



*Research article*

## **Path planning for mobile robots in complex environments based on improved ant colony algorithm**

**Yuzhuo Shi<sup>1</sup>, Huijie Zhang<sup>2,\*</sup>, Zhisheng Li<sup>2,\*</sup>, Kun Hao<sup>2</sup>, Yonglei Liu<sup>2</sup> and Lu Zhao<sup>2</sup>**

<sup>1</sup> College of Information Technology, Tianjin College of Commerce, Tianjin 300350, China

<sup>2</sup> School of Computer and Information Engineering, Tianjin Chengjian University, Tianjin 300384, China

\* **Correspondence:** Email: zhanghj9807@163.com, lzs\_jeff@qq.com; Tel: +8618854805202, +8613820244500.

**Abstract:** Aiming at the problems of the basic ant colony algorithm in path planning, such as long convergence time, poor global path quality and not being suitable for dynamic environments and unknown environments, this paper proposes a path planning method for mobile robots in complex environments based on an improved ant colony (CBIACO) algorithm. First, a new probability transfer function is designed for an ant colony algorithm, the weights of each component in the function are adaptively adjusted to optimize the convergence speed of the algorithm, and the global path is re-optimized by using the detection and optimization mechanism of diagonal obstacles. Second, a new unknown environment path exploration strategy (UPES) is designed to solve the problem of poor path exploration ability of the ant colony algorithm in unknown environment. Finally, a collision classification model is proposed for a dynamic environment, and the corresponding dynamic obstacle avoidance strategy is given. The experimental results show that CBIACO algorithm can not only rapidly generate high-quality global paths in known environments but also enable mobile robots to reach the specified target points safely and quickly in a variety of unknown environments. The new dynamic obstacle avoidance strategy enables the mobile robot to avoid dynamic obstacles in different directions at a lower cost.

**Keywords:** path planning; ant colony algorithm; unknown environment; path exploration; dynamic obstacle avoidance

---

## 1. Introduction

With the rapid development of science and technology, mobile robots have been widely used in people's work and life. For example, in autonomous navigation on the ground [1], resource exploration and development [2], emergency rescue and disaster relief [3], etc., mobile robots can replace or assist humans in completing various complex tasks. In addition, with the continuous expansion of application scenarios and service modes of mobile robots, people have increasingly higher requirements for the intelligence of mobile robots, and the intelligent core technology of mobile robots is path planning technology. Path planning refers to a mobile robot exploring a high-quality collision-free path from the starting point to the end point in a complex spatial environment [4]. A good path planning method can effectively improve the utilization of resources, reduce the wear and tear of mobile robots and prolong the service life of mobile robots [5,6].

At present, the mainstream algorithms of mobile robot path planning are divided into two categories: traditional algorithms and intelligent bionic algorithms. Among them, the traditional algorithms mainly include the A\* algorithm and artificial potential field method [7]. Intelligent bionic algorithms mainly include genetic algorithms [8], ant colony algorithms, particle swarm algorithms and immune algorithms [9]. Traditional algorithms have good performance in simple map environments, but they are not suitable for complex map environments. However, intelligent bionic algorithms have some problems, such as premature convergence, poor global path quality and easily falling into local extrema. In addition, there are many unknown environments in practical application environments, such as earthquake relief, emergency rescue and other actual scenes, which are usually unknown environments. Unknown environments are more complex and uncertain than known environments. However, most of the current path planning algorithms are only applicable to path planning of known environments [10], while the applicability to path planning of unknown environments [11] is poor. Therefore, it is very necessary to study a path planning method for mobile robots which is suitable for complex environments (including known environments, unknown environments and dynamic environments).

## 2. Related work

In the actual environment, the movement of a mobile robot is restricted by many factors. For example, the map scale is large, the terrain is complex, there are many static and dynamic obstacles, and some map environment information is unknown. Therefore, there will be many problems. For example, the global path generated by mobile robots is poor in quality and easily crosses diagonal obstacles, and mobile robots are not suitable for dynamic environments and unknown environments. To solve these problems, experts and scholars have carried out in-depth research.

