Mathematical Biosciences and Engineering

MBE, 19(12): 12405–12426.
DOI: 10.3934/mbe.2022579
Received: 18 February 2022
Revised: 20 May 2022
Accepted: 02 June 2022
Published: 25 August 2022

http://www.aimspress.com/journal/MBE

*Research article*

# An improved ant colony algorithm for integrating global path planning and local obstacle avoidance for mobile robot in dynamic environment

**Gong Chikun[1], Yang Yuhang[1], Yuan Lipeng[2,*] and Wang Jiaxin[3]**

[1] College of Mechanical Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China
[2] School of Mechanical and Electrical Engineering, Harbin Institute of Technology, Harbin 15001, China
[3] Faculty of Western Languages, Heilongjiang University, Harbin 150080, China

* **Correspondence:** Email: hitylp@126.com; Tel: +8613611644083.

**Abstract:** To improve the path optimization effect and search efficiency of ant colony optimization (ACO), an improved ant colony algorithm is proposed. A collar path is generated based on the known environmental information to avoid the blindness search at early planning. The effect of the ending point and the turning point is introduced to improve the heuristic information for high search efficiency. The adaptive adjustment of the pheromone intensity value is introduced to optimize the pheromone updating strategy. A variety of control strategies for updating the parameters are given to balance the convergence and global search ability. Then, the improved obstacle avoidance strategies are proposed for dynamic obstacles of different shapes and motion states, which overcome the shortcomings of existing obstacle avoidance strategies. Compared with other improved algorithms in different simulation environments, the results show that the algorithm in this paper is more effective and robust in complicated and large environments. On the other hand, the comparison with other obstacle avoidance strategies in a dynamic environment shows that the strategies designed in this paper have higher path quality after local obstacle avoidance, lower requirements for sensor performance, and higher safety.

**Keywords:** path planning; mobile robot; ant colony algorithm; local obstacle avoidance strategy

## 1. Introduction

As the application of mobile robots gradually shifts from industrial environments to medical handling, navigation assistance, warehousing and transportation, and catering services, there are higher requirements for mobile robot navigation algorithms. At present, the algorithms for path planning are divided into two categories: traditional path planning algorithms which mainly include the D* algorithm, fuzzy logic, potential field method, the A* algorithm, and intelligent optimization algorithms, such as the genetic algorithm [1–5], particle swarm algorithm, ant colony optimization and so on [6–9]. Ant colony algorithm has attracted much attention in mobile robot path planning because it can optimize many aspects of path quality, is better suited to the needs of practical robot applications, and have the advantage of being easily combined with other search methods.

For the research on path planning of mobile robots in a static environment with known global information, Liu et al. [10] combined the ant colony algorithm with geometric optimization to eliminate the cross paths generated during the pathfinding process. Viseras et al. [11] added fast search random tree method into the ant colony algorithm to accelerate the search speed of the algorithm. Wang et al. [12] added initial pheromone to all nodes between the starting point and the ending point, which improves the adaptability and convergence speed of the algorithm. However, in complex environments, due to the roulette transfer method's randomness, pre-planning using other algorithms and adding a region of the initial pheromone cannot guarantee that ants will always follow this path or region and will increase the running time and complexity of the algorithm. The result is not as good as expected. Zheng et al. [13] proposed an adaptive update heuristic function to improve the performance. Jiao et al. [14] used an adaptive state transition method and the adaptive information updating strategy to adjust parameters in real-time, which guarantee the relative importance of pheromone intensity and desirability and the ability to jump out of local optimal. Akka et al. [15] designed a piece-wise multi-heuristic function, rewarding and punishing the optimal and the worst path respectively, which expands the search space of the algorithm and reduces the influence of cluttered pheromone. You et al. [16] introduced a multiple evolution strategy mechanism into the state transition rules of the ant colony algorithm that increase the diversity of search results to improve search quality and convergence speed. Wang et al. [17] proposed a method that if the ants are trapped in deadlock, the pheromone of the path is not updated to avoid other ants being attracted to the deadlock. Qu et al. [18] adopt the regression and penalty strategy, which allows ants to return one step and punish the pheromones on the path that ants have traveled. Luo et al. [19] devised a dynamic punishment method to reduce the number of lost ants and improve the diversity of the solution. Miao et al. [20] transformed the path planning problem into a multi-objective optimization problem by introducing the multi-objective performance indexes into the algorithm. However, the existing improved algorithms are still not well adapted to diverse, unstructured and complex environments because of the single heuristic information, still exist the excessive data fluctuations, the low stability of the algorithms, the long running time and can't better balance the global search ability and convergence speed of the algorithm problems in large-scale complex environments.

For the research on path planning in a dynamic environment, Zhu et al. [21] solved the path planning problem of the robot in the working environment that beyond its perception range and contains unknown static and dynamic obstacles. Yen et al. [22] avoided dynamic obstacles by finding the linear function of moving obstacles. Yin et al. [23] used the obstacle detection algorithm to obtain parameters of obstacles, then used the behavioral dynamics method to plan local obstacle avoidance path. Xu et al. [24] proposed three obstacle avoidance strategies according to the volume and speed of different dynamic obstacles in the environment. However, the above studies did not specify the

magnitude of the volume and velocity of dynamic obstacles or only studied the case where dynamic obstacles and mobile robots were equally large or they do not consider the situation comprehensively enough. In practical applications, the size and speed of dynamic obstacles are critical to obstacle avoidance.

To overcome the shortcomings of existing algorithms and realize efficient and high-quality path planning for mobile robots, an improved ant colony algorithm is proposed in this paper. The algorithm sets the initial pheromone of uneven distribution according to the leading route which is created by the connection of the feature points, and it reduces the problem of blindness in the initial period. The bending suppression operator is introduced to improve heuristic information, which aims to optimize the smoothness of the path. The pheromone update rule is improved to increase the difference of pheromone concentration on different paths, and enhance the positive feedback effect of the algorithm. Three parameter dynamic adaptive control strategies are designed to coordinate the convergence and global search capability of the algorithm. The optimal parameters combination of the improved ant colony algorithm is determined by experimental and the experiments show that the aggregation pheromone structure can achieve a first complete solution and converge later more quickly, and has a higher efficiency and better performance.

Aiming at the problems existing in the path planning of the mobile robot in the dynamic environment. Firstly, using the improved algorithm to get the optimal path of the static environment, when the robot moves alone the optimal path, it can detect the dynamic obstacles' states, such as the volume and speed, then take corresponding obstacle avoidance strategies in time to avoid it until the robot reaches the end. Simulation researches show that, compared with other literatures, the obstacle avoidance strategies proposed in this paper is more comprehensive, and the path obtained after obstacle avoidance is better.

## 2. Description of basic ant colony algorithm

### *2.1. Environment model*

Grid maps are commonly used in mobile robot path planning due to the reasons that it is simple and can effectively represent the layout characteristics of the environment [25]. The grid map discretizes the environment into independent square grids, each grid has 2 states, free and occupied. The 0 value is used to represent the free area in the environment map, and the corresponding grid is filled with white; while the 1 value is used to represent the obstacle area, and the corresponding grid is filled with black. The grid model was placed into a two-dimensional coordinate system. And then serial number method is adopted to mark each grid, and the grid is numbered in order from upper to bottom, left to right to satisfy the needs of ant colony algorithm storage path, as shown in Figure 1.

In addition, for the convenience of research, the following assumptions are made about the mobile robot and its working environment:

(1) The location of the starting point, ending point, and static obstacles are known;

(2) Both the robot and the dynamic obstacle move in a straight line at a constant speed in units of grids and can switch between the constant straight-line motion and the stopped state;

(3) The robot is equipped with sensors can sense the state of dynamic obstacles, including moving direction, step length, size, and shape, but the states of different obstacles are randomly generated.

*2.2. The state transition probability*

In the process of ants searching the optimal path, the probability of the next node is selected by the heuristic value and the pheromone value of the path. At time $t$, the transition probability of ant $k$ selecting the next node $j$ from node $i$ is expressed in Eq (1):

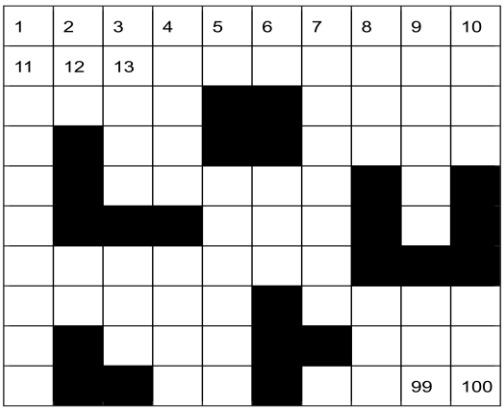


**Figure 1.** Grid environment map of mobile robot.

$$p_{ij}^{k}\begin{cases}\dfrac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}}{\sum\limits_{s\in allow_k}\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}}, s\in allow_k \\ 0 \qquad\qquad\qquad\qquad\qquad ,otherwise\end{cases} \tag{1}$$

where $allow_k$ is the set of next reachable nodes when ants in the grid $i$, $\alpha$ is the pheromone heuristic factor, indicating the degree of importance of pheromone, $\beta$ is the expected heuristic factor, which reflects the importance of the heuristic information on path. $\tau_{ij}$ is the pheromone concentration of path $(i, j)$, $\eta_{ij}$ is the heuristic value, and its expression is given in Eq (2):

$$\eta_{ij}(t)=\frac{1}{d_{ij}} \tag{2}$$

where $d_{ij}$ is the Euclidean distance between the current node $i$ and the candidate node $j$.

