processes

MDPI

Article

Path Planning of Mobile Robots Based on an Improved Particle Swarm Optimization Algorithm

Qingni Yuan, Ruitong Sun and Xiaoying Du *

Key Laboratory of Advanced Manufacturing Technology of the Ministry of Education, Guizhou University, Guiyang 550025, China
* Correspondence: duxiaoying12@163.com

Abstract: Aiming at disadvantages of particle swarm optimization in the path planning of mobile robots, such as low convergence accuracy and easy maturity, this paper proposes an improved particle swarm optimization algorithm based on differential evolution. First, the concept of corporate governance is introduced, adding adaptive adjustment weights and acceleration coefficients to improve the traditional particle swarm optimization and increase the algorithm convergence speed. Then, in order to improve the performance of the differential evolution algorithm, the size of the mutation is controlled by adding adaptive parameters. Moreover, a "high-intensity training" mode is developed to use the improved differential evolution algorithm to intensively train the global optimal position of the particle swarm optimization, which can improve the search precision of the algorithm. Finally, the mathematical model for robot path planning is devised as a two-objective optimization with two indices, i.e., the path length and the degree of danger to optimize the path planning. The proposed algorithm is applied to different experiments for path planning simulation tests. The results demonstrate the feasibility and effectiveness of it in solving a mobile robot path-planning problem.

Keywords: path planning; particle swarm optimization; differential evolution algorithm and self-adaption

Citation: Yuan, Q.; Sun, R.; Du, X. Path Planning of Mobile Robots Based on an Improved Particle Swarm Optimization Algorithm. *Processes* **2023**, *11*, 26. https://doi.org/10.3390/pr11010026

Academic Editors: Arkadiusz Gola, Izabela Nielsen and Patrik Grznár

Received: 14 November 2022
Revised: 17 December 2022
Accepted: 19 December 2022
Published: 23 December 2022

1. Introduction

Path planning is one of the important research directions in mobile robot technology. The path planning of a mobile robot refers to planning a collision-free path that meets certain conditions (usually the optimal) in a static or dynamic environment to reach a target point [1,2]. People can apply good path planning technology for mobile robots to robots to explore harsh environments that humans cannot reach [3], replace humans in high-risk rescues such as fire rescue [4], help visually impaired people with path guidance, and provide reliability for delicate surgery [5]. It can also be applied by people to the field of intelligent warehousing, improving material transportation efficiency and reducing manpower and material resources, which is a very meaningful research topic [6].

The traditional particle swarm optimization (PSO) was proposed by Kennedy in 1995 to search the global optimal value of objective function by imitating the foraging behavior of birds. It has the characteristics of simple structure, easy implementation, and wide application [7]. Its improved algorithms are widely used in path planning [8,9]. However, the PSO algorithm has shortcomings such as slow convergence speed, and it is easy to fall into local optimum.

This paper proposed an improved particle swarm optimization based on differential evolution (IPSO-IDE) for path planning in mobile robots. The IPSO-IDE is based on optimized differential evolution (IDE) and enhanced with improved particle swarm optimization (IPSO) to solve the limitations of the traditional particle swarm optimization. The main contributions of the text are as follows:

(1) The improved IPSO algorithm combines improved inertia weight ω*, adaptive parameter β, and the concept of corporate governance to improve the traditional particle swarm optimization and increase the algorithm convergence speed.

(2) Aiming at the shortcomings of the traditional DE algorithm, the scaling factor F and the cross-probability factor CR are adaptively optimized, so that the algorithm can adaptively control the search accuracy and the degree of mutation to improve the optimization accuracy of the algorithm.

(3) It proposes a new objective function applied to path planning, which is composed of a path length function and a penalty function, simplifying the path planning problem of mobile robots into an objective function optimization problem.

The remainder of this paper is arranged as follows. Section 2 describes a bibliographic review of related work that includes classical heuristic algorithms and provides the application of improved algorithms in path planning problem. Section 3 describes principles of PSO and DE. Section 4 describes the proposed improved particle swarm optimization based on differential evolution (IPSO-IDE). Section 5 describes its application to the path-planning problem and introduces experiments and result analysis. In the last section, conclusions are drawn.

2. Related Work

The path-planning algorithm of mobile robots has been deeply studied at home and abroad, and the results are remarkable. Traditional path planning algorithms mainly include artificial potential field method [10], element decomposition method [11], graph search algorithm [12], etc., but when the obstacles are complex, there are many disadvantages, such as a large amount of calculation, easy to fall into local optimum, and the obtained path is not smooth, easy to appear sharp points, which is not in line with the actual situation, increasing the workload of mobile robots [13,14]. At present, many experts use heuristic algorithms to optimize path planning [15], including genetic algorithm (GA) [16], particle swarm optimization (PSO) [17], artificial bee colony algorithm (ABC) [18], grey wolf algorithm (GWO) [19], ant colony algorithm (ACO) [20], differential evolution algorithm (DE) [21], etc., and obtain good results.

GA is an intelligent bionic algorithm proposed earlier, which is the theoretical basis of many algorithms. It simulates the calculation model of Darwinian biological evolution theory, mainly including the steps of establishing the initial population, calculating the individual fitness of the population, and iterating out the optimal individual through cross-mutation and other operations [22]. Since the mutation operation of GA is not targeted, the probability of forming a better offspring population is not high. As a classical algorithm, GA is often mixed with heuristic algorithms to solve problems, such as ant colony algorithm (ACO). Kamel et al. combined PSO and GA to improve the prediction performance of the model [23]. Memon et al. proposed a hybrid optimization algorithm based on GA and APSO [24]. However, the improved GA is still inefficient in solving the problem [25].

Among the heuristic algorithms, PSO and DE are simpler in structure and easier to implement, so they are widely used [26]. The improvement of PSO generally focuses on the adjustment of population structure and the optimization of update formulas of speed and position [27]. Burman R et al. proposed the democracy-inspired particle swarm optimizer with the concept of peer groups to increase the speed of convergence [28]. Zhao et al., in order to avoid falling into local optimal solutions and increase the diversity of particles, introduced a nonlinear recursive function to adjust the inertia weight [29]. Yu et al. proposed a novel hybrid particle swarm optimization (PSO) algorithm, namely SDPSO [30]. Pozna et al. proposed a hybrid metaheuristic optimization algorithm that combines Particle Filter (PF) and Particle Swarm Optimization (PSO) algorithms [31]. Mohammed Hussein et al. proposed a modified Particle Swarm Optimization (PSO), which is named MPSO [25]. Liu et al. proposed a hybrid path planning algorithm based on optimized reinforcement learning (RL) and improved particle swarm optimization (PSO) [32]. However, the improved algorithm still has some limitations, such as low convergence accuracy, easy precocity, and so on.

The differential evolution algorithm (DE) was proposed by R. Storn et al. in 1997. It inherits the evolutionary ideas of genetic algorithms and is more concise and effective. Due

to its outstanding optimization effect, it is widely developed and used by experts. The DE is also used to optimize other algorithms to achieve a more ideal effect of optimization [33]. R. Chai et al. used the theory of game theory to optimize the DE algorithm, combined with adaptive parameter adjustment to improve the convergence accuracy [34]. Lin C et al. introduced the concept of grouping into the DE to improve its local search ability [35]. Wang et al. proposed a distributed differential evolution (DDE) algorithm, which is called AED-DDE, for solving MMOPs [36]. Liu et al. proposed a hybrid DE algorithm based on the lion swarm optimization [37]. Xu M and Wang Y proposed a time series prediction study based on improved differential evolution and echo state network to optimize the echo state network [38].

In general, in view of the shortcomings of PSO, this paper makes the following improvements to PSO: introducing the concept of corporate governance, adding a leader particle to lead the population to the optimal position. In addition, to strengthen the convergence speed of the algorithm, the paper introduces the new update formulas of speed and position by proposing a kind of adaptive operator and introducing the adaptive parameters that can control the size of the degree of mutation to improve the DE. The improved DE algorithm is used to optimize the PSO model and improve the defect that PSO is easy to fall into local optimum. Finally, the improved algorithm is applied to solve the problem of path planning. The objective function is constructed in this paper to optimize the path planning, which is composed of path length function and penalty function, and the cubic spline interpolation method is used to smooth the path. The proposed algorithm can effectively improve the path planning ability of mobile robots in static and complex environment models, improve planning efficiency, and generate a collision-free path with the shortest path length to the target point.

3. Principles of PSO and DE

This section may be divided by subheadings. It should provide a concise and precise description of the experimental results, their interpretation, as well as the experimental conclusions that can be drawn. This section introduces the traditional particle swarm optimization (PSO) [17] and differential evolution algorithm (DE) [21], which form the basis of the proposed algorithm.

3.1. Particle Swarm Optimization (PSO)

In the target search space of dimension D, the number of particles in the population is set to N. The ith ($i = 1, 2, \ldots, N$) particle contains two attributes, namely position $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ and velocity $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$, and the individual optimal position experienced is $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$. The global optimal position obtained after judging the entire particle swarm $G = (g_1, g_2, \ldots, g_D)$. The update of velocity and position can be expressed as the following mathematical formulas:

$$V_i^{t+1} = \omega V_i^t + c_1 r_1 (P_i^t - X_i^t) + c_2 r_2 (G^t - X_i^t) \quad (1)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1}, i = 1, \ldots, N \quad (2)$$

where ω represents the inertia weight to suppress the inertial speed of the previous iteration, which enables the algorithm to adaptively adjust the inertial speed during the iteration process. As the number of iterations increases, ω decreases accordingly, which makes the algorithm have a larger search speed at the early stages of iterations to improve the ability of search and ensure the efficiency of operation. Then, at the later stages of iterations, the search speed is decreased to improve the accuracy of search. The specific formula is as follows:

$$\omega = \omega_{\max} - (\omega_{\max} - \omega_{\min}) \cdot \frac{t}{T} \quad (3)$$

where $\omega_{\max}$ and $\omega_{\min}$ represent the maximum and minimum values of ω, t is the number of iterations, and T is the maximum number of iterations.

In addition, r_1 and r_2 are random numbers in the range of [0,1], the purpose is to increase search range of the algorithm. c_1 and c_2 are learning factors, which are values greater than or equal to zero, where c_1 represents the cognition factor, which refers to the cognition of each particle to control the influence of ith particle in the local range. c_2 represents the social factor, which refers to the influence ability of social level to control the influence of the optimal particle in the particle swarm on the entire area. The two learning factors work together to promote the particles to continuously move closer to the optimal position.

P_i^t is the individual optimal position of the ith particle in iteration t, G_i^t is the global optimal position of the particle swarm in iteration t. The individual optimal position and the global optimal position can be expressed as the following mathematical formulas:

$$P_i^{t+1} = \begin{cases} P_i^t, & if \quad Fit(X_i^{t+1}) > Fit(P_i^t) \\ X_i^{t+1}, & if \quad Fit(X_i^{t+1}) < Fit(P_i^t) \end{cases} \tag{4}$$

$$G_i^t = \min\left\{Fit(P_1^t),\ Fit(P_2^t),\ \ldots,\ Fit(P_N^t)\right\} \tag{5}$$

3.2. Differential Evolution Algorithm (DE)

The principle of DE algorithm is similar to that of GA [16]. It also includes three operations: "mutation", "crossover" and "selection", but there are differences in mutation and crossover operations. Among them, GA adheres to the principle of "survival of the fittest" and directly compares the parents and offspring to select the individuals with higher fitness, while the DE algorithm introduces difference vector to mutate, which has better structure and higher efficiency.

