Hindawi
Journal of Control Science and Engineering
Volume 2020, Article ID 3857894, 12 pages
https://doi.org/10.1155/2020/3857894

Research Article

An Improved Method of Particle Swarm Optimization for Path Planning of Mobile Robot

Xun Li,[1,2] Dandan Wu ,[1] Jingjing He,[1] Muhammad Bashir,[1] and Ma Liping[1]

[1]*College of Electronics and Information, Xi'an Polytechnic University, Xi'an, China*
[2]*Bernoulli Institute, University of Groningen, Groningen, Netherlands*

Correspondence should be addressed to Dandan Wu; nakedbears@163.com

Received 6 February 2020; Revised 16 April 2020; Accepted 8 May 2020; Published 25 May 2020

Academic Editor: Carlos-Andrés García

The existing particle swarm optimization (PSO) algorithm has the disadvantages of application limitations and slow convergence speed when solving the problem of mobile robot path planning. This paper proposes an improved PSO integration scheme based on improved details, which integrates uniform distribution, exponential attenuation inertia weight, cubic spline interpolation function, and learning factor of enhanced control. Compared with other standard functions, our improved PSO (IPSO) can achieve better optimal results with less number of iteration steps than the different four path planning algorithms developed in the existing literature. IPSO makes the optimal path length with less than 20 iteration steps and reduces the path length and simulation time by 2.8% and 1.1 seconds, respectively.

1. Introduction

Path planning is different from motion planning where dynamics must be considered. Its purpose is to find the optimal path of motion in the least amount of time and to model the environment completely [1]. For the path planning problem of mobile agents, several researchers have proposed many algorithms, which can be classified into two categories, i.e., traditional path planning methods and bionic intelligent algorithm-based methods. The former method includes the A* algorithm [2], Dijkstra [3], RTT [4], and artificial potential field method [5]. Bionic intelligent algorithms include differential evolution algorithm [6], genetic algorithm [7], ant colony algorithm [8], artificial fish swarm algorithm [9], and PSO [10]. PSO algorithm is widely used in the practical application and theoretical research of mobile agent path planning due to its strong searchability, fast convergence speed, and high efficiency [11]. The PSO is a population-based randomization technique to find the optimal value for the foraging of the birds. PSO has the advantages of fast search speed, memory, few parameters, and simple structure and has easier implementation at the stage of validation. Besides, its shortcomings include global search and local search imbalance, low convergence precision, easy falling into optimal local solution, and poor robustness.

To obtain better optimization results, a PSO optimization technique is proposed in [12] to converge at the global minimum, and a custom algorithm is used to generate the coordinates of the search space. The coordinate values generated by the custom algorithm are passed to the PSO algorithm, which uses these coordinate values to determine the shortest path between two given end positions. Thus, it is not limited to only finding the optimal value. Still, it can improve the speed of the algorithm. However, the direct transfer of its two-point coordinates is easy to fall into the local optimum, and the obtained optimal value is often considered as the global suboptimal value; the literature in [13] proposed a random disturbance method.

The adaptive PSO optimization method introduces a disturbing global update mechanism to the optimal global position, which prevents the algorithm stalling. Besides, a new adaptive strategy is proposed to fine-tune the three control parameters in the algorithm. Moreover, the need for dynamic adjustment and exploration of the three parameters increases the computational complexity of the optimization algorithm as well as low execution efficiency. The work in

[14] proposed a modified particle swarm optimization (MPSO) with constraints to jointly obtain a smooth path, but the method does not improve the efficiency of the particle diversity, which makes the particles stagnate and fall into the optimal local solution. The studies in [15] proposed a fusion of chaotic PSO and ant colony algorithm (ACO). The algorithm effectively adjusts the particle swarm optimization parameters. It reduces the number of iterations of the ant colony algorithm, called the chaotic particle swarm algorithm, thereby effectively reduces the search time. However, due to the improved algorithm parameters, its speed and the position updates are proportional, and this can limit the particle global searchability. Moreover, it is a solution that can fall into the optimal local solution.

The hybrid genetic particle swarm optimization algorithm (GA-PSO) is proposed in [16] that established a path planning. The mathematical model of the problem proposed a time-first based particle-first iteration mechanism, which makes the evolution process more directional and accelerates the development of path planning problems. However, the hybrid algorithm has too many parameters that need control. This increases the computational complexity and has reduced the execution efficiency of the algorithm. It signifies that it is difficult to improve the diversity of particles due to the easy way of falling into the optimal local solution.

Therefore, to combine the advantages of the abovementioned various improved algorithms and improve their defects, this paper proposed a new method to improve the PSO integration scheme based on improved details, which is used to solve the global path planning of mobile robots in the indoor environment. In the proposed algorithm, the navigation point model is selected as the working area model of the mobile robot, and the uniform distribution, exponential attenuation inertia weight, cubic spline interpolation function, and learning factor of enhanced control are introduced to PSO to improve its performance. Finally, the advancement and effectiveness of the standard test function and obstacle environment are verified in the paper. The results show that, compared with other standard functions, IPSO achieves better optimal results and more iteration steps. Compared with the other four path planning algorithms, IPSO reaches the optimal path length with less than 20 iteration steps and reduces the path length and simulation time by 2.8% and 1.1 seconds, respectively.

2. Improved PSO (IPSO)

2.1. Classical PSO. The fundamental core of the PSO method is to share the information through the individuals in the group so that the movement of the whole group can be transformed from disorder to order in the problem of solution space to obtain the optimal solution of the problem. The answer to each optimization problem is the "particle" speed and position update through (1) and (2). The first term of (1) indicates that the next move of the particle is affected by the magnitude and direction of the last flight speed; the second term means that the following action of the particle originates from its own experience; the third term indicates that the next move of the particle originates from the population for the best companion to learn. That is, the next step of the particle is determined by the experience and the best experience of the companion:

$$V_{id}^{k+1} = \omega V_{id}^{k} + c_1 r_1 \left(P_{id}^{k} - X_{id}^{k}\right) + c_2 r_2 \left(P_{qd}^{k} - X_{id}^{k}\right), \quad (1)$$

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1}, \quad (2)$$

where V_{id}^{k} is the dth component of the flight velocity vector of the particle i of the kth iteration; X_{id}^{k} indicates the dth element of the position vector of the particle i of the kth iteration; c_1 and c_2 represent the learning factor, which is used to adjust the learned to the maximum step size; r_1 and r_2 are two random functions with a range of value from 0 to 1 and are used to increase the randomness searching; and ω is the inertia weight to adjust the searchability for the solution space. Although the classical PSO is simple to implement and has few adjustment parameters when it is used in the path planning method, it is prone to poor searchability, falls into the optimal local solution, and has reduced particle diversity, low convergence precision, and low accuracy of path planning. Therefore, this paper systematically improves the classical PSO algorithm by combining various improved methods.