*2.3. The pheromone update rule*

Ants will leave pheromone on the path they pass, and some of these pheromones will volatilize as time flows. Eqs (3) and (4) are used to update the pheromone concentration on the path $(i, j)$ at time $t+1$.

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t) \tag{3}$$

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) \tag{4}$$

where $\rho$ is the global pheromone volatilization coefficient, $\rho \in (0,1)$, $\Delta\tau_{ij}(t)$ is the increment of the pheromone on the path $(i, j)$ in the current iteration is the concentration of left pheromone by ant $k$ on the path $(i, j)$, and $\Delta\tau_{ij}^{k}(t)$ is calculated according to Eq (5):

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{Q}{L_k}, tour(i, j)\in tour_k\\ 0 \quad ,otherwise\end{cases} \tag{5}$$

Among them, $Q$ is a constant, $L_k$ represents the total length of the path searched by the $k^{th}$ ant in this iteration.

## 3. To improve ant colony algorithm

### *3.1. To improve initial pheromones*

In the early stage of path planning, the difference in pheromone concentration in each node is small, so the positive feedback effect of the algorithm is not obvious, and the ants have some blindness in the path search, which leads to poor convergence and more time-consuming. This paper proposes a method to achieve the uneven distribution of initial pheromone by creating a collar route [26] and adding pheromone to the nodes that the collar route passes.

#### 3.1.1. The extraction rules of feature points

In this paper, several points are used to describe the different shapes of obstacle grids, and these points are called the feature points of obstacle grids. The extraction rules of feature points are as follows:

(1) The feature points of the convex barrier grid are all its vertices, as shown A in Figure 2

(2) The feature points of the concave barrier grid are only part of its vertices, and the vertices at the concave corners are removed, as shown B in Figure 2.

(3) The U-shaped obstacle grid is shown in Figure 2 as C and D. The characteristic points of this type of obstacle grid are all vertices, but the red vertices are removed.

(4) The obstacles which are at the boundary of the map as shown in (E, F, G), Figure 2, so that the vertices at the boundary of the map will be removed.

#### 3.1.2. The creation of the collar line

At first, using a straight line connecting the starting point to the goal point, as shown in the green line in Figure 3(a). If the straight line does not intersect any obstacle, the final path is the collar line. Other- wise, the obstacle that intersects the straight line is defined as the Class I obstacle, and extract the feature points of the Class I obstacle grid according to the rules. Then, starting from the starting point, judge the connection between the feature point and the starting point, among which there may be more than one feature point that can be connected to the starting point, and give priority to the feature point that is close to the ending point and connect it with the starting point, and then start from the selected feature point and repeat until a collar line from the starting point to the ending point is generated as shown in the red line in Figure 3(a).

In some environments there may be cases where the feature point cannot be connected to the start

point, the feature point cannot be connected to the feature point, and the feature point cannot be connected to the ending point, as shown in Figure 3(b)–(d). For case b and c, it is necessary to connect the closest unconnectable feature point to the current start point or feature point as shown in the blue

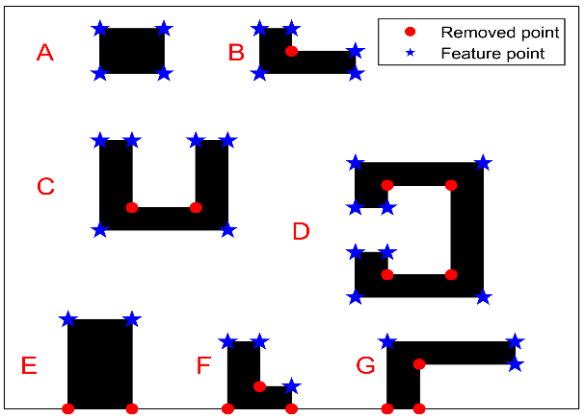


**Figure 2.** The feature points of obstacle grids.

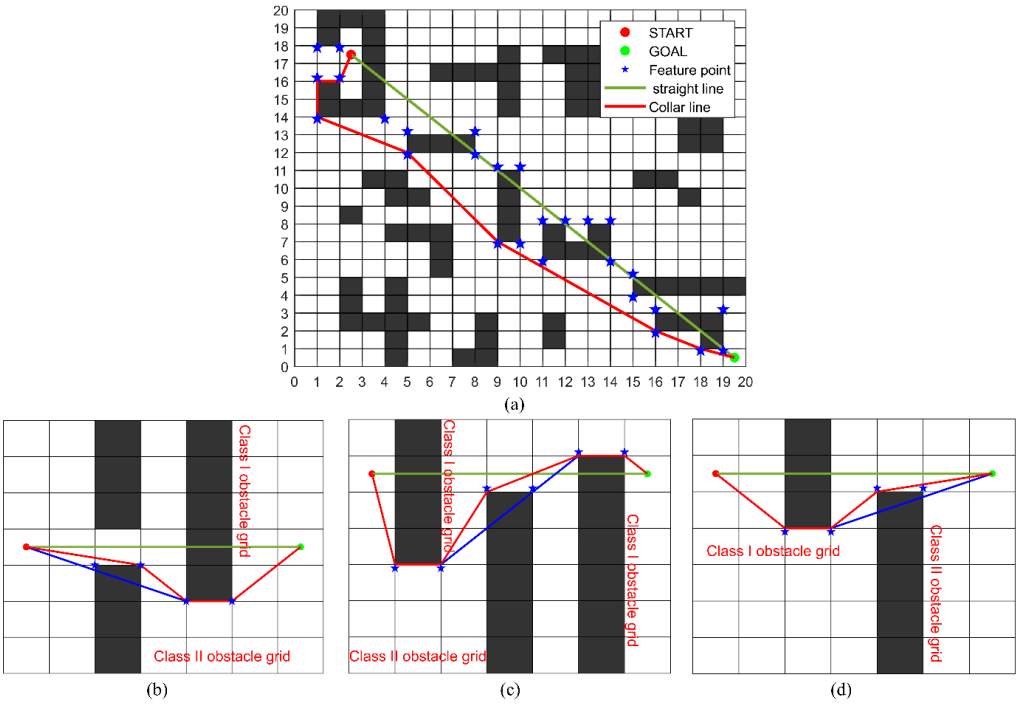


**Figure 3.** Creation of the collar line.

and d. For case d, it is necessary to connect the current feature point and the ending point, and the obstacles which the line passes are defined as class II obstacle and extract its feature points, and then continue to create the collar line from the current feature point.

### *3.2. To improve heuristic information function*

The heuristic information of the ACO is the reciprocal of the Euclidean distance between the

current node and the next optional node, which reduces the search efficiency and the ability to find the optimal path of ants. Therefore, this paper adds the guidance information of the target point and the bend suppression factor to the heuristic function to reduce the invalid path search and reduce the redundant turning point. The improvement heuristic information is

$$\eta_{ij}(t) = \frac{d_{ig}}{d_{ij} + d_{jg}}(\cos\omega + 1) \tag{6}$$

where $d_{ij}$ is the distance between the $i$ grid and the $j$ grid, $d_{jg}$ is the distance between the grid to be selected and the goal grid, $d_{ig}$ is the distance between the $i$ grid and the goal grid. $\omega$ is the angle between the line segment of node $i-1$ to node $i$ and the line segment of node $i$ to node $i+1$. The closer the angle is to 180°, the greater the $\eta_{ij}(t)$ is, which helps to reduce the turning points of the path.

### *3.3. To improve pheromone update rule*

After each iteration, the pheromone increment on different path of the ACO is the ratio of the pheromone intensity to the path length, this difference can be very small if the path lengths are close to each other, making it easy for the ants to fall into a local optimum in a complex environment. Therefore, this paper proposes an improved pheromone update rule, according to Eq (7) to distribute the pheromone intensity on different paths. After each iteration, only the pheromone of the ant path reaching the target point is updated according to Eqs (3),(4) and (8).

$$Q_m = t_c \cdot e^{(L_{GL} - L_m)^3} \cdot Q \tag{7}$$

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q_m}{L_m + turn}, tour(i,j) \in tour_m \\ 0 \quad ,otherwise \end{cases} \tag{8}$$

where $t_c$ is a constant, $L_{GL}$ denotes the length of the collar line, $L_m$ represents the length of the path searched by the ants that reach the end point, $\boldsymbol{turn}$ is the Total number of turns on the path.