The specific implementation steps of DE are as follows:

1. Establish the initial population and initialize the parameters:

Randomly generate N individuals uniformly in the solution space $\{x_{j\_\min}, x_{j\_\max}\}$, $j = 1, 2, 3, \ldots, D$. The individual is the candidate solution vector. The dimension is set to D. The expression of the ith candidate solution vector of the initial population is as follows:

$$X_i(0) = (x_{i1}(0), x_{i2}(0), \ldots, x_{iD}(0)), \quad i = 1, 2, \ldots, \ N$$

The numerical initialization formula for the ith individual in dimension j is as follows:

$$x_{ij}(0) = x_{j\_\min} + rand(0,1) \cdot (x_{j\_\max} - x_{j\_\min})$$

2. Mutation operation:

The DE algorithm randomly selects individuals for differential mutation based on the parent individuals $x_i(t)$, $i = 1, 2, \ldots, M$. The mutation strategy is as follows:

$$h_i(t) = x_{p1}(t) + F \cdot (x_{p2}(t) - x_{p3}(t)) \tag{6}$$

where $h_i(t)$ is the generated variation vector, $x_{p1}(t)$, $x_{p2}(t)$, $x_{p3}(t)$ are solution vectors numbered $p1$, $p2$, $p3$ of the population in iteration t, the numbers are randomly selected and are different from i. F is the scaling factor to appropriately scale the difference vector, and the value range is generally controlled at (0, 1.2].

3. Cross operation:

The parent vector $x_i(t)$, i = 1, 2, ..., M (M represents the dimension of the parent vector) crosses the mutation vector $h_i(t)$ with the crossover probability CR to generate a new individual vector that is the test vector v_{ij}(t), and the test vector in dimension j

(j = 1, 2, ..., D) is selected from the parent and the mutation vector according to CR; the formula is as follows:

$$v_{ij}(t+1) = \begin{cases} h_{ij}(t), & if \quad rand \le CR \quad or \quad j = j_{rand} \\ x_{ij}(t), & else \end{cases} \tag{7}$$

where $rand$ is a uniform random number in the range of [0,1]. CR is crossover probability factor, $CR \in$ [0,1]. j_{rand} is a random positive integer in the range of [0,1], so that at least one component is produced by the mutation vector, thereby ensuring that a new vector is generated.

4. Select operation.

The vector generated after cross mutation operation $v_i(t+1)$ is compared with the parent vector $x_i(t)$, and the vector with better fitness value ($fit(u)$) is retained. The formula is as follows:

$$x_i(t+1) = \begin{cases} v_i(t+1) \ , & if \quad fit(v_i(t+1)\ ) \ge fit(x_i(t)\ ) \\ v_i(t)\ , & else \end{cases} \tag{8}$$

4. Algorithm Improvement

4.1. Improved Particle Swarm Optimization Based on Corporate Governance Idea (IPSO)

The traditional PSO [17] contains the idea of leadership: an optimal particle is selected from the particle swarm, representing the global optimal solution G, which is the only leader to lead the particle swarm to the optimal position. However, the thought of single leadership has its limitations and cannot guarantee that the direction of leader guidance will always be correct, which causes the premature convergence and fall into a local optimum. In response to this problem, this paper adds the idea of corporate governance into the particle swarm optimization and optimizes its parameters.

In economics, the idea of corporate governance [39] means that there are two rights in an enterprise, namely, Ownership and Management rights. Good owners and operators are administrators with strong ability. The owner gives the operator the right to manage a company and lead the company's employees to develop in a better direction. If the company's profits are not good under the management of the operator, the owner will come forward to make decisions and deprive the operator of the management right. Two rights check and balance to promote enterprise development. Based on the above concepts, this paper improves the algorithm. The particles temporarily occupying the management position are named as the administrator particle (Adm). The two possible administrator particles, the Operator particle and the Owner particle, are elected through the voting mechanism. The Owner particle does not often come forward to make decisions. It plays a role of supervision and control, checking whether the Operator particle has always led the particle swarm to the optimal position. In addition, based on the corporate culture that a good working atmosphere can improve the work efficiency of employees, and the behavior of employees affects each other, the concept of peer group is proposed. The neighboring particles are regarded as a peer group, and their optimal positions influence each other. The local optimal value (Lbest) is selected in the peer group.

In this paper, the traditional PSO is improved by introducing the concept of corporate governance and adding adaptive parameters and acceleration coefficients. The update of particle positions in two successive iterations can be shown in Figure 1, and the specific update formulas of velocity and position are as follows:

$$V_i^{t+1} = \omega^* \cdot V_i^t + c_1 \cdot r_1 \cdot (P_i{}^t - X_i{}^t) + c_2 \cdot r_2 \cdot (Lbest_i{}^t - X_i{}^t) + \beta \cdot c_3 \cdot r_3 \cdot \psi \cdot (Adm^t - X_i{}^t) \tag{9}$$

$$X_i^{t+1} = X_i^t + c_4 \cdot r_4 \cdot V_i^{t+1} \tag{10}$$

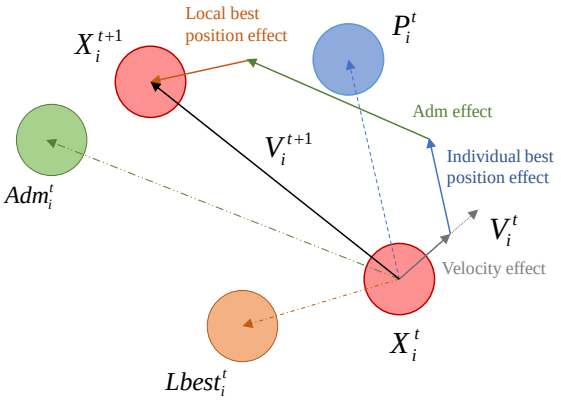


Figure 1. The update of particle positions in two successive iterations (i.e., t to $t+1$).

1. $\omega^* \cdot V_i^t$

The first term of (9) is the factor of inertia, which is the product of an adaptive inertia weight the paper proposed ω^* and the velocity of the previous iteration t. The adaptive inertia weight ω^* is based on the original inertia weight ω, adding a trigonometric function. This is because, after adding the trigonometric function, the step size can be adjusted to increase and the speed can be accelerated to avoid the precocious search and enter the local optimal state, thus improving the accuracy, which is expressed by following formula:

$$\omega^* = {\omega^*}_{\max} - ({\omega^*}_{\max} - {\omega^*}_{\min}) \cdot \sin(\frac{\pi \cdot t}{2 \cdot T})$$

where ${\omega^*}_{\max}$ and ${\omega^*}_{\min}$ represent the maximum and minimum values of ω.

2. $c_1 \cdot r_1 \cdot (P_i^t - X_i^t)$

The second term of (9) calculates the distance between the current position X_i^t of ith particle iterating to the iteration t and the individual best position of the particle P_i^t, multiplied by the acceleration factor c_1 and the random number $r_1 \in [0, 1]$.

3. $c_2 \cdot r_2 \cdot (Lbest_i^t - X_i^t)$

The third term of (9) calculates the distance between the current position X_i^t and the best position of ith particle in the peer ring or peer group area $Lbest_i^t$ when it is iterated to the iteration t, multiplied by the acceleration factor c_2 and random number $r_2 \in [0, 1]$.

4. $\beta \cdot c_3 \cdot r_3 \cdot \psi \cdot (Adm^t - X_i^t)$

The fourth term of (9) calculates the distance between the X_i^t and the global best position of the particle swarm Adm^t when it is iterated to the iteration t and multiplied by the acceleration factor c_3 and the random number $r_3 \in [0, 1]$. In addition, in order to better control the influence of Adm^t on the particle swarm, this paper introduces a trigonometric function and proposes an adaptive parameter β to speed up the algorithm's convergence speed. The parameter ψ means administrator regulatory factor.

Since the speed of the traditional particle swarm optimization algorithm is fixed when searching on a path, the adaptive parameter β can be adjusted adaptively to accelerate the speed and avoid falling into the local optimal. The formula of β is as follows:

$$\beta = \beta_{\min} + (\beta_{\max} - \beta_{\min}). \cdot \cos(\frac{\pi \cdot t}{2 \cdot T}) \quad (11)$$

where $\beta_{\min}$ and $\beta_{\max}$ are the maximum and minimum values of β.

Adm is generated based on the concept of voting [40], and particles follow the leader elected by voting. *Opvote* is the number of votes cast by Operator, and *Owvote* is the number of votes cast by Owner.

As the iteration progresses, the leadership of the Operator gradually increases, i.e., *Opvote* increases and *Owvote* decreases, so that the influence of the Owner weakens and the convergence speed increases. However, if the Operator cannot lead the particles to the optimum, the Owner needs to control the power. In this case, even if the number of *Owvote* is small, the influence of the Owner must be increased, so the Administrator regulatory factor $\psi = e^{(1-vote_{Adm})}$ is introduced.

The initial population is randomly generated, so it is necessary to artificially impose a preference first by asymmetric processing of the initial voting range of two candidates of administrator particles. The asymmetric range is controlled within [0, 1] by standardization. Therefore, in the initial iteration, the particles choose the Operator as *Adm* with a greater probability, and the Owner is their second choice that plays a supervisory role. The initial value of *Opvote* is set to φ, and *Owvote* is 1-φ, which makes the number of votes biased, where $\varphi \in [0, 1]$.

The voting mechanism is realized by roulette algorithm [16]. Each particle votes to select a leader $vote_i^d$ that is a random number within the range of [0,1]. The selection of *Adm* can be expressed as follows:

$$Adm_i^d = \begin{cases} Operator^d, & if\ vote_i^d \le Opvote \\ Owner_d, & else \end{cases} \tag{12}$$

The number of votes is accumulated and updated to record the influence of Operator and Owner. The expressions are as follows:

$$\begin{aligned} if \quad & vote_i^d < Opvote: \\ & Opvote = Opvote + \tfrac{1}{M \cdot N} \\ else: \quad & Owvote = Owvote + \tfrac{1}{M \cdot N} \end{aligned}$$

where M represents the number of votes that particles cast to the corresponding candidate leader in a given dimension. The cumulative value of support votes is the total number of support votes, which can intuitively indicate which candidate can be elected as the *Adm*. In order to facilitate further calculations, the voting should be standardized after each iteration. The expressions used for standardization are as follows:

$$Opvote = \frac{Opvote}{Opvote + Owvote}, \quad Owvote = \frac{Owvote}{Opvote + Owvote}$$

$$if \quad Opvote = \varphi: \ Owvote = 1 - \varphi$$

$vote_{Adm}$ refers to the standardized number of votes obtained by a specific administrator (*Operator* or *Owner*). The factor $e^{(1-vote_{Adm})}$ is calculated as follows:

$$\begin{aligned} & Case1 \\ & Adm_i^d = Operator^d, vote_{Adm} = Opvote \approx 1 \\ & e^{(1-vote_{Adm})} \approx e^0 = 1 \\ & Case2 \\ & Adm_i^d = Owner^d, vote_{Adm} = Owvote \approx 0 \\ & e^{(1-vote_{Adm})} \approx e^1 \approx 2.3 \end{aligned}$$

The update formulas of Operator and Owner are as follows:

$$Owner_d = \begin{cases} \phi \cdot (X_{\max} - X_{\min}) + X_{\min}, & if\ \ rand_d < pro \\ Operator_d, & else \end{cases}$$

$$Operator_d = gbest_d$$

where the parameters of ϕ and $rand_d$ are uniform random numbers in the range [0,1], $pro = 1/N$, $d \in \{1, 2, 3, \ldots, D\}$. See Algorithm 1 at the end of the article.