[12] proposed an improved global path planning method combining ant colony algorithm and A\* algorithm, which solved the problems of the low quality of the path generated by the traditional ant colony algorithm and easily falling into local optima. However, this method did not take into account the problem that the deadlock of ants in complex maps would lead to invalid paths generated by the algorithm. [13] proposed a multi-ant colony cooperative optimization algorithm based on cooperative game mechanism (CCACO) to speed up the algorithm to generate global paths, but this algorithm is not suitable for large-scale maps and multi-obstacle maps. A terrain-assisted path planning algorithm based on particle swarm optimization was proposed in [14]. The algorithm integrates a terrain

recognition strategy and an obstacle avoidance strategy in a path planning algorithm based on a particle swarm, and it can plan a suitable path in unknown terrain. However, this algorithm is only applicable to unknown environments with relatively simple obstacle distributions and not applicable to unknown environments with high complexity. In [15], a real-time online path planning method based on a deep neural network (DNN) was proposed. The method is suitable for cluttered unknown environments and has significant improvements in efficiency, success rate and path quality. However, the algorithm does not consider special terrain such as concave areas, which will make the mobile robot stagnate or oscillate. [16] proposed a path planner based on the hybrid A\* algorithm. The path planner tree pruning process was given to improve the operation of an autonomous underwater vehicle (AUV) in an unknown environment and keep an effective and feasible search tree in the operation process. To solve the problem of online cooperative path planning of a multi-quadrotor unmanned aerial vehicle (UAV) in an unknown dynamic environment, an online collision avoidance strategy based on local environment information and a distributed online path planning strategy were proposed in [17]. This strategy can obtain the feasible path of each quadrotor UAV. However, the dynamic obstacle avoidance effect of this strategy is poor, and the decision time of obstacle avoidance is long. Aiming at the multi-intelligent, multi-objective navigation problem in unknown dynamic environments, an evolutionary algorithm was proposed in [18]. The algorithm combines an artificial swarm neighborhood search planner with an evolutionary planner to smooth the generated intermediate feasible paths. However, the algorithm has a long execution time, low execution efficiency and high instability in complex maps.

To sum up, this paper proposes a path planning method for mobile robots in complex environments based on an improved ant colony algorithm (CBIACO). The innovations and contributions of this paper are as follows: 1) The ant colony algorithm is improved. By improving the probability transfer function and designing adaptive component weights, the generation time of the global path is reduced, and the generation quality of the global path is improved. 2) A global path optimization strategy based on diagonal obstacle detection and optimization mechanism is proposed. The problem of the global path crossing diagonal obstacles is solved, and the re-optimization of the global path is realized. 3) The unknown environment path exploration strategy (UPES) is proposed. By using the real-time local information obtained by the sensor and the corresponding path exploration strategy, the mobile robot can reach the destination safely, efficiently and quickly. 4) A collision classification model is proposed, and corresponding dynamic obstacle avoidance strategies are given, considering more comprehensive potential collision situations. The mobile robot can avoid various obstacles effectively by behavioral obstacle avoidance and local path replanning.

The remainder of the paper is organized as follows: Section 2 introduces environmental modeling. Section 3 introduces the path planning method for mobile robots based on the CBIACO algorithm in detail. Section 4 verifies the feasibility and effectiveness of the CBIACO algorithm proposed in this paper. Section 5 summarizes the paper.

### 3. Environmental model

In this paper, the grid method [19,20] is used to establish the spatial environment model. As shown in Figure 1, the entire two-dimensional spatial environment is divided into  $30 \times 30$  equal-size squares. Among them, the white grid area represents the free area, the black grid area represents the static obstacles, the green grid area represents the dynamic obstacles, and the blue grid area represents