2.2. Improvement of PSO Algorithm Integration

2.2.1. Uniform Distribution. We aim to ensure that the PSO algorithm does not lose the randomness of the particles when initializing the population and, at the same time, avoid the problem of excessive concentration of the initial positions of the particles, which is not conducive to global search and postprocessing, especially with the possibility that the search stagnation may occur in the late search. In this paper, continuous uniform random distribution is used to initialize the particles so that the particles are relatively uniformly distributed in the search space to facilitate later search and avoid particles falling into the local optimal solution. The formula for uniform position distribution is shown as follows:

$$x = \frac{x_{\max} - x_{\min}}{n-1}, \quad (3)$$

$$y = \frac{y_{\max} - y_{\min}}{n-1}, \quad (4)$$

where $x_{\max}$ and $y_{\max}$ are the upper limits of the variables, $x_{\min}$ and $y_{\min}$ are the lower limits of the variables, and n is the number of the decision variables.

2.2.2. Exponential Decay Inertia Weight. The change of the inertia weight w affects the position of the particle. The larger the value of w, the stronger the global searchability and the weaker the local mining ability. Therefore, good results can be obtained when the values of the w are dynamic (adjustable) compared to the fixed values. The value of w can vary linearly during the PSO search process, or dynamically based on a measure function of PSO performance. By analyzing and summarizing the linear decreasing inertia

weight proposed in [17], the improved particle swarm optimization algorithm based on the improved inertia weight proposed in [18], and the enhanced method of fuzzy inertia weight strategy proposed in [19], this paper presents an improvement in the above-mentioned methods. The fixed, predictable, and iterative step adopted in the previous ways is a global transformation that is not conducive to the particle, such as fixed transformation that can narrow the search range of the particle and affect the diversity of the particle. This paper uses the study in [20] to introduce the inertia weight of exponential decay which is more in line with the characteristics of the exponential function. By adopting the changes of the presteps and poststeps to the search area, its speed can be successfully updated at different periods. This eliminates the premature particles and improves the robustness of the algorithm. Henceforth, the global search and local mining ability are maintained, and this can solve the problems mentioned above to some extent. The expression is as follows:

$$A = it\frac{\ln \omega_{\max} - \ln \omega_{\min}}{MaxIt} - \ln \omega_{\max}, \tag{5}$$

$$\omega = \exp(-A), \tag{6}$$

where $\omega_{\min}$ is the minimum weight, $\omega_{\max}$ is the maximum weight, as well as the current number of iterations, *MaxIt* is the maximum number of iterations, and *it* is the current number of iterations.

2.2.3. Improved Learning Factors. The standard PSO expressed in (1) represents the acceleration weight of each particle moving to the optimal local value and the optimal global value. Lower values allow the particles to linger outside the target area, while the higher values are causing the particles to suddenly rush toward or cover the target area. Several methods have been effectively improved by the learning factor, although the learning factor and inertia weight weaken the uniformity of the optimization algorithm to a certain extent. Hence, it is not conducive to the global and local search algorithms.

The work in [21] analyzes the PSO with a compression factor that considered the learning factor as a function of inertia weight. This kind of method has the advantage of transforming the three variables involving inertia weight and learning factor into one variable, which not only facilitates practical application but also enhances the uniformity of the algorithm optimization process. However, the function used is complicated, costly, and time-consuming. Besides, the possibility of particle stagnation in the later search is high, increasing the risk of particles falling into optimal local values. Therefore, according to the characteristics of high self-learning ability in the early search period and high demand for "social learning ability," this paper adopts the dynamic method to improve the learning factor; its expressions are as shown in (7) and (8). It is shown that this expression is constrained by (9):

$$c_1 = \cos(\omega), \tag{7}$$

$$c_2 = \sin(\omega), \tag{8}$$

$$\begin{cases} 1 - \omega > 0, \\ 2\omega + 2 > rand_1 \times c_1 + rand_2 * c_2. \end{cases} \tag{9}$$

The cosine function in (7) is a subtraction function in the interval $(0, \pi/2)$, which has a significant value at the beginning of the search, and the value is decreasing in the latter part of the search. The sine function in (8) is an increasing function in the interval $(0, \pi/2)$ with a small value before the quests, and its value increases after searching. Not only does it satisfy the condition that the PSO algorithm has better learning ability in the optimization process, but it also changes the inertia weight and the learning factor into one variable, which is convenient for practical application and strengthens the uniformity of the algorithm optimization process. Equation (9) is the guiding condition for selecting and adjusting the PSO parameters given in [22].

2.2.4. Improvement Based on Robot Dynamics Requirements

(1) Cubic Spline Interpolation. The experimental results show that (see Section 4.2 for details) the path planning of the improved PSO has more turning points with a rough path, which will affect the dynamic characteristics of the robot when it is moved. Therefore, it is necessary to improve the aforementioned improved PSO algorithm further to have a smooth path with the improved robot dynamic adaptability of the algorithm requirements. The cubic spline curve is fitted by several interpolation intervals based on cubic polynomials to provide a smooth curve, defined as follows:

On the range [a, b], $n+1$ takes a node, and the node coordinates are $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. If the following conditions are met, it is called a cubic spline function.

(i) Within each cell (x_i, x_{i+1}), where $i = 0, 1, \ldots, n$, there is a cubic polynomial:

$$z_i = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3. \tag{10}$$

(ii) The function and its first and second derivatives are continuous at the interpolation point.

(iii) $z_i(x)$ commonly uses endpoint conditions that can satisfy the following three requirements:

(a) Free boundary: the second derivative at the endpoint is zero.
(b) Fixed limitation: the range value of the differential function from the beginning to the end is specified.
(c) Nonnode boundary: the third derivative at the 2nd to the last node is continuous.

(2) Particle Coding. Particle coding is the assignment of coordinate positions to several path nodes. The path node is

the intersection of any two cubic spline intervals. Path nodes can be selected arbitrarily or according to the environment. Suppose that the coordinates of n path nodes are (x_{n1}, y_{n1}), (x_{n2}, y_{n2}), ..., (x_{nn}, y_{nn}). The start and end coordinate of the trail are (x_s, y_s), (x_t, y_t). Interpolating the coordinates of the interpolation points with cubic spline interpolation is on the interval of (x_1, y_1), (x_2, y_2), ..., (x_m, y_m). The path of the particle code planning is the connection of the interpolation points. This paper uses the path node, the interpolation point, and the link of the start and endpoints of the path as the running trajectory of the robot.