5. The improved mathematical formula of pos.ition update formula

In (10), the particle position X_i^{t+1} in iteration $t + 1$ is equal to the sum of the historical position X_i^t and the velocity V_i^{t+1} multiplied by the acceleration factor c_4 and the random number $r_4 \in [0, 1]$.

In addition, since particles search in the solution space at velocity v, if the particle velocity exceeds the range and flies out of the solution space, it affects the algorithm solution. In order to solve the above problem, this paper performs boundary treatment on the velocity ${v_d}^i$, which can quickly absorb the particle whose velocity exceeds the velocity boundary:

$$\begin{array}{ll} if & v_d^i > v_{\max} : v_d^i = v_{\max}; \\ if & v_d^i < v_{\min} : v_d^i = v_{\min} \end{array} \tag{13}$$

where ${v_d}^i$ is the component of the ${v_i}^t$ of generation t in dimension d, and $v_{\min}$ and $v_{\max}$ are the minimum and maximum values of velocity.

4.2. Improved Differential Evolution Algorithm Based on Adaptive Parameters

The DE [21] has a simple structure and fast convergence speed. The algorithm has two main parameters, namely the scaling factor F and the probability factor of cross CR. In the standard DE, these two parameters are fixed values. In order to improve the convergence performance and iterative accuracy of the algorithm, these parameters can be adjusted to adaptive parameters.

1. Adaptive optimization of scaling factor F

F can control the degree of variation. A larger value of F represents a larger degree of mutation, which can expand the search range of the algorithm to be conducive to the overall progress, but it may be premature in the later stage of the iteration. A smaller value of F represents a smaller degree of variation, which is conducive to the local search to improve the search accuracy, but it is easy to fall into the local optimal solution.

Therefore, F is adaptively improved in this paper, and the expression is as follows:

$$F_i = F_{\min} + (F_{\max} - F_{\min}) \cdot \frac{fit(x_{p2}(t)) - fit(x_{p1}(t))}{fit(x_{p3}(t)) - fit(x_{p1}(t))} \tag{14}$$

where F_i represents the scaling factor of the ith vector of the population. $fit(x_{p1}(t))$, $fit(x_{p2}(t))$, $fit(x_{p3}(t))$ represent the fitness values of the vectors x_{p1}, $x_{p2,}$ x_{p3}. $F_{\min}$, $F_{\max}$ are the minimum and maximum values of the scaling factors. If the difference between the fitness values of x_{p2} and x_{p3} is too large, it expands the search range of the algorithm, which is not conducive to the improvement of search accuracy, so let the algorithm have a smaller F_i. On the contrary, the two vectors with similar fitness values are easy to fall into the local optimal solution, so F_i is needed to expand to improve the degree of variation.

2. Adaptive optimization of cross probability factor CR

In order to improve the convergence speed of the algorithm, the crossover probability factor is improved. CR is a factor that affects the degree of crossover between the parent vector and the mutation vector. If CR is too large, the degree of crossover is increased, but it may make the individuals with better fitness suffer from damage due to a too-large mutation degree. If CR is too small, the degree of cross-mutation is not enough, which may fall into local convergence and reduce the search efficiency. This paper proposes to compare the fitness of a specified individual with the average of the population fitness values. If it is smaller than the average, it means that the individual is better, and then its cross-variation is reduced. If it is larger than the average, it means that the individual needs to be further

optimized, i.e., to increase its variation degree and to promote the search for the optimal individual. The specific expression is as follows:

$$CR_i = \begin{cases} if \quad fit(x_i) \geq fit_{mean} : \\ \qquad CR_{\min} + (CR_{\max} - CR_{\min}) \cdot \frac{fit(x_i) - fit_{\min}}{fit_{\max} - fit_{\min}} \\ else : CR_{\min} \end{cases} \tag{15}$$

where CR_i is the cross-probability factor corresponding to the ith vector of the population, $CR_{\min}$ and $CR_{\max}$ are the minimum and maximum values of CR. $fit(x_i)$ is the fitness value of the vector x_i, $fit_{\min}$ and $fit_{\max}$ are the minimum and maximum values obtained after comparison of fitness values. fit_{mean} is the average of fitness values.

4.3. Hybrid IPSO with IDE (IPSO-IDE)

4.3.1. The Principle of IPSO-IDE Algorithm

In order to improve the optimization ability of PSO, this paper combines the concepts of corporate governance and voting and introduces adaptive factors to optimize the convergence speed. However, the principle of PSO is to update the velocity and position of the particles through continuous iterations to move closer to the optimal position, and the movement of the particles is simple and easily affected by other particles, i.e., the properties of the algorithm itself make it possible for PSO to converge in a non-optimal position when solving the optimization problem. In addition, DE compares the parent vector with the target vector generated by mutation and crossover in the early stage, so that it has high convergence speed and can be used to optimize PSO.

In summary, the paper improves traditional PSO and DE and integrates them. The DE is used to break the limitation of PSO and propose a new hybrid PSO-DE optimization mechanism, which includes the idea of "mutual benefit and win-win", i.e., the two algorithms are cooperative partnerships, which means that after one party optimizes its own capabilities through the partner, in turn, it provides greater benefits to the partner to achieve mutual benefit results for the two algorithms. Based on this idea, this paper proposes an improved IPSO-IDE algorithm, i.e., an improved particle swarm algorithm based on differential evolution. Both IPSO and IDE are in cooperative modes. IDE is used to optimize the *Adm* of IPSO to make the *Adm* position closer to the optimal position and improve its ability to guide particles to the optimal position. In this way, the performance of the population after iteration is better. This kind of population is called "elite population", and it can obtain better results by using elite population to train IDE algorithm.

The implementation steps of proposed IPSO-IDE are as follows:

step1 Initialization parameters, including acceleration factor, number of support votes *Opvote*, and number of negative votes *Owvote*, etc.

step2 Initialize the particle swarm randomly, including dimension D, number of population particle N, position X, velocity V, etc.

step3 Calculate the particle fitness value *Fit* according to the set objective function.

step4 Based on the fitness value *Fit*, calculate the individual best position P_i^t, the local best position $Lbest_i^t$, and elect *Adm* according to (12) to obtain the global best position.

step5 Update position X and velocity V according to the improved (9) (10) to generate the elite population with high quality.

step6 Use (13) to process the boundary.

step7 Use the elite population as the initial population of IDE, combine the adaptive parameters (14) (15), and use (6) (7) (8) to perform "high intensity" iterative optimization.

step8 Apply the optimized result of IDE algorithm to Leader of the updated particle swarm.

step9 If the termination condition is met, stop the algorithm and output the optimal results. Otherwise, go to Step2.

See Algorithm 2 at the end of the article.

Among them, the proposed algorithm is divided into two parts, which are cooperated by IPSO and IDE. DE has the advantage of fast convergence in the early stage. Updating

the particle position by the elite population optimized by IDE can make the algorithm have faster optimization speed and reduce the overall influence of IPSO. In addition, because the algorithm has two possible administrator particles, the selected *Adm* can better lead the particles to the optimal position. Therefore, the paper increases the influence of *Adm*, and sets up the following parameters: $c_1 = 0.5$, $c_2 = 0.5$, $c_3 = 1.2$, $c_4 = 1$, $\phi = 0.7$.

4.3.2. Experimental Verification of IPSO-IDE Model

In order to prove the performance superiority of IPSO-IDE proposed, the paper selects nine classical test functions for verification. The specific form of the function is given in Table 1, including the name of test function, expression, value range of independent variables, the minimum value (min) in the interval, and the dimension of function (Dim).

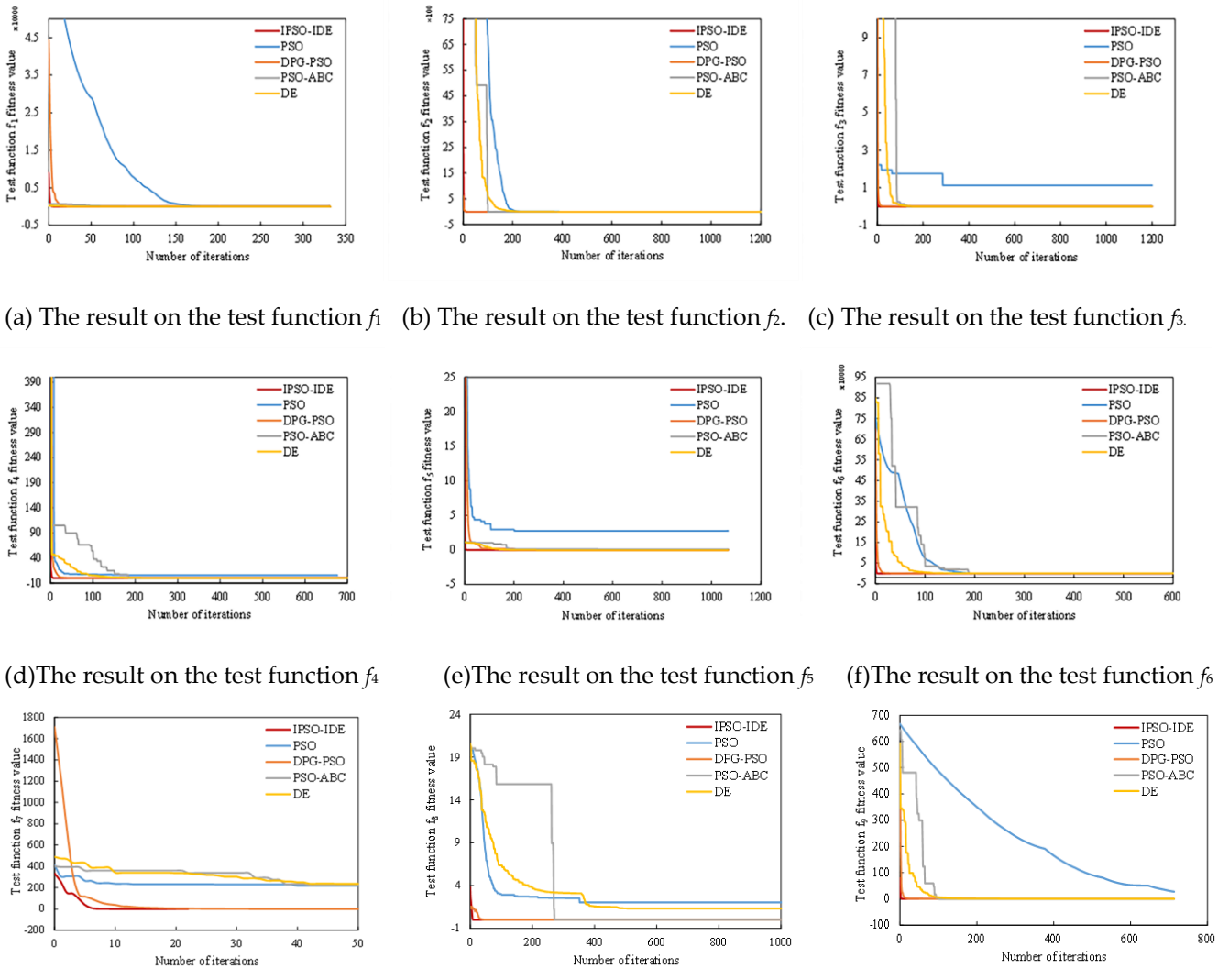
Table 1. Performance of IPSO-IDE with other algorithms based on test functions.