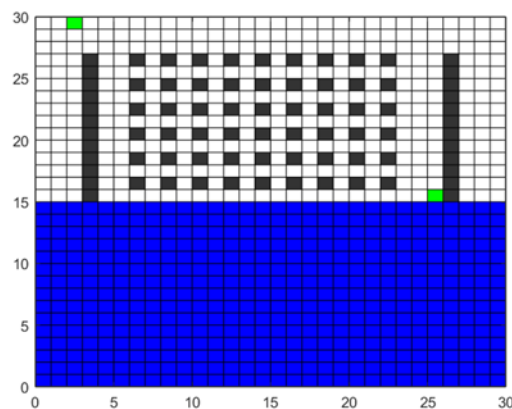
the unknown environment area (i.e., the distributions of static obstacles and dynamic obstacles in motion in the blue area are unknown). From left to right and from top to bottom of the map, we start numbering the squares from 1 until reaching the 900th square in the lower right corner. The coordinates of each square in the grid map are represented by its center point coordinates  $(x, y)$ , and the conversion between the square number and the grid coordinate is calculated as follows:

$$x = \begin{cases} \text{mod}(m, n) - 0.5, & \text{if } \text{mod}(m, n) \neq 0 \\ n - 0.5, & \text{if } \text{mod}(m, n) = 0 \end{cases} \quad (1)$$

$$y = n + 0.5 - \text{ceil}(m/n) \quad (2)$$

$$m = \text{floor}(l - y) * n + \text{ceil}(x) \quad (3)$$

Equations (1) and (2) convert the square number into grid coordinates. Equation (3) converts the grid coordinates to the square number.  $\text{mod}()$  is the remainder function,  $\text{floor}()$  is the downward rounding function, and  $\text{ceil}()$  is the upward rounding function. Here,  $m$  is the square number,  $n$  is the total number of grid columns,  $l$  is the total number of grid rows, and  $(x, y)$  are the horizontal and vertical grid coordinates.



**Figure 1.**  $30 \times 30$  grid map.

#### 4. Path planning method for mobile robots in complex environments based on improved ant colony algorithm

Based on previous research [21], this paper considers using the CBIACO algorithm to study path planning in complex environments. When the mobile robot is located in the known environment, the improved ant colony (IACO) algorithm is directly used to generate the global path from the start point to the target point. When the mobile robot encounters the unknown environment, the IACO algorithm and UPES are used for path exploration, so that the mobile robot can safely and quickly traverse the unknown environment and reach the specified target point. When the mobile robot encounters dynamic obstacles, the dynamic obstacle avoidance strategy is used to avoid obstacles, so that the mobile robot can avoid all kinds of dynamic obstacles at a low cost.

The pseudo-code of the CBIACO algorithm is shown in Table 1.



**Table 1.** The pseudo-code of the CBIACO algorithm.**Algorithm 1**

Initialization of algorithm parameters;

If the mobile robot is located in the known environment

    The mobile robot uses IACO algorithm to generate the global path;

If the mobile robot encounters the unknown environment

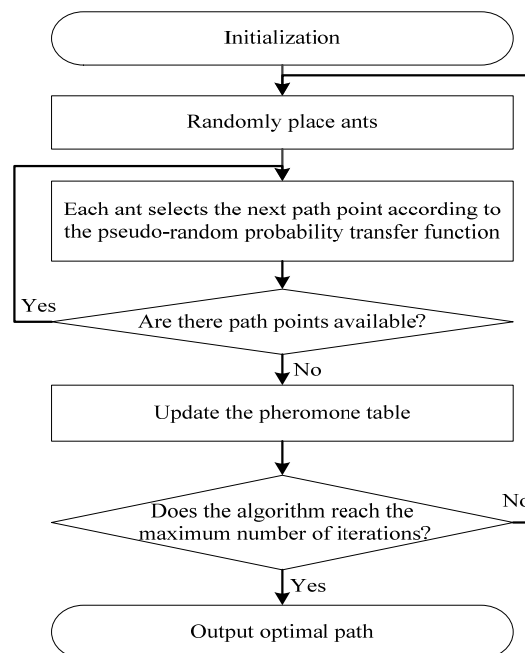
    The mobile robot uses IACO algorithm and UPES to explore path;

If the mobile robot encounters dynamic obstacles

    The mobile robot uses dynamic obstacle avoidance strategy to avoid the dynamic obstacle;

**4.1. IACO algorithm**

The flow of the IACO algorithm is shown in Figure 2. First, several ants are randomly placed at the starting point, and each ant selects the next path point according to the pseudo-random probability transfer function. Next, if the ant does not reach the target point, repeat the above process. If the ants reach the target point, the pheromone table is updated according to the paths generated by each ant. If the final algorithm does not reach the maximum number of iterations, repeat the above process. Otherwise, the algorithm outputs the globally optimal path.

