

RESEARCH ARTICLE

APSO: An A*-PSO Hybrid Algorithm for Mobile Robot Path Planning

CHANGSHENG HUANG¹, YANPU ZHAO¹, MENGJIE ZHANG², AND HONGYAN YANG¹¹School of Economics and Management, China University of Petroleum (East China), Qingdao, Shandong 266580, China²Dareway Software Company Ltd., Jinan, Shandong 250200, China


Corresponding author: Yanpu Zhao (Z21080319@s.upc.edu.cn)

ABSTRACT Aiming at the problems of the A* algorithm in mobile robot path planning, such as multiple nodes, low path accuracy, long running time and difficult path initialization of particle swarm optimization, an APSO algorithm combining A* and PSO was proposed to calculate the optimal path. First, a redundant point removal strategy is adopted to preliminarily optimize the path planned by the A* algorithm and obtain the set of key nodes. Second, a stochastic inertia weight is proposed to improve the search ability of PSO. Third, a stochastic opposition-based learning strategy is proposed to further improve the search ability of PSO. Fourth, the global path is obtained by using the improved PSO to optimize the set of key nodes. Fifth, a motion time objective function that is more in line with the actual motion requirements of the mobile robot is used to evaluate the algorithm. The simulation results of path planning show that the path planned by APSO not only reduces the running time of the mobile robot by 17.35%, 14.84%, 15.31%, 15.21%, 18.97%, 15.70% compared with the A* algorithm in the six environment maps but also outperforms other path planning algorithms to varying degrees. Therefore, the proposed APSO is more in line with the actual movement of the mobile robot.

INDEX TERMS A* algorithm, mobile robot, path planning, particle swarm optimization, stochastic inertia weight, stochastic opposition-based learning.

I. INTRODUCTION

In the new industrial revolution, also known as Industry 4.0, mobile robots are used in a wide range of scenarios. Path planning has always been a key research problem in the field of mobile robots. Path planning refers to planning a safe and collision-free optimal or near-optimal path from the starting point to the target point in an environment with obstacles [1]. The path planning of mobile robots needs to solve the following three problems: first, the path can make the mobile robot move from the starting point to the target point; second, the mobile robot should avoid obstacles in the environment in the path planning algorithm; finally, on the basis of solving the first two problems, the motion trajectory of the mobile robot is optimized. The optimization objectives usually include the shortest path, the fewest path nodes, and the smallest turning amplitude [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Sotirios Goudos .

Path planning of mobile robots can be divided into static path planning (global path planning) and dynamic path planning (local path planning) based on a known static environment and an unknown dynamic environment [3]. Local path planning is the real-time path planning carried out by the mobile robot in the process of movement according to the surrounding environment information fed back by its sensor, which can cope with frequent and random changes in the environment [4]. However, because dynamic path planning is very dependent on the local environment, the path planned by only using the dynamic path planning algorithm may not be the globally optimal path [5]. Global path planning uses a static path planning algorithm to search the global optimal path in the established environment map model before the mobile robot moves. What we study in this paper is the global path planning problem based on a static environment.

As seen from the above description, the core of the path planning problem is the path planning algorithm. Likewise, static path planning is no exception. Static path planning

algorithms can be subdivided into heuristic methods and artificial intelligence algorithms [6]. Widely acclaimed heuristic methods include the Dijkstra algorithm [7], A* algorithm [8], LPA* algorithm [9], D* algorithm [10], PRM algorithm [11], RRT algorithm [12], etc. In recent years, researchers have performed more in-depth research on heuristic methods in static path planning problems. Examples include the A* algorithm based on guideline [13], A* algorithm based on hybrid heuristic function [14], A* algorithm combined with B-spline curve [15], A* algorithm based on bidirectional alternating search [16], D*Lite algorithm based on inverse distance weighted interpolation [17], improved PRM algorithm [18], RRT algorithm based on APF [19], and RRT-connect algorithm based on triangle inequality [20].

In static path planning problems, commonly used artificial intelligence algorithms include the genetic algorithm (GA) [21], differential evolution (DE) [22], ant colony optimization (ACO) [23], particle swarm optimization (PSO) [24], etc. Additionally, to make these intelligent algorithms better at static path planning problems, researchers have also proposed a GA that integrates the Bessel curve [25], an adaptive GA that integrates the collision detection method [26], adaptive ACO [27], improved PSO based on the minimum-maximum normalization method [28], PSO based on the smoothing principle [29], etc. In recent years, with the increase in intelligent optimization algorithms, some new intelligent algorithms have been gradually applied to the global path planning problem. Yuan et al. [30] designed a bat algorithm that combined a logarithmic decline strategy and Cauchy perturbation and applied it to path planning problems. Li et al. [31] combined differential evolution with the whale optimization algorithm and applied it to the vehicle routing problem. Li et al. [32] applied the improved artificial fish swarm algorithm to the path planning of mobile robots and smoothed the path by using the continuous segmented Bessel curve. Pattnaik et al. [33] applied a chemical reaction optimization algorithm to the global path planning problem.

Due to the limitations of a single path planning algorithm [34], in recent years, researchers have developed many hybrid path planning algorithms that combine two static path planning algorithms to solve this problem. Hilal ARSLAN et al. [35] proposed a hybrid Dijkstra-BFS algorithm, which not only improved the accuracy of the path but also the computational efficiency of the algorithm. Leila Pasandi et al. [36] proposed a hybrid A*-ACO algorithm, which not only improved the path planning ability of the algorithm in a complex environment but also reduced the operation time of the algorithm. Zhou et al. [37] proposed a Dijkstra-ACO algorithm. In this algorithm, the Dijkstra algorithm was used to plan the initial path, and then the ACO algorithm was used to optimize the path, which not only solved the problem of insufficient path accuracy of the Dijkstra algorithm. It also solves the problem that the ACO algorithm easily falls into deadlock in a complex environment. Yu et al. [38] proposed a D*Lite-GWO algorithm and

proved its strong applicability and effectiveness. And Omar et al. [39], [40] proposed an ACPSO combining A* algorithm heuristic function with PSO.

Although the single A* algorithm has the advantage of fast computing speed, it also has the disadvantages of too many turning points, redundant nodes and low path accuracy [41]. In contrast, although the path searched by a single PSO in the static path planning problem has high accuracy, it is often difficult to initialize the population in a complex environment, resulting in low efficiency of the algorithm. Previous studies on the A* algorithm in static scenes mostly focused on the optimization of its heuristic function or path smoothness [42], [43], [44], [45], [46], while studies on particle swarm optimization mainly focused on the optimization of its flight mode, inertia weight and other aspects [47], [48], [49], [50], [51], [52] and rarely combined the two algorithms to optimize each other.

Based on this, this paper combines the A* algorithm with PSO, named APSO. First, the A* algorithm was used to calculate the initial path, a redundant point removal strategy was used to extract the key nodes in the initial path, and the key nodes on the path were used as the initial particles of the PSO. Then, the PSO combined with random inertia weight and random opposition-based learning strategy was used to obtain the global optimal path. This not only makes up for the low path accuracy of the A* algorithm but also solves the problem that PSO has difficulty initializing the population in the static path planning problem. Finally, a more comprehensive objective function is adopted to evaluate the effect of the algorithm when modeling the path planning problem of mobile robots.

The structure of this paper is as follows: Section II briefly describes the A* algorithm and the PSO algorithm; Section III introduces the specific content of APSO in detail; and Section IV shows the optimization results of stochastic inertia weight and stochastic opposition-based learning strategy in APSO in classical benchmark functions and the statistical test analysis results. In Section V, after modeling the path planning problem of a mobile robot, APSO is applied to it, and the result of path planning is analyzed. Section VI summarizes the thesis and looks forward to the next step.

II. BASIC A* ALGORITHM AND PSO

A. A* ALGORITHM

The A* algorithm combines the Dijkstra algorithm and the BFS algorithm, which is a global search heuristic algorithm with a wide range of application scenarios. Because of its high search efficiency and strong robustness, it is often used in path planning of mobile robots.

The steps of the A* algorithm are as follows: first, the starting node is taken as the first parent node to traverse its surrounding child nodes, and then the $f(n)$ value of the surrounding child nodes is calculated. The child node with the smallest $f(n)$ value is taken as the next parent node and put into *openList* until the target node is included in *openList*

to stop the search. The cost estimation function formula is shown as:

$$f(n) = g(n) + h(n), \quad (1)$$

where $f(n)$ is the estimated cost of reaching the target node from the starting node through the current node n ; $g(n)$ is the actual cost from the starting node to the current node n ; and $h(n)$ is the estimated cost from the current node n to the target node, that is, the heuristic function of the algorithm, which is usually represented by Manhattan distance or Euclidean distance. The Manhattan distance is the sum of the absolute values of the horizontal and vertical coordinates between two nodes, and the Euclidean distance is the straight-line distance between two nodes. Because the Euclidean distance has higher accuracy than the Manhattan distance, Euclidean distance is often used as the heuristic function when using the A* algorithm. In this paper, the eight-way search method is adopted in the raster map, and the map contains obstacles, so the Euclidean distance is used as the heuristic function of the algorithm. $g(n)$ and $h(n)$ can be shown as:

$$g(n) = \sqrt{(x_s - x_n)^2 + (y_s - y_n)^2}, \quad (2)$$

$$h(n) = \sqrt{(x_n - x_t)^2 + (y_n - y_t)^2}, \quad (3)$$

where x_s and y_s are the abscissa and ordinate of the starting node, respectively, and x_n and y_n are the abscissa and ordinate of the current node, respectively. x_t and y_t denote the abscissa and ordinate of the target node, respectively.

B. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) was proposed by Kennedy and Eberhart [24] and was inspired by the foraging behavior of birds. PSO simulates the process of a particle swarm moving toward the optimal solution in the multidimensional search space, where each particle represents a candidate solution and the initial particle is randomly generated [53]. Due to its simple steps and fast convergence speed, the algorithm is widely used in many practical scenarios, such as robot path planning [54], nonlinear optimization of power systems [55], medical image classification [56], and flow shop scheduling [57].

The PSO algorithm first initializes a certain number of random particle populations as available solutions and then searches for the optimal solution by iterating and updating the particle swarm. In the iteration process, each particle updates its speed and position through its own optimal solution ($pbest$) and the group optimal solution ($gbest$). The updating process of its speed and position is shown as follows:

$$v_i^{k+1} = \omega v_i^k + c_1 r_1 (pbest_i^k - x_i^k) + c_2 r_2 (gbest^k - x_i^k), \quad (4)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1}, \quad (5)$$

where v_i^{k+1} is the velocity of the i -th particle in the $k + 1$ -th iteration; ω stands for inertia weight; c_1 , the self-learning factor, represents the weight of a bird flying to its

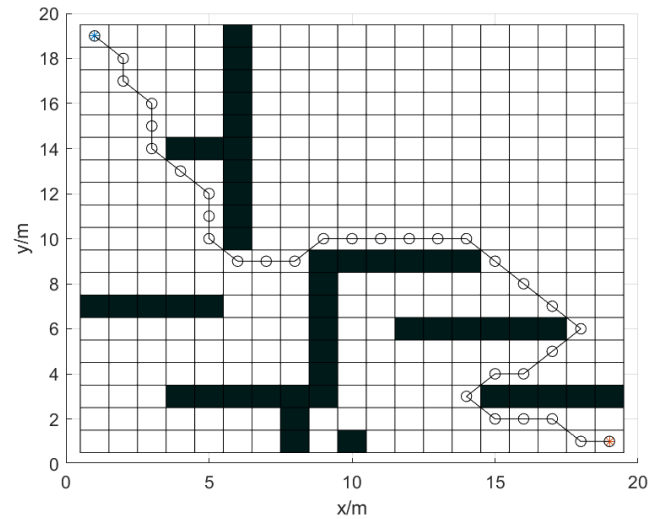


FIGURE 1. A* algorithm example.

optimal position, that is, the greater the value of c_1 , the stronger the bird's willingness to fly to its optimal position. c_2 , the social learning factor, is the weight of a bird flying to the optimal position that the group has been to, and the greater the c_2 value, the greater the bird's willingness to fly to the optimal position of the group. r_1 and r_2 are random numbers uniformly distributed in the interval $(0,1)$; $pbest_i^k$ is the optimal solution of the i -th particle searched after the k -th iteration. $gbest^k$ is the group optimal solution after the k -th iteration. x_i^k denotes the position of the i -th particle after the k -th iteration.