Test Function	Expression	Value Range	Min	Dim
Sphere	$f_1 = \sum_{i=1}^{D} x_i^2$	[−100,100]	0	30
Step	$f_2 = \sum_{i=1}^{D} (floor(x_i + 0.5))^2$	[−10,10]	0	30
H14	$f_3 = \exp(0.5 \cdot \sum_{i=1}^{D} x_i^2) - 1$	[−1.28,1.28]	0	30
Schwefel's P2.22	$f_4 = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	[−10,10]	0	30
Alpine	$f_5 = \sum_{i=1}^{D} \lvert x_i \cdot \sin x_i \rvert + 0.1 \cdot \lvert x_i \rvert$	[−10,10]	0	30
Quadric	$f_6 = \sum_{j=1}^{D} (\sum_{i}^{j} x_i)^2$	[−100,100]	0	30
Rastrigin	$f_7 = \sum_{i=1}^{D} [x_i^2 - 10 \cdot \cos 2\pi x_i + 10]$	[−5.12,5.12]	0	30
Ackley	$f_8 = -20 \cdot \exp(-0.2 \cdot \sqrt{\frac{1}{D} \cdot \sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D} \cdot \sum_{i=1}^{D} \cos 2\pi x_i) + 20 + e$	[−32,32]	0	30
Griewank	$f_9 = \frac{1}{4000} \cdot \sum_{i=1}^{D} x_i^D - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$	[−600,600]	0	30

The algorithm is tested on each test function, and it makes a comparison with other representative algorithms such as traditional particle swarm optimization (PSO) [17], democracy-inspired particle swarm optimizer with the concept of peer groups (DPG-PSO) [37], hybrid algorithm based on PSO and ABC (PSO-ABC) [9], and improved differential evolution algorithm (IDE). The parameters are shown in Table 2. The initial parameters for all competitor algorithms are set the same as the proposed algorithm. The maximum number of iterations is 2000, and the population size is set to 20. In order to ensure the validity of the algorithm results, each test function is independently tested 20 times. The experimental results are shown in Table 3, where Mean denotes the average of output results from 20 tests, Best represents the best value in the results of 20 tests, Miter denotes the number of iterations used for the best value, and Std denotes the standard deviation, which can determine the stability of the algorithms. Figure 2 shows the relationship between the optimal fitness of the test functions and the number of iterations.

Table 2. Parameter settings of the different test algorithms (meaning of the symbols are provided in the respective literates).

Algorithm	Parameter
IPSO-IDE	$c_1 = 0.5$, $c_2 = 0.5$, $c_3 = 1.2$, $c_4 = 1$; $\omega = 0.4$–0.2; $\phi = 0.7$; $\beta = 1$–1.5; Vmax = 0.6 × Range; F = 0.6; CR = 0.9–0.1
PSO	$c_1 = 2$; $c_2 = 2$; $\omega = 1$; Vmax = 0.5 × Range; Vmax = 0.1 × Range
DPG-PSO	$c_1 = 2$, $c_2 = 1.5$, $c_3 = 0.5$, $c_4 = 0.8$; $\omega = 0.2$; $\phi = 0.7$; Vmax = 0.5 × Range
PSO-ABC	$c_1 = 2$, $c_2 = 2$; $\omega = 0.95$–0.4; Vmax = Range
IDE	F = 0.6; CR = 0.9–0.1; Vmax = Range



(a) The result on the test function f_1 (b) The result on the test function f_2. (c) The result on the test function f_3.

(d)The result on the test function f_4 (e)The result on the test function f_5 (f)The result on the test function f_6

(g)The result on the test function f_7 (h)The result on the test function f_8 (i)The result on the test function f_9

Figure 2. The optimal fitness curves obtained based on test functions f1−f9 by five algorithms.

Table 3. Test function.

Algorithm	Evaluation Index	IPSO-IDE	PSO	DPG-PSO	PSO-ABC	IDE
f_1	Mean	0	1.539	0.075	2.9×10^{-5}	1.7×10^{-27}
	Best	0	1.330	0.002	2.83×10^{-5}	1.12×10^{-30}
	Worst	0	1.86	0.236	3.0×10^{-5}	1.6×10^{-26}
	Std	0	0.1809	0.0573	4.29×10^{-7}	3.93×10^{-27}
	M-iters	332	1180	214	425	1993
f_2	Mean	0	0	0	0	0
	Best	0	0	0	0	0
	Worst	0	0	0	0	0
	Std	0	0	0	0	0
	M-iters	15	1118	21	100	384
f_3	Mean	0	1.1045	2.79×10^{-6}	5.36×10^{-6}	0
	Best	0	0.88	1.36×10^{-8}	3.82×10^{-6}	0
	Worst	0	1.15	1.21×10^{-5}	6.84×10^{-6}	0
	Std	0	0.1067	4.06×10^{-6}	9.82×10^{-7}	0
	M-iters	16	286	279	1065	1173
f_4	Mean	0	5.36	0.362	0.029	1.01×10^{-21}
	Best	0	5.04	0.19	0.029	1.94×10^{-22}
	Worst	0	5.83	0.7	0.029	2.04×10^{-21}
	Std	0	0.2892	0.1968	3.88×10^{-18}	7.3×10^{-22}
	M-iters	19	1248	135	186	1997
f_5	Mean	0	3.6045	0.0116	1.7435	2.88×10^{-19}
	Best	0	1.99	0.002	0.039	8.01×10^{-28}
	Worst	0	4.8	0.04	2.41	6.73×10^{-19}
	Std	0	1.1845	0.0123	1.1265	6.73×10^{-19}
	M-iters	18	335	126	2000	2000
f_6	Mean	0	9.582	0.923	4.11×10^{-4}	3.98×10^{-29}
	Best	0	7.5	0.105	4×10^{-4}	3.14×10^{-29}
	Worst	0	10.27	2.56	4.2×10^{-4}	4.76×10^{-29}
	Std	0	1.0278	1.2421	1.15×10^{-5}	1.13×10^{-29}
	M-iters	400	1432	463	765	2000
f_7	Mean	0	38.066	26.629	0.0051	15.17
	Best	0	27.98	19.14	0.004	10.94
	Worst	0	59.34	35.63	0.0058	19.9
	Std	0	12.1852	7.5275	0.001	4.4048
	M-iters	25	503	208	665	738
f_8	Mean	3.09×10^{-16}	8.432	9×10^{-4}	0.004	0.4575
	Best	4.4×10^{-17}	7.18	6.58×10^{-5}	0.004	7.55×10^{-15}
	Worst	4.44×10^{-16}	9.51	1.8×10^{-3}	0.004	1.34
	Std	2.31×10^{-16}	0.8422	8.26×10^{-4}	0	0.6338
	M-iters	51	430	182	548	1831
	Grade	N	-	-	-	-
f_9	Mean	0	35.96	0.0664	2.01×10^{-6}	0.0147
	Best	0	31.05	0.037	2.01×10^{-6}	0
	Worst	0	41.02	0.147	2.01×10^{-6}	0.0172
	Std	0	3.97	0.0495	0	0.0218
	M-iters	21	2000	341	112	1222

5. Path Planning Based on IPSO-IDE Algorithm

Through functional tests, the new IPSO-IDE optimization algorithm proposed in this paper has good performance. In order to verify the practicability of the algorithm, different scenarios are set to carry out the path-planning simulation experiment of the algorithm.

5.1. Design of Fitness Function

Fitness function has an important impact on the function of evolutionary algorithms. According to the actual needs of the path-planning problem, the evaluation index mainly includes two factors: path length and degree of risk. The fitness function needs to be constructed based on the evaluation index, so the objective function of this paper is composed of path length function and penalty function. The paper sets the coordinate of starting point as Start (x_0, y_0) and the coordinate of target point as Goal (x_0, y_0). Each particle in the particle swarm represents a set of node coordinates along a path, denoted by $H = \{Start, (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), Goal\}$.

1. Path length function

The path length function f_L is used to calculate the path length of the mobile robot from the start pointing *Start* to the target pointing *Goal*, which can be expressed by the following formula:

$$CR_i = \begin{cases} if \quad fit(x_i) \geq fit_{mean}: \\ \quad CR_{\min} + (CR_{\max} - CR_{\min}) \cdot \frac{fit(x_i) - fit_{\min}}{fit_{\max} - fit_{\min}} \\ else: CR_{\min} \end{cases} \tag{16}$$

2. Penalty function

The path planning of the mobile robot must ensure the safety of the generated path. The more times a path intersects an obstacle, the higher the degree of risk. The degree of risk is used to set a penalty function to punish the path nodes that intersect with obstacles.

To facilitate the calculation, the obstacles in the environment model are regarded as circles, denoted as C_k. The center is O_k, and c is the number of obstacles. For irregular obstacles, the method of circular approximation is used to simplify them. The obstacle radius is set as the safety threshold, expressed as $R = \{r_0, r_1, \ldots, r_c\}$. In order to obtain a collision-free path, it is necessary to ensure that the distance between the path node and the obstacle is greater than the safety threshold, and the line of adjacent path nodes does not intersect the obstacles. Based on this requirement, the concept of mid-node is introduced: take m points on the connecting line of two adjacent path nodes, and this kind of path point is called mid-node. The paper uses the mid-nodes to judge whether the path intersects with obstacles, as Figure 3 shows. The Euclidean distance Dis_k (k = 1, 2, ..., c) between each node (i.e., mid-node and path node) and the obstacle center is calculated, and the calculation method of penalty degree between node i and node $i-1$ is expressed by the following formulas:

$$Dis_k = \sqrt{(x_i - O_k^x)^2 + (y_i - O_k^y)^2}$$

$$risk_k = \begin{cases} 0, & Dis_k \leq r_k \\ 1, & Dis_k > r_k \end{cases}, \quad Risk(x_i, y_i) = \sum_{k=1}^{c} risk_k$$

where $risk_k$ represents the penalty factor under the C_k, which is set to two values of 0 and 1. If distance between the node (i.e., mid-node and path node) is less than the safety threshold r_k, the node is punished by setting the $risk_k$ to 1. Otherwise, the $risk_k$ is set to 0. Then, the formula of penalty function f_P is as follows:

$$f_P = \eta \cdot \sum_{j=1}^{n+1} \sum_{i=1}^{m+2} Risk(x_i, y_i) \tag{17}$$

where n is the number of path nodes, m is the number of mid-nodes, and η is the weight coefficient, indicating the degree of influence of the penalty degree $Risk$ (x_i, y_i) on the path nodes.