**Figure 2.** Flow chart of IACO algorithm.**4.1.1. Pseudo-random probability transfer function**

The pseudo-random probability transfer function enhances the selection degree of high-quality path points in the algorithm. It adjusts the selection probability of high-quality path points by a pseudo-random probability transfer adjustment factor (as shown in Eqs (4) and (5)), aiming at ensuring the diversity of the population and strengthening the utilization of the current best neighbor information.

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \eta_{ij}^\beta}{\sum_{s \in allowed_k} \tau_{is}^\alpha \eta_{is}^\beta}, & \text{if } j \in allowed_k \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

$$w = \begin{cases} \mathit{arg}_{s \in allowed_k} \max(\tau_{is}^\alpha \eta_{is}^\beta), & \text{if } q \leq q_0 \\ \mathit{Roulette}(p_{ij}^k), & \text{if } q > q_0 \end{cases} \quad (5)$$

Among them,  $p_{ij}^k$  represents the transfer probability of the  $k$ -th ant between path point  $i$  and path point  $j$ ,  $\tau_{ij}$  represents pheromone concentration,  $\eta_{ij}$  represents heuristic information,  $\alpha$  represents the pheromone factor,  $\beta$  represents the heuristic factor,  $allowed_k$  represents the optional set of the next path point,  $w$  represents the path point obtained by the pseudo-random probability transfer function,  $\mathit{arg}_{s \in allowed_k} \max(\tau_{is}^\alpha \eta_{is}^\beta)$  is the function of selecting the best neighbor path point,  $\mathit{Roulette}()$  represents the function of selecting the path point by roulette strategy,  $q$  is a random number, and  $q_0$  is a pseudo-random probability transfer adjustment factor. When  $q \leq q_0$ , the current optimal neighbor path point is selected as the next path point. When  $q > q_0$ , the roulette strategy is used to select the next path point.

In the traditional probability transfer function, assuming that the probability of a high-quality path point being selected is  $p$ , the pseudo-random probability transfer adjustment factor  $q_0$  is introduced. Assuming that the probability of the pseudo-random probability transfer function selecting the high-quality path point is  $p_1$ , according to Eq (4),  $p_1 = q_0 + (1 - q_0)p$ . Because  $p_1 - p = q_0 - pq_0 = q_0(1 - p) \geq 0$ ,  $p_1 \geq p$ , that is, after introducing the pseudo-random probability transfer adjustment factor  $q_0$ , the probability of selecting a high-quality path point in the pseudo-random probability transfer function is greater than that of the traditional probability transfer function. When  $p$  is known, the selection probability of the pseudo-random probability transfer function for a high-quality path point depends on  $q_0$ , so it is feasible to adjust the selection probability of the high-quality path point by introducing the pseudo-random probability transfer adjustment factor  $q_0$ .

The IACO algorithm makes full use of prior information such as target points to strengthen the guiding role of heuristic information, as shown in Eq (6), where  $d_{jg}$  represents the Euclidean distance between the path point  $j$  and the target point  $g$ .

$$\eta_{ij} = \frac{1}{d_{jg}} \quad (6)$$

According to the iterative rule of the ant colony algorithm, this paper adaptively adjusts pheromone factors and heuristic factors to accelerate the convergence rate of the algorithm and improve the performance of the algorithm (as shown in Eqs (7) and (8)).

$$\alpha_1 = (N_{cmax} + N_c) / N_{cmax} * \alpha \quad (7)$$

$$\beta_1 = \beta * \exp(-2 * (N_c / N_{cmax})^2) \quad (8)$$

where  $\alpha_1$  is the adaptive pheromone factor,  $\beta_1$  is the adaptive heuristic factor,  $N_{cmax}$  is the maximum iteration number, and  $N_c$  is the current iteration number.