(3) Evaluation Function. This paper provides the shortest way that does not intersect with the obstacle. The design evaluation function of our proposed algorithm is as shown in (11), where L is the path length of the mobile agent from the ith path point to the $i+1$th path point, which is often used as an index in the path planning, and its mathematical expression is given in (12); x_i and y_i are the coordinates of the ith path point; x_{i+1} and y_{i+1} are the coordinates of the $i+1$th path point; α is the weight coefficient and is set to 100 here, which is used to exclude the illegal path, i.e., the way through the obstacle; and P is a penalty function of obstacle avoidance constraint in the selected indoor environment model, which is used to set up a safe distance. The calculation formula is as shown in (13), where R_m is the radius of the mth obstacle, m is the number of obstacles, and c and d are the center coordinates of the obstacle; the smaller the value of P, the higher the safety coefficient of the final path. If the trail passes the mth obstacle, it is a number greater than 0. If the obstacle is avoided, then the value is 0:

$$F = L \times (1 + \alpha \times P), \tag{11}$$

$$L = \sum_{i=1}^{n} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, \tag{12}$$

$$P = \sum_{m=1}^{m} \left(MAX\left(1 - \frac{\sqrt{(x_i - c)^2 + (y_i - d)^2}}{R_m}, 0 \right) \right). \tag{13}$$

3. Improved Algorithm Flow

This section provides the detailed step by step explanation of how to improve the PSO algorithm, as stated in Section 2.

Step 1: the number of path nodes, together with the number of interpolation, is determined according to the actual environment.

Step 2: set the parameters of the particle uniformly and initialize the particle position using (3) and (4), respectively. Then, the particle population and velocity are initialized.

Step 3: compute the coordinates of m interpolation points in x and y directions of each particle.

Step 4: calculate the fitness value of the particle according to (11).

Step 5: update the velocity and position of the particle according to (1), (2), and (5) to (9), respectively. Update the local optimal value Pbest and the global optimal value Gbest.

Step 6: determine whether the updated particle intersects the obstacle according to (13), and obtain a path composed of the path node coordinates after updating. The number of iterations is increased by one.

Step 7: if the termination condition is met (the threshold error is good enough to be negligible.), then the algorithm ends, and the optimal path is output. Otherwise, it goes to Step 3 and repeats the procedure. The flowchart of the algorithm is shown in Figure 1.

4. Simulation Experiment and Result Analysis

To verify the effectiveness and feasibility of the improved algorithm, the improved algorithm of this paper (denoted as IPSO), classical PSO algorithm (referred to as PSO), literature [23] stochastic inertia weight PSO algorithm (indicated as Rand PSO), literature [24] trigonometric learning factor PSO algorithm (denoted as TFPSO), and research [25] PSO with contraction factor algorithm (recorded as NCFPSO), the optimal values of the typical functions are compared and analyzed through path planning experiment. The simulation experiment environment is carried out on Windows 10, core i7, CPU (2.2 GHZ), memory 8 GB, Matlab (2018a).

4.1. Standard Test Function Optimization. At present, several researchers work on intelligent bionic algorithms to evaluate and compare the performance of algorithms by finding the optimal values for the five typical functions. The Ackley function is generally used to detect the global convergence rate of the algorithm, the Rastrigin function is used to find the optimal global value of the algorithm, and the Griewank function detects the ability of the algorithm to jump out of the optimal local value. The Sphere and Rosenbrock functions are unimodal functions that are used to solve optimal global solutions.

In this paper, the above five functions are used as objects to compare and evaluate the effectiveness of our proposed algorithm. The basic mathematical of the basic functions properties are shown in Table 1.

4.1.1. Standard Test Function and Parameter Setting. This section compares the performance of our proposed algorithm with some of the developed algorithms in the current literature. The maximum number of iterations of each algorithm is set to $MaxIt = 1000$, the population size is set to $Npop = 50$, the dimension is set to $D = 30$, and the lower and upper limit of the dynamic inertia weight are set as $w_{\min} = 0.4$ and $w_{\max} = 0.7$, respectively. The five algorithms were run 20 times, and the optimal values, mean values, standard deviations, and average simulation run times are computed based on the experimental results for each algorithm, respectively. The performance of our proposed (IPSO)

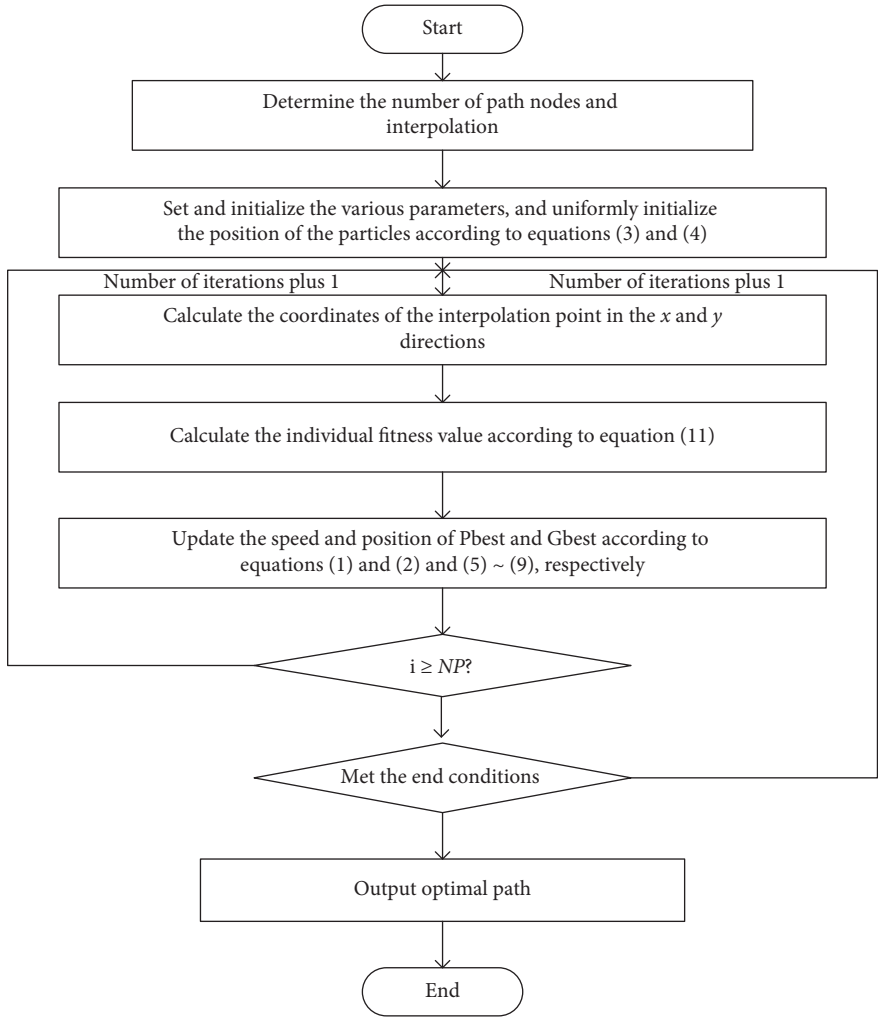

Figure 1: Flowchart of improved PSO algorithm.

