An Improved Hybrid Particle Swarm Optimization Path Planning Algorithm Based on Particle Reactivation

Yuan Luo, Xianfeng Zhang, Jinke Wu

***Abstract*—The particle swarm optimization (PSO) path planning algorithm often suffers from low population diversity and rapid early convergence, which can lead to the algorithm easily falling into local optima and affecting its stability. This paper proposes an improved PSO algorithm with two key enhancements: (1) the introduction of a particle reactivation module during the iteration process, and (2) the incorporation of the Simulated Annealing (SA) concept during the global optimal solution update phase. These improvements enhance the diversity of the particle population, slow down the early convergence rate of the algorithm, reduce the probability of the algorithm getting trapped in local optima, and increase its overall stability. Experiments were conducted to compare the proposed algorithm with the standard PSO algorithm, genetic algorithms, and other improved PSO path planning algorithms. The results indicate that the improved algorithm shows superior performance in both average path length and algorithm stability.**

***Index Terms*—path planning, particle swarm optimization, particle reactivation, simulated annealing**

I. Introduction

INTELLIGENT mobile robots hold tremendous potential for development and find extensive applications in various fields, including industrial automation, logistics, military operations, and the Internet of Things (IoT). Path planning involves determining an optimal route for a robot to reach its destination while avoiding obstacles in either static or dynamic environments, based on specific performance metrics[1]. An excellent path planning algorithm not only saves significant response time for robots but also reduces wear and tear, thereby lowering operational costs. Consequently, path planning is a critical component of robotics technology, possessing significant practical value, and has emerged as a research hotspot in recent years.

Generally, path planning is divided into local path planning and global path planning based on the information provided to the robot. Global path planning, typically conducted in static environments, provides the robot with comprehensive environmental information and generates a complete path from the starting point to the destination. In global path planning, the robot is aware of the optimal path it should follow[2]. Conversely, local path planning involves navigating within a short-term, small-scale environment to avoid obstacles and ultimately reach the destination guided by a reference line. Local path planning encounters unknown environmental factors, such as the size and position of dynamic obstacles[3].

For intelligent mobile robot path planning algorithms, they are typically categorized into graph-based path planning, potential field-based path planning, and intelligent bio-inspired algorithm-based path planning [4].

Graph-based algorithms for path planning involve deterministic search methods that traverse graphs. They are commonly employed in low-dimensional spaces, such as two-dimensional spaces. The performance of such algorithms relies on the chosen resolution of the map. If the map resolution is too low, suitable paths may not be generated, while excessively high resolution may lead to excessive time consumption. Moreover, as the scale of the map increases, the search performance tends to deteriorate. Notable examples of graph-based algorithms include Dijkstra's algorithm [5], the A* algorithm, the D* algorithm [6], and various improved variants of the A* algorithm. Graph-based path planning algorithms are characterized by their simplicity and are often used for static path planning, computing initial paths, or optimizing other algorithms using heuristic approaches.

Potential field-based path planning involves finding appropriate paths by tracking the steepest descent of a potential field composed of both attractive and repulsive components. The attractive force originates from the goal point, while the repulsive force arises from obstacles. The resultant force direction indicates the desired direction of movement, and its magnitude represents the desired speed. The artificial potential field method, first proposed by Khatib in 1986 [7], is one of the potential field-based path planning algorithms. Its advantages lie in its simple structure, ease of computation, and generation of smooth and safe paths.

Manuscript received March 27, 2024; revised August 19, 2024.

This work was supported in part by the the Youth Fund Program of the NationalNatural Science Foundation of China(Grant No.61703067), the ChongqingBasic Science and Frontier Technology Research Program (Grant No.Cstc2017jcyjAX0212), and the Science and Technology Research Programof Chongqing Municipal Education Commission (KJ1704072).

Yuan Luo is a Professor at the Key Laboratory of Optical InformationSensing and Technology, School of Optoelectronic Engineering,ChongqingUniversity of Posts and Telecommunications,Chongqing 400065 China(e-mail: 2217793866@qq.com)

Xianfeng Zhang is a graduate student of the School of OptoelectronicEngineering , Chongqing University of Posts and Telecommunications,Chongqing 400065 China (corresponding author phone: 139-831-73933; e-mail: 2294502789@qq.com)

Jinke Wu is a graduate student of the School of OptoelectronicEngineering , Chongqing University of Posts and Telecommunications,Chongqing 400065 China (corresponding author phone: 180-806-00289; e-mail: 393932884@qq.com)

However, it suffers from issues such as acceleration oscillations and high energy consumption when multiple obstacles are present in the vicinity.

Intelligent bio-inspired algorithms represent a class of stochastic search methods that mimic natural biological evolution or collective social behaviors [8]. Examples of such algorithms include genetic algorithms (GA), ant colony optimization (ACO), particle swarm optimization (PSO) [9], simulated annealing, firefly algorithms, and grey wolf optimization algorithms. These bio-inspired algorithms, characterized by their strong adaptability to the environment, are commonly employed for global planning tasks. However, they are susceptible to issues such as high iteration counts, slow convergence speeds, and low path quality.

The Particle Swarm Optimization (PSO) algorithm was introduced by Eberhart et al. in 1995 and represents a type of swarm optimization algorithm. It is inspired by the foraging behavior of bird flocks and other animal populations, where each particle possesses only position and velocity attributes. The fundamental concept of PSO revolves around leveraging information sharing within the swarm to cooperatively search for the optimal solution to a problem. Since its inception, the PSO algorithm has been applied to numerous optimization problems, with particularly widespread utilization in the field of mobile robot path planning [10].

Path planning based on the PSO algorithm offers advantages such as simplicity, ease of implementation, and wide applicability. However, it suffers from insufficient population diversity during the iterative process, leading to rapid convergence to local optima and thus instability in the algorithm [11]. To address these issues, subsequent researchers have primarily focused on improving the parameters of the PSO algorithm (such as inertia weight coefficients and learning factors), particle population initialization methods, and integration with concepts from other algorithms. These improvements have led to some degree of enhancement in the aforementioned problems, yet each approach also has its limitations.

This paper primarily focuses on three aspects: (1) The introduction of an adaptive particle reactivation module. This module performs "reactivation" operations on the population particles based on their state, maintaining population diversity and mitigating the issue of the PSO algorithm's rapid early convergence, which often leads to local optima; (2) The improvement of the global best solution update strategy by incorporating the concept of Simulated Annealing (SA), further enhancing population diversity; (3) The design of simulation experiments to compare the proposed method with the standard PSO algorithm, the Genetic Algorithm (GA) path planning method, and other improved PSO path planning algorithms from the past two years that operate under similar experimental conditions and integrate other algorithmic concepts.

The remaining structure of this paper is as follows: Section II provides an overview of standard PSO path planning algorithms and their evolutionary development. In Section III, the environmental modeling method employed in this study is introduced, followed by descriptions of the adaptive particle reactivation module and the updated strategy for the global best solution, which incorporates the simulated annealing concept. Section IV outlines the simulation experiment environment and presents the comparative experimental results. Finally, Section V summarizes the findings of this paper and outlines future research directions.

II. Particle swarm optimization path planning

A. Standard PSO Algorithm for Path Planning

The PSO algorithm, proposed by Kennedy in 1995, is a bio-inspired intelligent algorithm that mimics the foraging behavior of bird flocks. It conceptualizes the search space for problem-solving as analogous to the spatial domain of bird flight. Each bird in the flock is abstracted as a particle with volume and mass, representing a potential solution. The process of birds in a population searching for food is analogous to solving an optimization problem. Similar to other evolutionary algorithms, particle swarm optimization (PSO) is based on the concepts of population and evolution. Through collaboration and competition among individuals, it seeks to find the optimal solution in complex spaces. In the context of path planning problems based on the PSO algorithm, each potential solution is considered a particle. Through continuous evolution and iteration, the optimal solution, or the optimal path, is obtained.

In PSO-based path planning algorithms, several critical parameters must be considered[12-14]:

1) Dimension (D) of the search space for the target solution: This parameter denotes the number of control points, indicating that the target path comprises multiple points.

2) Population size (N): This signifies the quantity of particles within the solution space, representing the number of paths that are initialized, generated, and continuously evolving within the solution space.