#### 4.1.2. Initialization and updating of pheromones

In the aspect of pheromone initialization, this paper considers generating uniformly distributed initial pheromones (that is, the initial pheromones of each feasible path segment are the same fixed constant). In the aspect of pheromone update, the method of single pheromone update is adopted, and the update is based on the path fitness. Path fitness is composed of path length and penalty information. The pheromone updating process is shown in Eqs (9)–(11), the fitness calculation formula is shown in Eq (12), and the penalty information calculation formula is shown in Eq (13).

$$\tau_{ij}(t + 1) = (1 - \rho)\tau_{ij}(t) + \Delta\tau_{ij}, \rho \in (0, 1) \quad (9)$$

$$\Delta\tau_{ij} = \sum_{k=1}^K \Delta\tau_{ij}^k \quad (10)$$

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{Fitness}, & \text{If ant } k \text{ passes through the path segment } (i, j) \\ 0, & \text{Otherwise} \end{cases} \quad (11)$$

where,  $\Delta\tau_{ij}$  is the pheromone increment on the path segment  $(i, j)$ ,  $\rho$  is the pheromone volatilization factor,  $\Delta\tau_{ij}^k$  is the pheromone increment of the  $k$ -th ant on the path segment  $(i, j)$ ,  $K$  is the total number of ants,  $Q$  is the pheromone constant, and  $Fitness$  is the fitness of the path passed by the  $k$ -th ant.

$$Fitness = L_k + punish_k \quad (12)$$

$$punish_k = \begin{cases} cons, & \theta_k > \theta \\ 0, & \theta_k \leq \theta \end{cases} \quad (13)$$

$L_k$  is the length of the path passed by the  $k$ -th ant, and  $punish_k$  is the punishment information obtained by the  $k$ -th ant.  $cons$  is a fixed constant,  $\theta_k$  is the maximum rotation angle of the path passed by the  $k$ -th ant, and  $\theta$  is the steering angle constraint of the mobile robot. When the maximum rotation angle of the path is greater than the steering angle constraint of the mobile robot, the  $k$ -th ant gets the punishment information. Otherwise, no penalty information is obtained.

#### 4.1.3. Deadlock problem handling strategy

In this paper, the method of combining a backtracking mechanism and a path length zeroing mechanism is adopted to solve the deadlock problem of the ant colony algorithm. Suppose that the current ant travels to the  $n$ -th path point, and the ant is in a deadlock state at the  $n$ -th path point. The specific process of executing the method is as follows:

- 1) Let  $i = n$ ;

2) The path between the  $i$ -th path point and the  $i-1$  th path point is marked as an infeasible path (i.e., the path length zeroing mechanism);

3) The ant returns to the  $i-1$  th path point (i.e., the backtracking mechanism), i.e.,  $i = i - 1$ .

4) Determine whether the current path point of the ant is trapped in the deadlock phenomenon. If the current path point of the ant is trapped in the deadlock phenomenon, go to step (2). If the current path point of the ant is not trapped in the deadlock phenomenon, the process ends. The ant selects the next path point according to the probability transfer function.

This method readjusts the driving direction of ants through the backtracking mechanism and the path length zeroing mechanism, which ensures the 100% ant survival rate and improves the ability of the ants to explore the solution space.

#### 4.1.4. Pseudo-code of IACO algorithm

The pseudo-code of the IACO algorithm is shown in Table 2.

**Table 2.** The pseudo-code of the IACO algorithm.

---

#### Algorithm 2

---

Initialize the algorithm parameters; //The starting point grid number  $S$ , the target point grid number  $G$ , the maximum number of iterations  $Nc\_max$ , the current number of iterations  $Nc$ , the total number of ants  $K$ , the current number of ants  $k$ , the pheromone constant  $Q$ , the pheromone factor  $\alpha$ , the heuristic factor  $\beta$ , the pheromone volatilization factor  $\rho$  and the pseudo-random probability transfer adjustment factor  $q_0$ .

