PSO	1.05	1.35	4.58×10^{-2}
	LDWPSO	1.01×10^{-2}	2.77×10^{-2}	8.21×10^{-5}
	APSO	1.76×10^{-1}	3.10×10^{-1}	6.40×10^{-3}
f_4	IDCPSO	0	6.86×10^{-31}	3.14×10^{-60}
	DCPSO	1.37×10^{-4}	3.01×10^{-4}	9.54×10^{-9}
	PSO	1.38	1.61	3.54×10^{-2}
	LDWPSO	7.63×10^{-2}	1.18×10^{-1}	1.85×10^{-3}
	APSO	4.52×10^{-1}	7.10×10^{-1}	2.53×10^{-2}
f_5	IDCPSO	0	0	0
	DCPSO	3.33×10^{-29}	1.01×10^{-21}	7.96×10^{-42}
	PSO	9.97×10^{-9}	3.39×10^{-6}	2.53×10^{-11}
	LDWPSO	6.37×10^{-16}	1.28×10^{-13}	8.26×10^{-26}
	APSO	2.97×10^{-11}	6.60×10^{-9}	1.94×10^{-16}
f_6	IDCPSO	1.05×10^{-14}	2.18×10^{-14}	5.11×10^{-29}
	DCPSO	6.03×10^{-4}	3.77×10^{-3}	7.89×10^{-6}
	PSO	8.75	1.05×10^1	1.63
	LDWPSO	1.64×10^{-1}	3.07×10^{-1}	2.50×10^{-2}
	APSO	3.35	7.12	9.07
f_7	IDCPSO	0	3.55×10^{-52}	2.71×10^{-103}
	DCPSO	1.90×10^{-11}	3.62×10^{-10}	1.94×10^{-19}
	PSO	6.00×10^{-3}	1.45×10^{-2}	4.35×10^{-5}
	LDWPSO	7.39×10^{-06}	3.26×10^{-05}	4.17×10^{-10}
	APSO	5.32×10^{-4}	1.19×10^{-3}	3.04×10^{-7}
f_8	IDCPSO	2.56×10^{-2}	8.60×10^{-2}	3.51×10^{-4}
	DCPSO	1.51	1.90	8.59×10^{-2}
	PSO	1.09×10^1	1.16×10^1	6.18×10^{-1}
	LDWPSO	5.77	6.76	4.17×10^{-1}
	APSO	5.20	7.20	1.30
f_9	IDCPSO	7.30×10^{-19}	1.48×10^{-10}	1.06×10^{-18}
	DCPSO	1.34×10^{-4}	8.70×10^{-4}	3.14×10^{-7}
	PSO	8.02	1.03×10^1	1.52
	LDWPSO	6.20×10^{-1}	9.14×10^{-1}	9.03×10^{-2}
	APSO	4.19	5.27	3.26×10^{-1}
f_{10}	IDCPSO	0	3.10×10^{-89}	1.32×10^{-176}
	DCPSO	3.55×10^{-28}	3.98×10^{-26}	2.98×10^{-51}
	PSO	3.66×10^{-11}	1.37×10^{-9}	1.81×10^{-18}
	LDWPSO	1.92×10^{-20}	1.30×10^{-17}	6.29×10^{-34}
	APSO	5.29×10^{-12}	8.31×10^{-11}	1.07×10^{-20}

Table 5. Cont.

Functions	Algorithms	Minimum	Mean	Variance
f_{11}	IDCPSO	7.46×10^{-69}	2.88×10^{-39}	2.03×10^{-76}
	DCPSO	8.79×10^{-7}	1.01×10^{-1}	9.43×10^2
	PSO	5.44×10^1	2.28×10^2	1.08×10^4
	LDWPSO	5.57×10^1	2.45×10^2	2.13×10^4
	APSO	5.25×10^{-1}	1.07×10^1	9.42×10^2
f_{12}	IDCPSO	0	9.24×10^{-33}	3.33×10^{-64}
	DCPSO	1.54×10^{-7}	1.25×10^{-6}	8.44×10^{-13}
	PSO	2.97	5.42	2.00
	LDWPSO	5.50×10^{-2}	1.29×10^{-1}	3.30×10^{-3}
	APSO	1.09	1.36	2.19×10^{-2}