Table 1: Standard test function.

Function name	Expression	Optimal solution	Range of independent variables
Ackley	$f(x) = -20\exp(-0.2\sqrt{(1/D)\sum_{i=1}^{D} x_i^2}) - \exp((1/D)\sum_{i=1}^{D}\cos 2\pi x_i) + 20 + e$	0	(−32, 32)
Griewank	$f(x) = (1/4000)\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(x_i/\sqrt{i}) + 1$	0	(−600, 600)
Rastrigin	$f(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos 2\pi x_i + 10)$	0	(−5.12, 5.12)
Rosenbrock	$f(x) = \sum_{i=1}^{D-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$	0	(−2.048, 2.048)
Sphere	$f(x) = \sum_{i=1}^{D} x_i^2$	0	(−100, 100)

algorithm was evaluated using the experimental results of these four aspects and the iterative curve.

4.1.2. Comparative Analysis of Algorithm Simulation Results, Standard Function Test, and Result Analysis. The results for the five standard functions test curves are shown in Figure 2. The test curve for the Ackley function is shown in the subgraph of Figure 2(a). The Ackley function is an n-dimensional function with several local optimal values. Therefore, it is difficult to find the optimal global value, but it can be seen from the test curve in the subgraph of Figure 2(a) that our proposed IPSO algorithm has the fastest convergence and highest precision among all the four algorithms developed in the literature. The subgraph depicted in Figure 2(b) is the iterative curve of the Griewank function. Griewank function is a typical nonlinear multimode stating function with a large search space and has many local advantages. As can be seen from the graph shown in Figure 2(b), our proposed IPSO algorithm can obtain the optimal value better than the other developed algorithms in the existing literature, and its

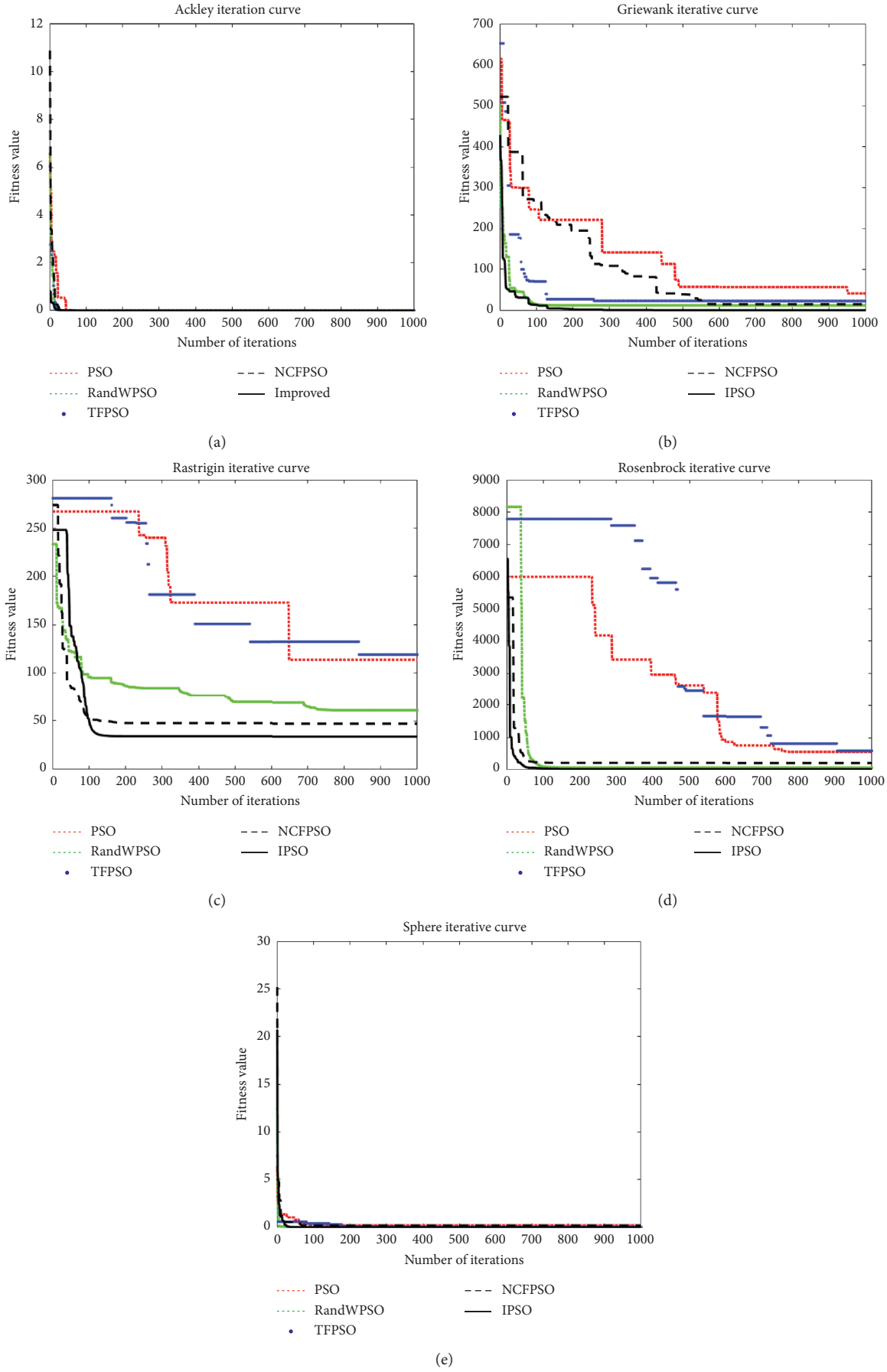

Figure 2: (a) Ackley iteration curve. (b) Griewank iteration curve. (c) Rastrigin iteration curve. (d) Rosenbrock iteration curve. (e) Sphere iteration curve.

convergence speed is fast. Its optimal value is received at the iteration step of less than 120. This shows that our proposed algorithm has better local optimal value bounce ability. The subgraph shown in Figure 2(c) represents the iterative curve of the Rastrigin function. Rastrigin function, which is similar to Griewank function, is a multipeak function. Its algorithm is easy to fall into the local maximum with an excellent solution. From the subgraph of Figure 2(c), our proposed IPSO algorithm approaches the optimal global value in a shorter iterative step when compared with the other four algorithms existing in the literature. Our proposed algorithm converges at 110 iteration steps, which verifies its ability to jump out from the optimal local value.

The subgraph in Figure 2(d) is the iterative curve of the Rosenbrock function. The global advantage of the Rosenbrock function lies in a smooth and narrow parabolic valley. The function optimization algorithm provides limited information, and it is challenging to distinguish the search direction. This implies that it is challenging to solve optimal global value. Nevertheless, it can be seen from the iteration curve that our proposed IPSO algorithm can find the optimal global value in less than 60 iterations, which is the fastest among the compared four existing algorithms in the literature. This indicates that our proposed algorithm has better global searchability.