For  $Nc=1:Nc\_max$  //Enter an iterative loop.

    For  $k=1:K$  //Enter ant iteration.

        Place the  $k$ -th ant at the starting point;

        While !isgoal( $k$ ) //Judge whether the  $k$ -th ant has reached the target point, and if not, the loop //is performed.

            Probability\_transfer\_function(); //Select the next path point according to the pseudo-random //probability transfer function.

            While IsDeadlock( $k$ ) //Judge whether the  $k$ -th ant is trapped in a deadlock, and if so, loop.

                Deadlock\_handling\_strategy(); //The backtracking mechanism and path length zeroing //mechanism are invoked to handle deadlock.

                Probability\_transfer\_function(); //Select the next path point according to the //pseudo-random probability transfer function.

                Update tabu list(); //The tabu table is updated after the next path point is selected.

                Pheromone update strategy(); //Execute pheromone update strategy.

            If IsConvergence() //Judge whether the algorithm converges.

                The convergence path is obtained; //If the algorithm has converged, the convergence path is //obtained.

                Path\_optimization\_strategy(); //Execute path optimization strategy for convergence path.

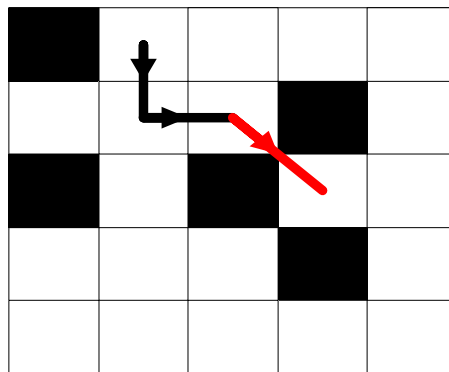
                The best path is obtained;

                Break;

---

#### 4.2. Diagonal obstacle optimization strategy

When the grid method is used to model the map, some diagonal obstacles may be generated (as shown in Figure 3). When the map complexity is low, diagonal obstacles will not have a substantial impact on the result of path planning. However, when the map complexity is high, the traditional ant colony algorithm may generate a path that crosses diagonal obstacles, which will adversely affect the feasibility and security of the path. In order to ensure the feasibility and security of the path, this paper designs a diagonal obstacle detection and optimization method suitable for the two-dimensional grid map.