III. APSO ALGORITHM

A. THE IMPROVEMENTS OF A* ALGORITHM

In the application scenarios of mobile robot path planning, although the A* algorithm can efficiently provide the lowest cost path, it also has obvious limitations. Fig. 1 shows the path node diagram planned by algorithm A*.

As seen from the figure, the path provided by the algorithm often has the problems of too many nodes, too many turning points, and poor path smoothness, which does not meet the realistic motion requirements of mobile robots. In view of the problem of redundant path nodes and redundant transition nodes in the path planned by the A* algorithm, we adopted a redundant node removal strategy to solve the problem.

1) DELETE REDUNDANT PATH NODES

As shown in Fig. 2, solid points are the starting nodes and target nodes of the path, and hollow points are other nodes on the path. Fig. 2-(a) shows the path before deleting the redundant path node, and Fig. 2-(b) shows the path after deleting the redundant path node.

If the current node is on a line with the previous node and the next node of the current node, it means that the node is a redundant path node, and then the current node will be deleted. For example, node An in Fig. 2-(a) is collinear with

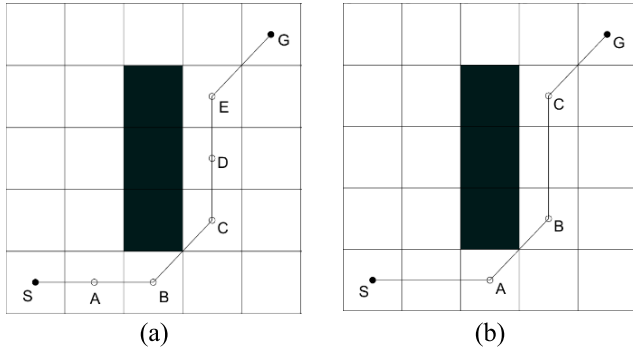


FIGURE 2. Comparison before and after deleting redundant path nodes.

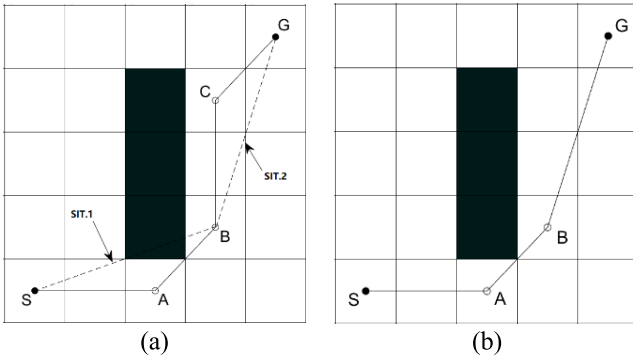


FIGURE 3. Comparison before and after deleting redundant transition nodes.

the previous node S and the next node B; then, node A will be deleted. The judgment function of the redundant path node is expressed as:

$$fr = \frac{y_n - y_{n-1}}{x_n - x_{n-1}}, \quad (6)$$

$$af = \frac{y_{n+1} - y_n}{x_{n+1} - x_n}, \quad (7)$$

where x_n and y_n are the abscissa and ordinate of the current node n . x_{n-1} and y_{n-1} are the abscissa and ordinate of the previous node $n - 1$. x_{n+1} and y_{n+1} are the abscissa and ordinate of the next node $n + 1$. If $fr = af$, the current node is collinear with the previous node and the next node; that is, the current node is a redundant path node. If $fr \neq af$, the current node is not collinear with the previous node and the next node; that is, the current node is not a redundant path node. The path after deleting the redundant path node is shown in Fig. 2-(b).

2) DELETE REDUNDANT TRANSITION NODES

After eliminating the redundant path nodes, except the starting point and the ending point, the remaining nodes on the path are all transition nodes. There are two situations in Fig. 3-(a) for the connection between two separated turning points: in SIT.1, there is an obstacle between two separated turning points S and B. For this situation, the intermediate turning point A between two separated turning points is reserved, and whether there is an obstacle between turning

points A and C is judged. In SIT.2, there is no obstacle between two separated turning points B and G. For this situation, the redundant turning point C between B and G should be deleted. For redundant transition nodes, the following implicit function is used to judge:

$$path = f(x, y) = \left(\frac{y - y_{n-1}}{y_{n+1} - y_{n-1}} \right) - \left(\frac{x - x_{n-1}}{x_{n+1} - x_{n-1}} \right), \quad (8)$$

$$obs = f(x, y) = (x - x_{obs})^2 + (y - y_{obs})^2 - r_{obs}^2, \quad (9)$$

where x_{n-1} and y_{n-1} represent the abscissa and ordinate of the previous node $n - 1$ of the current node, respectively; x_{n+1} and y_{n+1} represent the abscissa and ordinate of the next node $n + 1$ of the current node, respectively; x_{obs} and y_{obs} represent the abscissa and ordinate of each obstacle, respectively; and r_{obs} is the area radius of each obstacle after expansion. When $path = obs$, there are obstacles between the two turning points; otherwise, there are no obstacles between the two turning points, that is, the intermediate turning points between the two turning points are redundant turning points. The path after deleting the redundant turning point is shown in Fig. 3-(b).

B. THE IMPROVEMENTS OF PSO

1) STOCHASTIC INERTIA WEIGHT

The inertia weight ω is the core parameter for particle velocity and position update in PSO, which has an important impact on the convergence performance of the algorithm [58]. To improve the performance of the PSO algorithm, the adaptive inertia weight (AIW) [58] is often used, as shown in Eq. nonlinear decreasing inertia weight (NDIW) [59], as shown in Eq. 11; linearly decreasing inertia weight (LDIW) [60], as shown in Eq. 12, etc.

$$\omega = (\omega_{max} + \omega_{min}) \times \left(\frac{p_{gbest}^k}{p_{gbest}^{k-1}} \right) - \frac{w_{max} \times k}{T_{max}}, \quad (10)$$

$$\omega = \exp \left[- \exp \left(\frac{T_{max} - k}{T_{max}} \right) \right], \quad (11)$$

$$\omega = \omega_{max} - (\omega_{max} - \omega_{min}) \times \left(\frac{k}{T_{max}} \right), \quad (12)$$

where ω_{min} is the minimum value of inertia weight, often with a value of 0.4; w_{max} is the maximum value of inertia weight, often with a value of 0.9; p_{gbest}^k is the global optimal fitness at the k -th iteration; p_{gbest}^{k-1} is the global optimal fitness at the $k - 1$ -th iteration; c is a constant; k is the current iteration number; and T_{max} is the maximum iteration number.

If the inertia weight is set as a random number obeying a certain distribution, adjusting the inertia weight by using the characteristics of random variables can make the algorithm jump out of the local optimum quickly, which is conducive to maintaining the diversity of the population and improving the global search performance of the algorithm [61]. Therefore, based on the widely used linear decreasing inertia weight, this paper proposes a stochastic inertia weight (SIW), whose

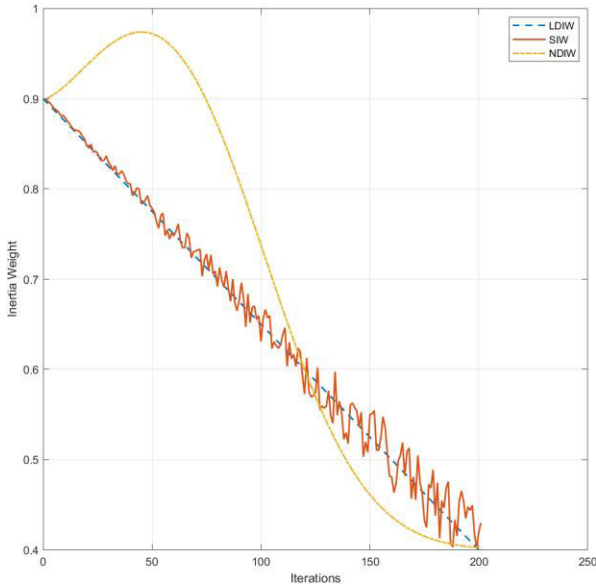


FIGURE 4. Inertia weight curve comparison.

formula is shown as:

$$\omega = \omega_{max} - (\omega_{max} - \omega_{min}) \times \left(\frac{k}{T_{max}} \right) \times e^{r_b}, \quad (13)$$

where r_b is a random number in the interval range $(-0.1, 0.1)$. A comparison of the inertia weight change curves is shown in Fig. 4.

2) STOCHASTIC OPPOSITION-BASED LEARNING

Opposition-based learning (OBL), proposed by Tizhoosh et al. [62], is an optimization tool with an excellent optimization effect. The main idea is to compare the fitness of the current feasible solution and its reverse feasible solution and select the better solution of the two to enter the subsequent iteration. The opposition-based learning strategy has been widely used to improve the convergence speed of swarm intelligence algorithms [63], [64]. The basic definition of opposition-based learning is as follows:

Definition 1: Opposite Numbers. Let x be a real number and $x \in [lb, ub]$; the opposite number \tilde{x} obtained after opposition-based learning of x is defined as:

$$\tilde{x} = lb + ub - x. \quad (14)$$

Definition 2: Opposite Points. Let point $P(x_1, x_2, \dots, x_n)$ be a point in n -dimensional coordinates, where x_1, x_2, \dots, x_n are real numbers and $x_1, x_2, \dots, x_n \in [lb_i, ub_i]$. The opposite point \tilde{P} can be defined as:

$$\tilde{x}_i = lb_i + ub_i - x_i. \quad (15)$$

The introduction of the OBL strategy into PSO is helpful to expand the search range of the algorithm by diversifying particles and improving the performance of the algorithm. If each particle uses the OBL strategy after every position change, the time complexity of the algorithm will be greatly

improved, and the algorithm search efficiency will decrease. Therefore, OBL can be applied to the individual and global optimal particles in PSO at the end of each iteration, which can not only expand the search scope of the algorithm but also not greatly improve the time complexity of the algorithm. According to the definition of OBL, a point has and only has one opposite point. However, once the individual most particle or global optimal particle falls into the local optimum and the fitness of its opposite solution is not as good as the original feasible solution, then the implementation of the OBL strategy will be invalid every time. To avoid this situation, we propose a stochastic opposition-based learning strategy (SOBL) to prevent the failure of the OBL strategy when the global optimal particle falls into the local optimum. SOBL is defined as follows:

Definition 3: Stochastic Opposite Points On the basis of the opposite points, a stochastic perturbation is taken to x_i :

$$\tilde{x}_i = lb_i + ub_i - x_i \times r, \quad (16)$$

where r is the random value conforming to a Gaussian distribution in the interval $(0, 1)$.

C. THE OPTIMIZATION PROCESS OF APSO

We take the route planned by the improved A* algorithm as the route to be optimized. The improved PSO will continue to optimize the route locally from the starting node until the target node enters the iteration.

First, the initial particles with a certain population size are randomly generated according to the random strategy, which can be shown as:

$$posx = x_n + ((x_{n+2} - x_n) \times r), \quad (17)$$

$$posy = y_n + ((y_{n+2} - y_n) \times r), \quad (18)$$

where $posx$ and $posy$ represent the x and y coordinates of randomly generated particles, respectively, and x_n and y_n represent the x and y coordinates of node n , respectively. x_{n+2} and y_{n+2} represent the x and y coordinates of nodes separated from node n , respectively. r stands for a random number in the interval $(0, 1)$.

After the population is initialized, the velocity of each particle in the population is initialized again by a random strategy, which can be shown as:

$$v_x = v_{max} \times r, \quad (19)$$

$$v_y = v_{max} \times r, \quad (20)$$

where v_x and v_y are the velocity of each particle on the x coordinate and the flying speed on the y coordinate, respectively; v_{max} indicates the maximum flight speed; and r stands for a random number in the interval $(0, 1)$.

After the population and population velocity are initialized, the fitness of each particle is calculated. The particle with the highest fitness is regarded as the group optimal solution $gbest$, the particle without obstacle avoidance has a fitness value of 0, and the other particles without a fitness value

of 0 are regarded as the individual optimal solution $pbest$. The fitness calculation is shown as:

$$f = d \times c, \quad (21)$$

$$d = \sum_{m=1}^n \sqrt{(x_{m+1} - x_m)^2 + (y_{m+1} - y_m)^2}, \quad (22)$$

$$c = \begin{cases} 1, & path \neq obs \\ 0, & path = obs \end{cases}, \quad (23)$$

where f is the particle fitness value; d is the particle path distance; c is the collision test function; and x_m and y_m represent the abscissa and ordinate of node m , respectively. x_{m+1} and y_{m+1} represent the abscissa and ordinate of adjacent nodes of node m , respectively. When $m = 1$, node m is the starting node of the local path, and when $m = n$, node m is the target node of the local path. $path$ and obs are the judgment functions in Equations (8) and (9), respectively.