The subgraph shown in Figure 2(e) is the iteration curve of the Sphere function. From Figure 2(e), our proposed algorithm has shown a certain advantage in terms of convergence speed and convergence precision that are not obvious. Still, the optimal value obtained by our proposed algorithm is the smallest, because the Sphere function is a unimodal function with a unique global minimum value, which does not make it difficult to find the optimal value. Hence, the degree of discrimination of the algorithm is not obvious.

Table 2 presents the performance comparison for the test result data obtained by the standard functions Ackley, Griewank, Rastrigin, Rosenbrock, Sphere, and our proposed algorithm for the verification. It can be seen from Table 2 that, for the five functions, except for the Ackley function, the best result is the same as the other four comparison algorithms. However, the other four functions are optimized by our proposed algorithm, and the results are improved by at least 45%. After the optimization of the five functions by the proposed algorithm in this paper, the average value is increased by at least 0.1%, and the standard deviation is increased by at least 0.2%. The experimental data shows that the IPSO algorithm is superior to the other four comparison algorithms in terms of optimal value, average value, and standard deviation. However, the average running time is more than that of the classical PSO. This is because the function of the parameters of our proposed algorithm is improved, while the classical PSO improved the parameters by constant values. Moreover, our proposed algorithm will increase the function to some extent. The limitation of the IPSO algorithm in this paper is being time-consuming. Our future studies would further consider the issue of achieving a reduced time. Although the IPSO algorithm of this paper optimizes the Rastrigin and Sphere functions with the slower convergence speed at the beginning, as the optimization process continues, the IPSO algorithm in this paper performs better than the other four algorithms at the middle and later stages.

4.2. Path Planning

4.2.1. Experimental Environment and Parameter Setting. The IPSO algorithm of this paper uses the navigation point model of [26] to construct the experimental environment model and the obstacle expansion treatment. The experimental environment is divided into a simple environment and a complex environment. The number of simple environmental obstacles is set to 5, the number of path nodes is 3, and the number of interpolation points is set to 100. Similarly, the number of complex environmental obstacles is set to 11, the number of path nodes is 5, and the number of interpolation points is set to 100. The boundary is a nonnode boundary.

Simulation parameter selection: the population size and the maximum number of iterations of the five algorithms are consistent: $Npop = 150$, $MaxIt = 100$; $w_{\min} = 0.4$, and $w_{\max} = 0.9$.

4.2.2. Path Planning and Algorithm Performance Analysis

(1) Simple Environment. In a simple environment, the starting point (■) is the map coordinate system of (0, 0), and the endpoint (★) coordinates are (9, 9), as shown in Figure 3(a).

It can be seen from Figure 3(a) that the path of the IPSO algorithm is smoother and the dynamics of the robot motion is improved; the path of the classical PSO planning does not find the optimal solution at the beginning of optimization and encounters stagnation along its path. It should be easy for the algorithm to fall into the optimal local solution. The ability to find the optimal global solution is weak due to poor particle diversity with small disturbance. The TFPSO algorithm starts to search for a short stagnation phenomenon. Finally, the localized optimal solution is obtained by the disturbance of the algorithm, which signifies that the algorithm has a weak ability to jump out from the optimal local solution.

Figure 3(b) shows that our proposed IPSO algorithm has the fastest convergence rate and converges at 13 iteration steps; the RandWPSO algorithm converges at around 20 iterative steps; the TFPSO converges at the 48th iteration; the NCFPSO algorithm converges at the 14th iterative step; the classical PSO converges at the 34th iteration.

Table 3 provides the performance comparison based average path length and the average simulation run time for the five algorithms. As compared with the other four algorithms, the longest path length of our proposed algorithm is reduced by at least 3.3%, the shortest path length is reduced by at least 2.9%, the average path length is reduced by at least 5.2%, and the average simulation time is reduced by 1.2 s.

Table 2: Optimization results of five functions.

Function	Performance index	PSO	RandWPSO	TFPSO	NCFPSO	IPSO
Ackley	Average value	0.34	0.15	0.35	0.17	0.10
	Best result	0	0	0	0	0
	Standard deviation	1.45	1.14	1.52	1.22	0.97
	Running time (s)	0.31	0.46	0.43	0.51	0.31
Griewank	Average value	130.56	19.33	45.80	98.19	8.00
	Best result	36.73	12.71	23.62	15.93	1.10
	Standard deviation	9.95×10^{3}	1.10×10^{3}	7.35×10^{3}	1.40×10^{4}	1.01×10^{3}
	Running time (s)	0.33	0.46	0.50	0.61	0.40
Rastrigin	Average value	179.70	79.53	175.79	52.4226	47.54
	Best result	113.62	61.48	118.97	47.55159	34.09
	Standard deviation	3.64×10^{3}	6.52×10^{2}	3.62×10^{3}	1.10×10^{3}	2.21×10^{2}
	Running time (s)	0.25	0.34	0.37	0.26	0.24
Rosenbrock	Average value	2.77×10^{3}	4.38×10^{2}	4.11×10^{3}	3.37×10^{2}	8.91×10^{-1}
	Best result	5.67×10^{2}	7.95×10^{2}	6.00×10^{2}	2.22×10^{2}	2.86×10^{1}
	Standard deviation	4.51×10^{6}	2.52×10^{6}	9.64×10^{6}	4.73×10^{5}	2.38×10^{5}
	Running time (s)	0.21	2.40	3.98	0.24	0.23
Sphere	Average value	2.70×10^{-1}	2.10×10^{-2}	1.20×10^{-2}	2.65×10^{-2}	5.8×10^{-3}
	Best result	2.00×10^{-1}	1.51×10^{-9}	2.88×10^{-2}	1.87×10^{-2}	7.97×10^{-192}
	Standard deviation	1.00×10^{-1}	1.71×10^{-1}	8.88×10^{-2}	7.60×10^{-1}	5.3×10^{-2}
	Running time (s)	0.88	3.36	5.33	0.95	1.09