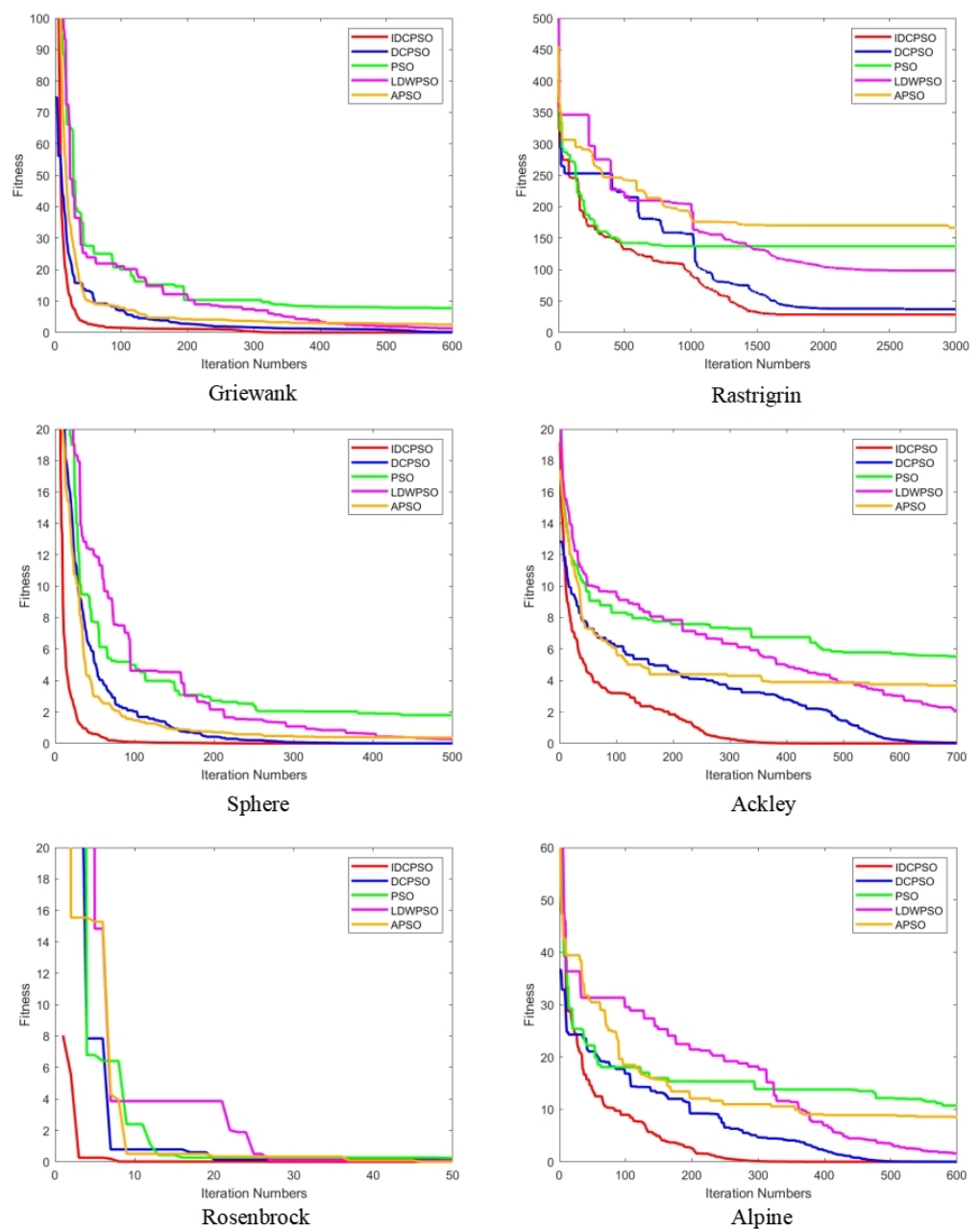


Figure 5. Cont.