After the particle swarm initialization is completed, the particle velocity and position are updated continuously through iteration to achieve the goal of continuous particle optimization. The velocity and position update formulas are expressed as:

$$v_i^{k+1} = \omega v_i^k + c_1 r_1 (pbest_i^k - x_i^k) + c_2 r_2 (gbest^k - x_i^k), \quad (24)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1}. \quad (25)$$

After the position movement of the particle swarm is completed, SOBL is applied to all individual optimal particles and global optimal particles in the next iteration, and their opposite positions are obtained. Then, the fitness of the opposite position of each particle is calculated. If the fitness of the current position of the particle is inferior to the fitness of the reverse position, the particle is moved to the opposite position; if the fitness of the current position of the particle is superior to the fitness of the opposite position, the current position of the particle is retained. The process for using SOBL on particles is as follows:

$$x_i^k = \begin{cases} x_i^k, & f(x_i^k) \leq f(\tilde{x}_i^k) \\ \tilde{x}_i^k, & f(x_i^k) > f(\tilde{x}_i^k) \end{cases} \quad (26)$$

Taking path S-A-B-G in Fig. 5-(a) as an example, node S and node B are first regarded as the starting node and the target node of local path S-A-B, respectively, so the search for a more suitable intermediate node A is the goal of this optimization. As shown in Fig. 5-(b), after the improved PSO is used to find the optimal node A, node A and node G are taken as the starting node and target node of local path A-B-G, and the improved PSO is used to optimize the local path A-B-G. After the optimization of the local path A-B-G is completed, the target node G of the local path is the same node as that of the global path, so the algorithm ends. The path optimized by APSO is finally shown in Fig. 5-(c).

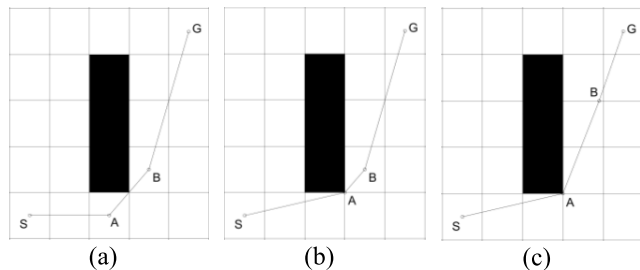


FIGURE 5. The example of APSO optimization process.

D. ALGORITHM IMPLEMENTATION PROCESS

STEP 1: The raster map is established by the raster method, and the parameters of the A* algorithm are initialized;

STEP 2: Create two empty sets, $openList$ and $closeList$. $openList$ is used to store the nodes to be searched, and $closeList$ is used to store the searched nodes. Add the starting node to $openList$;

STEP 3: Traverse the nodes in $openList$, search for the node with the smallest $f(n)$ value and set it as the current node n ;

STEP 4: Delete the current node n from $openList$ and add it to $closeList$;

STEP 5: Judge whether node n is the target node. If it is the target node, trace back to find the initial path and skip to step (7); if it is not the target node, the current node n is used as the parent node to search the surrounding eight neighborhoods.

STEP 6: Judge whether the points in the eight neighbors of node N that are not obstacles are in $openList$. If so, calculate the $f(n)$ value of these nodes, select the node with the smallest $f(n)$ value as the next node $n + 1$, and return to step 4. If it is not in $openList$, add it to $openList$ and calculate $f(n)$;

STEP 7: The redundant path nodes and redundant transition points in the node set $pathPoints$ in the initial path are removed by the redundant node removal strategy;

STEP 8: Traverse the remaining nodes in the $pathPoints$ set, and the separated nodes n and $n+2$ are used as the starting point and ending point of the PSO optimization path in each iteration.

STEP 9: To initialize the PSO parameters, input the correlation coefficients of inertia weights w_{max} and w_{min} , the self-learning factor c_1 , the social learning factor c_2 , the population number $possiz$, and the maximum number of iterations T_{max} , and enter the iteration.

STEP 10: Each particle in the population is disturbed according to the individual optimal $pbest$ and the overall optimal $gbest$ to generate a new historical optimal population;

STEP 11: Calculate the fitness value of the individual optimal $pbest$ and the global optimal $gbest$ and compare them with the fitness value of the reverse position. If the fitness value of the particle in the reverse position is better, the particle will be moved to its reverse position; otherwise, the particle will not be moved.

Algorithm 1 Pseudo-code of APSO

```

1: Input the information of start node, target node and environment map
2: Create grid map
3: Create a collection to store the search node
4: Set the minimum node of  $f(n)$  as the current node  $n$ 
5: while The current node is not the target node do
6:   Search for eight-neighbor of current nodes
7:   if The eight-neighbor points are in openList then
8:     Calculate the  $f(n)$  of these points and select the minimum as the current node
9:   else if The eight-neighbor points are not in openList then
10:    Add them in openList
11:    Calculate the  $f(n)$  of these points and select the minimum as the current node
12:   end if
13: end while
14: Trace to find the initial path and place the path node in pathPoints
15: Delete non-start and non-destination nodes on local paths in pathPoints
16: Delete the intermediate node when obstacles exist between the separated nodes on pathPoints
17: while The target node  $n$  does not enter the iteration do
18:    $n = n + 1$ 
19:   Take  $n$  and  $n+2$  isolated nodes in pathPoints as starting and destination nodes for local path optimization
20:   Initialize the PSO parameters
21:   while  $t \leq \text{maxIter}$  do
22:      $t = t + 1$ 
23:     Calculate the objective function values  $F$  of each particle
24:     Update individual and group optimal values
25:     Calculate the objective function values  $F_o$  of each particle's opposite position
26:     if  $F_o \leq F$  then
27:       Move the particle to its opposite position
28:     end if
29:     Update the velocity and position of each particle
30:   end while
31:   Output a global optimization path composed of local optimization paths
32: end while
33: Output nodes of the final route

```

FIGURE 6. Inertia weight curve comparison.

STEP 12: Judge whether the iteration number T_{max} is reached; if so, terminate the iteration and save the local optimization path; if not, return to STEP 11.

STEP 13: Judge whether the target node enters the iteration; if so, end the algorithm and output the overall path composed of each local path. If not, return to STEP 8. The algorithm pseudocode is shown in Fig. 6:

IV. THE PERFORMANCE EVALUATION EXPERIMENT OF THE IMPROVED PSO

To test the improvement degree of the PSO by the two strategies of stochastic inertia weight and stochastic opposition-based learning adopted in APSO, 15 benchmark functions are introduced in this section, and these 15 benchmark functions are popular problems used in the optimization literature [65], [66], [67], [68], [69]. The basic PSO is denoted as PSO, and the improved PSO that adopts stochastic inertia weight and stochastic opposition-based learning strategy is denoted as OBLPSO. In addition, for a more intuitive comparison, we reproduced the differential evolution algorithm (DE) of Storn et al. [22], the gravity search algorithm (GSA) of Esmat Rashedi et al. [70], the biogeography algorithm (BBO) of Simon [71], the Salp swarm algorithm (SSA) of Esmat Rashedi et al. [72], the sine and

cosine algorithm (SCA) of Mirjalili [73], and the gray wolf optimization (GWO) of Mirjalili et al. [74]. Tab. 1 shows the hyperparameters of all the comparison algorithms. After analyzing the optimization results of eight algorithms in the benchmark functions, the optimization ability of the proposed OBLPSO is evaluated. The experiments are conducted using a computer Core i7-1165G7 with 16 GB RAM and 64-bit for Microsoft Windows 11. The source code is implemented using MATLAB (R2021b).

The details of the 15 benchmark functions are shown in Tab. 2, where $f_1 - f_7$ are unimodal functions, which are used to test the global search ability of the algorithm in the case of high dimensions. $f_8 - f_{11}$ are multimodal functions, which are used to test the local search ability and the ability of the algorithm to jump out of the local optimum in the case of high dimensions. $f_{12} - f_{15}$ are dimension fixed functions used to test the ability of the algorithm to solve simple problems.

Tab. 3 shows the comparison results between OBLPSO and other algorithms after 20 independent runs in the benchmark function, where the data in bold are the optimal values in each data.

Both the average value and the best value can reflect the optimization accuracy of the algorithm, while the standard deviation value can reflect the stability of the algorithm in

TABLE 1. Hyperparameters of algorithms.

Hyperparameter	PSO	OBLPSO	DE	GSA	BBO	SSA	SCA	GWO
Population size	50	50	50	50	50	50	50	50
c_1 and c_2	2	2	-	-	-	-	-	-
Inertia weight ω	0.9-0.4	0.9-0.4	-	-	-	-	-	-
Crossover rate	-	-	0.2	-	-	-	-	-
Mutation scale	-	-	0.8-0.2	-	0.1	-	-	-
Power of R	-	-	-	1	-	-	-	-
Norm of R	-	-	-	2	-	-	-	-
Keep rate	-	-	-	-	0.2	-	-	-
Migration value α	-	-	-	-	0.9	-	-	-

TABLE 2. Benchmark functions.

Function	Formulation	Dim	Search range	f_{min}
Sphere	$f_1(X) = \sum_{i=1}^D x_i^2$	30	[-100,100]	0
Schwefel 2.22	$f_2(X) = \sum_{i=1}^D x_i + \prod_{i=1}^D x_i $	30	[-10,10]	0
Schwefel 12	$f_3(X) = \sum_{i=1}^D (\sum_{j=1}^D x_j)^2$	30	[-100,100]	0
Schwefel 2.21	$f_4(X) = \max\{ x_i , 1 \leq x_i \leq D\}$	30	[-100,100]	0
Rosenbrok	$f_5(X) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	0
Step	$f_6(X) = \sum_{i=1}^D ([x_i + 0.5])^2$	30	[-100,100]	0
Quartic	$f_7(X) = \sum_{i=1}^D ix_i^4$	30	[-1.28,1.28]	0
Rastrigin	$f_8(X) = \sum_{i=1}^D [x_i^2 - 10 \cos(2\pi x_i) + 10]$	30	[-5.12,5.12]	0
Ackley	$f_9(X) = \sum_{i=1}^D -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}\right) - \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)\right) + 20 + e$	30	[-32,32]	0
Griewank	$f_{10}(X) = \frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600,600]	0
Kowalik	$f_{11}(X) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	4	[-5,5]	0.00030
Six Hump Camel	$f_{12}(X) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[-5,5]	-1.0316
Branin	$f_{13}(X) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	[-5,5]	0.398
GoldStein-Price	$f_{14}(X) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[-2,2]	3
Hartman 6	$f_{15}(X) = -\sum_{i=1}^4 c_i \exp\left(-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2\right)$	6	[0,1]	-3.32

the search process. Overall, the average value, best value and standard deviation of OBLPSO in $f_1 - f_4$ and $f_7 - f_{15}$ after 20 independent runs were better than those of the other algorithms. OBLPSO can simultaneously search for the

theoretical optimal value in $f_1 - f_4, f_8 - f_{10}$ and $f_{12} - f_{15}$. As seen from the convergence curve in Fig. 7, OBLPSO shows the fastest convergence speed among the 15 benchmark functions and has higher search accuracy compared

TABLE 3. Comparison of experimental results under the benchmark functions.