### *3.4. Multiple parameter control strategies*

For ACO, $\alpha$ and $\beta$ are two important parameters that coordinate the algorithm's global search capability and convergence speed. In the early stage of the algorithm, the heuristic function is mainly used to guide the ants to search for the path. As the number of iterations increases, the pheromone concentration of each node increases. The pheromone should play a leading role in the selection of the path. In the later iterations, in order to avoid falling into the local optimization, the guiding effect of pheromone and heuristic function should be weakened at the same time to improve the randomness of the algorithm, therefore, $\alpha$ should be increased first and then decreased, and $\beta$ should be gradually decreased. Since most parameter control strategies only consider the number of iterations to judge the operating conditions of the algorithm and then to adjust the value of the parameters while not consider the actual operating conditions of the algorithm, it is likely to cause the adjustment of the parameters to lag or lead. Therefore, this paper designs a variety of parameter control strategies to adjust the

parameter values based on the length of the leading route, the length of the path found by the ants, and the number of iterations. The parameter control strategies proposed in this paper are as follows:

For $\alpha$:

$$\alpha(Nc) = a_{inital} + \delta \cdot \sin\left(\frac{\left(\frac{Nc}{N} + \frac{L_{GL}}{LF_{best} + L_{GL}}\right)}{2} \cdot \pi\right) \tag{9}$$

For $\beta$:

$$\beta(Nc) = \beta_{inital} \cdot e^{-2\left(\frac{\left(\frac{Nc}{N} + \frac{L_{GL}}{LF_{best} + L_{GL}}\right)}{2}\right)^2} \tag{10}$$

where $a_{inital}$ and $\beta_{inital}$ respectively represents the initial value of the parameter $\alpha$ and $\beta$, $\delta$ is the adjustment coefficient, the value is a positive number bigger than 0, $N$ is the maximum number of iterations, $N_c$ is the current number of iterations, $L_{GL}$ is the length of the collar line, $LF_{best}$ is the optimal path length found in the previous iterations. It can be seen from Eqs (9) and (10) that the parameters are jointly controlled by the number of iterations and path length, and the parameters change nonlinearly and dynamically with the number of iterations and path length to coordinate the algorithm's global search capability and convergence speed.

Meanwhile, in the process of finding the optimal path for the ant colony algorithm, the pheromone volatilization factor also affects the performance of the algorithm. This paper uses adaptive dynamic adjustment of pheromone volatilization factor is used to adjust the global search ability and convergence speed of the algorithm, and shown in Eq (11):

$$\rho = \kappa\left(\frac{L_{GL}}{L_{best} + L_{GL}}\right) \tag{11}$$

where $\kappa$ is the adjustment coefficient, the value is a positive number less than 1, with the higher quality of the found path, the volatilization factor $\rho$ gradually increases, which increases the pheromone concentration difference on different paths, enhances the guiding role of high-quality ants, which can improve the search speed of ant colony and make the algorithm converges rapidly.

### *3.5. Application of improved ant colony algorithm in path planning*

Step1: Build the grid map and initialize the number of ants m, the maximum number of iterations $N$, the pheromone heuristic factor $\alpha$, the expected heuristic factor $\beta$, the pheromone volatilization factor $\rho$, the strength of the pheromone $Q$ and other parameters.

Step2: Create the collar line and add the initial pheromone to the grid it passes through, then, record the length of the collar line.

Step3: Place the ant at the starting point, calculate the heuristic information according to Eq (6), and calculate the probability function according to Eq (11), finally, using the roulette method to select the next feasible grid. Then, turn to Step4.

Step4: If the ants reach the target grid, it will turn to Step5, otherwise it will turn to Step3.

Step5: The number of ants m = m + 1, and if all the ants have completed the path search, turn to Step6, otherwise turn to Step3.

Step6: Record the optimal path of this iteration, and update the path pheromone according to Eqs (3), (4) and (8).

Step7: Adaptively adjust parameters according to the Eqs (9)–(11).

Step8: Determine whether the end condition is satisfied. If it is satisfied, the optimal path length is output; otherwise, output optimal path and iterative curve; otherwise, let $N_c = N_c + 1$, go to Step3. and continue to circularly execute until the maximum number of iterations is reached.

## 4. To improve the local obstacle avoidance strategy

### *4.1. Collision types and regular processing method*

In a grid environment, the following collisions may occur between robots and dynamic obstacles. (Only obstacles in uniform linear motion are considered in this paper)

(1) The trajectory of the dynamic obstacle is not in the same line as the trajectory of the robot.

(2) The trajectory of the dynamic obstacle and the trajectory of the robot is in the same line and in the opposite direction. (At this point, the obstacle and the robot will collide face to face)

(3) The trajectory of the dynamic obstacle and the trajectory of the robot is in the same line and in the same direction (The dynamic obstacle is chasing the robot and moves faster than robot).

(4) The trajectory of the dynamic obstacle and the trajectory of the robot is in the same line and in the same direction (The robot is chasing the dynamic obstacle and moves faster than dynamic obstacle).

Four corresponding dynamic obstacle avoidance strategies are proposed for each of the above four scenarios.

In order to facilitate the design of obstacle avoidance strategies for dynamic obstacles with different shapes, first conduct regularization processing to obtain the trajectory of dynamic obstacles. As shown in Figure 4, it shows the original shape of the obstacle and the red line represents the moving direction of the obstacle, then extract the feature points of the dynamic obstacle based on the information detected by the sensor and the method proposed, and then select the two feature points that are the furthest vertical distance from the direction of movement on both sides of the dynamic obstacle, and according to the selected feature points, the dynamic obstacles are regularized into a part of the $n \times n$ grid, as shown in Figure 4(b), the blue dotted line represents the trajectory of the dynamic obstacle.

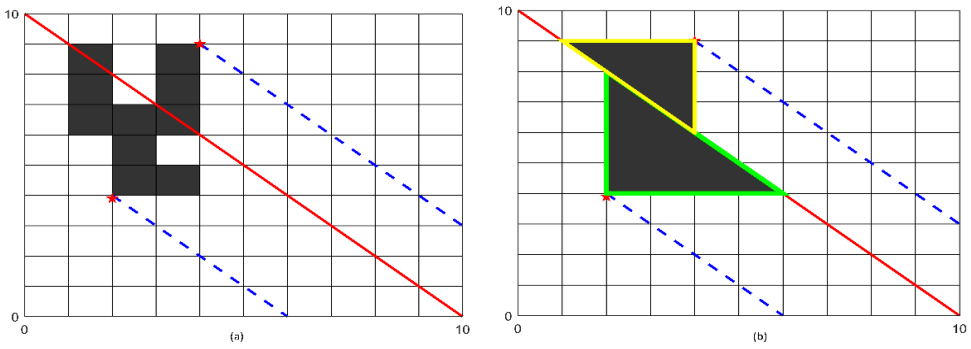


**Figure 4.** Regular processing method.

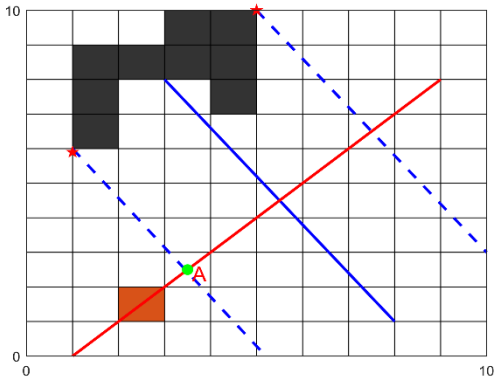


**Figure 5.** Obstacle avoidance strategy for scenario 1.

*4.2. Obstacle avoidance strategies in different situations*

For scenario1, the wait-and-avoid strategy proposed by [24] can be used, as shown in Figure 5, A represents the collision point, and the robot stops at the red grid waits for the obstacle to pass, and then continues to move along the original path after ensuring that it will not collide with the obstacles.

$$X = floor\left(\frac{Ld}{\sqrt{2}}\right)Step_{scale} - mod\left(\frac{Grid_{scale}}{2}\right) \le ng \tag{12}$$

For scenario 2, for the convenience of research, suppose that the step length of the robot is 1 grid and it occupies 1 grid. The ratio of the step length of the dynamic obstacle to the step length of the robot is $Step_{scale}$ and the speed of the dynamic obstacle is greater than the speed of the robot. After the regularization process, the side length of the grid occupied is $Grid_{scale}$, and the distance from the robot to the dynamic obstacle when the robot finds the dynamic obstacle is $ng$ grids. At the same time, assuming that the grid coordinates where the robot is located is $(X_{rob}, Y_{rob})$.The precondition for a robot to avoid dynamic obstacles is shown in Eq (12):

If the side length of the obstacle grid after regularization is an even multiple of the robot, it still needs to satisfy that there are non-obstacle grids in the 6 grids in Eq (13) and there is a path to the ending point in these grids

$$\left.\begin{array}{l}(x_{rob}-2, y_{rob}-1)、(x_{rob}-2, y_{rob}-2)\\(x_{rob}-1, y_{rob}-2)、(x_{rob}+2, y_{rob}+1)\\(x_{rob}+2, y_{rob}+2)、(x_{rob}+1, y_{rob}+2)\end{array}\right\} \not\subset obsgrid \tag{13}$$