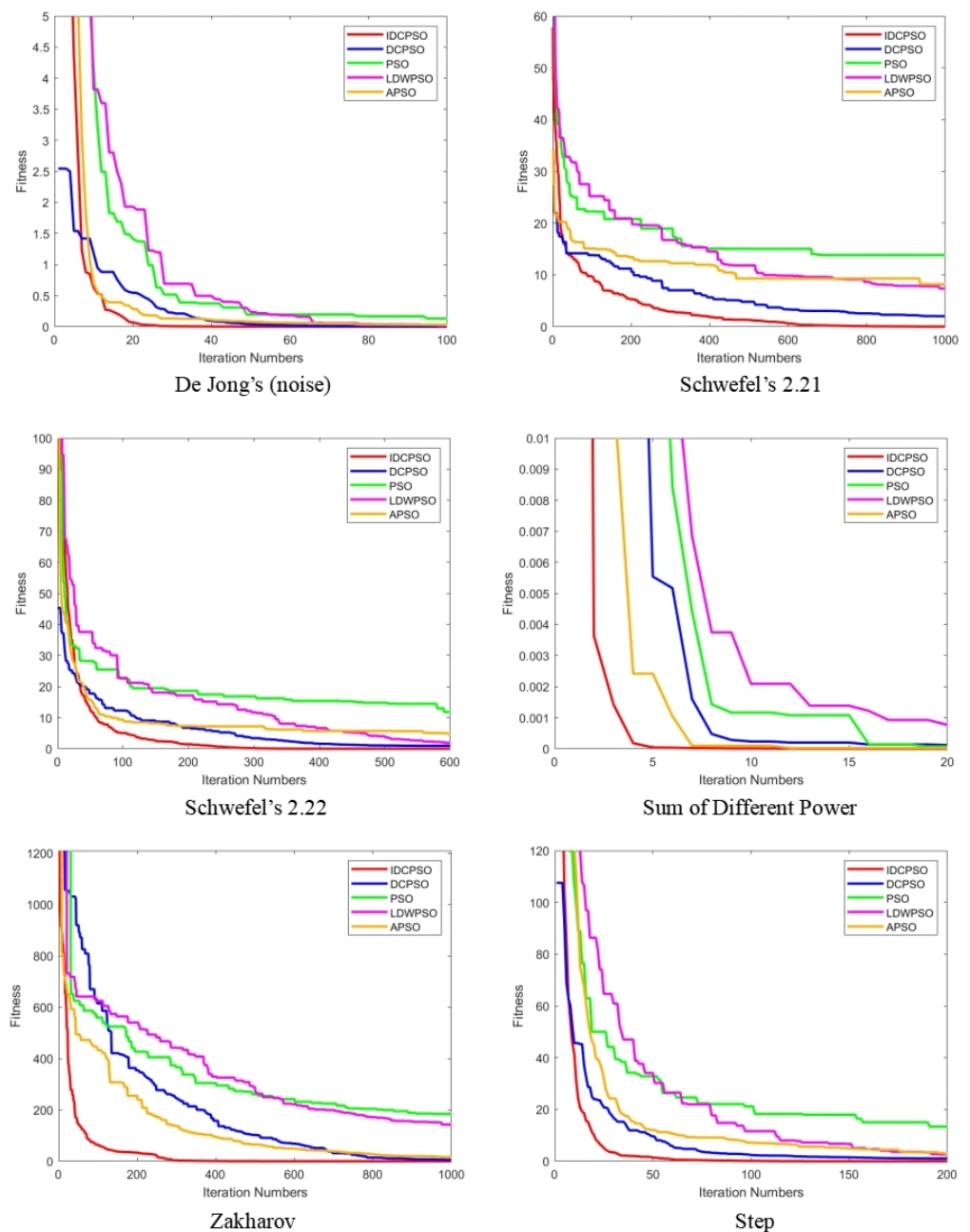


Figure 5. Particle distribution map.

5. Conclusions

In this work, we proposed an improved dual-center particle swarm optimization (IDCPSO) algorithm by adding five particle motion routes and other optimization strategies to the DCPSO algorithm. The algorithm analyzes the flight trajectories of vector-decomposed particles in depth by updating the velocity formula of the particles, constructs the center particles reasonably and, finally, combines with the introduction of the mutation factor, which accelerates the convergence speed of the algorithm, greatly optimizes the quality of the global extremes and strengthens the quality of the average solution of the population. The comparison test of this paper's algorithm with four other algorithms showed that the IDCPSO algorithm has a better optimization effect.

We have not yet applied the algorithm to practical engineering optimization problems such as the traveler's problem, the backpack problem and the shop floor scheduling problem. This provides us a future research direction for testing the IDCPSO algorithm further in

future work. For example, to study the Vehicle Path Problem with Time Window (VRPTW), one can consider designing the position of the particle as a three-part problem that contains information about the sequence of the customers, the path segmentation information and the number of paths, which can then be solved using the IDCPSO algorithm.

Author Contributions: Conceptualization, D.P. and Z.Q.; methodology, D.P.; software, Z.Q.; validation, D.P.; formal analysis, D.P.; data curation, Z.Q.; writing—original, draft preparation, Z.Q.; writing—review, and editing, D.P.; visualization, Z.Q.; supervision, D.P.; project administration, D.P.; funding acquisition, D.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China grant number 11871059 and Natural Science Foundation of Sichuan Education Department grant number 18ZA0469.

Data Availability Statement: All data that can reproduce the results in this study can be requested from the corresponding author.