		OBLPSO	PSO	DE	GSA	BBO	SSA	SCA	GWO
f_1	Mean	0	59.98492	1556.199	1464.338	11.00474	487.7355	2195.277	0.000889
	Best	0	27.07148	1174.521	658.0592	6.60988	182.4681	245.2675	0.000202
	Std	0	27.47243	324.2394	387.2942	2.438866	222.1886	1383.687	0.00062
f_2	Mean	0	25.38466	14.92852	9.15086	0.994957	11.31854	4.820405	0.005049
	Best	0	2.22172	12.09259	4.345062	0.727204	6.322514	1.522111	0.001838
	Std	0	27.66012	1.492589	2.298738	0.132853	3.143674	2.824907	0.00166
f_3	Mean	0	5291.085	51054.78	3170.572	8314.926	3842.14	20159.91	83.16757
	Best	0	2647.364	37326.27	1576.297	4041.166	1456.762	11603.23	7.102674
	Std	0	2275.654	6506.998	1180.897	2656.892	1371.662	6152.922	72.18655
f_4	Mean	0	5291.085	51054.78	3170.572	8314.926	3842.14	20159.91	83.16757
	Best	0	2647.364	37326.27	1576.297	4041.166	1456.762	11603.23	7.102674
	Std	0	2275.654	6506.998	1180.897	2656.892	1371.662	6152.922	72.18655
f_5	Mean	28.76884	3492.43	679708.6	73216.01	400.8734	25312.22	8330735	28.67001
	Best	28.22667	895.5643	330671.4	11734.38	198.8075	3491.892	1010486	27.23888
	Std	0.185271	2449.992	308761.3	43950.92	174.8286	18014.33	10424156	0.623422
f_6	Mean	2.597954	52.08739	1533.473	1549.502	10.32633	406.8296	3038.364	1.767749
	Best	1.369457	19.438	1049.989	876.0261	4.860053	130.0141	318.5764	0.785553
	Std	0.467535	24.27206	315.7947	348.6209	2.472663	152.5397	2989.365	0.584952
f_7	Mean	0.00021	0.10738	0.815291	0.669622	0.037774	0.311646	4.480562	0.013324
	Best	4.73E-06	0.058504	0.612142	0.218124	0.013258	0.09637	0.211172	0.00716
	Std	0.000165	0.039645	0.127169	0.340633	0.01556	0.147642	6.893648	0.004871
f_8	Mean	0	131.381	180.3758	62.96596	38.32645	65.64815	140.0432	28.52005
	Best	0	90.6863	159.1573	22.61029	19.94158	45.31275	39.93395	13.19636
	Std	0	26.83688	8.803982	21.51963	10.2807	11.32027	44.70859	9.620302
f_9	Mean	8.88E-16	3.309219	10.02752	6.271073	1.488909	6.652548	16.16813	0.006244
	Best	8.88E-16	2.632204	8.762817	5.09957	1.049942	5.075771	5.609717	0.003355
	Std	0	0.392204	0.757625	0.760892	0.223235	0.906416	4.711604	0.001767
f_{10}	Mean	0	1.432526	14.8766	393.8569	1.098906	4.558771	22.51232	0.027311
	Best	0	1.121902	9.787167	294.2885	1.052336	2.514127	2.628546	0.000555
	Std	0	0.204225	2.59239	44.90672	0.025311	1.250413	14.20406	0.032823
f_{11}	Mean	0.002869	0.004796	0.00121	0.014169	0.003943	0.002395	0.001212	0.008012
	Best	0.000307	0.000389	0.000768	0.002995	0.00074	0.000345	0.000549	0.000484
	Std	0.005848	0.007792	0.000289	0.008271	0.005631	0.004308	0.000422	0.009243
f_{12}	Mean	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	-1.03149	-1.03163
	Best	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	-1.03162	-1.03163
	Std	2.18E-15	3.95E-09	3.23E-10	1.38E-05	2.54E-09	1.03E-13	0.000113	4.49E-07
f_{13}	Mean	0.397887	0.397887	0.397887	0.397887	0.397887	0.397887	0.403509	0.39989
	Best	0.397887	0.397887	0.397887	0.397887	0.397887	0.397887	0.398056	0.397888
	Std	1.11E-16	1.54E-15	3.74E-08	2E-11	8.75E-08	2.1E-13	0.008223	0.008586
f_{14}	Mean	3	3	3	3.076071	5.700069	3	3.000488	3.000281
	Best	3	3	3	3	3	3	3.000032	3.000001
	Std	1.26E-14	5.48E-14	1.75E-13	0.331587	8.100206	1.19E-12	0.000508	0.00039
f_{15}	Mean	-3.31846	-3.28038	-3.28633	-2.62726	-3.28633	-3.21538	-2.90322	-3.27495
	Best	-3.322	-3.322	-3.32199	-3.322	-3.322	-3.322	-3.13749	-3.32196
	Std	0.054484	0.056708	0.005913	0.636738	0.054484	0.087971	0.164791	0.076293

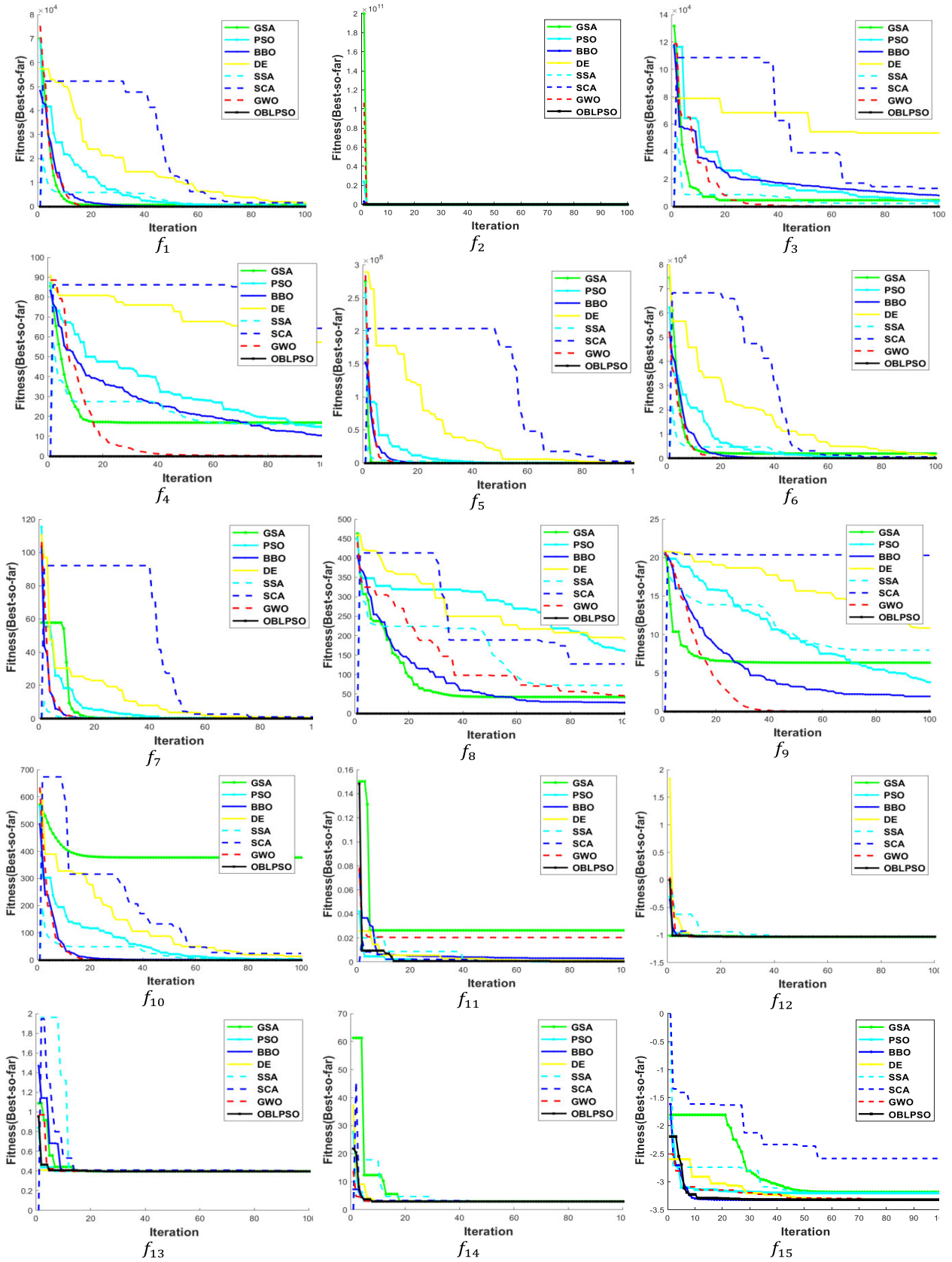


FIGURE 7. Comparison of convergence curves.

TABLE 4. Wilcoxon rank sum test under the benchmark function.

OBLPSO vs.	PSO	DE	GSA	BBO	SSA	SCA	GWO
f_1	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05
f_2	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05
f_3	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05
f_4	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05
f_5	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	0.881293
f_6	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	0.000338
f_7	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05
f_8	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05
f_9	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05
f_{10}	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05
f_{11}	0.295835	0.370261	0.001162	0.022769	0.135357	0.654159	0.331723
f_{12}	0.25	0.25	0.875	0.625	0.25	8.86E-05	8.86E-05
f_{13}	1	0.0625	1	0.000244	1	8.86E-05	8.86E-05
f_{14}	1	1	0.5	0.00013	1	4.01E-05	4.01E-05
f_{15}	0.262722	1	0.000517	0.092963	0.009996	8.86E-05	0.079322
+/-/=	10/0/5	10/0/5	12/0/3	13/0/2	11/0/4	14/0/1	11/1/3

with other algorithms. Therefore, OBLPSO not only outperforms basic PSO in terms of optimization accuracy and stability but also outperforms some other metaheuristic algorithms.

The above data analysis only visually describes the performance of the algorithm from the optimal value, average value, and standard difference. Although the performance differences between some algorithms can be seen, the analysis process may not be comprehensive, so it is difficult to explain the optimal performance of the algorithm. Therefore, we can conduct certain statistical analyses on the data, evaluate the performance of the algorithm from the statistical perspective and judge whether there are significant differences between the algorithms. First, the Wilcoxon rank sum test [75] was used to calculate the P -value for significance analysis. Then, the Friedman statistical test was used to derive the algorithm ranking and comprehensive evaluation.

Tab. 4 shows the comparison results of each algorithm and the Wilcoxon rank sum test of OBLPSO under the benchmark test function at the significance level of $\alpha = 0.05$. If the test result $P < 0.05$, it indicates that there is a significant difference between the algorithm results; otherwise, there is no significant difference. “+/-/=” in Tab. 3 means that OBLPSO is more “superior/inferior/comparable” compared with the performance of the algorithm. In Tab. 3, the four functions $f_{12} - f_{15}$ are dimension fixed functions, and the problem is relatively simple, so a certain number of “1” and P -values less than 0.05 are presented in the results. In addition, most of the P -values of the Wilcoxon rank sum test after comparing OBLPSO with other algorithms are less than 0.05, indicating that OBLPSO has a significant difference compared with other algorithms on the whole.

To further analyze the performance of each algorithm, the Friedman test was used to rank the algorithms. The ranking formula can be shown as:

$$R_j = \frac{1}{N} \sum_{i=1}^N r_i^j, \quad (27)$$

where i represents the i -th function, j represents the j -th algorithm, N is the number of test functions, and r_i^j is the ranking of algorithm j in function i . A smaller value of R_j indicates a higher ranking algorithm and better performance. It can be seen from Tab. 5 that OBLPSO, which adopts stochastic inertia weight and stochastic opposition-based learning, ranks first in the experiment, which is sufficient to show that OBLPSO has strong search performance.

In summary, OBLPSO can show good performance in a variety of test functions, with advantages, stability and effectiveness.

V. PATH PLANNING SIMULATION EXPERIMENT

To test the feasibility and effectiveness of APSO in the path planning scenario of mobile robots, this paper introduces A*, PRM, RRT and Bi-RRT and the proposed APSO algorithm to carry out a simulation experiment of mobile robot static path planning. Due to the dense obstacles in the environment map, we found that it was difficult for a single intelligent optimization algorithm to initialize an effective path population in the experiment, so we did not introduce the relevant path planning algorithm for comparison. Each algorithm is set to run independently 20 times in each environment map. The inertia weight correlation coefficients in APSO are $\omega_{max} = 0.9$, $\omega_{min} = 0.4$, self-learning factor $c_1 = 2$, social learning

TABLE 5. Ranking of the Friedman test algorithm under the benchmark function.

Algorithm	OBLPSO	PSO	DE	GSA	BBO	SSA	SCA	GWO
mean	1.333333	4.066667	5.2	5.2	3.533333	4.133333	6.4	2.866667
rank	1	4	6	6	3	5	8	2

factor $c_2 = 2$, population $possiz = 40$, and maximum number of iterations $T_{max} = 200$. The experiment also uses MATLAB R2021b software to run on a Lenovo X1 Carbon Gen 9 computer Windows11 system, in which the computer hardware is an Intel Core i7 processor and 16G memory.

A. PROBLEM DESCRIPTION

The environment of path planning is set in the intelligent warehouse. After the mobile robot obtains the task, it starts from the starting point and arrives at the target point after obstacle avoidance to complete the task. The path planning process requires the shortest path length and minimum total turning range.

The following assumptions are made for the environment map:

- (1) Assuming that the environment map is a two-dimensional map, the height information of mobile robots and obstacles can be ignored;
- (2) Assume that the position and area of obstacles are determined and do not change;
- (3) Assume that the mobile robot is a particle and moves at a constant speed.