3) Position of the i-th particle (X_i, Y_i): This indicates the current coordinates of the i-th particle. As this study focuses on paths within a two-dimensional plane, the particle positions consist of two axes: X and Y.

$$\begin{cases} X_i = \left(x_{i1}, x_{i2}, \cdots, x_{iD}\right) \\ Y_i = \left(y_{i1}, y_{i2}, \cdots, y_{iD}\right) \end{cases} i = 1, 2, \cdots N_{\circ} \quad (1)$$

4) The velocity of the i-th particle (VX_i, VY_i) represents the current velocity of the i-th particle, with both X and Y components. Typically, there is a limit imposed on the particle's velocity, and if it exceeds the maximum limit, it is replaced by the maximum velocity.

$$\begin{cases} VX_i = \left(vx_{i1}, vx_{i2}, \cdots, vx_{iD}\right) \\ VY_i = \left(vy_{i1}, vy_{i2}, \cdots, vy_{iD}\right) \end{cases} i = 1, 2, \cdots N_{\circ} \quad (2)$$

5) The individual historical best solution of the i-th particle (PXi, PYi) represents the best position encountered by the i-th particle during the iteration process.

$$\begin{cases} PX_i = \left(px_{i1}, px_{i2}, \cdots, px_{iD}\right) \\ PY_i = \left(py_{i1}, py_{i2}, \cdots, py_{iD}\right) \end{cases} i = 1, 2, \cdots N_{\circ} \quad (3)$$

6) The global historical best solution (gbestX, gbestY) denotes the optimal position experienced by the entire population during the iteration process.

$$\begin{cases} gbestX = \left(gbestx_1, gbestx_2, \cdots, gbestx_D\right) \\ gbestY = \left(gbesty_1, gbesty_2, \cdots, gbesty_D\right) \end{cases} \quad (4)$$

The standard PSO path planning algorithm is shown in Figure 1 [15].

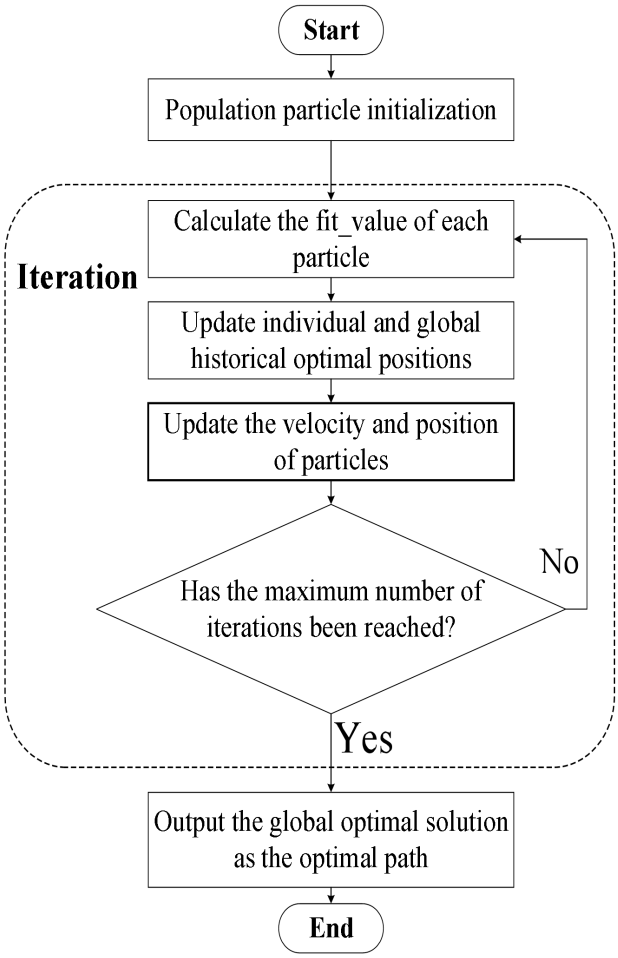


Fig. 1. Standard Particle Swarm Optimization Path Planning Algorithm Process

The PSO path planning algorithm commences by initializing the population, generating N valid paths randomly at the outset to ensure they avoid collisions with obstacles. Following initialization, the algorithm proceeds to the iteration phase. Each iteration computes the fitness value of every particle using a fitness function, typically associated with the path's length. Subsequently, it updates the historical optimal positions of individuals and groups based on these fitness values. Next, the algorithm updates the velocity and position of particles according to equations (5), (6), (7), and (8).

$$\begin{cases} VX_i^{t+1} = \omega VX_i^t + c_1 r_1\left(pbestX_i^t - X_i^t\right) + c_2 r_2\left(gbestX^t - X_i^t\right) \\ VY_i^{t+1} = \omega VY_i^t + c_1 r_1\left(pbestY_i^t - Y_i^t\right) + c_2 r_2\left(gbestY^t - Y_i^t\right) \end{cases} \quad (5)$$

$$\begin{cases} X_i^{t+1} = X_i^t + VX_i^t \\ Y_i^{t+1} = Y_i^t + VY_i^t \end{cases} \quad (6)$$

In the formula, ω is the inertia weight coefficient, which ensures the global convergence performance of the algorithm. The parameters c1 and c2 are learning factors, while pbestX and pbestY are the X and Y components of the individual's historical optimal position, respectively. gbestX and gbestY represent the X and Y components of the global historical optimal position of the entire particle population. The first term of the formula represents the ability to maintain the original velocity. The second term indicates the tendency to approach the individual's optimal position, and the third term reflects the tendency to approach the global optimal position. r1 and r2 are random numbers in the range of 0 to 1, providing a degree of randomness for velocity updates.[16]. After meeting the iteration count condition, the loop will exit, output GX and GY, and form the optimal path.

B. The Evolution of PSO Algorithm

In the PSO algorithm, the inertia weight coefficient (ω) and the two learning factors (c1 and c2) are crucial parameters. In the traditional PSO algorithm, these three parameters are typically fixed and remain constant throughout the iterations. However, subsequent research has shown that adapting the inertia weight coefficient and learning factors based on the number of iterations can lead to improved performance. As a result, various strategies for adjusting the inertia weight coefficient have been developed, including linear, nonlinear, fuzzy rule-based, and adaptive approaches. Additionally, strategies such as nonlinear adjustment and compression of learning factors have been proposed for optimizing the learning process. Moreover, recent research has focused on improving population initialization techniques and integrating them with other heuristic algorithms, which has emerged as a prominent research direction[17].

1) Improvement of Inertia Weight Coefficient

For the inertia weight coefficient, in equations (5) for velocity update, the part where it is located is called the "inertia" part, representing the tendency of the particle to maintain its previous velocity[18]. The larger the value, the stronger the global convergence ability and the weaker the local convergence ability. Conversely, the weaker the global convergence ability and the stronger the local convergence ability. In the process of searching for the best path, the requirements for global convergence ability and local convergence ability vary at different stages. Generally speaking, at the beginning of the algorithm, a larger value should be given to the inertia weight coefficient, so that particles can approach better regions faster on a global scale. And in the later stages of the search, we need smaller ones ω Value is used to ensure that particles can perform a more precise search near the optimal solution, making the algorithm more likely to converge towards the global optimal position. There are several improvements and evolutions for the inertia weight coefficient as follows:

Linear decreasing inertia weight coefficient:The linearly decreasing inertia weight coefficient is shown in equation (7).

$$\omega_{next} = \omega_{\max} - \left(\omega_{\max} - \omega_{\min}\right) \cdot \frac{t}{T_{\max}} \quad (7)$$

Compared with the previously fixed inertia weight coefficients, the current weight coefficients will decrease linearly with increasing iteration times. among ω Max is generally taken as 0.9, ω Take 0.4 for min. The advantage of linear descent method is that the algorithm is simple and easy to implement. But it's not precise enough, and the effect is not good enough.