If the side length of the obstacle grid after regularization is an odd multiple of the robot, it still needs to satisfy that there are non-obstacle grids in the 2 grids in Eq (14) and there is a path to the ending point in these grids.

$$\left.\begin{array}{l}(x_{rob}-2, y_{rob}-2)、\\(x_{rob}+2, y_{rob}+2)\end{array}\right\} \not\subset obsgrid \tag{14}$$

$$X \le ng < X + V_{rob} + V_{obs} \tag{15}$$

If $ng$ s atisfies Eq (15), then the obstacle avoidance strategy designed below needs to be

implemented at the grid where it is currently located.

$$X + V_{rob} + V_{obs} \le ng \tag{16}$$

If $ng$ satisfies Eq (16), the robot can continue walking along the previously planned path with the number of grids D as shown in Eq (17) and then execute the obstacle avoidance strategy designed below.

$$ng - X + V_{rob} + V_{obs} \in \left[ \{0,1,2,3\} + D(V_{rob} + V_{obs}) \right] \tag{17}$$

The obstacle avoidance strategies designed in this paper will be illustrated in conjunction with the examples shown in Figure 6. Since the ratio of the size of the dynamic obstacle to the volume of the robot is different, the obstacle avoidance strategy adopted will be different. Therefore, the obstacle avoidance strategies are designed in two situations.

The obstacle avoidance strategies designed in this paper will be illustrated in conjunction with the examples shown in Figure 6. Since the ratio of the size of the dynamic obstacle to the volume of the robot is different, the obstacle avoidance strategy adopted will be different. Therefore, the obstacle avoidance strategy is designed in two situations.

Assuming that the size of the obstacle is 4 times or even times that of the robot, and the step length is 3 times that of the robot, as shown in Figure 6(a). In the Figure 6, the blue square represents the dynamic obstacle, the green square represents the robot, the red solid line represents the planned path before the robot encounters the obstacle, and the blue dotted line is the movement trajectory of the dynamic obstacle. At a certain moment, the position of the obstacle and the robot is shown in Figure 6(a). At this time, the two objects are 6 grids apart and satisfy the conditions of Eqs (12) and (14), so the obstacle avoidance strategy is executed at the current position, the next moment, the position of the obstacle and the robot is shown in Figure 6(b), and there will be no collision at this time. At the next moment, the robot can move to one of the three positions 3–1, 3–2 and 3–3 in Figure 6c, the turning angle is limited to $\pm 90°$. At this time, the three grids available for selection plus the three grids on the upper right A total of 6 grids need to meet the condition of Eq (12), and then replan the path with the non-obstacle grid as the starting point, and take the shortest path as the path for the robot to move.

If the size of the obstacle is 9 times or odd times that of the robot, and the step length is 3 times that of the robot, as shown in Figure 7, the obstacle avoidance strategy is different. The robot only needs to move 2 grids along the $\pm 90°$ direction to avoid dynamic obstacles, and then replan the path in the grid at the position in Figure 7(c). ③

The difference between the obstacle avoidance strategy adopted when the dynamic obstacle is an odd or even multiple of the robot's size is the position the robot is in after it has moved one grid after discovering the dynamic obstacle. As shown in Figure 6, for the odd multiple case, the robot is at position 2 in the Figure 6(b) after moving one grid, which is inside the trajectory of the dynamic obstacle , while for the even multiple case, the robot is at position in Figure 7(b), which is on the trajectory of the dynamic obstacle, the next position can be moved towards 3–1, 3–2 and 3–3 in the Figure 8(c) to avoid the dynamic obstacle, while the robot in position 2 in Figure 6(b) can only avoid the dynamic obstacle by moving towards the position where 3 is located Figure 6(c).

For the odd multiple case, if adopts the same avoidance strategy as the even multiple case would require the robot to find the obstacle further away from it or the robot will hit the dynamic obstacle as shown in Figure 8. In Figure 8, from Figure 8(a) to Figure 8(d), it represents the process of avoiding obstacles when the robot adopts the even-numbered strategy in odd-numbered situations, and will

eventually collide with the obstacle, so the obstacle-avoiding strategy in even-numbered situations cannot be adopted, or which would undoubtedly place greater demands on the robot's performance.

For scenario 3, the same obstacle avoidance strategy is adopted as for case 2, simply by replacing Eqs (15)–(17) with Eqs (18)–(20) and then implementing the corresponding designed obstacle avoidance strategies.

$$X \le ng < X + V_{obs} - V_{rob} \tag{18}$$

$$X + V_{obs} - V_{rob} \le ng \tag{19}$$

$$ng - X + V_{obs} - V_{rob} \in \left[\{0,1,2,3\} + D(V_{obs} - V_{rob})\right] \tag{20}$$

For scenario 4, in this situation, the robot only needs to use an obstacle avoidance strategy similar to that in case 2 at a distance of *rg* grids from the obstacle. (*rg* means that the speed of the robot is *rg* times the speed of the obstacle)

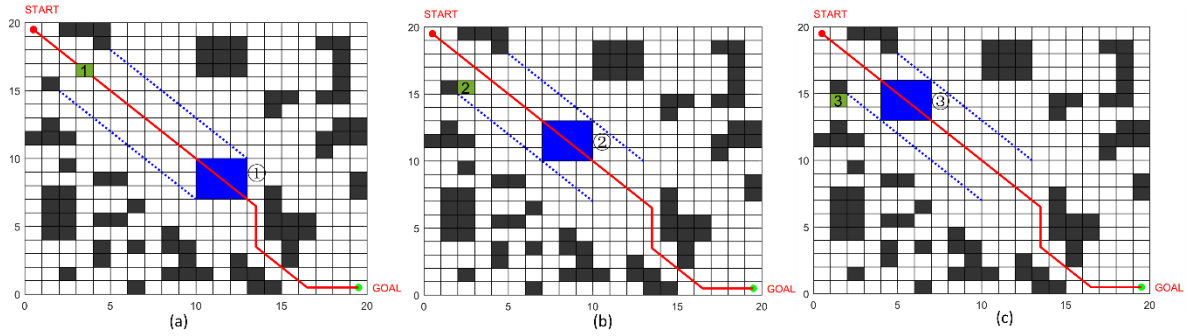


**Figure 6.** Obstacle avoidance strategy for scenario 2(odd times of the robot).

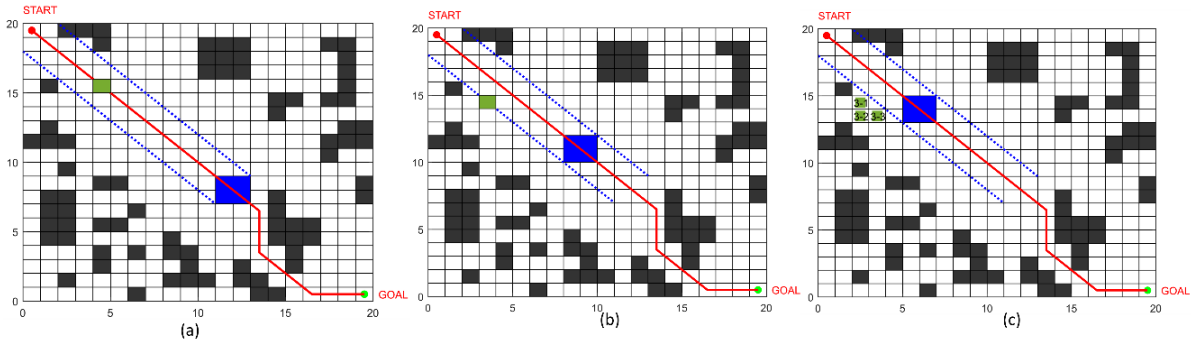


**Figure 7.** Obstacle avoidance strategy for scenario 2(even times of the robot).

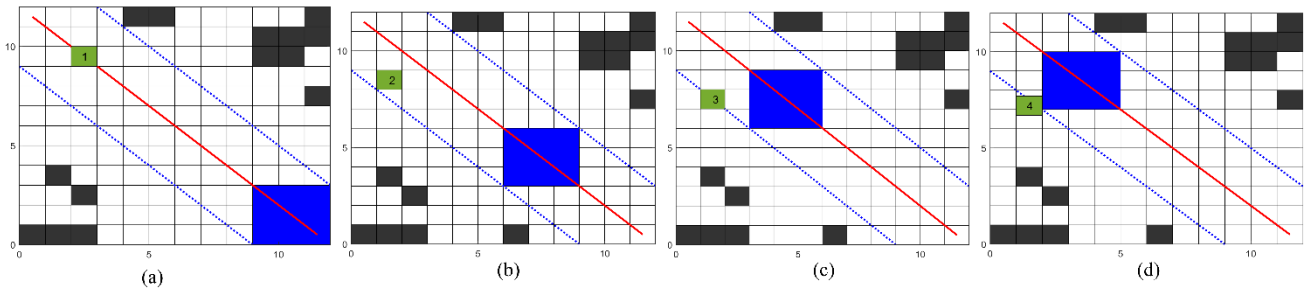


**Figure 8.** The situation when an even-time obstacle avoidance strategy is adopted when the volume of the obstacle is an odd multiple of the robot.