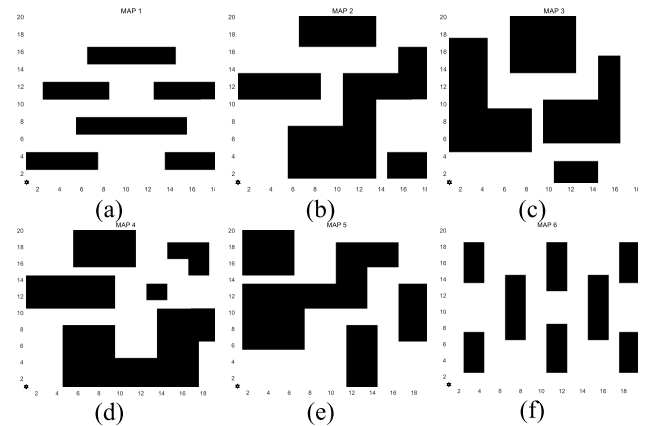
B. MODEL CONSTRUCTION

1) ENVIRONMENT MODELING

The common environment modeling methods of path planning include the raster method, viewable method, topology diagram method and so on. In this paper, the raster method is adopted to create the environment map model, and a two-dimensional raster map of 20×20 is adopted. Fig. 8 lists three types of environment map models created by the raster method. Among them, the obstacle coverage rate of environment map 1 in Fig. 8-(a) is 22%, that of environment map 2 in Fig. 8-(b) is 41.75%, that of environment map 3 in Fig. 8-(c) is 45.75%, that of environment map 4 in Fig. 8-(d) is 48.50%, that of environment map 5 in Fig. 8-(e) is 45.25% and that of environment map 6 in Fig. 8-(f) is 24%. The black area is the area where the obstacle is located, and the white area is the driving area. Set the start node of mobile robot movement as (1,1) and the target node as (20,20).

2) PROBLEM MODELING

In the path planning problem of mobile robots, the primary goal is to reduce the overall running time of mobile robots. In most studies, only the path length is used as the objective function to measure the effect of the path planning algorithm. However, in reality, the factors that affect the running time of mobile robots are not only the path length but also the turning

**FIGURE 8.** Obstacle environment map.

amplitude, the number of nodes in the path and the operation time of the path planning algorithm. Therefore, we set these factors into the following objective functions:

a: SHORTEST PATH LENGTH OBJECTIVE FUNCTION

The goal of the path planning algorithm is to minimize the path length of the mobile robot, and the path length can directly affect the linear movement time of the mobile robot. Therefore, the shortest path length objective function should be set, and its formula is shown as:

$$\min D(x, y) = \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, \quad (28)$$

where n is the number of nodes and x_i and y_i are the abscissa and ordinate of node i , respectively. x_{i+1} and y_{i+1} are the abscissa and ordinate of the next node of node i , respectively.

b: PATH NODE MINIMUM OBJECTIVE FUNCTION

When the mobile robot moves on the path, it will judge the direction of the next movement every time it reaches a node. The more motion nodes there are, the more judgment times of the mobile robot will increase, thus affecting its motion fluency. Therefore, the minimum objective function of the path node should be set, and its formula is:

$$\min P(n) = n, \quad (29)$$

c: MINIMUM TURNING AMPLITUDE OBJECTIVE FUNCTION

Each turning amplitude of the mobile robot has a direct impact on its motion time, smoothness of path and motion

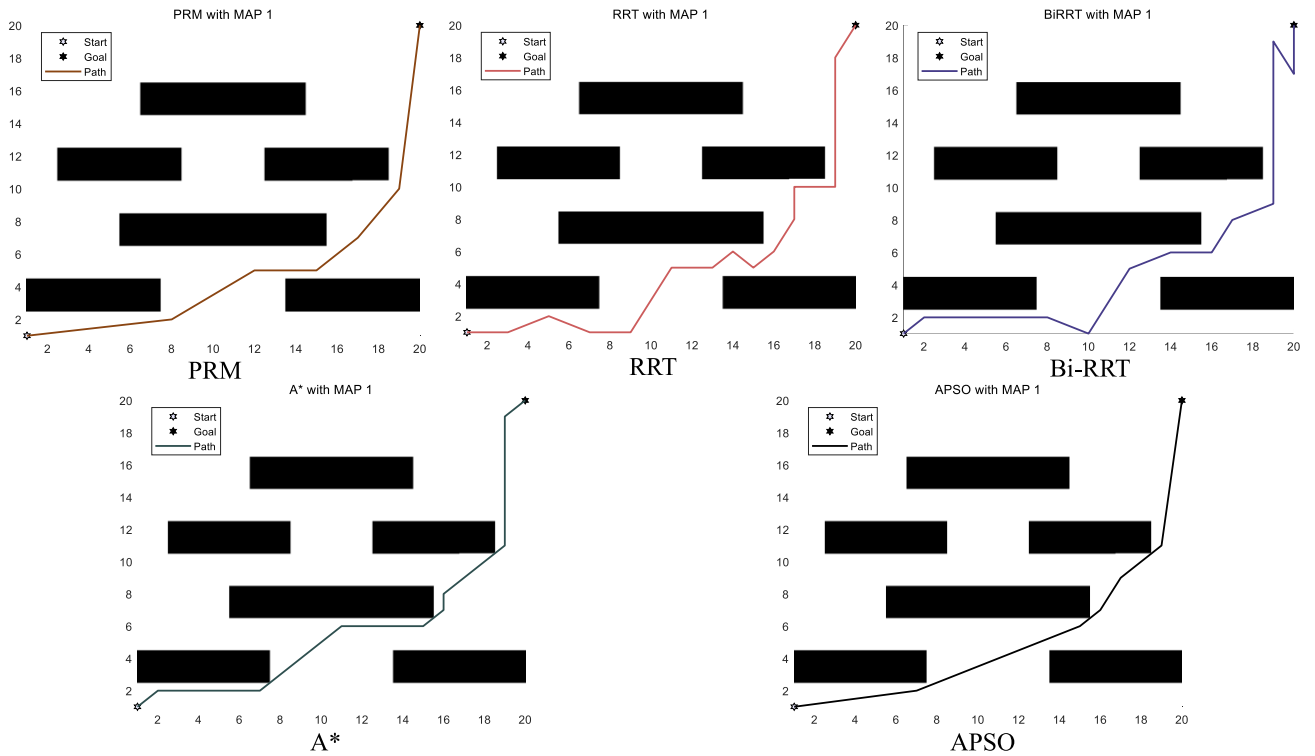


FIGURE 9. Simulation experiment of the obstacle environment in map 1.

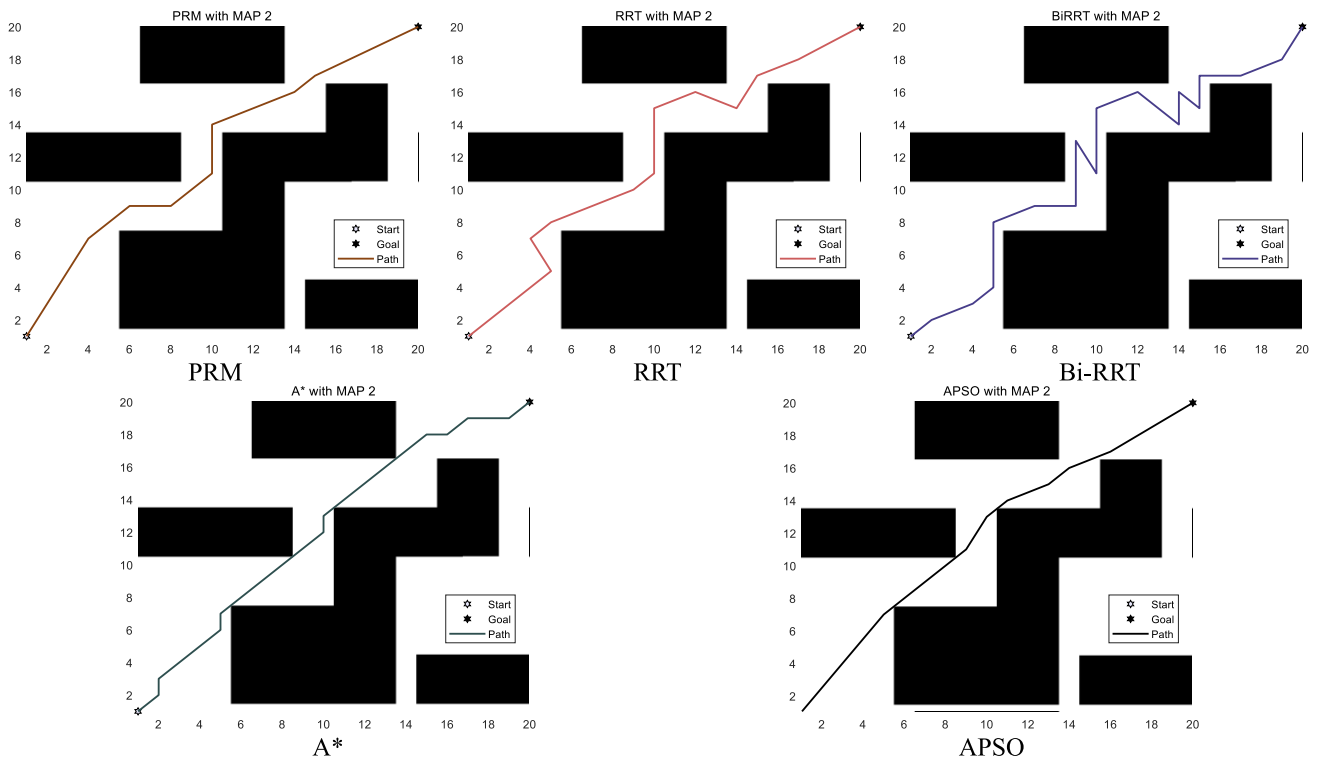


FIGURE 10. Simulation experiment of the obstacle environment in map 2.

stability. The smaller the turning amplitude is, the shorter the motion time, the better the path smoothness and the higher

the motion stability. Therefore, the mobile robot should be set to minimize the turning amplitude, and its objective function

TABLE 6. The objective function values of each algorithm in different maps.

	APSO	A*	PRM	RRT	Bi-RRT	
Map1	operation time (s)	0.48	0.05	0.76	0.01	0.01
	turn amplitude (rad)	1.94	6.28	2.21	11.15	8.56
	path length (m)	30.56	32.14	31.54	38.91	39.00
	node number	7.00	29.00	7.00	21.00	19.00
	motion time (s)	33.05	39.99	34.40	46.67	45.54
Map2	operation time (s)	0.45	0.06	0.79	0.01	0.01
	turn amplitude (rad)	1.99	7.85	3.84	11.08	11.40
	path length (m)	27.40	28.63	29.00	34.83	31.22
	node number	10.00	23.00	10.00	18.00	18.00
	motion time (s)	30.48	35.79	33.02	41.97	38.46
Map3	operation time (s)	0.29	0.05	0.77	0.01	0.01
	turn amplitude (rad)	2.27	6.28	3.91	10.93	8.66
	path length (m)	28.99	29.80	31.14	34.48	33.73
	node number	6.00	25.00	11.00	19.00	23.00
	motion time (s)	31.21	36.85	35.36	41.77	41.10
Map4	operation time (s)	1.35	0.07	0.76	0.01	0.01
	turn amplitude (rad)	2.69	7.85	3.26	17.87	14.86
	path length (m)	29.20	30.97	30.93	38.17	31.10
	node number	8.00	27.00	7.00	36.00	34.00
	motion time (s)	33.02	38.94	34.13	51.07	42.64
Map5	operation time (s)	0.62	0.05	0.80	0.00	0.02
	turn amplitude (rad)	0.39	7.85	3.29	15.16	14.45
	path length (m)	27.25	28.63	28.56	32.91	28.34
	node number	5.00	23.00	8.00	29.00	29.00
	motion time (s)	29.00	35.78	32.01	43.55	38.76
Map6	operation time (s)	0.30	0.07	0.81	0.01	0.02
	turn amplitude (rad)	1.43	6.28	4.00	17.32	14.68
	path length (m)	27.60	28.63	29.56	36.00	31.60
	node number	7.00	23.00	11.00	34.00	33.00
	motion time (s)	29.75	35.29	33.85	48.32	42.89

TABLE 7. Wilcoxon rank sum test.