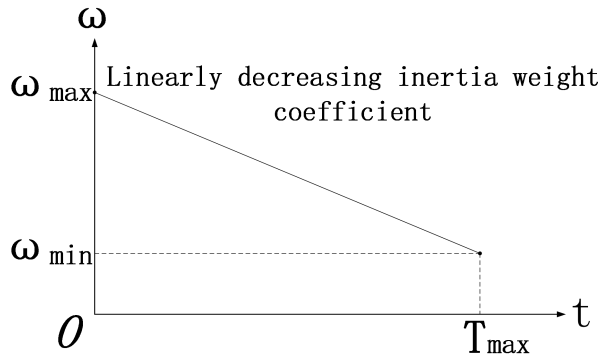


Fig. 2. Linear decreasing inertial weight system

Nonlinear decreasing inertia weight coefficients:There are mainly two forms of non-linear decreasing inertia weight coefficients, as shown in equations (6) and (7).

$$\omega_{next} = \omega_{\max} - \left(\omega_{\max} - \omega_{\min}\right) \cdot \left(\frac{t}{T_{\max}}\right)^2 \tag{8}$$

$$\omega_{next} = \omega_{\max} - \left(\omega_{\max} - \omega_{\min}\right) \times \left(\frac{2t}{T_{\max}} - \left(\frac{t}{T_{\max}}\right)^2\right) \tag{9}$$

The inertia weight coefficient will decrease nonlinearly with increasing iteration times. Among them ω Max and ω Min is also generally taken as 0.9 and 0.4. This inertia weight coefficient adjustment method is currently the most commonly used form, and selecting a more suitable formula based on different situations can achieve better results. However, this method has not effectively improved the population diversity of the PSO algorithm, and it still frequently leads to the problem of getting stuck in local optima too early, resulting in the algorithm being not stable enough.

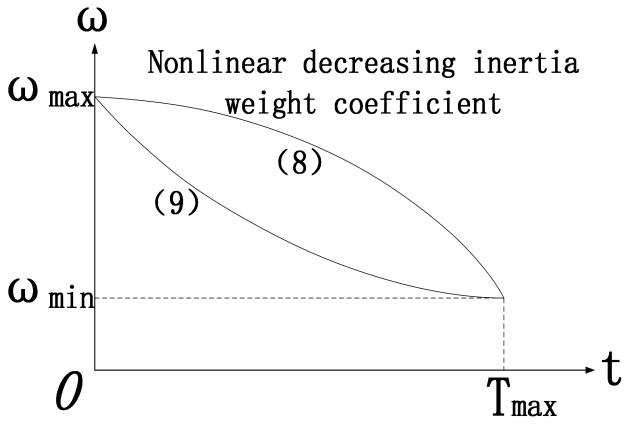


Fig. 3. Nonlinear decreasing inertia weight coefficient

Random inertia weight coefficient[19]:The random inertia weight coefficients are shown in equation (10).

$$\omega = 0.5 + \frac{\left(rand\left(0,1\right)\right)}{2} \tag{10}$$

rand (0,1) is a random number between 0 and 1 in the formula, ω The range of values for is 0.5 to 1. The random inertia weight coefficient is mainly used to improve the randomness of speed updates, in order to increase the population diversity of particles.

Adaptive inertia weight coefficients:There is also an adaptive inertia weight coefficient, and the basic method is shown in equation (11).

$$\omega_i^t = \begin{cases} \omega_{\min} + \left(\omega_{\max} - \omega_{\min}\right) \cdot \dfrac{f_{\max}^t - f\left(x_i^t\right)}{f_{\max}^t - f_{average}^t}, & f\left(x_i^t\right) \geq f_{average}^t \\ \omega_{\max}, & f\left(x_i^t\right) < f_{average}^t \end{cases} \tag{11}$$

Compared with the first two more common linear and nonlinear decreasing methods, the biggest feature of adaptive inertia weight coefficients is that all particles have one inertia weight coefficient, while the first two are shared by all particles. The inertia weight coefficient is now related to the fitness of each particle, and the inertia weight coefficient of the particles is adjusted based on the average fitness value of the population particles in the current iteration round. This is very helpful for global optimization and fast convergence. However, adopting an adaptive adjustment method may result in premature convergence of the algorithm to the local optimal solution, and may also lead to oscillation and instability of the algorithm[20].

Yang et al. proposed a new particle swarm optimization method, LHNPSO, based on the basic nonlinear inertia weight reduction method. This method uses non-linear functions with significant order changes to adjust inertia weights, cognitive parameters, and social parameters. After performance analysis, it was found that this method can converge faster. However, there has been no improvement in the issue of insufficient population diversity[21].

Yan et al. proposed an improved nonlinear adaptive inertia weight particle swarm optimization algorithm supplemented by linear change learning factors. By combining the improved particle swarm optimization algorithm with a coevolutionary algorithm, the convergence speed of the algorithm is improved, while also improving the problem of standard PSO algorithms easily getting stuck in local optima[22].

2) Acceleration coefficient

For the learning factor, in equations (1) of velocity update, the part where c1 is located is the "cognitive" part, which represents the particle's experience of its best position and represents the trend of the particle moving towards its historical best position. C1 is also known as a cognitive factor; The part where c2 is located is the "social" part, which reflects the mutual cooperation and information sharing between particles, representing the trend of particles moving towards the optimal position of the group. Generally speaking, a larger c1 value and a smaller c2 value are required at the beginning of the algorithm to enable particles to approach the global optimal position faster. At the end of the algorithm, a larger c2 value and a smaller c1 value are required to facilitate better convergence of particle fitness [23]. How to adjust learning factors in the algorithm process is also a worthwhile research question.

For learning factors, non-linear adjustment strategies are commonly used nowadays, as shown in equations (12) and (13).

$$c_1 = c_{1\max} \cdot \left(\frac{c_{1\min}}{c_{1\max}} \right)^{\frac{t}{T_{\max}}} \tag{12}$$

$$c_2 = c_{2\min} \cdot \left(\frac{c_{2\max}}{c_{2\min}} \right)^{\frac{t}{T_{\max}}} \tag{13}$$

This strategy can to some extent balance the search ability within its own local range and the ability to converge to the local optimal value, so that the algorithm cannot converge too early and will not always search within its own range. But the effect is not particularly outstanding.

Compressing learning factors is also a strategy for adjusting two learning factors, which can ensure the convergence of the PSO algorithm to a certain extent by selecting appropriate parameters. The speed update formula with the introduction of compressed learning factors is shown in equation (14).

$$v_i^{t+1} = K \cdot \left[\omega v_i^t + c_1 r_1 \left(P_i^t - X_i^t \right) + c_2 r_2 \left(G^t - X_i^t \right) \right] \tag{14}$$

The value of K is shown in equation (15), and generally speaking, both c1 and c2 are taken as 2.05.

$$K = \frac{2}{\left| 2 - \phi - \sqrt{\phi^2 - 4\phi} \right|}, \quad \phi = c_1 + c_2 \tag{15}$$

The compressed learning factor method can control the forward speed of particles to balance the ability of global search and local search. However, there is still no significant improvement in improving the population diversity of the PSO algorithm and avoiding the problem of getting stuck in local optima too early[24].

3) Improvements for population initialization

Improving the initialization steps of PSO algorithm is also one of the research directions in recent years. The population initialization of the standard PSO algorithm randomly generates the attributes of the population particles, and this relatively simple processing method has become a point of improvement for subsequent researchers.

In 2022, Chu et al. proposed an improved unmanned aerial vehicle path planning algorithm based on PSO. This method introduces speed adaptive adjustment for chaotic initialization, and introduces an improved logistic chaotic mapping in the algorithm to improve the convergence speed and reduce the initialization time of the algorithm[25].

Yang et al. proposed an improved hybrid PSO path planning algorithm for population initialization in 2023. This algorithm divides particles into three categories based on their fitness values: excellent performance, average performance, and poor performance. At the beginning of each iteration, the algorithm initializes poorly performing particles to increase population diversity and prevent premature convergence[26].

4) Hybrid PSO algorithm

Integrating concepts from other algorithms to enhance the PSO algorithm has emerged as a popular strategy for improvement in recent years. For instance, in 2023, Huang et al. introduced a hybrid path planning algorithm for mobile robots, integrating A* and PSO. This method employs a redundant point removal strategy to optimize the path initially planned by the A* algorithm, obtaining a set of key nodes to guide the PSO algorithm in searching for the optimal solution. This approach effectively reduces computation time .