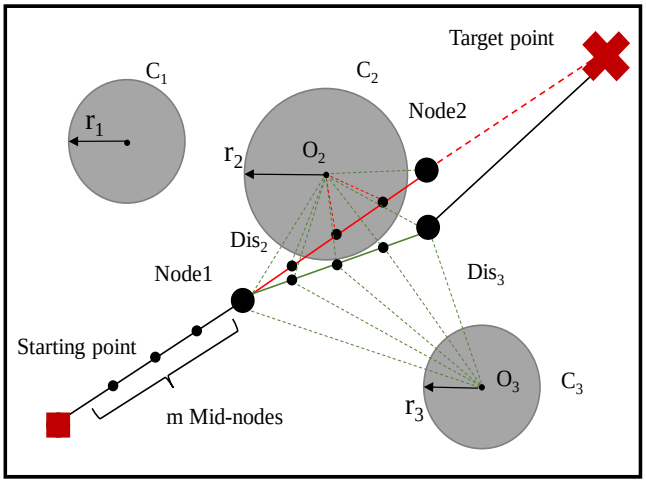


Figure 3. The concept of mid-node introduced to avoid obstacles.

In summary, the fitness function of IPSO-IDE is expressed by the following formula:

$$Fit = f_L + f_P \tag{18}$$

5.2. Path Smoothing

Combined with the analysis of the actual situation of the mobile robot operation, the path turning cannot be too sharp, and the generated path needs to be smoothed. In this paper, cubic spline interpolation [41] is added to IPSO-IDE for path planning to obtain a smooth path.

1. Basic principle of cubic spline interpolation

Cubic spline interpolation is abbreviated as Spline interpolation, which is an effective piecewise interpolation method to obtain smooth curves. A set of points is assumed to be: $\begin{cases} X = \{x_0, x_1, \ldots x_n\} \\ Y = \{y_0, y_1, \ldots y_n\} \end{cases}$, where X and Y correspond one to one.

The interval of the spline curve is set to $[x_{\min}, x_{\max}]$. The $n + 1$ data points are taken in the interval to generate n subintervals. $S(x)$ as cubic spline interpolation function needs to meet the following stipulations:

- In n subintervals $[x_i, x_{i+1}]$ $(i = 0, 1, \ldots, n-1)$, $S(x)$ is a cubic polynomial.
- $S(x_i) = y_i$ $(i = 0, 1, 2, \ldots, n-1)$.
- The first derivative and the second derivative of $S(x)$ in the interval $[x_{\min}, x_{\max}]$ are continuous.

2. Determine the equation of Spline Interpolation

Based on the stipulations that the cubic spline interpolation function must meet, the calculation method is proposed as follows:

- According to $S(x)$, which is composed of n cubic polynomials, the polynomial expression can be obtained as:

$$S_i(x) = a_i + b_i \cdot (x - x_i) + c_i \cdot (x - x_i)^2 + d_i \cdot (x - x_i)^3$$

where a_i, b_i, c_i, d_i $(i = 0, 1, 2, \ldots, n-1)$ are undetermined coefficients, so $S(x)$ has $4n$ undetermined coefficients in total.

- According to $S(x_i) = y_i (i = 0, 1, 2, \ldots, n-1)$, it can be concluded that:

$$S(x_{i+1}) = y_{i+1}, \quad (i = 0, 1, 2, \ldots, n-1),\ a_i = y_i$$

Determine step size: $h_i = x_{i+1} - x_i, i = 0, 1, \ldots, n-2$

$$a_i + b_i \cdot h_i + c_i \cdot {h_i}^2 + d_i \cdot {h_i}^3 = y_{i+1}$$

- According to the continuity of the differential, it can be concluded that:

$$S_i\prime(x_{i+1}) = S_{i+1}\prime(x_{i+1}), (i = 0, 1, 2, \ldots, n-1)$$
$$S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}), (i = 0, 1, 2, \ldots, n-1)$$

To sum up, according to the differential continuity, $2(n-1)$ conditions can be obtained. In addition, according to the free boundary condition, the second derivatives of the two endpoints of interval are 0, so $S(x)$ can be determined. Figure 4 shows the trajectory with spline through n points.

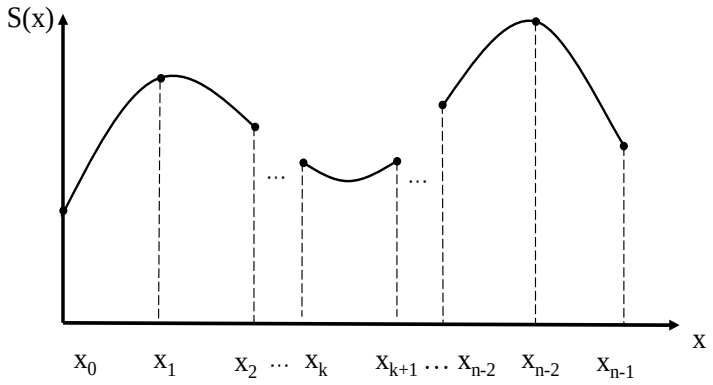


Figure 4. Trajectory with spline through n points.

3. Smoothing by spline interpolation

Since the path generated by IPSO-IDE is represented by H = {Start, (x_1, y_1), (x_2, y_2), . . . , (x_n, y_n), Goal}, i.e., there are n + 2 path nodes including the starting point and the target point. The calculation is performed by cubic spline interpolation on the intervals $(x_0, x_1, \ldots, x_{n+2})$ and $(y_0, y_1, \ldots, y_{n+2})$. The path is obtained by the connecting lines of the adjacent nodes (i.e., path nodes, interpolation points, starting point and target point).

The paper carries out a path-planning experiment based on the IPSO-IDE fused with cubic spline interpolation, which is conducted by PYTHON 3.7.5 (Beijing, China) software. The path is planned in a simple environment model, and it is smoothed using cubic spline interpolation to obtain the final optimal path that is shown in Figure 5, where the green curve is the original path before smoothing, and the red line is the final path after smoothing. It can be seen that the final path has no sharp points and is smoother, which meets the requirement of dynamics and kinematics, and fits the actual situation better. The direction and shape of the final path are roughly consistent with the original path, and almost no change.

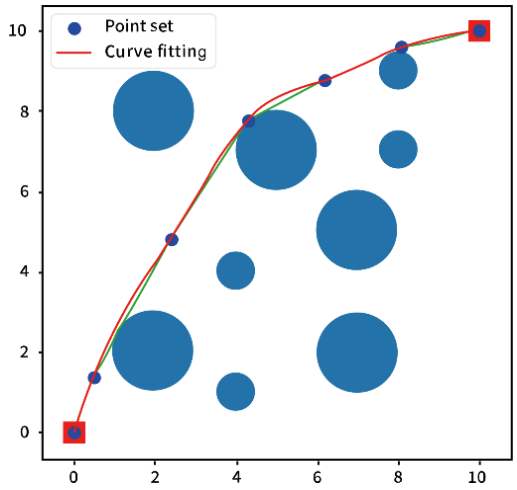


Figure 5. Trajectory with spline through n points. Comparison of generated paths before and after path smoothing.

Figure 6 is a detailed flowchart of the improved IPSO-IDE fused with cubic spline interpolation applied to path planning. The improved algorithm can efficiently find the optimal position and improve the path-planning ability.

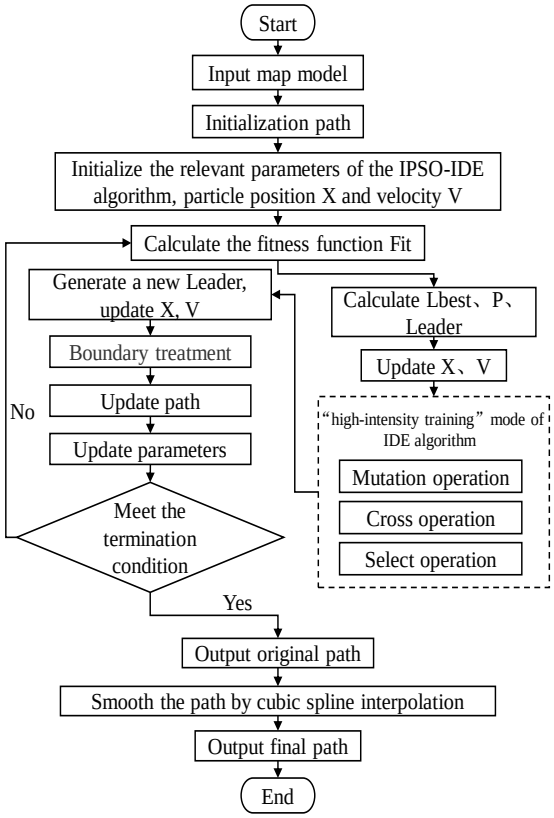


Figure 6. Flowchart of robot path planning based on the proposed algorithm.

In order to verify the practicability of the algorithm, different scenarios are set to carry out the path-planning simulation experiment of the algorithm, so that the path planning algorithm can be applied to the mobile robots in the future. The mobile robot uses the path-planning algorithm package Move_base delivered with ROS to complete the path-planning task based on its dynamic model, which mainly includes global path planning and local

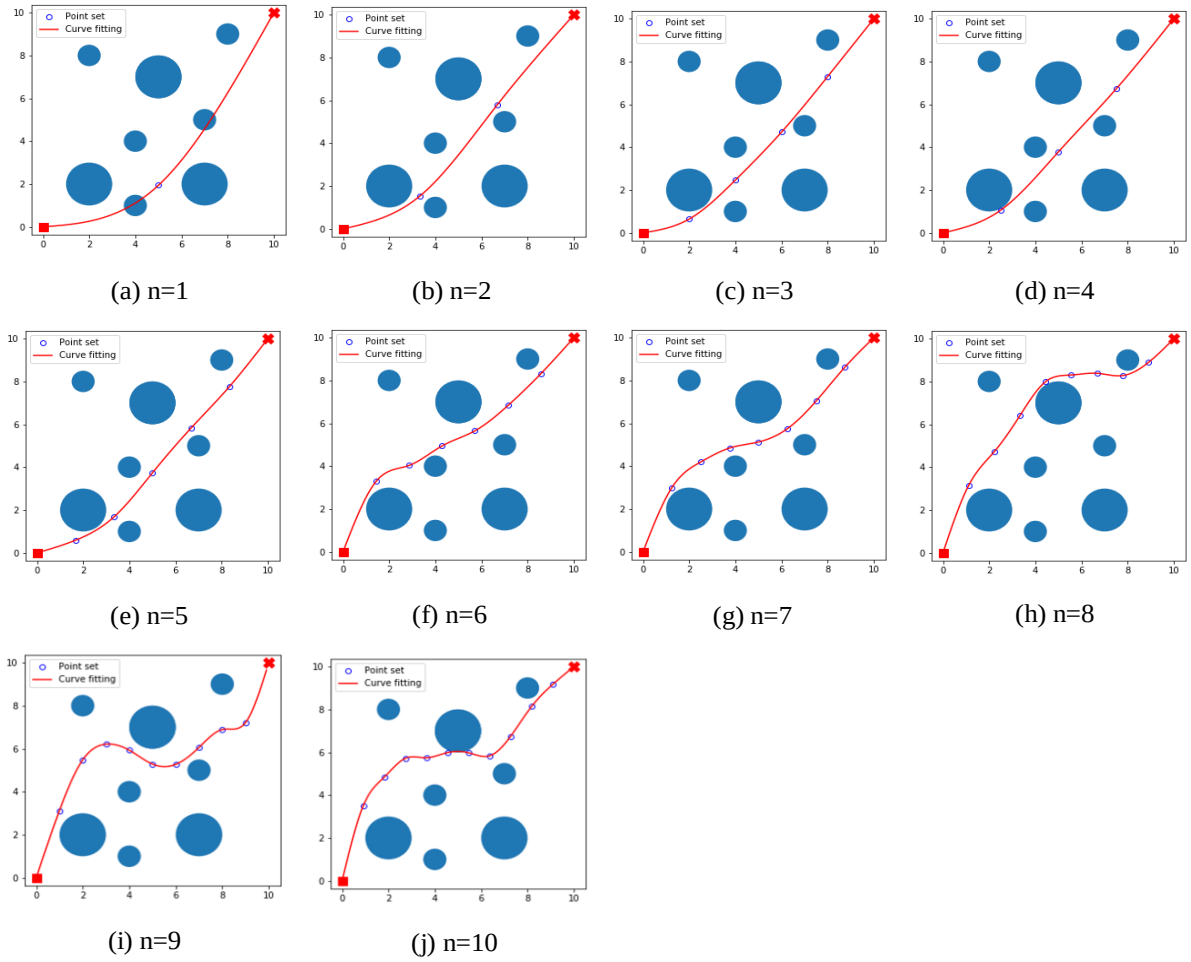
path planning and design. Global path planning constructs the shortest distance from the start to the end based on global map information.