### *4.3. Path planning method based on improved ant colony and dynamic environment*

Combining the environmental model, the improved ant colony algorithm and the dynamic obstacle avoidance strategy, the main steps of the path planning method proposed in this paper are as follows.

Step1: Build the grid map and create the collar route based on known static environmental information.

Step2: The robot uses the improved ant colony algorithm in this paper to plan an optimal path in a static environment.

Step3: The robot obtains the information of the dynamic environment at the current starting point through its sensors, and executes the corresponding collision avoidance strategy according to different dynamic obstacles based on the dynamic environment obstacle avoidance strategies described in Sect 4, and reaches a new starting point; otherwise, follow the original path to reach the new starting point.

Step4: If the robot reaches the ending point, the path planning ends; otherwise, return to Step3.

## 5. Simulation experiments and analysis

The simulation environment is as follows: Windows10 64 bit; processor AMD Ryzen 7 5800H; main frequency 3.2 GHz; memory 8 GB; simulation software: Matlab R2018b.

### *5.1. Main parameter selection*

Since there are no precise mathematical analysis methods of the ACO model has so far been developed to generate optimal parameter settings in different situation, the parameters are generally chosen empirically. Nowadays, the commonly used methods are mainly based on experience and repeated trial through many experiments.

This paper mainly selects several important parameters that play a role in determining the performance of the algorithm: pheromone heuristic factor, expected heuristic factor, adaptive pheromone volatility，adaptive pheromone intensity.

Firstly, set the value range of each parameter: $\alpha \in \{0, 0.5, 1, 2, 4\}$ , $\beta \in \{0, 2, 5, 7, 9\}$ , $t_c \in \{1, 1.5, 2, 3, 5\}$ , $\kappa \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ . The default value of a group of parameters is set to $\alpha = 1$ , $\beta = 5$ , $t_c = 2$ , $\kappa = 0.5$ . Only one parameter is changed for each experiment. E.g. when calculating simulation results for $\alpha = 0$ , all other parameters are taken as default values. Each parameter setting is simulated 10 times to avoid chance, and compares and analyzes the mean values of the simulation results of the algorithm. The corresponding experimental results for each group are shown in Table 1.

From the experimental results in Table 1, it can be seen that the optimal value of $\alpha$ is around 1, the optimal value of $\beta$ is around 7, the optimal value of $t_c$ is around 1.5, the optimal value of $\kappa$ is around 0.7.

### *5.2. Algorithm comparison and analysis in a static environment*

The experiment was divided into four parts according to four types of maps and three algorithms [19], the algorithm in [24] and the improved ant colony algorithm proposed in this paper are simulated on each map in turn. Each algorithm was executed 50 times to obtain the algorithm performance comparison table. The values of the experimental parameters are shown in Table 2.

**Table 1.** Main parameter optimization experiment results.

| $\alpha$ | 0 | 0.5 | 1 | 2 | 4 |
|---|---|---|---|---|---|
| Mean value of path length | 58.254 | 51.298 | 48.531 | 49.378 | 53.854 |
| $\beta$ | 0 | 2 | 5 | 7 | 9 |
| Mean value of path length | 65.624 | 53.274 | 49.591 | 50.325 | 52.951 |
| $t_c$ | 1 | 1.5 | 2 | 3 | 5 |
| Mean value of path length | 57.936 | 49.292 | 50.836 | 51.951 | 53.469 |
| $\kappa$ | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
| Mean value of path length | 52.861 | 50.197 | 49.178 | 48.568 | 49.625 |

**Table 2.** Main parameters of simulation experiment.

| Parameter | $\alpha$ | $\beta$ | $\rho$ | $Q$ | $m$ | $N$ | $t_c$ | $\delta$ | $\kappa$ |
|---|---|---|---|---|---|---|---|---|---|
| Value | 1 | 5 | 0.3 | 100 | 50 | 100 | 1.5 | 2 | 0.7 |

Figure 9 is a path map generated by three algorithms in a simple environment. As shown in Figure 9 and Table 3, all three algorithms find the shortest path with a length of 28.6274 m, however, there are only 6 turning points in the algorithm of this paper, 7 in [19] and 7 in [24]. The algorithm in this paper has certain advantages in convergence speed and path smoothness.

Figure 10 is a path map generated by three algorithms in a complex environment. As shown in Figure 10 and Table 3, the optimize length of the improved ant colony algorithm is 73.9828, and the number of bending times is 12. In the shortest path length, the improved ant colony algorithm is basically as the same as the algorithm in [19], but the algorithm in [24] can't find the global optimal path. In the number of bending times, it is 14% decrease than the algorithm in [19], and as shown in Table 3, the algorithm in this paper has obvious advantages in convergence speed.

Figure 11 is a path map generated by three algorithms in the baffle environment. The baffle environment is prone to occur deadlock problem due to the small space. As shown in Figure 11 and Table 3, the algorithm in this paper still plans the shortest path without redundant turning points, which length is 115.87m. Both the algorithm in literature [19] and [24] can't search the global optimized path, the algorithm in paper has faster convergence speed and less running time than the algorithm of this paper.

Figure 12 is a path map generated by three algorithms in the trough environment. As shown in Figure 12 and Table 3, compared with the algorithm in [19], the path length is reduced by 1.415, compared with the algorithm in [24], the turning point of this algorithm is reduced by 12 and the path length is reduced by 6.928. The optimal path length of the algorithm in this paper converges to 62.527, while the average path length of the algorithm in [19] is 70.847, which further illustrates that the algorithm of this paper is more stable.

Figure 13 is a path map generated by three algorithms in the complex environment. As shown in Figure 13 and Table 3, the algorithm in this paper can still quickly search for the optimal path. The optimal path length of the algorithm in [19] is 84.0833, while the optimal path length is 78.8112 in this paper. The path length is shortened 5.2721; the average iteration in [19] is 75times, and the average iteration of this algorithm is 30 times. And the convergence speed is increased nearly double times. The number of bending times is shortened 14.

Figure 14 is a path map generated by three algorithms in the environment which combine the maps above. As shown in Figure 14 and Table 3, compared with the algorithm in [19], the path length

is reduced by 12.4437, the average iteration in [19] is 80 times, and the average iteration of this algorithm is 35 times. And the time consuming also decreased a lot compared with the algorithm in [19] and in this paper.