In the realm of hybrid PSO algorithms, fusion with other heuristic algorithms is deemed as one of the most significant hybridization methods. Tao et al. proposed an enhanced path planning algorithm for PSO mobile robots, integrating the cross-mutation operation from the genetic algorithm to update particle positions, thereby addressing the issue of the PSO algorithm becoming easily trapped in local optimal solutions [27].

Additionally, Ayad et al. introduced an improved hybrid PSO path planning algorithm. This algorithm combines two naturally inspired metaheuristic algorithms—namely, a blend of grey wolf optimization and particle swarm optimization—to enforce distance limitations and adhere to path optimization criteria, refining the algorithm's original structure and enhancing its performance [28].

III. Methods

A. The environmental model of this paper

This paper uses a two-dimensional plane coordinate system to represent the space of robot motion. In order to simplify the model, obstacles are represented by circles of different radii, and the radius of the k-th obstacle is r_k. The center coordinates of the circle (x_k, y_k) are shown in Figure 4.

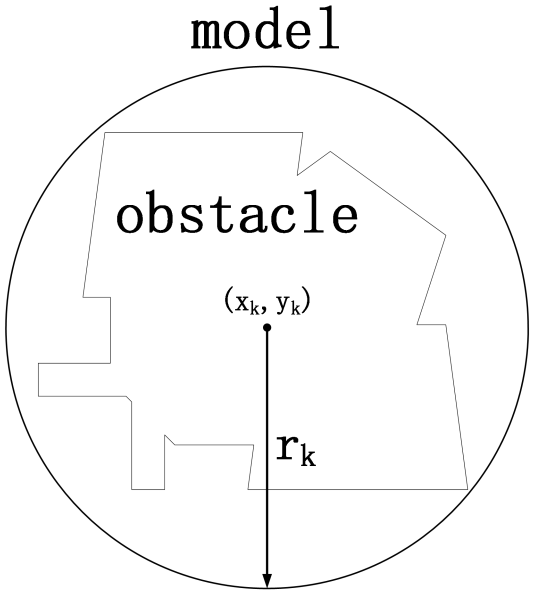


Fig. 4. Obstacle Model

The path length *dis* is represented by equation (16).

$$dis = \sum_{m=0}^{D+1} \sqrt{\left(x_{m+1} - x_m \right)^2 + \left(y_{m+1} - y_m \right)^2} \tag{16}$$

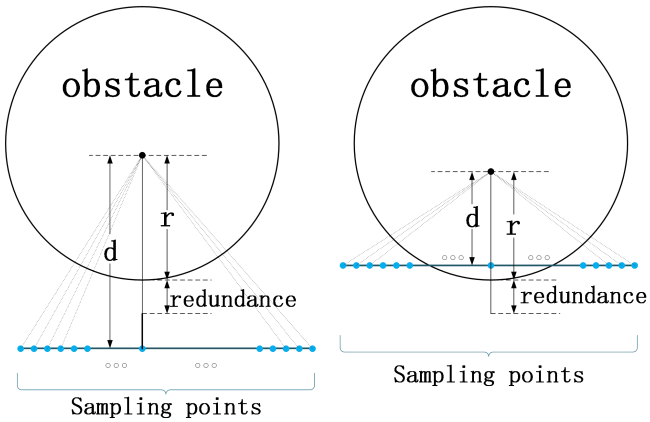
This method uses the Euclidean distance method. Where D represents the particle dimension, (x_0, y_0) and (x_{D+1}, y_{D+1}) represent the starting and ending points, respectively.

The fitness function used in this paper is represented by equation (17).

$$fit = \frac{100 \cdot collision}{dis} \tag{17}$$

In the equation, *dis* is the path length in the above equation; Collision is an indicator that determines whether the path collides with an obstacle. If the path collides with an obstacle, the value is 0; if the path does not collide with an obstacle, the value is 0.1; The function is multiplied by 100 in order to retain more decimal places in the simulation software and improve the accuracy of the algorithm.

Under the setting of the fitness function, the higher the fitness, the better the path; If colliding with obstacles, the fitness is 0, which is an illegal path. The following mathematical model is used to determine whether the path collides with obstacles, as shown in Figure 5.



(a) (b)

Fig. 5. Determining whether the path collides with obstacles. (a) Indicates not colliding with obstacles, (b) indicates colliding with obstacles

Determine whether the line segment formed by every two points collides with a circular obstacle: first, uniformly sample s points from the line segment, and then use equation (18) to calculate the d^2, which is the square of the distance from each sampling point to the center of the circle.

$$d^2 = \left(xs_i - x_k\right)^2 + \left(ys_i - y_k\right)^2 \tag{18}$$

If equation (19) holds, then it means that the distance from the sampling point to the center of the obstacle circle is less than the redundancy, then the line segment collides with the obstacle, otherwise it does not collide (redundancy is a redundant value set by the user according to their needs).

$$d^2 - r^2 \le redundance^2 \tag{19}$$

We judge the line segment formed by every two adjacent points in each path and each obstacle. If they do not collide, the entire path is legal, with a collision value of 0.1. Although theoretically there is a possibility of judgment errors in this mathematical model, in reality, as long as the number of sampling points is appropriate, the probability of errors can be reduced to a very low level, while also ensuring computational efficiency. At the same time, the mathematical model of the obstacle takes into account a certain degree of redundancy, and the actual obstacle must be smaller than the constructed model. The algorithm generally does not use a redundancy value of 0, but rather 0.01, 0.04, or 0.09. So the possibility of the model failing can be ignored.

B. The method of this paper

Compared to traditional PSO path planning algorithms, the enhanced algorithm framework presented in this paper exhibits the following distinctions: (1) integration of an adaptive particle reactivation module within the iteration process, aimed at mitigating the issue of premature convergence to local optima frequently encountered by PSO path planning algorithms due to inadequate population diversity; (2) incorporation of the simulated annealing algorithm to refine the updating strategy of global historical optimal solutions, thereby augmenting the diversity of the population.

The improved algorithm framework is shown in Figure 6.

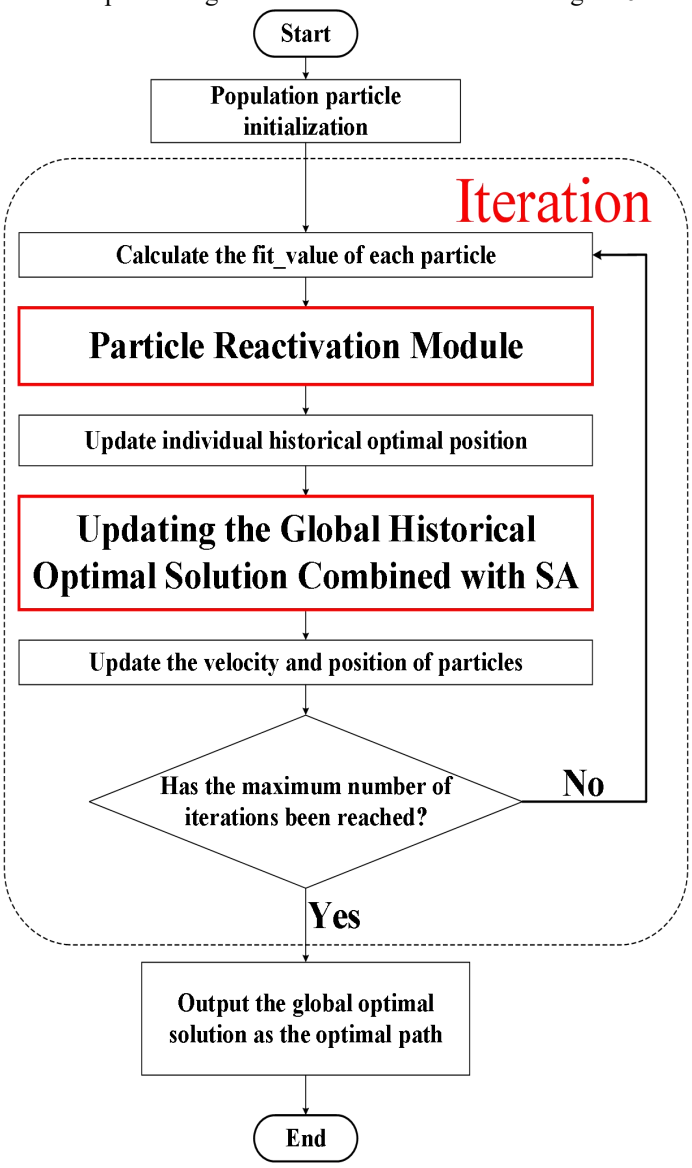


Fig. 6. Improved PSO path planning algorithm framework

Initially, the population is initialized, with "position" and "velocity" vectors of N particles randomly generated. Each particle's "position" is determined as a path comprising D points in a predetermined sequence, ensuring randomness while avoiding collision with obstacles within a constrained range. Concurrently, the initialization of each particle's "velocity" entails the generation of a D-dimensional random row vector within the limits of velocity [Vmin, Vmax]. As the paths are situated on a two-dimensional plane, the "position" and "velocity" of each particle are represented by two components, denoted as "X" and "Y".