	APSO vs.	A*	PRM	RRT	Bi-RRT
MAP 1	operation time	8.61E-05	8.86E-05	8.86E-05	8.86E-05
	turn amplitude	7.74E-06	1.52E-02	8.83E-05	8.86E-05
	path length	7.74E-06	8.76E-05	8.84E-05	8.84E-05
	node number	7.74E-06	1.00E+00	7.74E-06	7.74E-06
	motion time	8.61E-05	8.86E-05	8.86E-05	8.86E-05
MAP 2	operation time	8.68E-05	1.63E-04	8.86E-05	8.86E-05
	turn amplitude	7.74E-06	8.83E-05	8.86E-05	8.86E-05
	path length	7.74E-06	8.77E-05	8.86E-05	8.86E-05
	node number	7.74E-06	1.00E+00	7.74E-06	7.74E-06
	motion time	8.68E-05	8.86E-05	8.86E-05	8.86E-05
MAP 3	operation time	8.56E-05	1.20E-04	8.86E-05	8.86E-05
	turn amplitude	7.74E-06	3.73E-04	8.84E-05	8.86E-05
	path length	7.74E-06	8.40E-05	8.84E-05	8.83E-05
	node number	7.74E-06	7.74E-06	7.74E-06	7.74E-06
	motion time	8.56E-05	8.86E-05	8.86E-05	8.86E-05
MAP 4	operation time	7.74E-06	8.70E-05	8.86E-05	8.86E-05
	turn amplitude	7.74E-06	7.44E-05	8.76E-05	8.49E-05
	path length	7.74E-06	7.45E-05	8.78E-05	7.90E-05
	node number	7.74E-06	0.089272	8.56E-05	8.41E-05
	motion time	7.74E-06	8.83E-05	8.86E-05	8.86E-05
MAP 5	operation time	7.74E-06	0.001157	8.86E-05	8.86E-05
	turn amplitude	7.74E-06	8.60E-05	8.70E-05	8.63E-05
	path length	7.74E-06	8.60E-05	8.77E-05	8.29E-05
	node number	7.74E-06	7.35E-05	8.33E-05	8.08E-05
	motion time	7.74E-06	8.84E-05	8.86E-05	8.86E-05
MAP 6	operation time	7.74E-06	8.72E-05	8.86E-05	8.86E-05
	turn amplitude	7.74E-06	8.83E-05	8.65E-05	8.83E-05
	path length	7.74E-06	8.86E-05	8.76E-05	8.83E-05
	node number	7.74E-06	8.37E-05	8.61E-05	8.66E-05
	motion time	7.74E-06	8.86E-05	8.86E-05	8.86E-05

formula is as follows:

$$\min I(\theta) = \sum_{i=1}^{n-2} \theta_n, \tag{30}$$

$$\theta_n = \left| \arctan \left(\frac{\frac{y_{n+1}-y_n}{x_{n+1}-x_n} - \frac{y_{n+2}-y_{n+1}}{x_{n+2}-x_{n+1}}}{1 + \frac{y_{n+1}-y_n}{x_{n+1}-x_n} \times \frac{y_{n+2}-y_{n+1}}{x_{n+2}-x_{n+1}}} \right) \right|, \tag{31}$$

where n is the n -th node in the path.

d: SHORTEST ALGORITHM OPERATION TIME OBJECTIVE FUNCTION

The operation time of the path planning algorithm of the mobile robot can directly affect the working efficiency of the mobile robot by affecting the operation efficiency. Therefore, the operation time of the minimization algorithm of the mobile robot should be set, and its objective function formula

is shown in Eq.32

$$\min C(a) = t_a, \tag{32}$$

where t_a is the running time of algorithm a .

The above objective functions can directly or indirectly affect the running time of the mobile robot, and there is no conflict between them. Therefore, after processing these several objective functions, we can combine them into the final objective function with the shortest running time of the mobile robot. The combined objective function is shown as follows:

$$\min T(n) = \frac{D(x, y)}{v_m} + P(n) \times t_p + \frac{I(\theta)}{v_i} + C(a) \tag{33}$$

where v_m represents the linear moving speed of the mobile robot. In this paper, $v_m = 1\text{m/s}$. t_p represents the judgment time required by the mobile robot when it reaches a node.

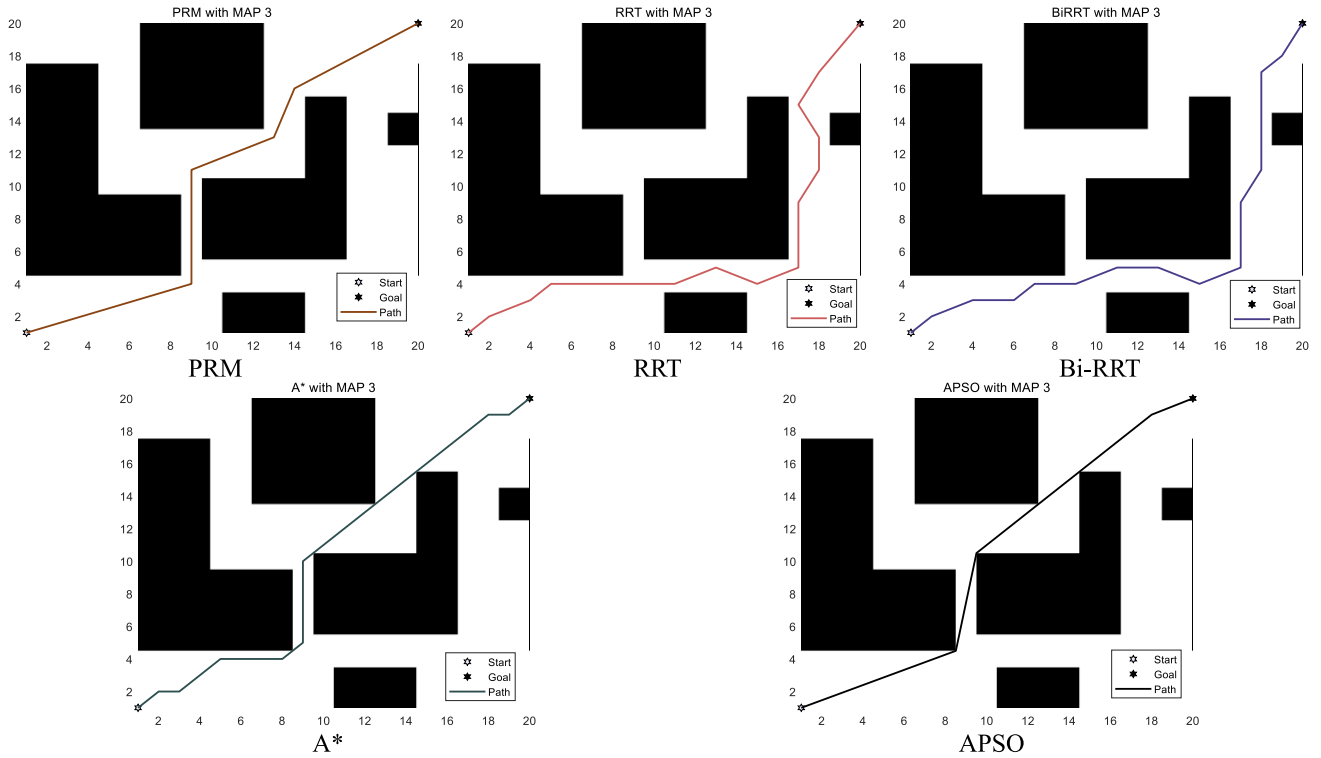


FIGURE 11. Simulation experiment of the obstacle environment in map 3.

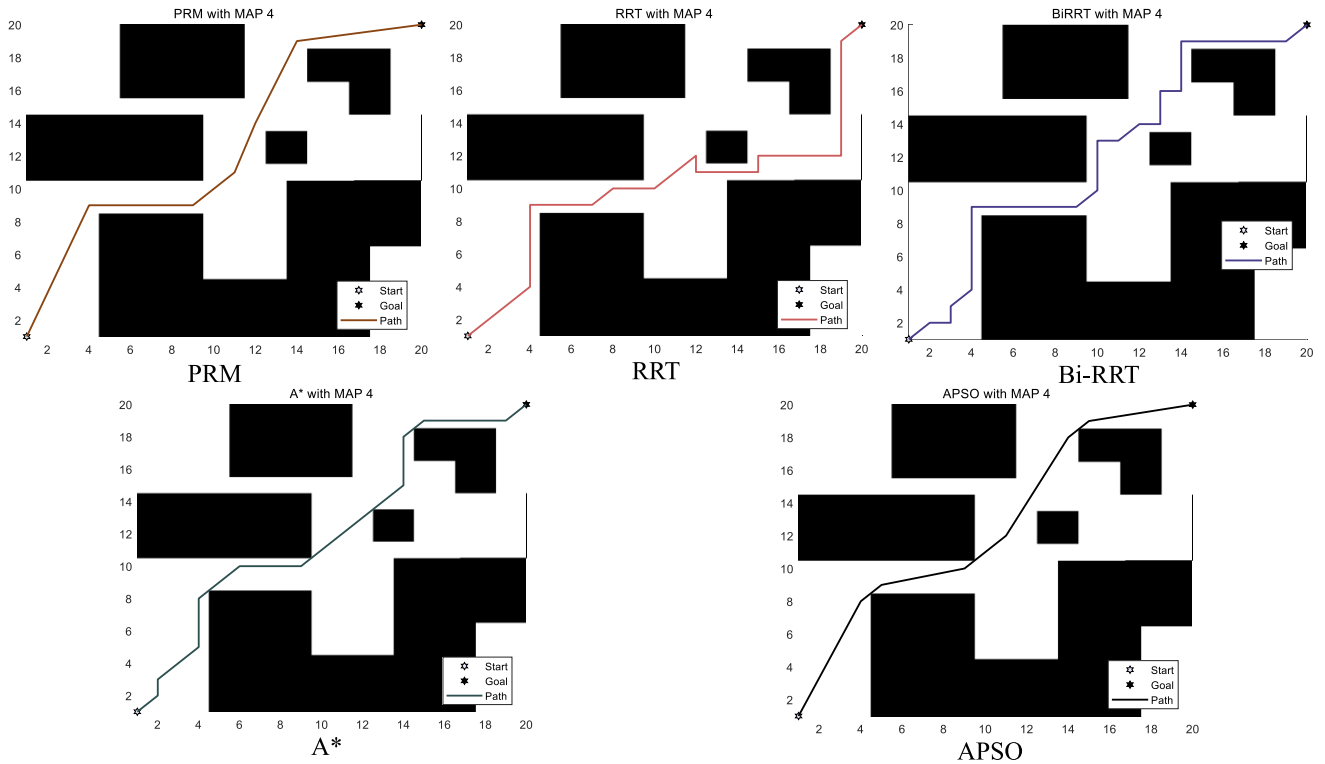


FIGURE 12. Simulation experiment of the obstacle environment in map 4.

In this paper, $t_p = 0.2s$. v_i represents the turning speed of the mobile robot, that is, the time consumed by each turning radian. In this paper, $v_i = \pi \text{ rad/s}$.