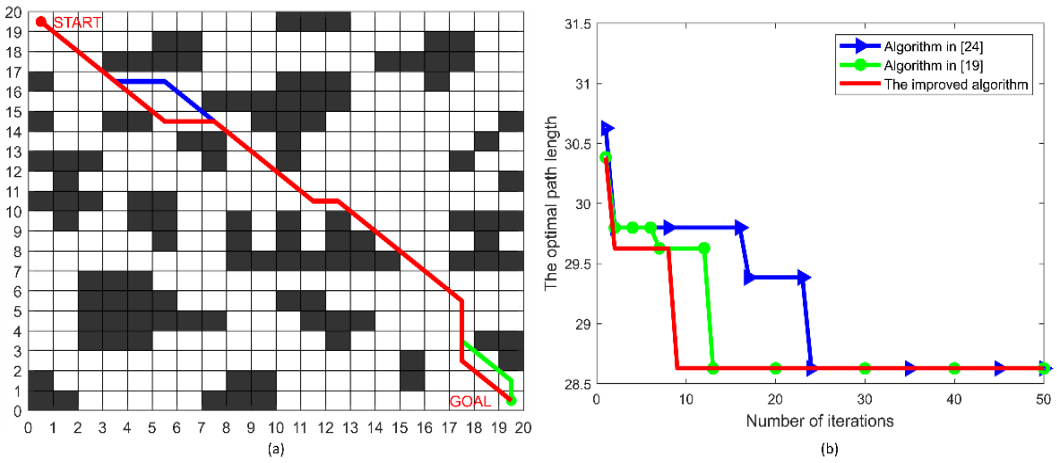


**Figure 9.** Comparison of path planning results on map 1.

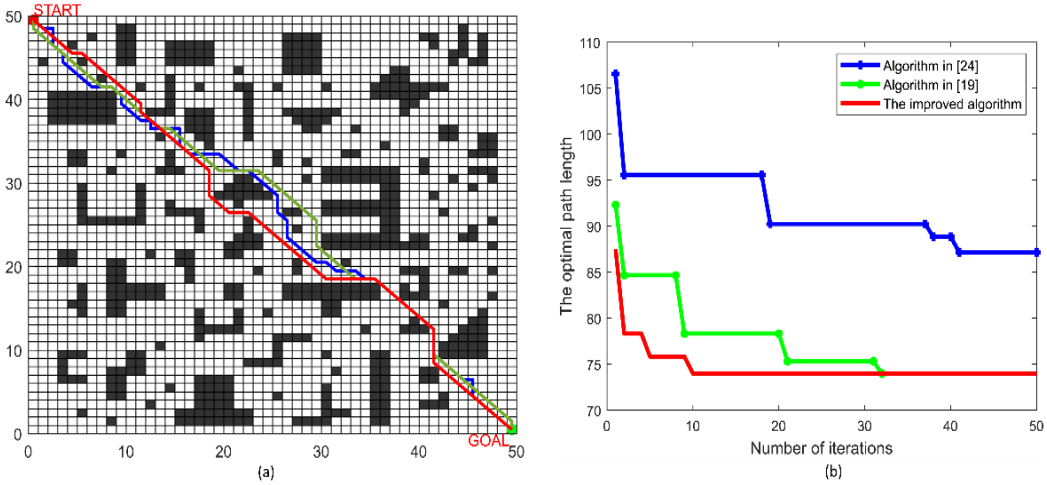


**Figure 10.** Comparison of path planning results on map 2.

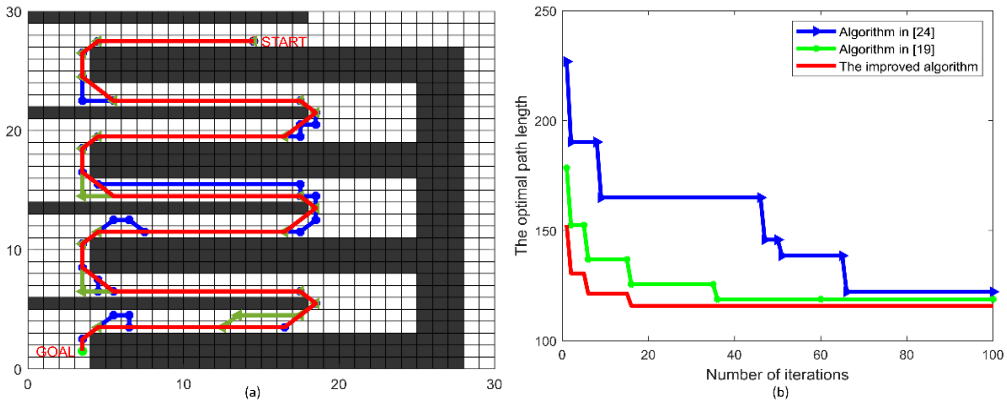


**Figure 11.** Comparison of path planning results on map 3.

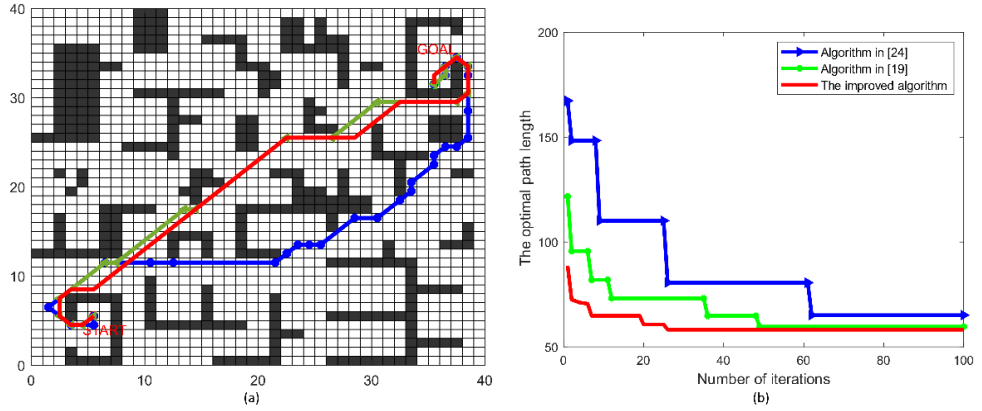


**Figure 12.** Comparison of path planning results on map 4.

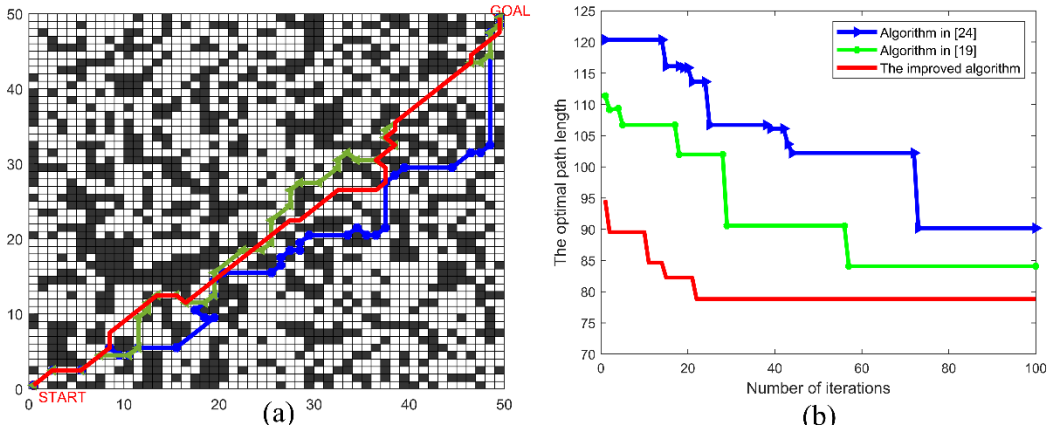


**Figure 13.** Comparison of path planning results on map5.

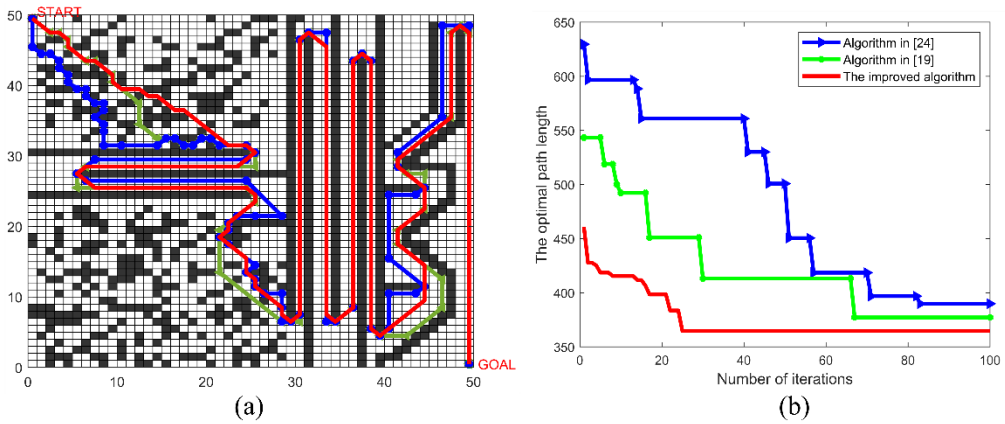


**Figure 14.** Comparison of path planning results on map 6.

The results of the three algorithms that run 50 times in the same map environments are shown in Table 3. The algorithm in [19] and [24] have large fluctuations in the path length and the number of inflection points during the operation of the algorithm. The algorithm in this paper can find the global optimal path more stably, with fewer inflection points, higher quality, and faster convergence, which significantly reduces the optimization time and the improvement of the algorithm is more obvious on

large-scale complex maps.

**Table 3.** Comparison of different algorithm performance on different map.

| Map | Algorithm | Optimal path length | Average of path length | Standard deviation | Average number of iterations | Average time-consuming(sec) | Number of bends |
|---|---|---|---|---|---|---|---|
| 1 | ① | 28.6274 | 28.6274 | 0 | 14 | 1.35 | 6 |
| | ② | 28.6274 | 28.6557 | 0.0283 | 20 | 1.48 | 8 |
| | ③ | 28.6274 | 29.0127 | 0.3853 | 32 | 3.37 | 11 |
| 2 | ① | 73.9828 | 74.2296 | 0.2468 | 13 | 4.04 | 12 |
| | ② | 73.9828 | 77.0538 | 3.071 | 35 | 7.91 | 18 |
| | ③ | 77.6694 | 90.5694 | 12.9 | - | 18.23 | 33 |
| 3 | ① | 115.87 | 119.35 | 3.48 | 27 | 4.68 | 32 |
| | ② | 118.799 | 132.26 | 13.461 | 45 | 11.01 | 35 |
| | ③ | 122.213 | 148.84 | 26.627 | 83 | 23.6 | 41 |
| 4 | ① | 58.183 | 62.527 | 4.344 | 38 | 12.32 | 18 |
| | ② | 59.598 | 70.847 | 11.249 | 59 | 19.56 | 23 |
| | ③ | 65.111 | 85.247 | 20.136 | 81 | 44.83 | 40 |
| 5 | ① | 78.8112 | 87.5614 | 8.7502 | 30 | 20.45 | 21 |
| | ② | 84.0833 | 103.1546 | 19.0713 | 75 | 33.54 | 35 |
| | ③ | 90.1838 | 126.154 | 35.9702 | 85 | 50.57 | 34 |
| 6 | ① | 364.7523 | 395.146 | 30.3937 | 35 | 40.56 | 59 |
| | ② | 377.196 | 426.545 | 49.349 | 80 | 70.48 | 55 |
| | ③ | 389.8712 | 460.254 | 70.3828 | 97 | 110.32 | 71 |