Then enter the part of the loop iteration. Firstly, the fitness

value (fit_value) of each particle will be calculated and stored in an N-dimensional column vector. After obtaining the fit_value of each particle, you will enter the core part of this paper: the adaptive particle reactivation module. This module will ensure that the particle swarm always maintains a certain level of population diversity, improving the problem of algorithms easily falling into local optima. This section will be explained in detail later. After the adaptive particle reactivation module, a global historical optimal solution update strategy that integrates the idea of simulated annealing algorithm is used for updating, and then the velocity and position are updated.

C. *Adaptive particle reactivation module*

During the research process, the author found that in PSO based path planning algorithms, only during the initialization phase will the generated random path be judged for legality. In the subsequent iteration process, updating the position of particles can cause collisions between the path and obstacles. And due to the existence of a speed limit, particles may not be able to update to their legal positions quickly and move towards a more optimal direction. This algorithm will generate a large number of "illegal" paths during the iteration process. Therefore, this paper introduces an adaptive particle reactivation module during the algorithm iteration process to solve this problem. The structure of this module is shown in Figure 7.

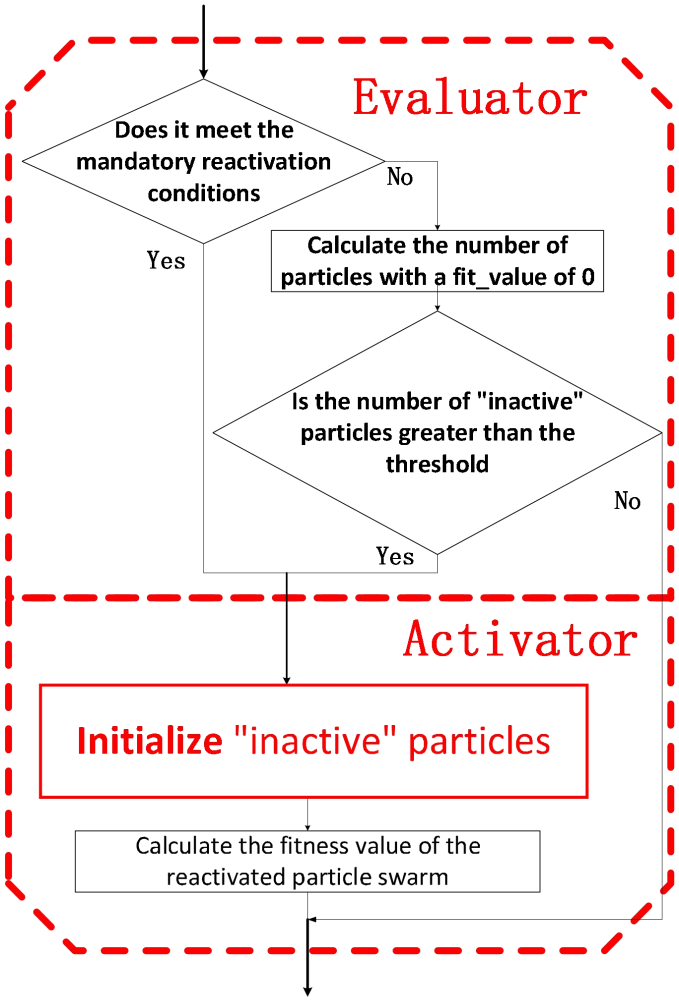


Fig. 7. Adaptation particle reactivation module

The particle reactivation module consists of two distinct segments: the discriminator at the outset and the activator towards the conclusion. Initially, the algorithm traverses the "discriminator" phase, which assesses whether particle reactivation is warranted. A successful assessment transitions the algorithm to the subsequent "activator" phase, whereas failure prompts a return to the iteration process.

The discrimination stage comprises two primary evaluations: (1) forced reactivation discrimination and (2) adaptive reactivation discrimination. Forced reactivation discrimination entails predefining a threshold for forced reactivation attempts before algorithmic execution, ensuring a minimum number of reactivation operations during each run. If the current iteration meets the criteria for forced reactivation, subsequent judgment stages are bypassed, and the reactivation operation within the "activator" section is promptly initiated.

In adaptive reactivation discrimination, an "inactive" state is initially defined. This state occurs when a path intersects with an obstacle during the position optimization update process, designating the corresponding particles as "inactive." If the mandatory reactivation condition is not met during the current iteration, the count of "inactive" particles within the population is computed. Success is declared when the proportion of "inactive" particles surpasses a predefined threshold, leading to entry into the "activator" phase. Conversely, failure prompts an immediate exit from the particle reactivation module to pursue alternative operations within the ongoing iteration.

Through iterative experimental testing, it was observed that setting the threshold too low leads to excessive reactivation operations, thereby consuming excessive computational resources, while setting it too high results in insufficient reactivation occurrences. Consequently, this paper introduces a linearly increasing threshold, as depicted in Equation (20), to strike a balance between computational efficiency and reactivation frequency.

$$Num_{th} = 0.5 + (0.95 - 0.5) \cdot \frac{t}{T} \tag{20}$$

The minimum threshold set in this paper is 50%, and the maximum threshold is 95%.

Upon entering the "Activator" phase, the algorithm reactivates the population particles by selecting the "inactive" ones and initializing them, transforming their represented paths from illegal to legal. This procedure reinstates previously "deactivated" particles while also preserving population diversity, thereby mitigating the risk of the algorithm converging prematurely to local optima owing to the stochastic nature of its initialization.

Finally, we will recalculate the fit value of the population particles to overwrite the original value, in order for the algorithm to perform subsequent operations.

D. *Global Historical Optimal Solution Update Mechanism Based on Simulated Annealing*

This paper integrates the simulated annealing algorithm to enhance the update strategy for the global historical optimal solution within the PSO algorithm framework. In the standard PSO algorithm, updating occurs whenever the fitness value of the current population's optimal solution

surpasses that of the global historical optimal solution[29]. However, such updates invariably diminish population diversity, thereby increasing the susceptibility of the algorithm to local optima. Refer to Figure 8 for an illustration of this process.

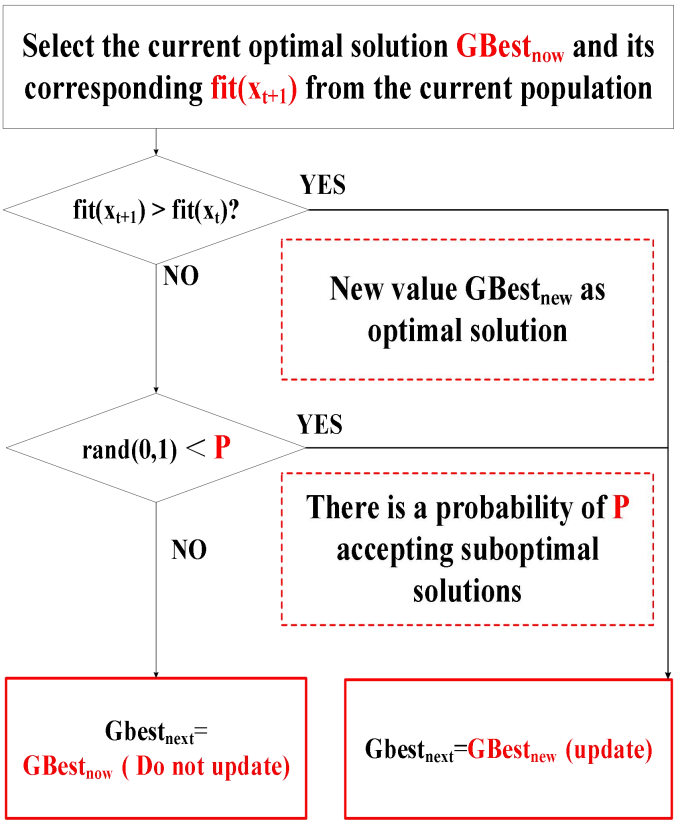


Fig. 8. Global Historical Optimal Solution Update Strategy Combined with Simulated Annealing Algorithm

Firstly, there is a probability P, which represents the likelihood of accepting a worse solution as the global historical optimal solution. In other words, if the newly calculated global historical optimal solution in this iteration is worse than the current global historical optimal solution, there is still a probability P that it will be updated as the global historical optimal solution[30].