C. OBSTACLE ENVIRONMENT SIMULATION EXPERIMENT
The paths planned by the APSO, A*, PRM, RRT and Bi-RRT algorithms in Maps 1 to 6 are shown in Figs. 9-14,

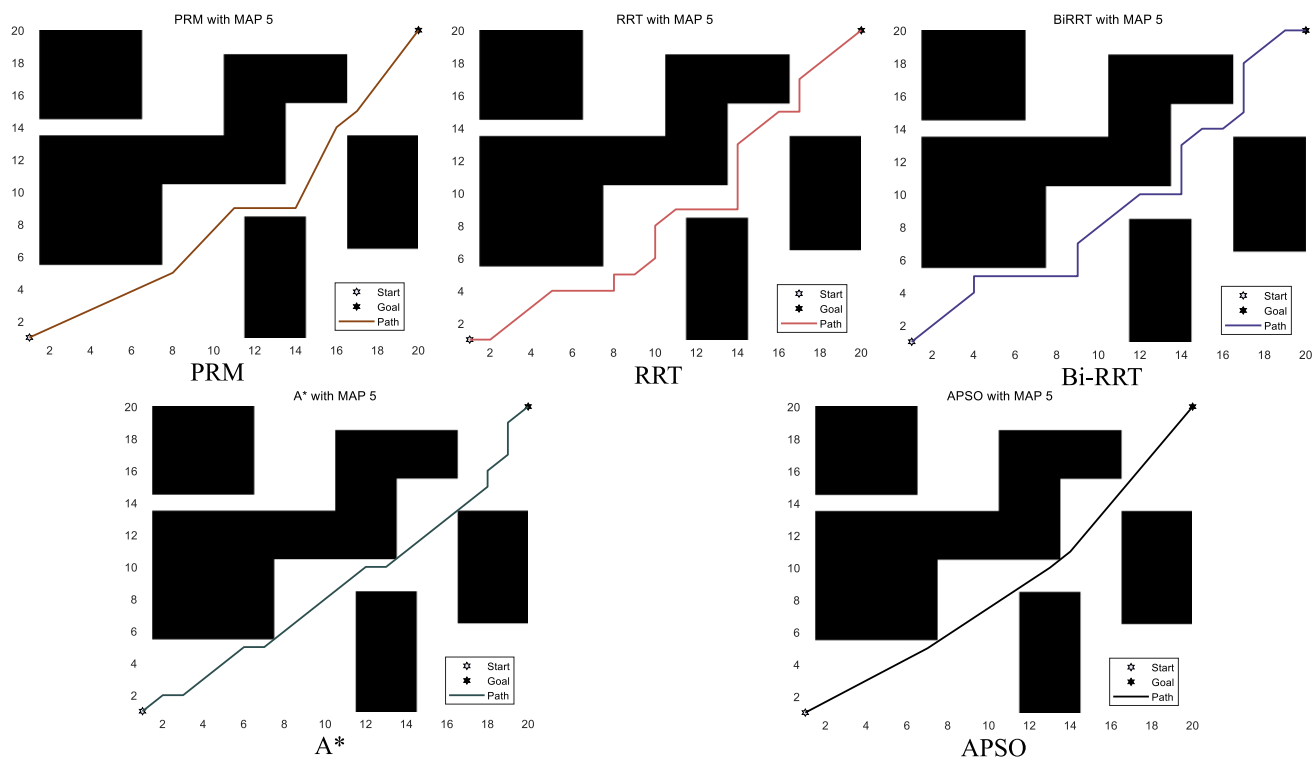


FIGURE 13. Simulation experiment of the obstacle environment in map 5.

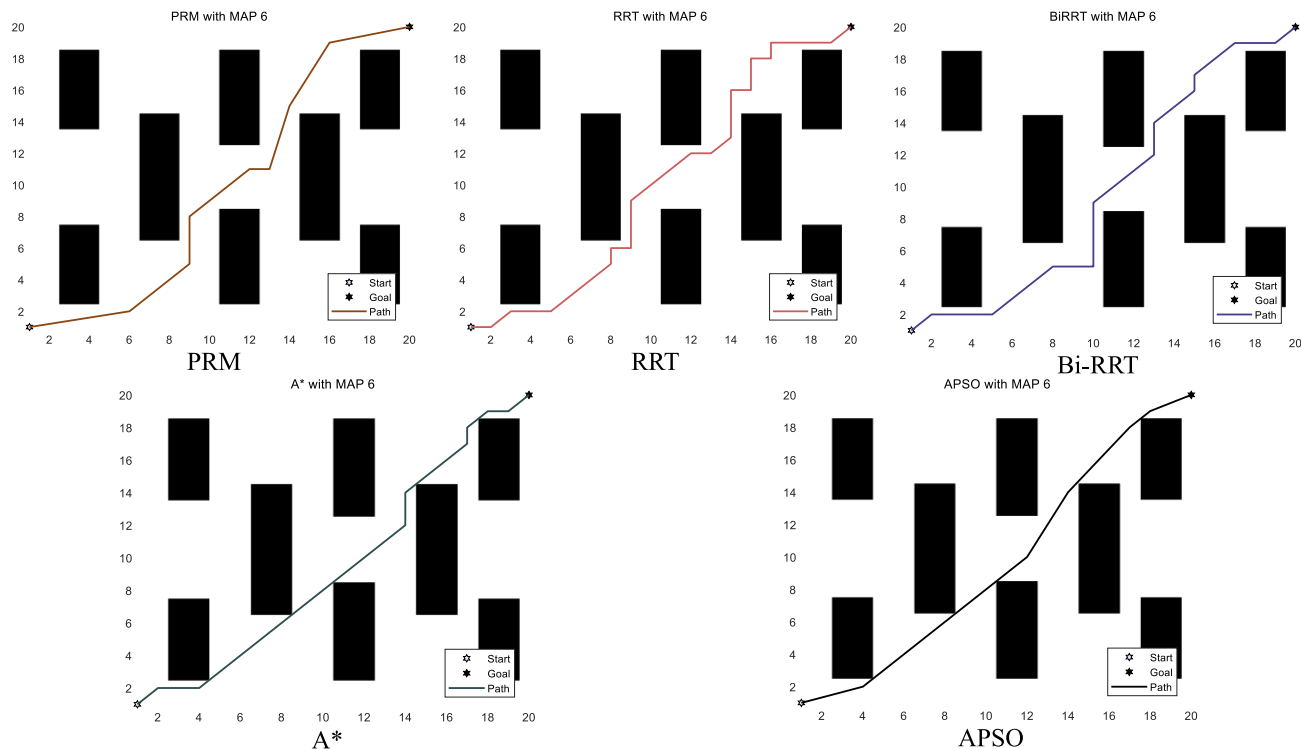


FIGURE 14. Simulation experiment of the obstacle environment in map 6.

respectively. Intuitively, compared with other comparison algorithms, the paths planned by the APSO in the six maps are obviously shorter in length, with fewer path nodes and

turning points, and have higher smoothness. Tab. 6 provides the average objective function value of each algorithm after 20 independent runs in different maps. Because the number

of nodes must be an integer, the average number of nodes is rounded upward. The bold data in the table are the final objective function values. It can be seen from Tab. 6 that in the six maps of APSO, except that the operation time of the algorithm is slightly higher than that of the A*, RRT and Bi-RRT algorithms due to the inevitable increase in time complexity, the other objective function values are the best among the algorithms. In map 1 with 22.00% obstacles, the running time of the mobile robot using APSO is reduced by 17.35% compared with A*, 3.92% compared with PRM, 28.76% compared with RRT and 27.43% compared with Bi-RRT. In map 2 with 41.75% obstacles, the running time of the mobile robot using APSO is reduced by 14.84% compared with A*, 7.69% compared with PRM, 27.38% compared with RRT and 20.75% compared with Bi-RRT. In map 3 with 45.75% obstacles, the running time of the mobile robot using APSO is reduced by 15.31% compared with A*, 12.23% compared with PRM, 25.28% compared with RRT and 24.06% compared with Bi-RRT. In map 4 with 48.5% obstacles, the running time of the mobile robot using APSO is reduced by 15.21% compared with A*, 3.26% compared with PRM, 35.35% compared with RRT and 22.57% compared with Bi-RRT. In map 5 with 45.25% obstacles, the running time of the mobile robot using APSO is reduced by 18.97% compared with A*, 9.41% compared with PRM, 33.41% compared with RRT and 25.19% compared with Bi-RRT. In map 6 with 24.00% obstacles, the running time of the mobile robot using APSO is reduced by 15.70% compared with A*, 12.09% compared with PRM, 38.42% compared with RRT and 30.63% compared with Bi-RRT. It can be seen that APSO has superior performance in environments with different obstacle densities.

To prove the validity of the conclusion that “APSO can search superior paths among obstacles of different densities”, the Wilcoxon rank sum test was used to conduct a statistical test on relevant data, and the test results are shown in Tab. 7. Since most of the P values are less than 0.05, the relevant data of APSO have statistically significant differences compared with other algorithms, so the conclusion is valid.

VI. CONCLUSION AND FUTURE WORK

Aiming at the problems of multiple nodes and poor path accuracy of the A* algorithm in the path planning problem of mobile robots, as well as the problems of a difficult initial population and fast convergence speed of the PSO algorithm in a complex environment, this paper proposed an APSO algorithm embedded with an improved A* algorithm and improved PSO algorithm and conducted an algorithm performance test and path planning simulation test. The performance test of the algorithm shows that the PSO using stochastic inertia weight and stochastic opposition-based learning strategy has statistical superiority, stability and effectiveness. Simulation results show that the APSO proposed in this paper can not only solve the problem of path planning but also has higher smoothness, shorter route length and stronger safety, which effectively reduces the running time of mobile

robots and is more in line with the movement conditions of mobile robots in the actual environment.

Since we focus on the path planning problem in a fixed environment, the influence of dynamic scenes on mobile robots is not considered. In dynamic scenarios, the hybrid algorithm in this paper should be combined with the dynamic path planning algorithm to meet the real-time obstacle avoidance requirements of mobile robots, which will become the focus of the next research direction.

REFERENCES

- [1] M. A. Hossain and I. Ferdous, “Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique,” *Robot. Auto. Syst.*, vol. 64, pp. 137–141, Feb. 2015.
- [2] W. Sun, Y. Lv, H. Tang, and M. Xue, “Mobile robot path planning based on an improved A* algorithm,” *J. Hunan Univ., Natural Sci.*, vol. 44, no. 4, pp. 94–101, 2017.
- [3] T.-W. Zhang, G.-H. Xu, X.-S. Zhan, and T. Han, “A new hybrid algorithm for path planning of mobile robot,” *J. Supercomput.*, vol. 78, no. 3, pp. 4158–4181, Feb. 2022.
- [4] D. Curiau and C. Volosencu, “A 2D chaotic path planning for mobile robots accomplishing boundary surveillance missions in adversarial conditions,” *Commun. Nonlinear Sci. Numer. Simul.*, vol. 19, pp. 3617–3627, Oct. 2014.
- [5] X. J. Fan, M. Y. Jiang, and Z. L. Pei, “Research on path planning of mobile robot based on genetic algorithm in dynamic environment,” *Basic Clin. Pharmacol.*, vol. 124, p. 54, Dec. 2018.
- [6] H.-Y. Zhang, W.-M. Lin, and A.-X. Chen, “Path planning for the mobile robot: A review,” *Symmetry*, vol. 10, no. 10, p. 450, 2018.
- [7] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Oct. 1959.
- [8] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.
- [9] S. Koenig and M. Likhachev, “Incremental A*,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 1539–1546.
- [10] D. Ferguson and A. Stentz, “Using interpolation to improve path planning: The field D* algorithm,” *J. Field Robot.*, vol. 23, pp. 79–101, Feb. 2006, doi: 10.1002/rob.20109.
- [11] L. E. Kavrakli, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.
- [12] S. Lavalle, “Rapidly-exploring random trees: A new tool for path planning,” Tech. Rep., 1999. [Online]. Available: https://www.researchgate.net/publication/2639200_Rapidly-Exploring_Random_Trees_A_New_Tool_for_Path_Planning
- [13] S. Erke, D. Bin, N. Yiming, Z. Qi, X. Liang, and Z. Dawei, “An improved A-star based path planning algorithm for autonomous land vehicles,” *Int. J. Adv. Robot. Syst.*, vol. 17, no. 5, Sep. 2020, Art. no. 172988142096226.
- [14] H. Liu and Y. Zhang, “ASL-DWA: An improved A-star algorithm for indoor cleaning robots,” *IEEE Access*, vol. 10, pp. 99498–99515, 2022.
- [15] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, “Geometric A-star algorithm: An improved A-star algorithm for AGV path planning in a port environment,” *IEEE Access*, vol. 9, pp. 59196–59210, 2021.
- [16] C. Li, X. Huang, J. Ding, K. Song, and S. Lu, “Global path planning based on a bidirectional alternating search A* algorithm for mobile robots,” *Comput. Ind. Eng.*, vol. 168, Jun. 2022, Art. no. 108123.
- [17] J. Yu, M. Yang, Z. Zhao, X. Wang, Y. Bai, J. Wu, and J. Xu, “Path planning of unmanned surface vessel in an unknown environment based on improved D*lite algorithm,” *Ocean Eng.*, vol. 266, Dec. 2022, Art. no. 112873.
- [18] W. Li, L. Wang, A. Zou, J. Cai, H. He, and T. Tan, “Path planning for UAV based on improved PRM,” *Energies*, vol. 15, no. 19, p. 7267, Oct. 2022.
- [19] H. Ma, W. Pei, and Q. Zhang, “Research on path planning algorithm for driverless vehicles,” *Mathematics*, vol. 10, no. 15, p. 2555, Jul. 2022.
- [20] J.-G. Kang, D.-W. Lim, Y.-S. Choi, W.-J. Jang, and J.-W. Jung, “Improved RRT-connect algorithm based on triangular inequality for robot path planning,” *Sensors*, vol. 21, no. 2, p. 333, Jan. 2021.
- [21] J. H. Holland, “Genetic algorithms,” *Sci. Amer.*, vol. 267, no. 1, pp. 66–73, 1992.