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

References

- Kennedy, J.; Eberhart, R. Particle swarm optimization. *Int. Conf. Neural Net.* **1995**, *4*, 1942–1948.
- Zhang, L.; Tang, Y.; Hua, C.; Guan, X. A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques. *Appl. Soft Comput. J.* **2015**, *28*, 138–149. [\[CrossRef\]](#)
- Taherkhani, M.; Safabakhsh, R. A novel stability-based adaptive inertia weight for particle swarm optimization. *Appl. Soft Comput.* **2016**, *38*, 281–295. [\[CrossRef\]](#)
- Xinliang, X.; Fu, Y. Random walk autonomous groups of particles for particle swarm optimization. *J. Intell. Fuzzy Syst.* **2022**, *42*, 1519–1545.
- Kang, Y.; Zang, S. Improved particle swarm optimization algorithm based on multiple strategies. *J. Northeast. Univ. Nat. Sci. Ed.* **2023**, *44*, 1089–1097.
- Ge, H.; Sun, L.; Tan, G.; Chen, Z.; Chen, C.P. Cooperative Hierarchical PSO with Two Stage Variable Interaction Reconstruction for Large Scale Optimization. *IEEE Trans. Cybern.* **2017**, *47*, 2809–2823. [\[CrossRef\]](#) [\[PubMed\]](#)
- Gou, J.; Lei, Y.; Guo, W.; Wang, C.; Cai, Y.Q.; Luo, W. A novel improved particle swarm optimization algorithm based on individual difference evolution. *Appl. Intell.* **2017**, *57*, 468–481. [\[CrossRef\]](#)
- Lai, X.; Zhou, Y. An adaptive parallel particle swarm optimization for numerical optimization problems. *Neural Comput. Appl.* **2019**, *31*, 6449–6467. [\[CrossRef\]](#)
- Xu, G.; Cui, Q.; Shi, X.; Ge, H.; Zhan, Z.H.; Lee, H.P.; Liang, Y.; Tai, R.; Wu, C. Particle swarm optimization based on dimensional learning strategy. *Swarm Evol. Comput.* **2019**, *45*, 33–51. [\[CrossRef\]](#)
- Rahman, C.M. Group learning algorithm: A new metaheuristic algorithm. *Neural Comput. Appl.* **2023**, *35*, 14013–14028. [\[CrossRef\]](#)
- Aziz, A.A.N.; Ibrahim, Z.; Mubin, M.; Nawawi, S.W.; Mohamad, M.S. Improving Particle Swarm Optimization via Adaptive Switching Asynchronous—Synchronous Update. *Appl. Soft Comput.* **2018**, *72*, 298–311. [\[CrossRef\]](#)
- Jiang, L.; Ye, R.; Liang, C.; Lu, W. Improved second-order oscillating particle swarm optimization. *Comput. Eng. Appl.* **2019**, *55*, 130–138+167.
- Tang, B.; Xiang, K.; Pang, M. An integrated particle swarm optimization approach hybridizing a new self-adaptive particle swarm optimization with a modified differential evolution. *Neural Comput. Appl.* **2020**, *32*, 4849–4883. [\[CrossRef\]](#)
- Ding, S.; Zhang, Z.; Sun, Y.; Shi, S. Multiple birth support vector machine based on dynamic quantum particle swarm optimization algorithm. *Neurocomputing* **2022**, *480*, 146–156. [\[CrossRef\]](#)
- Wang, Y.; Qian, Q.; Feng, Y.; Fu, Y. An improved particle swarm optimization algorithm combining attraction and repulsion and two-way learning. *Comput. Eng. Appl.* **2022**, *58*, 79–86.
- Chen, Y.; Li, L.; Xiao, J.; Yang, Y.; Liang, J.; Li, T. Particle swarm optimizer with crossover operation. *Eng. Appl. Artif. Intell.* **2018**, *70*, 159–169. [\[CrossRef\]](#)
- Tian, D.; Shi, Z. MPSO: Modified particle swarm optimization and its applications. *Swarm Evol. Comput.* **2018**, *41*, 49–68. [\[CrossRef\]](#)
- Ren, M.; Huang, X.; Zhu, X.; Shao, L. Optimized PSO algorithm based on the simplicial algorithm of fixed point theory. *Appl. Intell.* **2020**, *50*, 2009–2024. [\[CrossRef\]](#)
- Xu, F.; Zou, D.; Li, C.; Luo, H.; Zhang, M. An improved particle swarm optimization algorithm with Circle mapping and sine cosine factor. *Comput. Eng. Appl.* **2023**, *59*, 80–90.
- Predić, B.; Jovanovic, L.; Simic, V.; Bacanin, N.; Zivkovic, M.; Spalevic, P.; Budimirovic, N.; Dobrojevic, M. Cloud-load forecasting via decomposition-aided attention recurrent neural network tuned by modified particle swarm optimization. *Complex Intell. Syst.* **2024**, *10*, 2249–2269. [\[CrossRef\]](#)