The calculation method of probability P is shown in equation (21).

$$P = \exp\left(-\frac{fit(x_t) - fit(x_{t+1})}{Temp_t}\right) \tag{21}$$

Among them, fit (x_t) and fit (x_{t+1}) are the fitness values of the current global historical optimal solution and the current population optimal solution, collectively referred to as the previous and new values. If the new value is greater than the previous value, the new solution is better than the previous solution, and P is greater than 1, it will naturally be directly updated to the global historical optimal solution. If the new value is smaller than the previous value, there is a probability of P updating the new value to the new global historical optimal solution[31].

Tempt is the temperature. However, if the temperature is too high at the end of the algorithm iteration, resulting in a probability P that is too high and close to 1, it will cause the algorithm to be unstable and difficult to converge. So during the algorithm iteration process, it is necessary to gradually lower the temperature. We have set a shrinkage coefficient r here to reduce the temperature, as shown in equation (22).

$$Temp_{t+1} = rate \cdot Temp_t \tag{22}$$

Regarding the shrinkage coefficient rate in the equation, if its value is too large, the speed of temperature reduction will be too slow, and the algorithm is more likely to accept suboptimal solutions, resulting in the algorithm consuming too much time. If the rate value is too small, it will cause the algorithm to converge too quickly and easily fall into local optima. Therefore, we let the rate decrease as the number of iterations increases, as shown in equation (23).

$$rate = \frac{T_{max} - t}{T_{max}} \tag{23}$$

In the formula, Tmax is the maximum number of iterations, and t is the current number of iterations. This formula causes the temperature to decrease from the highest temperature MaxTempto the lowest temperature minTemp.

IV. Experiment

To assess the efficacy of the enhanced algorithm proposed in this study, we conducted two control experiments: (1) A performance evaluation comparing it with standard PSO and genetic algorithm (GA) within the experimental environment devised in this research. (2) Comparative analyses were carried out with fusion PSO algorithms IPSO [26] and HGWO-PSO [28], both of which were augmented for population diversity and evaluated under identical conditions as described in this paper.

A. experimental environment

The platform hardware and software specifications utilized in the experiments are as follows: The CPU is an AMD Ryzen 7 5800H with Radeon Graphics, operating at 3.2GHz, with 16GB of RAM. The GPU is an NVIDIA GeForce RTX3060 Laptop GPU. The system operates on a 64-bit Windows 11 operating system. The simulation software employed is MATLAB 2022b.

B. Performance comparison experiment with standard PSO algorithm and GA algorithm

This paper compares the path planning of standar PSO algorithm and GA algorithm with the improved algorithm in this paper.. After repeated experiments, the experimental parameters used in this paper are shown in Table 2. In the comparative experiment, each algorithm was run 50 times and data such as the generated path, algorithm convergence curve, path length, and computation time were recorded.

The experimental environment is set in a 10 × 10 Cartesian coordinate system, with a starting point at (0,0), represented by blue squares. The endpoint is at (10,10), represented by green blocks, and there are four circular obstacles set up, as shown in Figure 9. The coordinates and radius of the obstacles are shown in Table 1.

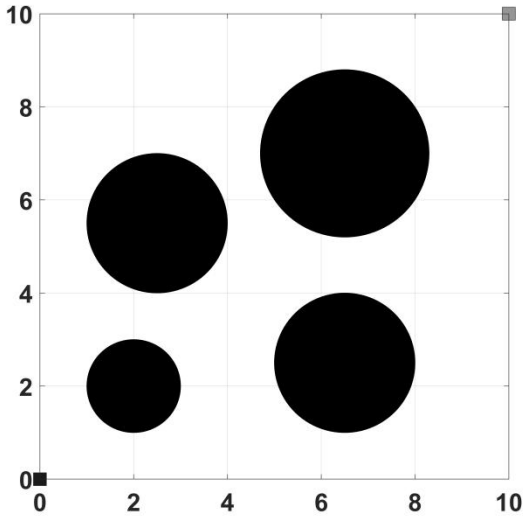


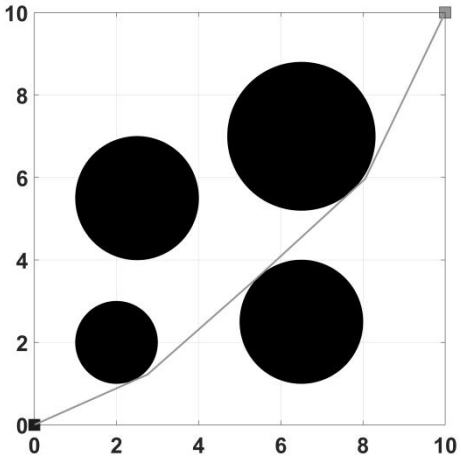
Fig. 9. Experimental Environment Diagram

TABLE I
OBSTACLE DATA

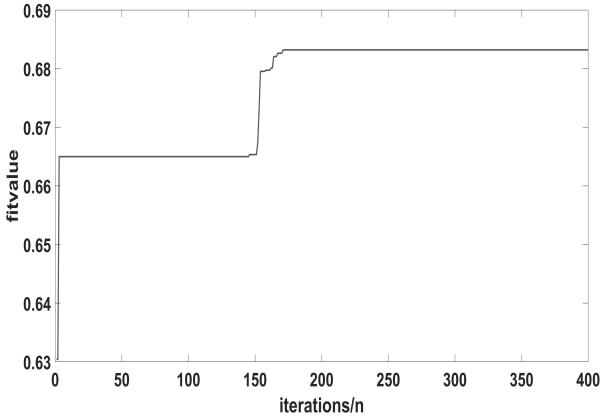
Obstacles	coordinate	radius
Obstacle1	(2.0, 2.0)	1.0
Obstacle2	(6.5, 7.0)	1.8
Obstacle3	(2.5, 5.5)	1.5
Obstacle4	(6.5, 2.5)	1.5

The experimental parameters are set as follows: the particle population size N is 80; the dimensionality D (number of control points) is 3; the maximum number of iterations $T_{\max}$ is 400; the maximum and minimum inertia weight coefficients $\omega_{\max}$ and $\omega_{\min}$ are 0.9 and 0.3, respectively; the maximum and minimum values of the learning factors $C_{1\max}$ and $C_{1\min}$ are 2 and 1, respectively, and $C_{2\max}$ and $C_{2\min}$ are also 2 and 1, respectively; the maximum velocity $V_{\max}$ is 1; and the minimum reactivation count $relife\_num$ is 4.

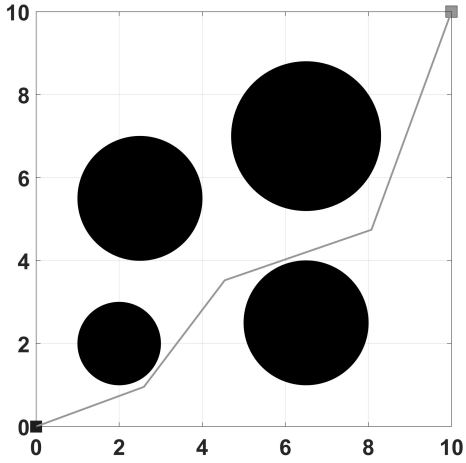
The experimental simulation results are shown in Figure 10.The experimental data comparison of each algorithm is shown in Table II.