Note: ①: Algorithm in this paper.②: Algorithm in paper [24]. ③: Algorithm in paper [19]

**Table 4.** Results of Wilcoxon signed rank tes.

| Map | Algorithm in paper [24] | | Algorithm in paper [19] | |
|---|---|---|---|---|
| | Z | P | Z | P |
| 1 | –4.298 | 1.700E–05 | –4.623 | 4.000E–06 |
| 2 | –4.141 | 3.500E–05 | –4.782 | 2.000E–06 |
| 3 | –4.247 | 2.200E–05 | –4.782 | 2.000E–06 |
| 4 | –3.795 | 1.480E–04 | –4.556 | 5.000E–06 |
| 5 | –4.433 | 9.000E–06 | –4.762 | 2.000E–06 |
| 6 | –3.034 | 2.000E–03 | –4.330 | 1.500E–05 |

### *5.3. Wilcoxon rank sum test*

Only using the above test method to test the algorithm in this paper cannot fully prove its significant optimization performance. Thus, statistical test is used to judge about the significance of results here, and 50 times results of three algorithms on the metric of path length are tested by Wilcoxon

rank sum test to obtain p value. That is, whether the algorithm in this paper is significantly different from other algorithms under the standard of p = 0.05, the original hypothesis H0 is that there is no significant difference between the two algorithms. When p < 0.05, the assumption of H0 is rejected, indicating that there is a significant difference between the two algorithms; when p > 0.05, H0 is accepted, indicating that the difference between the two algorithms is not significant, and the algorithms have the same optimization ability. The specific test results are shown in Table 4.

According to Table 4, the p-values are all less than 0.05, indicating that there is a significant difference between the algorithm in this paper and the other algorithms.

### *5.4. Comparison of obstacle avoidance strategies in a dynamic environment*

This paper will verify the feasibility of the proposed dynamic environment obstacle avoidance strategy through the following simulation examples. In this simulation experiment, the red grid represents the robot, and other color grids represent dynamic obstacles in different states. The starting and ending points have been marked, as shown in Figure 15.

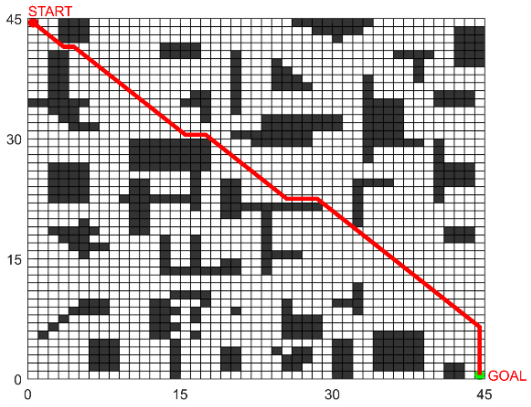


**Figure 15.** The optimal path in a static environment.

Firstly, the robot uses the improved ant colony algorithm to plan a global optimal path from the starting point to the ending point in the static environment, as shown by the red line in Figure 15. Then the robot moves along the planned path, and detects environmental information every time it moves a grid, then executes the corresponding obstacle avoidance strategy according to the information of the detected dynamic obstacles (The obstacle avoidance strategies in case 1 and case 4 are relatively simple, so no corresponding simulation experiments have been done).

At time $t_1$, the robot moves to the position shown in A in Figure 16(a) and the sensors detect a dynamic obstacle Obs1 (occupying 9 green grids) moving towards the robot at a rate of 3 grids per step, its initial and end positions are marked in the Figure 16a and the green dashed line indicates its trajectory. At this point, the robot is 5 grids away from the obstacle and chooses to execute the obstacle avoidance strategy for the odd number of grids in case 2, and the re-planned path is shown by the blue dotted line in the Figure 16(a). If the positive obstacle avoidance strategy proposed in [24] is used, the robot will not be able to avoid the obstacle regardless of when its sensors detect the state of the dynamic obstacle.

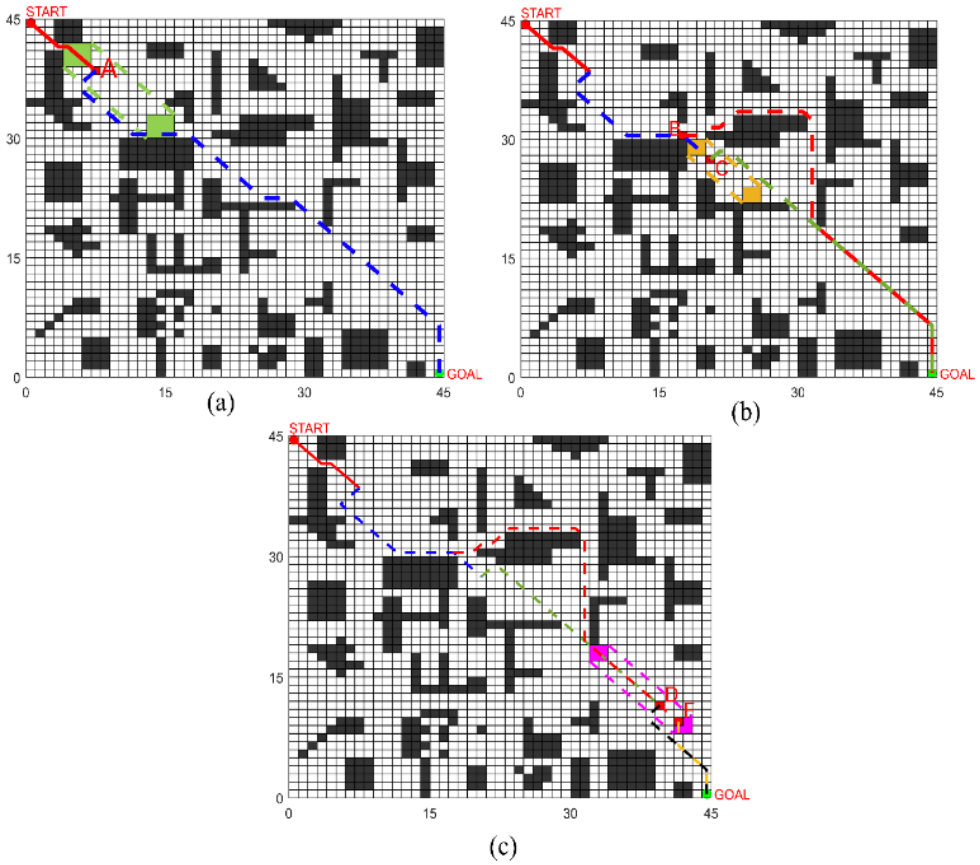


**Figure 16.** Comparison of obstacle avoidance strategies in a dynamic.

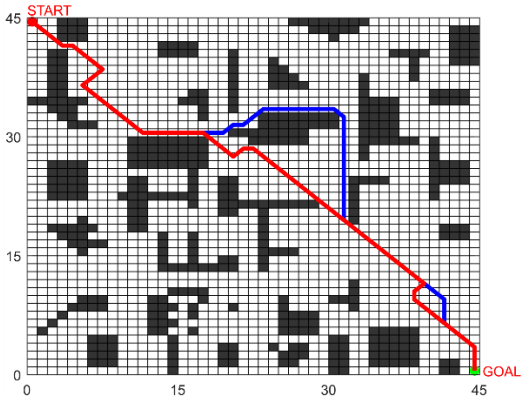


**Figure 17.** Path comparison after obstacle avoidance.

At time $t_2$, the robot moves to the position shown in C in Figure 16(b), and the sensors detect a dynamic obstacle Obs2 (occupying 4 yellow grids) moving towards the robot at a rate of 2 grids per step, its initial and end positions are marked in the Figure 16(b) and the yellow dashed line indicates its trajectory. At this point, the robot is 3 grids away from the obstacle and chooses to execute the obstacle avoidance strategy for the even number of grids in case 2, and the re-planned path is shown by the green dotted line in the Figure 16(b). If the collision avoidance strategy proposed in [24] is adopted, the robot's sensor needs to detect the state of the dynamic obstacle at position B in the Figure 16(b), and start to replan the path to avoid the obstacle at this position. The path is shown by the red

dotted line in the Figure 16(b). It can be seen from the Figure 16(b) that, compared with the obstacle avoidance strategy in [24], the path length re-planned after obstacle avoidance in this paper is shorter, and the number of inflection points is also less.

At time $t_3$, the robot moves to the position shown in D in Figure 16(c), and the sensors detect a dynamic obstacle Obs3 (occupying 4 purple grids) moving towards the robot from behind at a rate of 3 grids per step, its initial and end positions are marked in the Figure 16c and the purple dashed line indicates its trajectory. At this point, the robot is 5 grids away from the obstacle and chooses to execute the obstacle avoidance strategy for the even number of grids in case 3, and the re-planned path is shown by the black line in the Figure 16(c). If the positive obstacle avoidance strategy proposed in [24] is used, the robot's sensor needs to detect the state of the dynamic obstacle at position F in the Figure 16c and start to replan the path to avoid the obstacle at this position. The path is shown by the yellow dotted line in the Figure 16(c).