6. Simulation Experiments and Analysis

In order to validate the algorithm proposed in this paper in solving the path planning and analyze the influence of the number of path nodes on the performance of the algorithm, the simulation experiments are conducted by PYTHON 3.7.5 (Beijing, China) software. The performances of the proposed algorithm are compared with those of PSO [17], DE [21], ABC [18], PSO-ABC [9], DPG-PSO [28], hybrid algorithm based on PSO and DE (PSO-DE) [42,43], and IDE.

6.1. The Number of Path Nodes Experiment

The purpose of this experiment is to analyze the influence of the number of path nodes on the performance of path planning. The paper selects environment 1 for experiment, where the domains of x and y were between 0 and 10, the starting point coordinate is (0,0), which is represented by a red square, the end point coordinates are (10,10), which is represented by a red "X", the path node is represented by a blue circle, and the red line represents the final path. In this experiment, the population size is set to 15, the number of path nodes is 1 to 10, the maximum number of iterations is 100, and the output result is shown in Figure 7. Figure 8 shows the convergence curve of path length based on IPSO-IDE when the number of path nodes ranged from 1 to 10.



(a) n=1 (b) n=2 (c) n=3 (d) n=4

(e) n=5 (f) n=6 (g) n=7 (h) n=8

(i) n=9 (j) n=10

Figure 7. Flowchart of robot path planning based on the proposed algorithm.

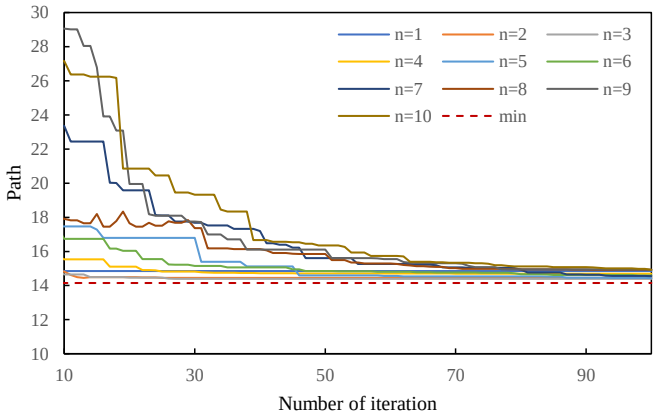


Figure 8. Convergence curves of optimal fitness values based on IPSO-IDE with n (n = 1–10) path nodes under the environment 1.

6.2. Path Planning Experiment

The path planning experiments are conducted based on IPSO-IDE in different environments, and the results are compared with other algorithms. In the experiments, the parameters of the Table 4 are used to test.

Table 4. Parameter settings of the different test algorithms (meaning of the symbols are provided in the respective literatures.

Algorithm	Parameter
IPSO-IDE	c_1 = 0.5, c_2 = 0.5, c_3 = 1.2, c_4 = 1; ω = 0.4–0.2; ϕ = 0.7; β = 1–1.5; Vmax = 0.6 × Range; F = 0.9–0.1; CR = 0.9–0.1
DE	F = 0.6; CR = 0.7; Vmax = Range
PSO	c_1 = 2, c_2= 2; ω = 1; Vmax = 0.5 × Range;Vmax = 0.1 × Range
ABC	$n_{OnLooker}$ = 10; φ = 1.2; P = 0.5; Vmax = Range
PSO-ABC	c_1 = 2, c_2 = 2; ω = 0.95–0.4; Vmax = Range
DPG-PSO	c_1 = 2, c_2 = 1.5, c_3 = 0.5, c_4 = 0.8; ω = 0.2; ϕ = 0.7; Vmax = 0.5 × Range
PSO-DE	c_1 = 2, c_2 = 2; F = 0.7; CR = 0.7; Vmax = 0.6 × Range
IDE	F = 0.9–0.1; CR = 0.9–0.1; Vmax = Range

6.2.1. First Experiment: Compared with Different Traditional Heuristic Algorithms

This experiment aims to analyze the path planning results of the proposed algorithm and compare with the traditional heuristic algorithms of PSO, DE, and ABC in environment 2. The environment 2 is a mixed map of square and circular obstacles, with a total of 10 obstacles, where the domains of x and y were between 0 and 10, and the red square and red "X" represent the start and end points. Since the obstacles are dense, the number of path nodes is selected as 5. The population size of all algorithms is 15, and the maximum number of iterations is 100. For each algorithm, each experiment is performed 20 times.

Figure 9 shows the optimal path in environment 2 generated by the different algorithms during 20 tests, and Figure 10 shows the convergence curves of optimal fitness values based on different algorithms during the 20 times. Based on four indicators of Mean, Best, Worst, Std, and Time, the algorithms are compared. Among them, Mean represents the average of the average of output results from 20 tests, Best and Worst represent the best value and the worst value in the results, respectively, Time represents the average of running time from 20 tests, and Std denotes the standard deviation. The experimental results including the above indicators are shown in Table 5. As can be seen from Figure 10, in the simple mixed scenario, DE and ABC algorithms fall into local optimal before iteration of the 20th generation, while the IPSO-IDE algorithm proposed in this paper converges only after iteration of the 40th generation. Compared with DE and ABC algorithms, IPSO-IDE algorithm has the advantage of not being precocious. As can be seen from the path planning results in Table 5, the experimental operation result of IPSO-IDE algorithm proposed in this

paper is the smallest, indicating that the convergence accuracy of the algorithm proposed in this paper is higher than that of DE, ABC, and ABC algorithms.

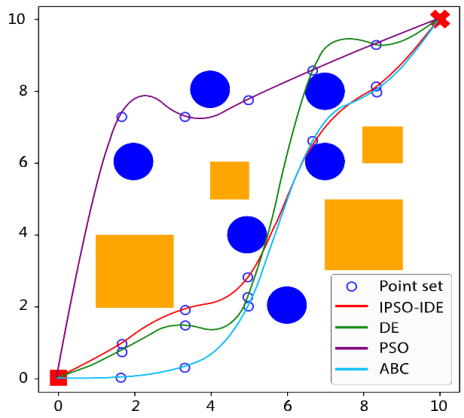


Figure 9. Optimal path results based on different algorithms in environment 2.

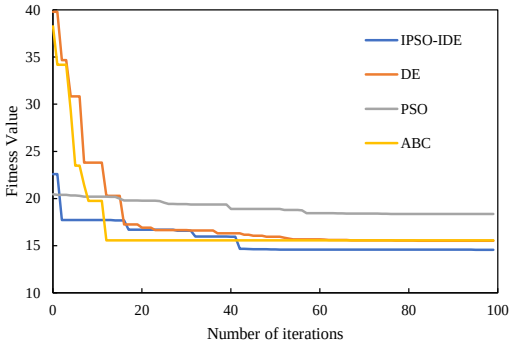


Figure 10. Convergence curves of optimal fitness values based on different algorithms in environment 2.

Table 5. Path-planning results of different algorithms in environment 2.

Algorithm	Mean	Best	Worst	Std	Time
IPSO-IDE	15.273	14.562	18.311	1.255	5.83
DE	16.461	15.514	18.317	1.359	7.52
PSO	22.735	18.919	33.489	4.833	6.53
ABC	16.865	15.557	18.356	1.330	7.90

6.2.2. Second Experiment: Compared with Different Improved Algorithms

This experiment aims to analyze the path-planning results of the proposed algorithm and compare with the improved algorithms of PSO-ABC, DPG-PSO, and ABC in environment 3. Compared with environment 2, the types of obstacles in environment 3 are increased, which is a more complex map, as shown in Figure 11. In here, the domains of x and y were between 0 and 10, and the red square and red "X" represent the start and end points, respectively. The algorithm parameters and the number of path nodes are the same as the first experiment. Figure 12 shows the convergence curves of optimal fitness values based on different algorithms during the 20 times. Based on four indicators of Mean, Best, Worst, and Std, the algorithms are compared, and the experimental results including the above indicators are shown in Table 6. As can be seen from Figures 11 and 12, in complex mixed scenarios, although IPSO-IDE algorithm proposed in this paper also falls into local optimal earlier, compared with PSO-ABC, DPGPSO, PSO-DE, and IDE algorithm, IPSO-IDE algorithm can find a better path. In addition, we can see from the path planning results in Table 6 that the proposed algorithm has higher convergence accuracy than other algorithms.

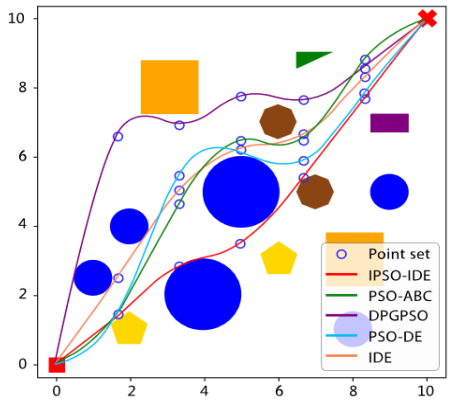


Figure 11. Optimal path results based on different algorithms in environment 3.

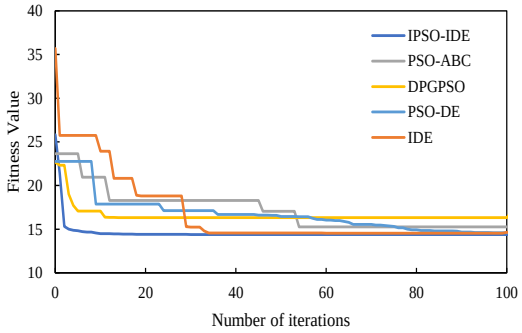


Figure 12. Convergence curves of optimal fitness values based on different algorithms in environment 3.

Table 6. Path planning results of different algorithms in environment 3.