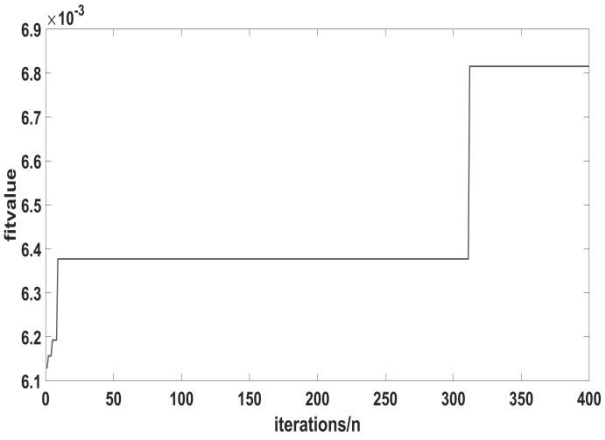
(a)



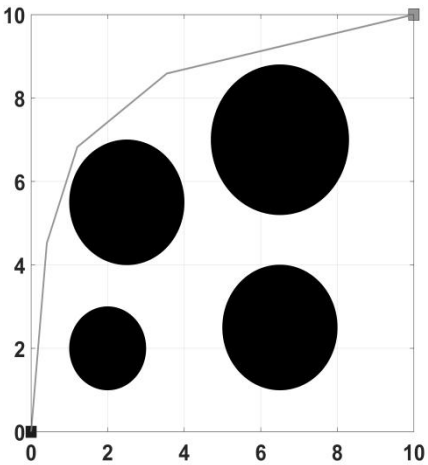
(b)



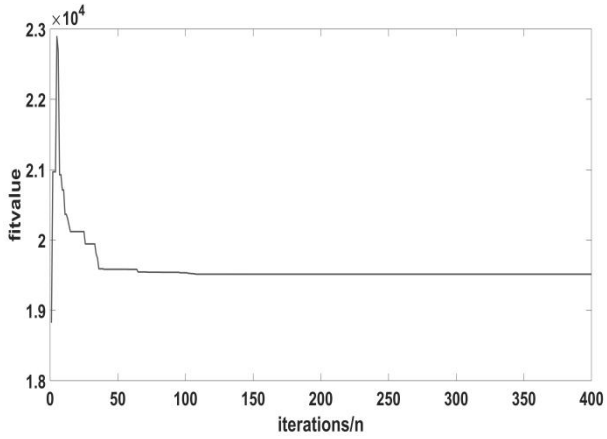
(c)



(d)



(e)



(f)

Figure. 10. Results and convergence curves of improved PSO path planning algorithm and traditional algorithm.(a) (b) represents the path planning results and convergence curve of the improved PSO algorithm; (c) and (d) represent the standard PSO algorithm path planning results and convergence curve; (e) and (f) represent the GA algorithm path planning results and convergence curve.

TABLE II
COMPARISON OF EXPERIMENTAL DATA

algorithm	Optimum distance	Average distance	STD
My algorithm	**14.5875**	**14.7269**	**0.2079**
PSO	14.6306	15.7243	0.4820
GA	14.9423	15.7776	0.4750

Based on the aforementioned experimental outcomes, the enhanced PSO path planning algorithm exhibits minimal variance in terms of iteration count when compared with traditional PSO variants. In terms of runtime, it marginally lags behind conventional PSO algorithms but outperforms GA algorithms in efficiency. Notably, the average path length and its standard deviation demonstrate notable advantages, underscoring the enhanced algorithm's enhanced stability.

C. Comparative experiments with other improved PSO path planning algorithms

This study contrasts two enhanced PSO path planning algorithms: IPSO [26], which integrates the dynamic window method (IDWA) to enhance PSO, and HGWO-PSO [28], a hybrid algorithm combining PSO with the grey wolf optimization algorithm. Experimental evaluations are conducted using an improved PSO path planning algorithm within obstacle environments and basic parameter configurations outlined in two prior studies. Comparative analyses are then performed against data extracted from these references.

1) Comparison experiment with IPSO

The experimental obstacle environment is shown in Figure 11, with a population of 50 and a maximum iteration of 300.

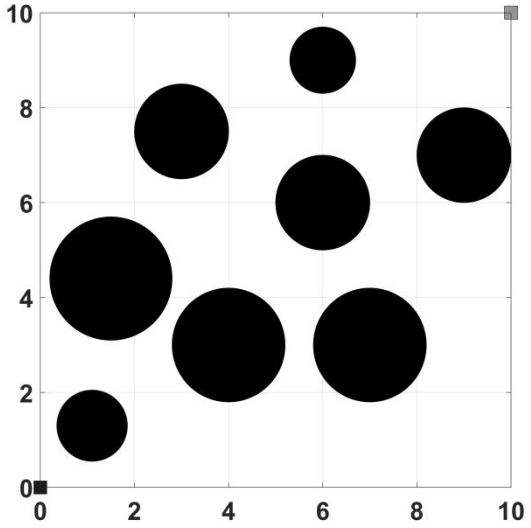


Figure. 11. Obstacle Environment Setting in Comparison Experiment with IPSO

The optimal path of the algorithm in this environment is shown in Figure 12, and the comparison with IPSO data is shown in Table III.

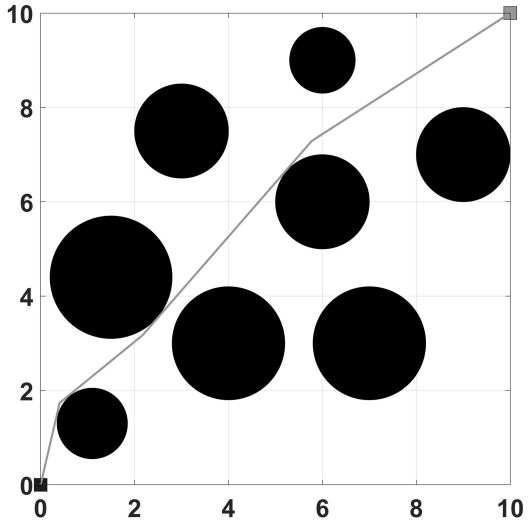


Figure. 12. The result chart of the improved algorithm in IPSO experimental environment in this article

TABLE III
COMPARISON OF ALGORITHM AND IPSO DATA IN THIS PAPER

algorithm	Optimum	Average
My algorithm	14.5509	**14.8106**
IPSO[26]	14.5374	15.0966

From the data in Table III, it can be seen that our algorithm is slightly inferior to the IPSO algorithm in the shortest distance, but superior to the IPSO algorithm in the average distance.

2) Comparison experiment with HGWO-PSO

The experimental obstacle environment is shown in Figure 13, with a population of 50 and a maximum iteration of 300.

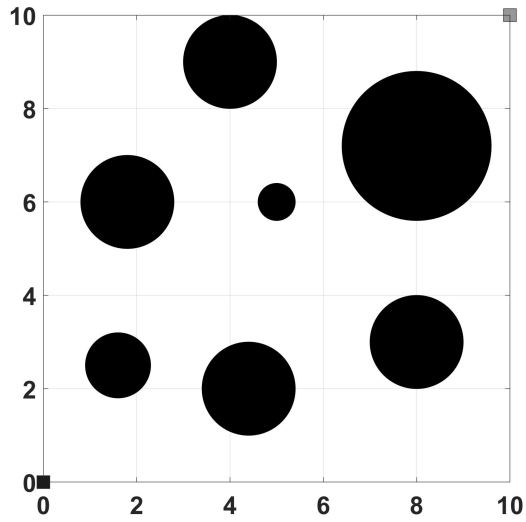


Figure. 13. Obstacle Environment Setting in Comparison Experiment with HGWO-PSO

The optimal path of the algorithm in this environment is shown in Figure 14, and the comparison with IPSO data is shown in Table IV.