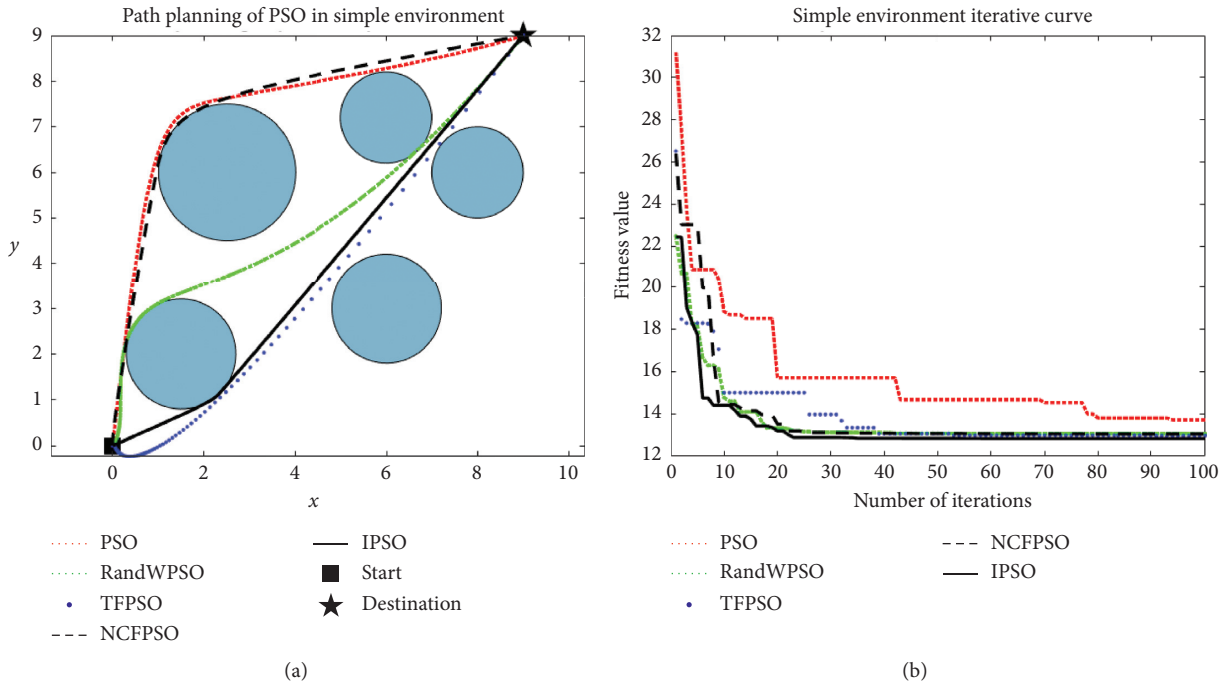

Figure 3: Experimental comparison of five algorithms in a simple environment. (a) Simple environment path plan. (b) Simple environment iteration graph.

Table 3: Comparison of path length of five algorithms in simple environment.

Algorithm	Longest path	Shortest path	Average path	Average time (s)
PSO	28.52	15.23	15.67	9.91
RandWPSO	22.89	13.51	13.90	14.10
TFPSO	27.06	13.29	14.36	15.16
NCFPSO	23.55	15.13	15.41	9.81
IPSO	22.14	12.91	13.18	8.71

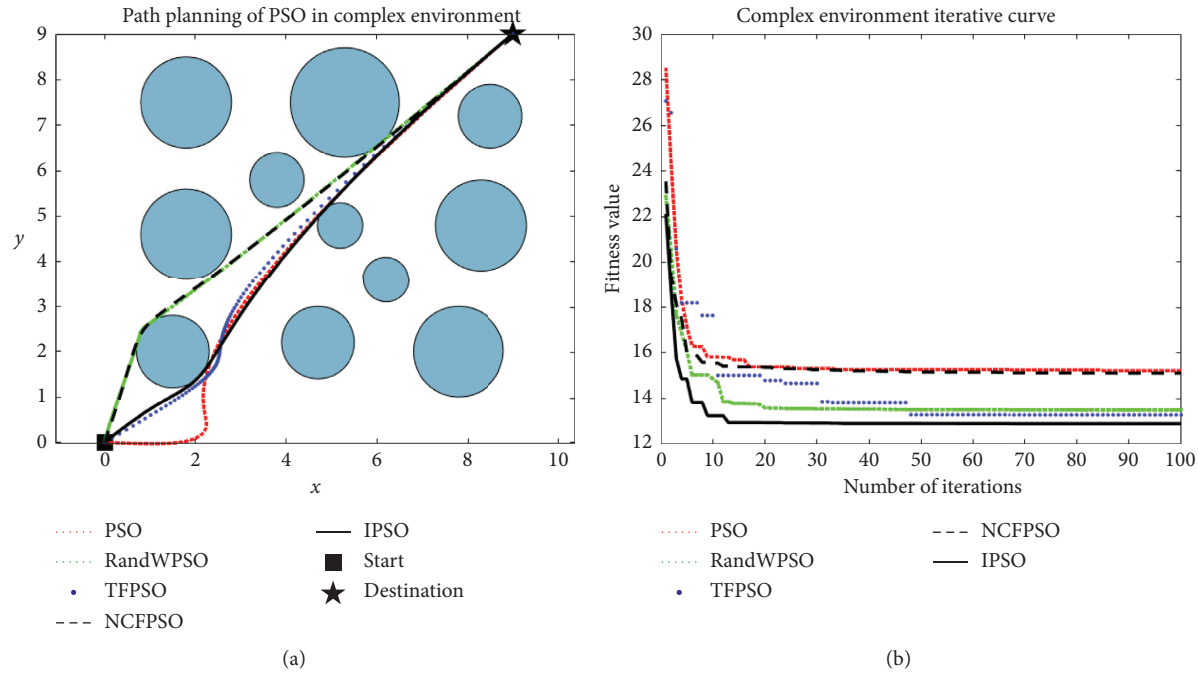

Figure 4: The first experimental comparison of five algorithms in the complex environment. (a) The first type of complex environment path planning diagram. (b) The first type of complex environment iteration graph.

(2) Complex Environment. To verify the universality of the experiment, two types of experiments were carried out. The first experiment is carried out from the starting point to the endpoint, while the second experiment is conducted from the endpoint to the starting point.

The first type of experiment: the starting point (■) is the map coordinate system (0, 0), and the endpoint (★) coordinates are (9, 9), as shown in Figure 4(a).

It can be seen from Figure 4(a) that the path planning of our proposed algorithm is not only the shortest but the smoothest. The path planning of a complex environment is relatively concentrated. This is because there are many obstacles in the complex environment with fewer paths to choose from. The classical PSO and TFPSO appear to be stagnant at the beginning of the algorithm, and the IPSO algorithm is more prominent. An excellent solution is achieved after the algorithm is optimized for a while. The optimal local solution is jumped out after its disturbance is settled. It shows that two algorithms (i.e., classical PSO and TFPSO) have small disturbances, few particle diversity, and weak ability to jump out from the local optimal solutions. Figure 4(b) presents the results obtained from the complex environment; from the iterative curves shown in Figure 4(b), the IPSO algorithm has the fastest convergence rate and the highest convergence precision with 22 iteration steps. The NCFPSO algorithm has 27 iterative steps. The RandWPSO algorithm has 28 iterations before it starts to converge. TFPSO iterates 39 times before the left and right algorithms start converging; the PSO algorithm starts to converge at around 93 iterative steps. Table 4 compares the path lengths of five algorithms in the first type of complex heterogeneous environment.