Figure 17 shows a comparison of the global optimal paths for the obstacle avoidance strategies designed in this paper and in [24]. The red solid line in the Figure 17 is the path planned after the obstacle avoidance strategy of this paper, and the blue solid line is the path after obstacle avoidance of the obstacle avoidance strategy in [24] (if the obstacle cannot be avoided, the strategy of this article will be used instead). Compared with the obstacle avoidance strategy in [24], this paper proposes a method of regular processing for different shapes of obstacles, and fully considers the position of static obstacles in the environment and the state of dynamic obstacles to design different obstacle avoidance strategies, the robot is able to avoid obstacles even when it is close to dynamic obstacles, so the requirements for sensors are lower, the safety is higher, the path distance after obstacle avoidance is also shorter, and the number of inflection points is also fewer.

## 6. Conclusions

This paper proposes an improved ant colony algorithm for robot path planning in a static environment. Firstly, the feature points of different types of obstacle grids are extracted according to the known environmental information, and then a leading route from the starting point to the ending point is planned to solve the problem that the blindly searches in the initial stage to improve the convergence speed. A heuristic function that considers the starting point, ending point and turning point at the same time is designed to improve search efficiency and smoothness of path. Three different parameter control strategies are designed to coordinate the convergence and global search ability of the algorithm. Different obstacle avoidance strategies are then proposed for dynamic obstacles of different shapes and states in the dynamic environment to plan a collision-free optimal path for the robot from the start to the end.

Simulation experiments under different scales and complexity environments show that, compared with the algorithm in [19] and the algorithm in [24], the algorithm in this paper has effectively improvement in reducing the path length and the number of redundant corners, accelerating the iterative convergence speed and operation efficiency. The grid model based on feature point extraction eliminates some poor solutions in the original grid model, so that the ant colony algorithm does not need to consider these poor solutions in path planning, and the path selection is more purposeful and can obtain better quality of the initial solution and greatly reduces the computational time of the algorithm.

Simulation and comparison with other dynamic obstacle avoidance strategies in the dynamic environment show that the proposed strategy can fully consider the location of static obstacles and the shape of dynamic obstacles in the environment, and the paths planned after obstacle avoidance are

shorter in length, have fewer turning points and can avoid obstacles even when they are close to the dynamic obstacles, which indicates that the strategy in this paper has lower requirements for sensor performance and higher security .

**Acknowledgments**

This work was funded by the National Key R&D Program of China. (2018YFB2000700).

**Conflict of interest**

The authors declare that there is no conflict of interest.

**References**

1. B. K. Patle, G. Babu L, A. Pandey, D. R. K. Parhi, A. Jagadeesh, A review: On path planning strategies for navigation of mobile robot, *Def. Technol.*, **15** (2019), 582–606. https://doi.org/10.1016/j.dt.2019.04.011
2. A. Bakdi, A. Hentout, H. Boutami, A. Maoudj, O. Hachour, B. Bouzouia, Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control, *Robot. Auton. Syst.*, **89** (2017), 95–109. https://doi.org/10.1016/j.robot.2016.12.008
3. G. Luo, J. Yu, Y. Mei, S. Zhang, UAV path planning in mixed-obstacle environment via artificial potential field method improved by additional control force, *Asian J. Control*, **17** (2015), 1600–1610. https://doi.org/10.1002/asjc.960
4. S. M. Persson, I. Sharf, Sampling-based A* algorithm for robot path-planning, *Int. J. Robot. Res.*, **33** (2014), 1683–1708. https://doi.org/10.1177/0278364914547786
5. M. Elhoseny, A. Tharwat, A. E. Hassaniene, Bezier curve based path planning in a dynamic field using modified genetic algorithm, *J. Comput. Sci.*, **25** (2018), 339–350. https://doi.org/10.1016/j.jocs.2017.08.004
6. E. S. Low, P. Ong, K. C. Cheah, Solving the optimal path planning of a mobile robot using improved Q-learning, *Robot. Auton. Syst.* **115** (2019), 143–161. https://doi.org/10.1016/j.robot.2019.02.013
7. S. Hosseininejad, C. Dadkhah, Mobile robot path planning in dynamic environment based on cuckoo optimization algorithm, *Int. J. Adv. Robot. Syst.*, **16** (2019), 1–13. https://doi.org/10.1177/1729881419839575
8. L. Guangsheng, C. Wusheng, Path planning for mobile robot using self-adaptive lea-rning particle swarm optimization, *Sci. China Inf. Sci.*, **61** (2018), 267–284. https://doi.org/10.1007/s11432-016-9115-2
9. F. H. Ajeil, I. K. Ibraheem, M. A. Sahib, A. J. Humaidi, Genetic algorithm based approach for autonomous mobile robot path planning, *Procedia Comput. Sci.*, **127** (2018), 180–189. https://doi.org/10.1016/j.procs.2018.01.113
10. L. Jianhua, Y. Jianguo, L. Huaping, T. Xingjun, G. Meng, An improved ant colony algorithm for robot path planning, *Soft Comput.*, **21** (2017), 5829–5839. https://doi.org/10.1007/s00500-016-2161-7
11. A. Viseras, R. O. Losada, L. Merino, Planning with ants: Efficient path planning with rapidly exploring random trees and ant colony optimization, *Int. J. Adv. Robot. Syst.*, **13** (2016), 1–16. https://doi.org/10.1177/1729881416664078

12. Y. Zheng, L. Qiang, H. Wang, C. Wang, X. Chen, Path planning based on improved ant colony algorithm with potential field heuristic, *Control Decision*, **33** (2018), 1775–1781.
13. Y. Zheng, L. Qiang, H. Wang, C. Wang, X. Chen, Path planning of mobile robot based on adaptive ant colony algorithm, *J. Intell. Fuzzy Syst.*, **1** (2020), 5329–5338. https://doi.org/10.3233/JIFS-189018
14. Z. Jiao, K. Ma, Y. Rong, P. Wang, H. Zhang, S. Wang, A path planning method using adaptive polymorphic ant colony algorithm for smart wheelchairs, *J. Comput. Sci.*, **25** (2018), 50–57. https://doi.org/10.1016/j.jocs.2018.02.004
15. K. Akka, F. Khaber, Mobile robot path planning using an improved ant colony optimization, *Int. J. Adv. Robot. Syst.*, **15** (2018),1–7. https://doi.org/10.1177/1729881418774673
16. Y. XiaoMing, L. Sheng, Z. Chen, An improved ant colony system algorithm for robot path planning and performance analysis, *Int. J. Robot. Autom.*, **33** (2018), 527–533.
17. D. Wang, H. Yu, Path planning of mobile robot in dynamic environments, in *International Conference on Intelligent Control and Information Processing (ICICIP)*, IEEE, **2** (2011), 691–696. https://doi.org/10.1109/ICICIP.2011.6008338
18. Q. Hong, H. Liwei, X. Ke, Research of improved ant colony based robot path planning under dynamic environment, *J. Univ. Electron. Sci. Technol. China*, **44** (2015), 260–265.
19. Q. Luo, H. Wang, Y. Zheng, J. He, Research on path planning of mobile robot based on improved ant colony algorithm, *Neural Comput. Appl.*, **32** (2020), 1555–1566. https://doi.org/10.1007/s00521-019-04172-2
20. C. Miao, G. Chen, C. Yan, Y. Wu, Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm, *Comput. Ind. Eng.*, **156** (2021), 107230. https://doi.org/10.1016/j.cie.2021.107230
21. Q. Zhu, J. Hu, W. Cai, Larry Henschen, A new robot navigation algorithm for dynamic unknown environments based on dynamic path re-computation and an improved scout ant algorithm, *Appl. Soft. Comput.*, **11** (2011), 4667–4676. https://doi.org/10.1016/j.asoc.2011.07.016
22. C. Yen, M. Cheng, A study of fuzzy control with ant colony algorithm used in mobile robot for shortest path planning and obstacle avoidance, *Microsyst Technol.*, **24** (2018), 125–135. https://doi.org/10.1007/s00542-016-3192-9
23. J. Yin, W. Fu, A safety navigation method for integrating global path planning and local obstacle avoidance for self-driving cars in a dynamic environment, *Sci. Iran.*, **28** (2021).
24. K. Xu, X. Lu, Y. Huang, S. Hu, Robot path planning based on double-layer ant colony optimization algorithm and dynamic environment, *Acta Electron. Sinica*, **47** (2019), 2166–2176.
25. Y. Weitao, S. Jing, G. Zhimin, Y. Chuanyi, G. Tong, Best grid size of the occupancy grid map and its accuracy, *Robot*, **42** (2020), 199–206.
26. X. Deng, R. Li, L. Zhao, K. Wang, X Gui, Multi-obstacle path planning and optimization for mobile robot, *Expert Syst. Appl.*, **183** (2021), 1–16. https://doi.org/10.1016/j.eswa.2021.115445


AIMS Press