Algorithm	Mean	Best	Worst	Std	Time
IPSO-IDE	14.656	14.374	15.487	0.376	7.08
PSO-ABC	15.896	15.257	16.364	0.463	9.13
DPG-PSO	17.636	16.325	18.689	0.971	7.52
PSO-DE	15.147	14.572	15.793	0.547	11.13
IDE	14.843	14.534	15.635	0.426	9.74

6.2.3. Third Experiment: Verification of Big Map

This experiment aims to analyze the path-planning results of the proposed algorithm and compare with the of the improved algorithms of PSO-ABC, DPG-PSO, and ABC in environment 4. Environment 4 is a map of dense obstacles, and the map range is expanded, as shown in Figure 13. In here, the domains of x and y were between 0 and 100, and the red square and red "X" represent the start and end points, respectively. The algorithm parameters and the number of path nodes are the same as the first experiment. Figure 14 shows the convergence curves of optimal fitness values based on different algorithms during the 20 times. Based on five indicators of Mean, Best, Worst, Std, and Time, where Time means the average of running time from 20 tests, the algorithms are compared, and the experimental results including the above indicators are shown in Table 7. It can be seen from Figures 13 and 14, in a scenario with dense obstacles, although the IPSO-IDE algorithm proposed in this paper also falls into local optimal earlier, compared with PSO-ABC, DPGPSO, PSO-DE, and IDE algorithms, IPSO-IDE can find a better path. In addition, we can see from the path planning results in Table 7 that the proposed algorithm has higher convergence accuracy than other algorithms.

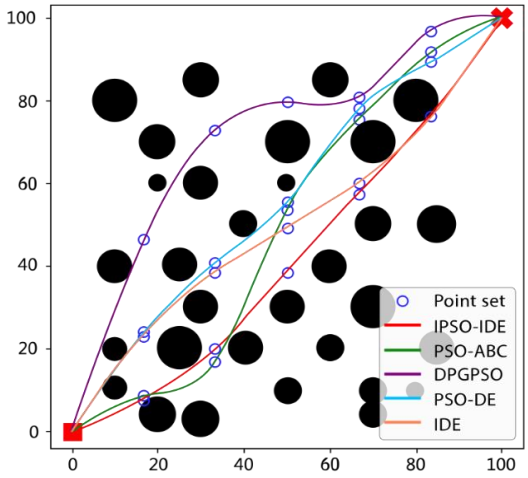


Figure 13. Optimal path results based on different algorithms in environment 4.

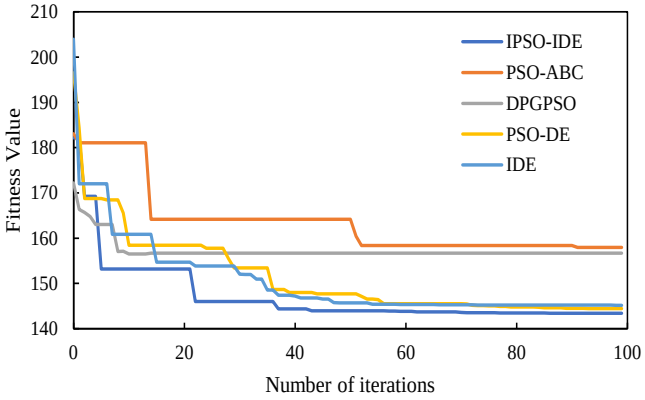


Figure 14. Convergence curves of optimal fitness values based on different algorithms in environment 4.

Table 7. Path planning results of different algorithms in environment 4.

Algorithm	Mean	Best	Worst	Std	Time
IPSO-IDE	143.982	143.362	144.527	0.738	16.51
PSO-ABC	158.504	157.924	151.457	1.120	43.02
DPG-PSO	157.771	157.158	159.090	0.819	35.21
PSO-DE	145.137	144.443	146.481	0.781	43.12
IDE	145.703	145.180	147.504	1.021	45.34

7. Conclusions

This paper studies the application of the PSO algorithm in the path planning of mobile robots and proposes an IPSO-IDE algorithm based on cubic spline interpolation. The proposed algorithm is mainly based on the traditional PSO algorithm to make the improvements. It combines improved inertia weight ω^*, adaptive parameter β, and the concept of corporate governance. Aiming at the shortcomings of the traditional DE algorithm, the scaling factor F and the cross-probability factor CR are adaptively optimized, so that the algorithm can adaptively control the search accuracy and the degree of mutation to improve the optimization accuracy of the algorithm. Then, the improved IDE algorithm is used to improve the global optimal position of the IPSO algorithm to prevent the IPSO algorithm from falling into the local optimal solution. A new objective function applied to path planning, which is composed of a path length function and a penalty function, simplifying the path planning problem of mobile robots into an objective function optimization problem.

The proposed algorithm is tested on nine classical test functions. The results show that the proposed algorithm has high optimization capabilities and search efficiency without requiring a large amount of sample data, and in addition to the f_9 test functions, the results of this algorithm have reached the optimum. Although the results of the f_9 test function have not reached the optimum, the accuracy of the worst results has also reached the 10^{-6} overall level, it shows superiority compared with comparison algorithms such as PSO, DPG-PSO, PSO-ABC, IDE, etc. Then, the algorithm is applied to various experimental environments for path-planning experiments. The experimental results show that, compared with the traditional path-planning algorithm, the proposed IPSO-IDE algorithm not only has higher convergence accuracy, but also has the advantages of not being precocious. At the same time, compared with the other two improved particle swarm optimization algorithms, although the IPSO-IDE algorithm proposed in this paper is also prone to fall into the precocious state, compared with other algorithms, the algorithm can find a better path and the final convergence accuracy is higher. The results show that the algorithm improves the global search ability and has certain practical value. However, this algorithm is only applicable to the path planning problem of mobile robots with complex static maps. In the subsequent research, this paper plans to use the proposed algorithm to solve the path planning problem under the dynamic scene model, increase the real-time scene acquisition and processing functions, and improve the effect of path planning.

Author Contributions: Q.Y. carried out the data curation and writing—original draft. R.S. carried out the conceptualization and methodology. X.D. carried out the validation and visualization. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Project No.52065010 and No.52165063), Department of Science and Technology of Guizhou Province (Project No. [2022] G140 and No. [2022]K024), Graduate Innovative Talents Program of Guizhou University (2021), Research on Industrial Robot Technology based on Patent Analysis (K19-0204-001).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Thanks for the computing support of the State Key Laboratory of Public Big Data, Guizhou University.

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

Abbreviations

List of Abbreviations.

Abbreviation	meaning
PSO	particle swarm optimization
IPSO	improved particle swarm optimization
DE	differential evolution
IDE	optimized differential evolution
IPSO- IDE	improved particle swarm optimization based on differential evolution
GA	genetic algorithm
CR	crossover probability factor
DPG-PSO	democracy-inspired particle swarm optimizer with the concept of peer groups
ABC	artificial bee colony
PSO-ABC	hybrid algorithm based on PSO and ABC

The pseudo code of manager selects.

Algorithm 1: **Code for manager selects**

//Initialize operator, Owner, operator vote, manager vote, administrator particle.
Initialize : $Opvote := Œ$, $ovote := 1 - Œ$, Operator, $Owner, Adm, \mathrm{Adm}_i^{\mathrm{d}}, vote_i^d$: [0,1].
1. **While** ($fit > fit_{\min}$)
2. **For** t = 1 to T //T is the number of iterations.
3. **For** i = 1 to N //N is the number of particles in the population.
4. **For** d = 1 to D //D for dimension.
5. **If (**$vote_i^d$ < Opvote)
6. $\mathrm{Adm}_i^{\mathrm{d}} \leftarrow Operator^d$ //If the particles vote for the operator.
7. Opvote ← Opvote + 1/(M.N) //Add up the votes.
8. Opvote ← Opvote/(Opvote+Owvote) //Standardize voting.
9. **Else**
10. $\mathrm{Adm}_i^{\mathrm{d}} \leftarrow Owner^d$ //If the particle votes for the owner.
11. Owvote ← Owvote + 1/(M.N) //Add up the votes.
12. Owvote←Owvote/(Opvote+Owvote) //Standardize voting.
13. **End if**
14. //The $e^{(1-vote_{Adm})}$ expressed as follows.
15. If $\left(\mathrm{Adm}_i^{\mathrm{d}} = Operator^d\right)$
16. $e^{(1-vote_{Adm})} \leftarrow 1$
17. **End if**
18. If $\left(\mathrm{Adm}_i^{\mathrm{d}} = Owner^d\right)$
19. $e^{(1-vote_{Adm})} \leftarrow 2.3$
20. **End if**
21. Update $Operator^d$ and V_t
22. **End for**
23. **End for**
24. **End while**

The pseudo code of IPSO-IDE.

Algorithm2: Code for IPSO-IDE

Initialize: pro = 1/N, $\phi \leftarrow 0.7$, $Opvote$: = ϕ, $ovote$: = 1− ϕ,
$c_1 \leftarrow 0.5$, $c_2 \leftarrow 0.5$, $c_3 \leftarrow 1.2$, $c_4 \leftarrow 1$, X, V, Operator, $Owner$, $Leader$, $Lbest$, P.
1. **While** ($fit > fit_{\min}$)
2. **For** t = 1 to T //T is the number of iterations.
3. **For** i = 1 to N //N is the number of particles in the population.
4. **For** d = 1 to D //D for dimension.
//The following is the calculation of the optimal position of an individual based on the fitness value fit.
5. **If** ($fit\ (X_i(t)) \leq fit\ (P_i(t))$)
6. $P_i(t) \leftarrow X_i(t)$
7. **Else**
8. $P_i(t) \leftarrow X_i(t)$
9. **End if**
//The following is the calculation of the local optimal position based on the fitness value fit.
10. **If** ($fit(X_i(t)) \leq fit(Lbest_i(t))$)
11. $Lbest_i(t) \leftarrow X_i(t)$
12. **Else**
13. $Lbest_i(t) \leftarrow Lbest_i(t-1)$
14. **End if**
//The following is the calculation of the global optimal position based on the fitness value fit.