Table 4 shows that the shortest path of the IPSO algorithm is reduced by at least 3% compared with the other four algorithms, the average path is reduced by at least 2.1%, and the average simulation running time is reduced by 1.1 s. Furthermore, path planning has higher accuracy.

The second type of experiment: the starting point (■) is the origin of the map with the coordinate system of (9, 9), and the endpoint (★) coordinates are (0, 0), as shown in Figure 5(a).

By considering Table 5, the performance of the path planning of our proposed algorithm in this paper is as follow: the average path length is reduced by at least 5.2%, the shortest path length is reduced by at least 1.5%, the longest path length is reduced by 7.2%, and the average simulation running time is reduced by 1.2 s.

In summary, it is verified that our proposed IPSO algorithm has the shortest path, the highest precision, and least processing time when compared with the four algorithms existing in the literature.

4.2.3. Summary of Path Planning. In the simple environment, the path planning by the five algorithms is relatively scattered. This is because there are fewer obstacles and ample space with many paths available for the algorithm to select. When comparing the first type of experiment with the second type experimental path planning diagram in a complex environment, the five types of experiments in the first type of experiment are more concentrated when the robot starts running, and appear to be more dispersed in the later stages. This is because the first type of experiment is near the starting point. In the presence of obstacles, the algorithms avoid the obstacles according to their optimization ability. At the later stages of the optimization, the obstacles at the endpoint are sparse, thereby making the path more concentrated.

Table 4: The first comparison of path length of five algorithms in the complex environment.

Algorithm	Longest path	Shortest path	Average path	Average time (s)
PSO	31.21	13.73	15.84	10.89
RandWPSO	22.49	13.07	13.67	15.10
TFPSO	26.50	13.00	14.00	16.22
NCFPSO	26.40	13.07	13.92	10.83
IPSO	22.44	12.84	13.38	9.70

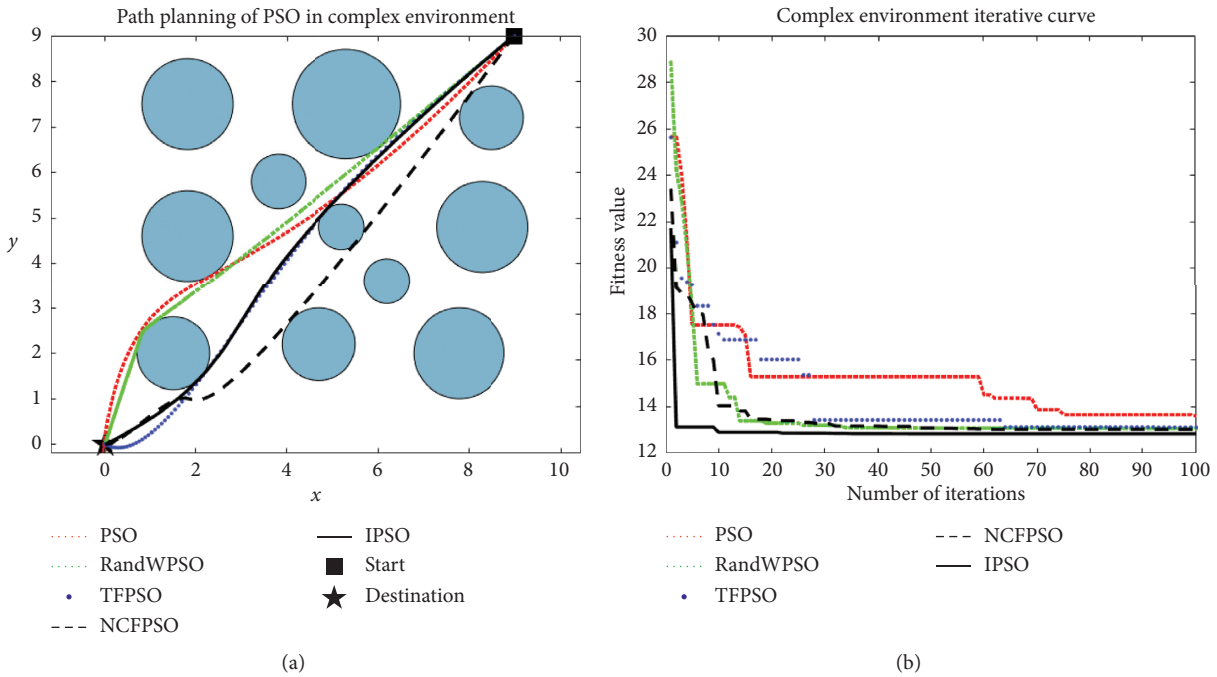

Figure 5: The second experimental comparison of five algorithms in the complex environment. (a) The second type of complex environment path planning diagram. (b) The second type of complex environment iteration graph.

Table 5: The second comparison of path length of five algorithms in the complex environment.

Algorithm	Longest path	Shortest path	Average path	Average time (s)
PSO	25.66	13.59	15.30	11.14
RandWPSO	28.86	13.07	13.76	15.07
TFPSO	25.62	13.12	14.41	16.42
NCFPSO	23.41	13.04	13.67	11.07
IPSO	21.72	12.84	12.96	10.00

The second type of experiment is the opposite of the first type. The three-path planning experiment is carried out to verify the effectiveness of our proposed IPSO algorithm among the algorithms existing in the literature. The performance comparison for the algorithms is based on the longest path, the shortest path, the average path, and the average simulation running time. In all the comparisons carried out, our proposed IPSO algorithm has performed better than the other four algorithms in the literature. This is because the algorithm proposed in this paper introduces a uniform distribution initialization strategy. Furthermore, the inertia weight and the learning factor interact in the optimization algorithm, which reduces the parameters of the proposed algorithm. Increasing the uniformity of the improved algorithm can better balance the global search of the algorithm. The introduction of the local optimization and exponential decay for inertia weight improves the searchability of our proposed algorithm and increases the disturbance of our algorithm, which subsequently improves the diversity of the particles.

5. Conclusion

In this paper, particle swarm optimization (PSO) and cubic spline interpolation are combined to solve the minimum of five test functions and robot path planning problems. In view of the problems of the classical particle swarm optimization algorithm, such as poor searchability, low convergence accuracy, easy falling into local optimal solution, poor robustness, and poor path smoothness, the classical

particle swarm optimization algorithm is improved from the following aspects: (1) The uniform initialization strategy is adopted for the population to improve the later searchability of our IPSO algorithm. This prevents the algorithm from falling into the local optimal solutions, because the random initialization of the particles is not evenly distributed, which is not conducive to the searchability at the later stages. (2) The introduction of the exponential decay for inertia weight makes the particles grow up in the early stage of the search and is beneficial to the global search. The particle step size in the later stages of the search is small in the local development, and the optimization accuracy is high. Experimental results show that the method increases the disturbance and diversity of particles to a certain extent. (3) The use of sine and cosine function can control the independent variable as the inertia weight. The learning factor makes the three variables become one variable; this reduces the parameters of the improved algorithm and thus reduces the complexity of our algorithm. The interaction of inertia weight and learning factor is increased, the unity of algorithm optimization is improved, and the performance of the algorithm is improved to a certain extent. (4) The evaluation function is constructed, and a smooth path is planned with cubic spline interpolation, which improves the accuracy and dynamic characteristics of the path. However, when compared with the classical PSO algorithm, the time advantage of the proposed algorithm is weak, which will be a crucial issue to be further studied in future research. Nevertheless, this does not affect the competitiveness of the improved algorithm. Besides, in the follow-up study, virtual obstacles will be used for the simulation experiment of multirobot path planning, and it will be used in the real environment of an operating system with hardware platform as turnabout and software platform as ROS (robot operation system), to improve the practicability of the algorithm.