21. Sulaiman, T.A.; Salau, B.H.; Onumanyi, J.A.; Mu'azu, M.B.; Adedokun, E.A.; Salawudeen, A.T.; Adekale, A.D. A Particle Swarm and Smell Agent-Based Hybrid Algorithm for Enhanced Optimization. *Algorithms* **2024**, *17*, 53. [\[CrossRef\]](#)
22. Kannan, S.K.; Diwekar, U. An Enhanced Particle Swarm Optimization (PSO) Algorithm Employing Quasi-Random Numbers. *Algorithms* **2024**, *17*, 195. [\[CrossRef\]](#)
23. Feng, D.; Li, Y.; Liu, J.; Liu, Y. A particle swarm optimization algorithm based on modified crowding distance for multimodal multi-objective problems. *Appl. Soft Comput.* **2024**, *152*, 111280. [\[CrossRef\]](#)
24. Tian, D.; Xu, Q.; Yao, X.; Zhang, G.; Li, Y.; Xu, C. Diversity-guided particle swarm optimization with multi-level learning strategy. *Swarm Evol. Comput.* **2024**, *86*, 101533. [\[CrossRef\]](#)
25. Tang, K.; Liu, B.; Yang, J.; Sun, T. Double Center Particle Swarm Optimization. *Comput. Res. Dev.* **2012**, *49*, 1086–1094.
26. Reza, M.B.; Zbigniew, M. Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review. *Evol. Comput.* **2017**, *25*, 1–54.
27. Jordehi, R.A. Particle swarm optimisation for dynamic optimisation problems: A review. *Neural Comput. Appl.* **2014**, *25*, 1507–1516. [\[CrossRef\]](#)
28. Arrison, R.K.; Engelbrecht, P.A. Ombuki-Berman, M.B. Inertia weight control strategies for particle swarm optimization: Too much momentum, not enough analysis. *Swarm Intell.* **2016**, *10*, 267–305. [\[CrossRef\]](#)
29. Octavio, R.; Marcela, Q.; Efrén, M.; Kharel, R. Variation Operators for Grouping Genetic Algorithms: A Review. *Swarm Evol. Comput.* **2021**, *60*, 100796.
30. Hassanat, A.; Almohammadi, K.; Alkafaween, E.; Abunawas, E.; Hammouri, A.; Prasath, V.S. Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach. *Information* **2019**, *10*, 390. [\[CrossRef\]](#)
31. Li, W.; Liang, P.; Sun, B.; Sun, Y.; Huang, Y. Reinforcement learning-based particle swarm optimization with neighborhood differential mutation strategy. *Swarm Evol. Comput.* **2023**, *78*, 101274. [\[CrossRef\]](#)
32. Duan, X.; Zhang, X. A hybrid genetic-particle swarm optimizer using precise mutation strategy for computationally expensive problems. *Appl. Intell.* **2021**, *52*, 8510–8533. [\[CrossRef\]](#)
33. Quanbin, Z.; Zhenyu, M. Adaptive differential evolution algorithm based on deeply-informed mutation strategy and restart mechanism. *Eng. Appl. Artif. Intell.* **2023**, *126*, 107001.
34. Jiang, M.; Luo, Y.; Yang, S. Stochastic convergence analysis and parameter selection of the modified particle swarm optimization algorithm. *Inf. Process. Lett.* **2006**, *102*, 8–16. [\[CrossRef\]](#)
35. Choudhary, S.; Sugumaran, S.; Belazi, A.; El-Latif, A.A.A. Linearly decreasing inertia weight PSO and improved weight factor-based clustering algorithm for wireless sensor networks. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *14*, 6661–6679. [\[CrossRef\]](#)
36. Lian, X.Q.; Liu, Y.; Chen, Y.M.; Huang, J.; Gong, Y.G.; Huo, L. Research on Multi-Peak Spectral Line Separation Method Based on Adaptive Particle Swarm Optimization. *Spectrosc. Spectr. Anal.* **2021**, *41*, 1452–1457.
37. Akkar, A.H.D.; Mahdi, R.F. Evolutionary Algorithms Performance Comparison For Optimizing Unimodal And Multimodal Test Functions. *Int. J. Sci. Technol. Res.* **2015**, *4*, 38–45.

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