If ($fit(X_i(t)) \leq fit(Lbest_{i\pm1}(t))$ or $(fit(X_i(t)) \leq fit(Lbest_{i\pm2}(t)))$
  $Lbest_{i\pm1}\ (t) \leftarrow Lbest_i(t)$
//Elect an *Adm* according to the expression (3-15)
  Leader, *Leader_fit*, *Opvote*, *Owvote* ← *Choose_Leader* (*Operator*, *Owner*)
  //Update position X and speed V to generate a better elite group, and take the elite group as the initial group of IDE algorithm.
  DE_list ← *X*
  $h(t) \leftarrow Mutation\ (DE\_list(t), fit(DE\_list(t)))$ //variation.
  $v(t) \leftarrow Crossover\ (DE\_list(t), h(t), fit(DE\_list(t)))$ //cross.
//*Selection*.
  $DE\_list(t) \leftarrow Selection\ (v(t), DE\_list(t))$ (Mutation, Crossover, and Selection: respectively refer to the mutation, crossover, and selection operations in the DE algorithm)
//Apply the optimized result of IDE algorithm to Leader of the updated particle swarm.
  DE_fitness ← *fit* (*DE_list*(*t*))
  min_*f* ← Minimum *DE_fitness*
  min_*position* ← Minimum *DE_fitness* position
  If (min_*f* ← *Leader_fit*)
  Leader ← min_*position*
  End if
  Normalize *Opvote* and *Owvote*
  If ($Leader^d(t) = Opvote_i^d(t)$)
  $e^{(1-vote_{leader})} \leftarrow 1$
  Else
  $e^{(1-vote_{leader})} \leftarrow 2.3$
  End if
  Update X_t and V_t //Update the position and velocity of the particle.
 End for
End for
End while

References

1. Chipade, V.S.; Panagou, D. Multiagent Planning and Control for Swarm Herding in 2-D Obstacle Environments Under Bounded Inputs. *IEEE Trans. Robot.* **2021**, *37*, 1956–1972. [CrossRef]
2. Ren, Z.; Rathinam, S.; Likhachev, M.; Choset, H. Multi-Objective Safe-Interval Path Planning With Dynamic Obstacles. *IEEE Robot. Autom. Lett.* **2022**, *7*, 8154–8161. [CrossRef]
3. Pei, M.; An, H.; Liu, B.; Wang, C. An Improved Dyna-*Q* Algorithm for Mobile Robot Path Planning in Unknown Dynamic Environment. *IEEE Trans. Syst. Man, Cybern. Syst.* **2022**, *52*, 4415–4425. [CrossRef]
4. Nguyen, V.-L.; Hwang, R.-H.; Lin, P.-C. Controllable Path Planning and Traffic Scheduling for Emergency Services in the Internet of Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 12399–12413. [CrossRef]
5. Favaro, A.; Segato, A.; Muretti, F.; De Momi, E. An Evolutionary-Optimized Surgical Path Planner for a Programmable Bevel-Tip Needle. *IEEE Trans. Robot.* **2021**, *37*, 1039–1050. [CrossRef]
6. Vagale, A.; Oucheikh, R.; Bye, R.T.; Osen, O.L.; Fossen, T.I. Path planning and collision avoidance for autonomous surface vehicles I: A review. *J. Mar. Sci. Technol.* **2021**, *26*, 1292–1306. [CrossRef]
7. Chen, P.; Li, Q.; Zhang, C.; Cui, J.; Zhou, H. Hybrid chaos-based particle swarm optimization-ant colony optimization algorithm with asynchro-nous pheromone updating strategy for path planning of landfill inspection robots. *Int. J. Adv. Robot. Syst.* **2019**, *16*, 255795084. [CrossRef]
8. Li, G.; Chou, W. Path planning for mobile robot using self-adaptive learning particle swarm optimization. *Sci. China Inf. Sci.* **2018**, *61*, 052204. [CrossRef]
9. Gul, F.; Rahiman, W.; Alhady, S.S.N.; Ali, A.; Mir, I.; Jalil, A. Meta-heuristic approach for solving multi-objective path planning for autonomous guided robot using PSO–GWO optimization algorithm with evolutionary programming. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 7873–7890. [CrossRef]
10. Xie, S.; Hu, J.; Bhowmick, P.; Ding, Z.; Arvin, F. Distributed Motion Planning for Safe Autonomous Vehicle Overtaking via Artificial Poten-tial Field. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 21531–21547. [CrossRef]
11. Gul, F.; Mir, I.; Abualigah, L.; Sumari, P.; Forestiero, A. A Consolidated Review of Path Planning and Optimization Techniques: Technical Per-spectives and Future Directions. *Electronics* **2021**, *10*, 2250. [CrossRef]
12. Jian, Z.; Zhang, S.; Chen, S.; Nan, Z.; Zheng, N. A Global-Local Coupling Two-Stage Path Planning Method for Mobile Robots. *IEEE Robot. Autom. Lett.* **2021**, *6*, 5349–5356. [CrossRef]

13. Qi, Z.; Wang, T.; Chen, J.; Narang, D.; Wang, Y.; Yang, H. Learning-based Path Planning and Predictive Control for Autonomous Vehicles With Low-Cost Positioning. *IEEE Trans. Intell. Veh.* **2021**, *early access*. [CrossRef]
14. Zhang, Z.; Wu, R.; Pan, Y.; Wang, Y.; Wang, Y.; Guan, X.; Hao, J.; Zhang, J.; Li, G. A Robust Reference Path Selection Method for Path Planning Algorithm. *IEEE Robot. Autom. Lett.* **2022**, *7*, 4837–4844. [CrossRef]
15. Wen, J.; Yang, J.; Wang, T. Path Planning for Autonomous Underwater Vehicles Under the Influence of Ocean Currents Based on a Fusion Heuristic Algorithm. *IEEE Trans. Veh. Technol.* **2021**, *70*, 8529–8544. [CrossRef]
16. Awad, A.; Hawash, A.; Abdalhaq, B. A Genetic Algorithm (GA) and Swarm Based Binary Decision Diagram (BDD) Reordering Optimizer Reinforced with Recent Operators. *IEEE Trans. Evol. Comput.* **2021**, *early access*. [CrossRef]
17. 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.
18. Karaboga, D.; Gorkemli, B.; Ozturk, C.; Karaboga, N. A comprehensive survey: Artificial bee colony (ABC) algorithm and applications. *Artif. Intell. Rev.* **2014**, *42*, 21–57. [CrossRef]
19. Liu, E.; Yao, X.; Liu, M.; Jin, H. AGV path planning based on improved grey wolf optimization algorithm and its implementation prototype platform. *Comput. Integr. Manuf. Syst.* **2018**, *24*, 2779–2791.
20. Tang, J.; Liu, G.; Pan, Q. A Review on Representative Swarm Intelligence Algorithms for Solving Optimization Problems: Applications and Trends. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 1627–1643. [CrossRef]
21. 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]
22. Zhang, H.-Y.; Lin, W.-M.; Chen, A.-X. Path Planning for the Mobile Robot: A Review. *Symmetry* **2018**, *10*, 450. [CrossRef]
23. Kamel, M.A.; Yu, X.; Zhang, Y. Real-Time Fault-Tolerant Formation Control of Multiple WMRs Based on Hybrid GA–PSO Algorithm. *IEEE Trans. Autom. Sci. Eng.* **2021**, *18*, 1263–1276. [CrossRef]
24. Memon, M.A.; Siddique, M.D.; Mekhilef, S.; Mubin, M. Asynchronous Particle Swarm Optimization-Genetic Algorithm (APSO-GA) Based Selective Harmonic Elimination in a Cascaded H-Bridge Multilevel Inverter. *IEEE Trans. Ind. Electron.* **2022**, *69*, 1477–1487. [CrossRef]
25. Mohammed Hussein, H.; Katzis, K.; Mfupe, L.P.; Bekele, E.T. Performance Optimization of High-Altitude Platform Wireless Communication Network Exploiting TVWS Spectrums Based on Modified PSO. *IEEE Open J. Veh. Technol.* **2022**, *3*, 356–366. [CrossRef]
26. Fan, Q.; Zhang, Y.; Li, N. An Autoselection Strategy of Multiobjective Evolutionary Algorithms Based on Performance Indicator and its Ap-plication. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 2422–2436. [CrossRef]
27. Patle, B.K.; Ganesh Babu, L.; Anish, P.; Parhi, D.R.K. A review: On path planning strategies for navigation of mobile robot. *Def. Technol.* **2019**, *4*, 582–606. [CrossRef]
28. Burman, R.; Chakrabarti, S.; Das, S. Democracy-inspired particle swarm optimizer with the concept of peer groups. *Soft Comput.* **2017**, *21*, 3267–3286. [CrossRef]
29. Zhao, C.; Guo, D. Particle Swarm Optimization Algorithm With Self-Organizing Mapping for Nash Equilibrium Strategy in Application of Multiobjective Optimization. *IEEE Trans. Neural Networks Learn. Syst.* **2021**, *32*, 5179–5193. [CrossRef]
30. Yu, Z.; Si, Z.; Li, X.; Wang, D.; Song, H. A Novel Hybrid Particle Swarm Optimization Algorithm for Path Planning of UAVs. *IEEE Internet Things J.* **2022**, *9*, 22547–22558. [CrossRef]
31. Pozna, C.; Precup, R.; Horvath, E.; Petriu, E.M. Hybrid Particle Filter-Particle Swarm Optimization Algorithm and Application to Fuzzy Con-trolled Servo Systems. *IEEE Trans. Fuzzy Syst.* **2022**, *30*, 4286–4297. [CrossRef]
32. Liu, X.; Zhang, D.; Zhang, T.; Zhang, J.; Wang, J. A new path plan method based on hybrid algorithm of reinforcement learning and particle swarm optimization. *Eng. Comput.* **2021**, *ahead of print*. [CrossRef]
33. Zhou, S.; Xing, L.; Zheng, X.; Du, N.; Wang, L.; Zhang, Q. A Self-Adaptive Differential Evolution Algorithm for Scheduling a Single Batch-Processing Machine With Arbitrary Job Sizes and Release Times. *IEEE Trans. Cybern.* **2021**, *51*, 1430–1442. [CrossRef] [PubMed]
34. Chai, R.; Savvaris, A.; Tsourdos, A.; Chai, S. Multi-objective trajectory optimization of Space Manoeuvre Vehicle using adaptive differential evolution and modified game theory. *Acta Astronaut.* **2017**, *136*, 273–280. [CrossRef]
35. Lin, C. An adaptive-group-based differential evolution algorithm for inspecting machined workpiece path planning. *Int. J. Adv. Manuf. Technol.* **2019**, *105*, 2647–2657. [CrossRef]
36. Wang, Z.-J.; Zhou, Y.-R.; Zhang, J. Adaptive Estimation Distribution Distributed Differential Evolution for Multimodal Optimization Problems. *IEEE Trans. Cybern.* **2022**, *52*, 6059–6070. [CrossRef]
37. Liu, H.; Chen, Q.; Pan, N.; Sun, Y.; An, Y.; Pan, D. UAV Stocktaking Task-Planning for Industrial Warehouses Based on the Improved Hybrid Differential Evolution Algorithm. *IEEE Trans. Ind. Informatics* **2022**, *18*, 582–591. [CrossRef]
38. Xu, M.; Wang, Y. Time Series Prediction Based on Improved Differential Evolution and Echo State Network. *Acta Autom. Sin.* **2019**, *45*, 1–9.
39. Zhang, B.; Lei, T. The Relationship between Corporate Governance and Corporate Performance in China's Civilian-Owned Listed En-terprise. In Proceedings of the 2009 International Conference on Business Intelligence and Financial Engineering, Beijing, China, 24–26 July 2009; pp. 782–785.

40. Kashyap, S.; Jeyasekar, A. A Competent and Accurate BlockChain based E-Voting System on Liquid Democracy. In Proceedings of the 2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), Paris, France, 28–30 September 2020; pp. 202–203.
41. Sadikin, R.; Swardiana, I.W.A.; Wirahman, T. Cubic spline interpolation for large regular 3D grid in cylindrical coordinate: (Invited pa-per). In Proceedings of the 2017 International Conference on Computer, Control, Informatics and its Applications (IC3INA), Jakarta, Indonesia, 23–26 October 2017; pp. 1–6.
42. Bogdanov, V.V.; Volkov, Y.S. Near-optimal tension parameters in convexity preserving interpolation by generalized cubic splines. *Numer. Algorithms* **2021**, *86*, 833–861. [CrossRef]
43. 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]