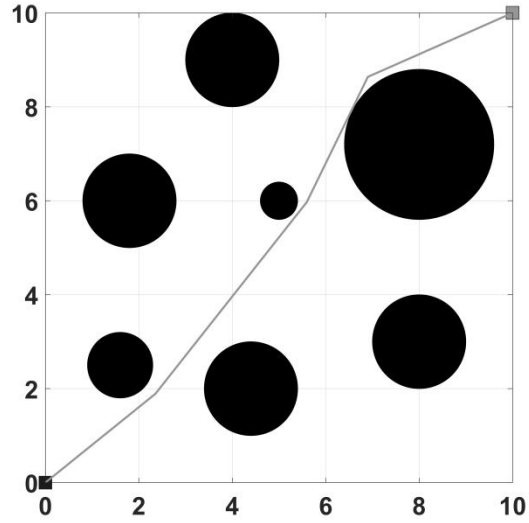


Figure. 14. The result chart of the improved algorithm in the HGWS-PSO experimental environment

TABLE IV
COMPARISON OF ALGORITHM AND IPSO DATA IN THIS PAPER

algorithm	Average distance	STD	Average time/s
My algorithm	**14.8990**	**0.2002**	**0.3402**
HGWO-PSO[28]	15.0858	1.0758	28

Based on the data presented in Table IV, it is evident that while the enhanced algorithm proposed in this study slightly underperforms compared to HGWO-PSO in terms of the shortest distance metric, it surpasses HGWO-PSO in average distance. Furthermore, the proposed algorithm demonstrates significantly lower runtime compared to HGWO-PSO. Additionally, the distance standard deviation of the proposed algorithm is superior to that of HGWO-PSO, indicating better stability in the algorithm presented in this study.

V. CONCLUSION

In this paper, we proposed an improved PSO path planning algorithm to address the path planning problem based on the Particle Swarm Optimization (PSO) algorithm. Firstly, a particle reactivation module was added during the algorithm iteration process. This module can adaptively reinitialize the particles based on their states, thus maintaining population diversity. Secondly, inspired by the Simulated Annealing algorithm, we improved the update strategy for the global historical optimal solution of the particles, further enhancing population diversity. The experimental results demonstrate that the proposed method effectively maintains population diversity, mitigates the problem of the PSO algorithm's tendency to quickly converge to a local optimum in the early stages, and improves the algorithm's stability. Future work aims to: (1) optimize the algorithm process to enhance its applicability; (2) further optimize the algorithm for complex environments; (3) apply the algorithm to actual intelligent mobile robots to verify its effectiveness and robustness, and study the algorithm's real-time performance.

REFERENCES

[1] Song, Baoye, Zidong Wang, and Lei Zou. "An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve." Applied Soft Computing 100 (2021): 106960.
[2] Karur, Karthik, et al. "A survey of path planning algorithms for mobile robots." Vehicles 3.3 (2021): 448-468.
[3] Yang, Zhenjian, et al. "Mobile Robot Path Planning Based on Improved Particle Swarm Optimization and Improved Dynamic Window Approach." Journal of Robotics 2023 (2023).
[4] Huang, Changsheng, et al. "APSO: An A*-PSO hybrid algorithm for mobile robot path planning." IEEE Access (2023).
[5] Dijkstra, Edsger W. "A note on two problems in connexion with graphs." Edsger Wybe Dijkstra: His Life, Work, and Legacy. 2022. 287-290.
[6] Stentz, Anthony. "Optimal and efficient path planning for partially-known environments." Proceedings of the 1994 IEEE international conference on robotics and automation. IEEE, 1994.
[7] Khatib, Oussama. "Real-time obstacle avoidance for manipulators and mobile robots." The international journal of robotics research 5.1 (1986): 90-98.
[8] Zhou, Huai-fang, Hua Zhang, and Meng-wen Qiu. "Radiation avoiding algorithm for nuclear robot path optimization." Annals of Nuclear Energy 169 (2022): 108948.
[9] Kennedy, James, and Russell Eberhart. "Particle swarm optimization." Proceedings of ICNN'95-international conference on neural networks. Vol. 4. ieee, 1995.
[10] Yu, Zhenhua, et al. "A novel hybrid particle swarm optimization algorithm for path planning of UAVs." IEEE Internet of Things Journal 9.22 (2022): 22547-22558.
[11] Wang, Haiyan, and Zhiyu Zhou. "A heuristic elastic particle swarm optimization algorithm for robot path planning." Information 10.3 (2019): 99.
[12] Huang, Xiongjian, et al. "Application of Oil-immersed Transformer Fault Diagnosis Based on Modified Neural Network Combined with PSO." 2021 IEEE 4th International Conference on Power and Energy Applications (ICPEA). IEEE, 2021.
[13] Pang, Wei, et al. "Fuzzy discrete particle swarm optimization for solving traveling salesman problem." The Fourth International Conference onComputer and Information Technology, 2004. CIT'04.. IEEE, 2004.
[14] Zhao, Chenglong, et al. "Design and Research of Series Actuator Structure and Control System Based on Lower Limb Exoskeleton Rehabilitation Robot." Actuators. Vol. 13. No. 1. MDPI, 2024.
[15] Xue, Han. "A quasi-reflection based SC-PSO for ship path planning with grounding avoidance." Ocean engineering 247 (2022): 110772.

[16] Yu, Along, et al. "Research on reinforced corrosion monitoring system based on Multi-sensor data fusion and wireless sensor network." 2023 4th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT). IEEE, 2023.
[17] Zhongda, Tian, et al. "SVM predictive control for calcination zone temperature in lime rotary kiln with improved PSO algorithm." Transactions of the Institute of Measurement and Control 40.10 (2018): 3134-3146.
[18] Gao, Mengqi, et al. "Multi-granularity competition-cooperation optimization algorithm with adaptive parameter configuration." Applied Intelligence 52.11 (2022): 13132-13161.
[19] Shami, Tareq M., et al. "Particle swarm optimization: A comprehensive survey." Ieee Access 10 (2022): 10031-10061.
[20] Li, Mi, et al. "An improved particle swarm optimization algorithm with adaptive inertia weights." International Journal of Information Technology & Decision Making 18.03 (2019): 833-866.
[21] Yang, Chengwei, et al. "Low-discrepancy sequence initialized particle swarm optimization algorithm with high-order nonlinear time-varying inertia weight." Applied Soft Computing 29 (2015): 386-394.
[22] Yan, L. I. N., et al. "Improved adaptive inertia weight PSO algorithm and its application in nuclear power pipeline layout optimization." Chinese Journal of Ship Research 18.3 (2023): 1-12.
[23] Tian, Zhongda, Yi Ren, and Gang Wang. "Short-term wind speed prediction based on improved PSO algorithm optimized EM-ELM." Energy Sources, Part A: Recovery, Utilization, and Environmental Effects 41.1 (2019): 26-46.
[24] Jain, Meetu, et al. "An overview of variants and advancements of PSO algorithm." Applied Sciences 12.17 (2022): 8392.
[25] Chu, Hongyue, Junkai Yi, and Fei Yang. "Chaos particle swarm optimization enhancement algorithm for UAV safe path planning." Applied Sciences 12.18 (2022): 8977.
[26] Yang, Zhenjian, et al. "Mobile Robot Path Planning Based on Improved Particle Swarm Optimization and Improved Dynamic Window Approach." Journal of Robotics 2023 (2023).
[27] Qiuyun, Tao, et al. "Improved particle swarm optimization algorithm for AGV path planning." IEEE Access 9 (2021): 33522-33531.
[28] Alabdalbari, Ayad Abdulrahem, and Issa Ahmed Abed. "New robot path planning optimization using hybrid GWO-PSO algorithm." Bulletin of Electrical Engineering and Informatics 11.3 (2022): 1289-1296.
[29] Vincent, F. Yu, et al. "A simulated annealing algorithm for the vehicle routing problem with parcel lockers." IEEE Access 10 (2022): 20764-20782.
[30] Shi, Weijie, et al. "Path planning of multi-robot systems with boolean specifications based on simulated annealing." IEEE Robotics and Automation Letters 7.3 (2022): 6091-6098.
[31] Vincent, F. Yu, et al. "A simulated annealing algorithm for the vehicle routing problem with parcel lockers." IEEE Access 10 (2022): 20764-20782.