Conflicts of Interest

The authors declare that they do not have any conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 51607133); Shaanxi Natural Science Basic Research Project (No. 2019JM567); China Textile Industry Federation Science and Technology Guidance Project (No. 2018094); and College Student Innovation and Entrepreneurship Training Program (No. 201910709019).

References

[1] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia, "Survey of robot 3D path planning algorithms," *Journal of Control Science and Engineering*, vol. 2016, Article ID 7426913, 22 pages, 2016.

[2] J. Votion and Y. Cao, "Diversity-based cooperative multivehicle path planning for risk management in costmap environments," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 8, pp. 6117–6127, 2019.

[3] A. Sedeño-noda and M. Colebrook, "A biobjective Dijkstra algorithm," *European Journal of Operational Research*, vol. 276, no. 1, pp. 106–118, 2019.

[4] X. Cao, X. Zou, C. Jia, M. Chen, and Z. Zeng, "RRT-based path planning for an intelligent litchi-picking manipulator," *Computers and Electronics in Agriculture*, vol. 156, pp. 105–118, 2019.

[5] A. Azzabi and K. Nouri, "An advanced potential field method proposed for mobile robot path planning," *Transactions of the Institute of Measurement and Control*, vol. 41, no. 11, pp. 3132–3144, 2019.

[6] X. Li and S. R. Qu, "An effective differential evolution algorithm for collision avoidance of vehicles under IOT (Internet of things)," *Journal of Northwestern Polytechnical University*, vol. 30, no. 5, pp. 729–733, 2012.

[7] J. W. Wu, D. Bin, X. B. Feng et al., "GA based adaptive singularity-robust path planning of space robot for on-orbit detection," *Complexity*, vol. 2018, Article ID 3702916, 11 pages, 2018.

[8] C. Xiong, D. Chen, D. Lu, Z. Zeng, and L. Lian, "Path planning of multiple autonomous marine vehicles for adaptive sampling using Voronoi-based ant colony optimization," *Robotics and Autonomous Systems*, vol. 115, pp. 90–103, 2019.

[9] Y. Q. Huang, Z. K. Li, Y. Jiang et al., "Cooperative path planning for multiple mobile robots via HAFSA and an expansion logic strategy," *Applied Sciences-Basel*, vol. 9, no. 4, p. 10, 2019.

[10] Li Xun, D. Wu, Z. Zhao et al., "Path planning method for indoor robot based on improved PSO," *Computer Measurement &Control*, vol. 28, no. 3, pp. 206–211, 2020.

[11] T. T. Gao, L. Zhang, B. D. Li et al., "Study on path planning of soccer robot based on improved particle swarm algorithm," *Journal of Xi'an Polytechnic University*, vol. 30, no. 5, pp. 609–615, 2016.

[12] P. I. Adamu, H. I. Okagbue, and P. E. Oguntunde, "Fast and optimal path planning algorithm (FAOPPA) for a mobile robot," *Wireless Personal Communications*, vol. 106, no. 2, pp. 577–592, 2019.

[13] B. Tang, Z. Zhanxia, and J. Luo, "A convergence-guaranteed particle swarm optimization method for mobile robot global path planning," *Assembly Automation*, vol. 37, no. 1, pp. 114–129, 2017.

[14] B. Song, Z. Wang, L. Zou, L. Xu, and F. E. Alsaadi, "A new approach to smooth global path planning of mobile robots with kinematic constraints," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 1, pp. 107–119, 2019.

[15] P. Chen, Q. Li, C. Zhang et al., "Hybrid chaos-based particle swarm optimization-ant colony optimization algorithm with asynchronous pheromone updating strategy for path planning of landfill inspection robots," *International Journal of Advanced Robotic Systems*, vol. 16, no. 4, p. 11, 2019.

[16] L.-Z. Du, Z. Wang, S. Ke et al., "Research on multi-load AGV path planning of weaving workshop based on time priority," *Mathematical Biosciences and Engineering*, vol. 16, no. 4, pp. 2277–2292, 2019.

[17] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of IEEE Congress on Evolutionary Computation*, IEEE Service Center, Piscataway, NJ, USA, pp. 69–73, May 1998.

[18] J. Q. Wang and X. D. Wang, "Particle swarm optimization algorithm with improved inertia weight," *Journal of Xi'an Polytechnic University*, vol. 31, no. 6, pp. 835–840, 2017.

[19] Y. Shi and R. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE, San Francisco, USA, pp. 101–106, May 2001.

[20] Y. G. Wang, T. T. Qu, and S. Li, "Disruption particle swarm optimization algorithm based on exponential decay weight," *Application Research of Computers*, vol. 37, no. 4, pp. 1020–1024, 2020.

[21] Y. Zhao and Z. Fang, "Particle swarm optimization algorithm with weight function's learning factor," *Journal of Computer Applications*, vol. 33, no. 8, pp. 2265–2268, 2013.

[22] M. G. C. Resende, C. C. Ribeiro, F. Glover et al., "Scatter search and path-relinking: fundamentals, advances, and applications," *Handbook of Metaheuristics*, vol. 146, pp. 87–107, Springer, Berlin, Germany, 2010.

[23] Z. G. Zhao, S. Y. Huang, and W. Q. Wang, "Simplified particle swarm optimization algorithm based on stochastic inertia weight," *Application Research of Computers*, vol. 31, no. 2, pp. 361–391, 2014.

[24] A. Khare and S. Rangnekar, "A review of particle swarm optimization and its applications in Solar Photovoltaic system," *Applied Soft Computing*, vol. 13, no. 5, pp. 2997–3006, 2012.

[25] S. Zhao, J. H. Zhang, and Y. F. Zhu, "An adaptive constriction factor particle swarm optimization algorithm using fuzzy inference engine," *Journal of Qingdao University*, vol. 33, no. 3, pp. 9–37, 2018.

[26] J. Yang and L. Li, "Improved biogeography-based optimization algorithm for mobile robot path planning," in *Proceedings of 2017 Chinese Intelligent Systems Conference*, Singapore, 2018.