- [22] R. Storn and K. V. Price, "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," 1995.
- [23] M. Dorigo and G. Di Caro, "Ant colony optimization: A new meta-heuristic," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 2, 1999, pp. 1470–1477.
- [24] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, vol. 4, 1995, pp. 1942–1948.
- [25] J. Ma, Y. Liu, S. Zang, and L. Wang, "Robot path planning based on genetic algorithm fused with continuous Bezier optimization," *Comput. Intell. Neurosci.*, vol. 2020, pp. 1–10, Feb. 2020.
- [26] K. Hao, J. Zhao, B. Wang, Y. Liu, and C. Wang, "The application of an adaptive genetic algorithm based on collision detection in path planning of mobile robots," *Comput. Intell. Neurosci.*, vol. 2021, pp. 1–20, May 2021.
- [27] C. Miao, G. Chen, C. Yan, and Y. Wu, "Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm," *Comput. Ind. Eng.*, vol. 156, Jun. 2021, Art. no. 107230.
- [28] M. Xie, Y. Bai, M. Huang, Y. Deng, and Z. Hu, "Energy- and time-aware data acquisition for mobile robots using mixed cognition particle swarm optimization," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7734–7750, Aug. 2020.
- [29] L. Zhang, Y. Zhang, and Y. Li, "Mobile robot path planning based on improved localized particle swarm optimization," *IEEE Sensors J.*, vol. 21, no. 5, pp. 6962–6972, Mar. 2021.
- [30] X. Yuan, X. Yuan, and X. Wang, "Path planning for mobile robot based on improved bat algorithm," *Sensors*, vol. 21, no. 13, p. 4389, Jun. 2021.
- [31] X. Li, Y. Xu, K. K. Lai, H. Ji, Y. Xu, and J. Li, "A multi-period vehicle routing problem for emergency perishable materials under uncertain demand based on an improved whale optimization algorithm," *Mathematics*, vol. 10, no. 17, p. 3124, Aug. 2022.
- [32] F.-F. Li, Y. Du, and K.-J. Jia, "Path planning and smoothing of mobile robot based on improved artificial fish swarm algorithm," *Sci. Rep.*, vol. 12, no. 1, p. 659, Jan. 2022.
- [33] S. K. Pattnaik, S. Panda, and D. Mishra, "A multi-objective approach for local path planning of autonomous mobile robot based on metaheuristics," *Concurrency Comput., Pract. Exp.*, vol. 34, no. 10, May 2022, Art. no. e6801.
- [34] Q. Song, S. Li, J. Yang, Q. Bai, J. Hu, X. Zhang, and A. Zhang, "Intelligent optimization algorithm-based path planning for a mobile robot," *Comput. Intell. Neurosci.*, vol. 2021, Sep. 2021, Art. no. 8025730.
- [35] H. Arslan and M. Manguoğlu, "A hybrid single-source shortest path algorithm," *TURKISH J. Electr. Eng. Comput. Sci.*, vol. 27, no. 4, pp. 2636–2647, Jul. 2019.
- [36] L. Pasandi, M. Hooshmand, and M. Rahbar, "Modified A* algorithm integrated with ant colony optimization for multi-objective route-finding; case study: Yazd," *Appl. Soft Comput.*, vol. 113, Dec. 2021, Art. no. 107877.
- [37] Y. Zhou and N. Huang, "Airport AGV path optimization model based on ant colony algorithm to optimize Dijkstra algorithm in urban systems," *Sustain. Comput., Informat. Syst.*, vol. 35, Sep. 2022, Art. no. 100716.
- [38] J. Yu, G. Liu, J. Xu, Z. Zhao, Z. Chen, M. Yang, X. Wang, and Y. Bai, "A hybrid multi-target path planning algorithm for unmanned cruise ship in an unknown obstacle environment," *Sensors*, vol. 22, no. 7, p. 2429, 2022.
- [39] A. Omar and S. Ahmed, "An optimal path planning algorithms for mobile robot," *Iraqi J. Comput., Commun. Control Syst. Eng.*, vol. 21, no. 2, pp. 44–58, 2021.
- [40] O. Wahhab and A. Al-Araji, "Path planning and control strategy design for mobile robot based on hybrid swarm optimization algorithm," *Int. J. Intell. Eng. Syst.*, vol. 14, no. 3, pp. 565–579, Jun. 2021.
- [41] H. Min, X. Xiong, P. Wang, and Y. Yu, "Autonomous driving path planning algorithm based on improved A* algorithm in unstructured environment," *Proc. Inst. Mech. Eng. D, J. Automobile Eng.*, vol. 235, nos. 2–3, pp. 513–526, Feb. 2021.
- [42] Y. Li, R. Jin, X. Xu, Y. Qian, H. Wang, S. Xu, and Z. Wang, "A mobile robot path planning algorithm based on improved A* algorithm and dynamic window approach," *IEEE Access*, vol. 10, pp. 57736–57747, 2022.
- [43] Y. Ou, Y. Fan, X. Zhang, Y. Lin, and W. Yang, "Improved A* path planning method based on the grid map," *Sensors*, vol. 22, no. 16, p. 6198, Aug. 2022.
- [44] C. Niu, A. Li, X. Huang, W. Li, and C. Xu, "Research on global dynamic path planning method based on improved A* algorithm," *Math. Problems Eng.*, vol. 2021, Aug. 2021, Art. no. 4977041.
- [45] J. Zhang, J. Wu, X. Shen, and Y. Li, "Autonomous land vehicle path planning algorithm based on improved heuristic function of A-star," *Int. J. Adv. Robot. Syst.*, vol. 18, no. 5, Sep. 2021, Art. no. 172988142110427.
- [46] H. Wang, X. Qi, S. Lou, J. Jing, H. He, and W. Liu, "An efficient and robust improved A* algorithm for path planning," *Symmetry*, vol. 13, no. 11, p. 2213, Nov. 2021.
- [47] T. Qiuyun, S. Hongyan, G. Hengwei, and W. Ping, "Improved particle swarm optimization algorithm for AGV path planning," *IEEE Access*, vol. 9, pp. 33522–33531, 2021.
- [48] X. Xu and F. Yan, "Random walk autonomous groups of particles for particle swarm optimization," *J. Intell. Fuzzy Syst.*, vol. 42, no. 3, pp. 1519–1545, Feb. 2022.
- [49] F. Kong, J. Jiang, and Y. Huang, "An adaptive multi-swarm competition particle swarm optimizer for large-scale optimization," *Mathematics*, vol. 7, no. 6, p. 521, Jun. 2019.
- [50] X. Chen, H. Tianfield, and W. Du, "Bee-foraging learning particle swarm optimization," *Appl. Soft Comput.*, vol. 102, Apr. 2021, Art. no. 107134.
- [51] A. Flori, H. Oulhadj, and P. Siarry, "Quantum particle swarm optimization: An auto-adaptive PSO for local and global optimization," *Comput. Optim. Appl.*, vol. 82, no. 2, pp. 525–559, Jun. 2022.
- [52] Y. Zhang, X. Liu, F. Bao, J. Chi, C. Zhang, and P. Liu, "Particle swarm optimization with adaptive learning strategy," *Knowl.-Based Syst.*, vol. 196, May 2020, Art. no. 105789.
- [53] V. Roberge, M. Tarbouchi, and G. Labonté, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 132–141, Feb. 2012.
- [54] B. K. Patle, A. Pandey, D. R. K. Parhi, and A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot," *Defence Technol.*, vol. 15, pp. 582–606, Aug. 2019.
- [55] Y. Valle, G. K. Venayagamoorthy, S. Mohagheghi, J. C. Hernandez, and R. G. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 171–195, Apr. 2008.
- [56] Y. Zhang, S. Wang, G. Ji, and Z. Dong, "An MR brain images classifier system via particle swarm optimization and kernel support vector machine," *Sci. World J.*, vol. 2013, Aug. 2013, Art. no. 130134.
- [57] M. F. Tasgetiren, Y. C. Liang, M. Sevkli, and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," *Eur. J. Oper. Res.*, vol. 177, no. 3, pp. 1930–1947, 2007.
- [58] X. Cheng, J. Li, C. Zheng, J. Zhang, and M. Zhao, "An improved PSO-GWO algorithm with chaos and adaptive inertial weight for robot path planning," *Frontiers Neurobot.*, vol. 15, Nov. 2021, Art. no. 770361.
- [59] A. T. Kiani, M. F. Nadeem, A. Ahmed, I. Khan, R. M. Elavarasan, and N. Das, "Optimal PV parameter estimation via double exponential function-based dynamic inertia weight particle swarm optimization," *Energies*, vol. 13, no. 15, p. 4037, Aug. 2020.
- [60] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evol. Comput.*, May 1998, pp. 69–73.
- [61] M. Li, H. Chen, N. Zhong, S. Lu, and X. Wang, "An improved particle swarm optimization algorithm with adaptive inertia weights," *Int. J. Inf. Technol. Decis. Making*, vol. 18, no. 3, pp. 833–866, 2019.
- [62] H. R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Proc. Int. Conf. Comput. Intell. Modeling, Control Autom. Int. Conf. Intell. Agents, Web Technol. Internet Commerce (CIMCA-IAWTIC)*, 2005, pp. 695–701.
- [63] X. Yu, W. Xu, and C. Li, "Opposition-based learning grey wolf optimizer for global optimization," *Knowl.-Based Syst.*, vol. 226, Aug. 2021, Art. no. 107139.
- [64] J. Li, Y.-X. Li, S.-S. Tian, and J. Zou, "Dynamic cuckoo search algorithm based on Taguchi opposition-based search," *Int. J. Bio-Inspired Comput.*, vol. 13, no. 1, pp. 59–69, 2019.
- [65] L. Abualigah, D. Yousef, M. A. Elaziz, A. A. Ewees, M. A. A. Al-Qaness, and A. H. Gandomi, "Aquila optimizer: A novel meta-heuristic optimization algorithm," *Comput. Ind. Eng.*, vol. 157, Jul. 2021, Art. no. 107250.
- [66] T. Sang-To, H. Le-Minh, S. Mirjalili, M. A. Wahab, and T. Cuong-Le, "A new movement strategy of grey wolf optimizer for optimization problems and structural damage identification," *Adv. Eng. Softw.*, vol. 173, Nov. 2022, Art. no. 103276.
- [67] Q. Askari, I. Younas, and M. Saeed, "Political optimizer: A novel socio-inspired meta-heuristic for global optimization," *Knowl.-Based Syst.*, vol. 195, May 2020, Art. no. 105709.

- [68] K. Zervoudakis and S. Tsafarakis, "A mayfly optimization algorithm," *Comput. Ind. Eng.*, vol. 145, Jul. 2020, Art. no. 106559.
- [69] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-verse optimizer: A nature-inspired algorithm for global optimization," *Neural Comput. Appl.*, vol. 27, no. 2, pp. 495–513, 2016.
- [70] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *J. Inf. Sci.*, vol. 179, no. 13, pp. 2232–2248, 2009.
- [71] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, Dec. 2008.
- [72] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.
- [73] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 2016.
- [74] M. Seyedali, M. M. Seyed, and L. Andrew, "Gray wolf optimizer," *Adv. Eng. Softw.*, vol. 69, no. 3, pp. 46–61, 2014.
- [75] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, 1945.



CHANGSHENG HUANG received the B.S., M.S., and Ph.D. degrees in management from the China University of Petroleum, in 1995, 1998, and 2010, respectively.

Since 1998, he has been a Professor with the School of Economics and Management, China University of Petroleum. He is the author of three books and more than 60 articles. His research interests include strategic management, marketing, and industrial engineering.



YANPU ZHAO received the B.S. degree in software engineering from Qingdao University, Qingdao, China, in 2021. He is currently pursuing the master's degree in industrial engineering and management with the School of Economics and Management, China University of Petroleum.

His research interests include path planning, intelligent computing, and metaheuristic algorithm.



MENGJIE ZHANG received the B.S. degree in software engineering from Qingdao University, Qingdao, China, in 2021.

Her research interests include software engineering, smart medical care, and intelligent computing.



HONGYAN YANG received the B.S. degree in industrial engineering from the Shandong University of Science and Technology, Qingdao, China, in 2021. She is currently pursuing the master's degree in industrial engineering and management with the School of Economics and Management, China University of Petroleum.

Her research interests include line balancing and intelligent computing.

...