**Figure 3.** Schematic diagram of path crossing diagonal obstacles.

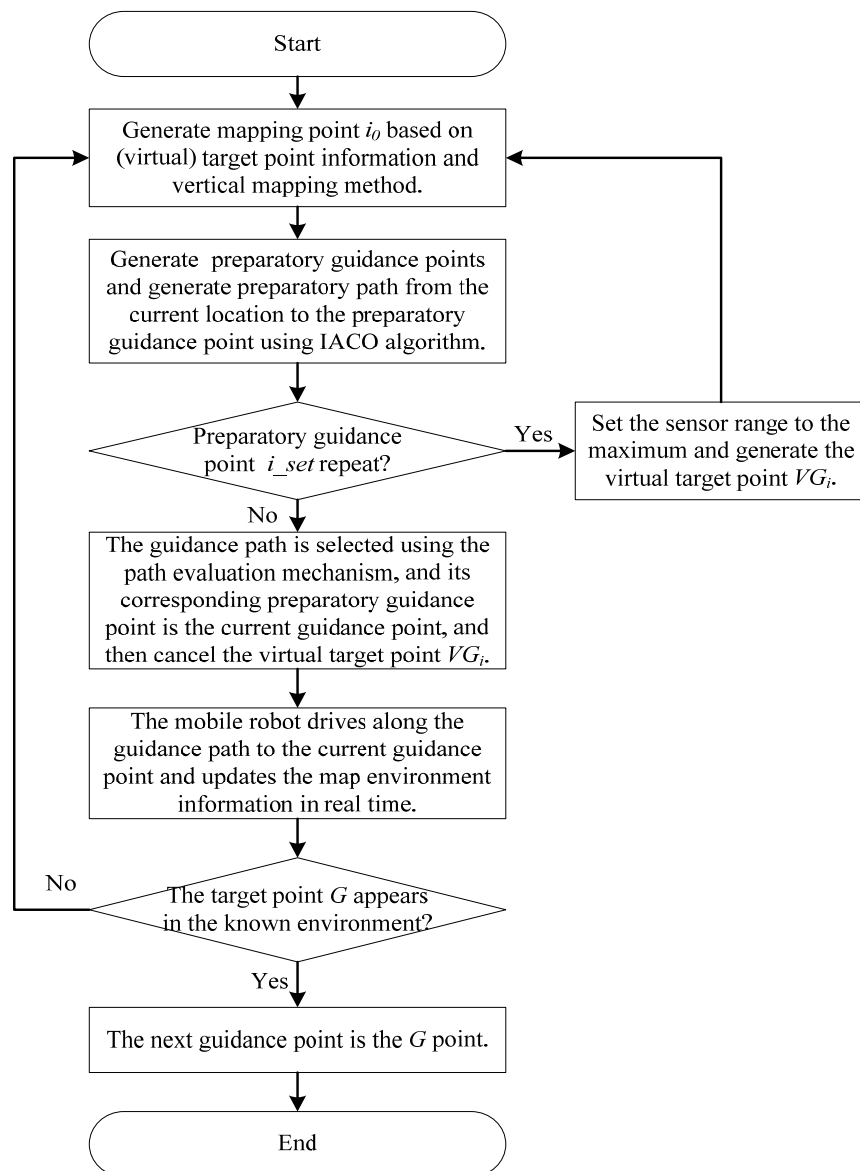
Assuming that a path consists of several path points, if we want to check whether a path crosses a diagonal obstacle, we only need to check whether each path segment composed of all adjacent path points in the path crosses a diagonal obstacle. If a path segment composed of any adjacent path points crosses a diagonal obstacle, it can be considered that the path crosses a diagonal obstacle. Otherwise, the path does not cross the diagonal obstacles.

First, the four-direction detection method is used to detect whether the path segment composed of adjacent path points crosses diagonal obstacles. Four-direction detection is a method to detect whether the path segment composed of adjacent path points crosses diagonal obstacles. Assuming that the path segment  $ij$  is a path segment composed of adjacent path points  $i$  and  $j$ , the steps of detecting whether the path segment  $ij$  crosses diagonal obstacles by adopting the four-direction detection method are as follows: 1) Detect whether the neighbor obstacle squares of the up, down, left and right directions of the path point  $i$  and the neighbor obstacle squares of the up, down, left and right directions of the path point  $j$  overlap. 2) If there are two or more overlapping obstacles squares, it can be judged that the path segment  $ij$  has crossed a diagonal obstacle. 3) Optimize the path segment.

The optimization method is as follows: 1) Find the path section  $ij$  that crosses the diagonal obstacle. 2) Mark the square where the path point  $j$  is located as a temporary obstacle, and use the heuristic strategy to generate a new path segment from the path point  $i$  to the path point  $j + 1$ . 3) In order to avoid the new path segment still crossing the diagonal obstacle, it is necessary to use the four-direction detection method to detect the new path segment again. 4) If the new path segment does not cross the diagonal obstacle, the new path segment will replace the old path segment. Otherwise, the above method is adopted to continue optimization until the new path segment does not cross the diagonal obstacle.

Note: When the path point  $i$  is the penultimate path point in the global path,  $j$  is the target point, and the path segment  $ij$  just crosses the diagonal obstacle, it cannot be optimized according to the above method because there is no path point  $j + 1$ . At this time, the path point  $i$  can be marked as a temporary obstacle, and a new path segment between the path point  $i - 1$  and the path point  $j$  can be generated by heuristic strategy, and then the above optimization method can be followed.

#### 4.3. Path exploration strategy in unknown environment



**Figure 4.** UPES execution flow chart.

Compared with the known environment, the unknown environment has high complexity, strong uncertainty and a large amount of global prior information missing, but it exists widely in the actual environment. Actual scenes such as earthquake relief and emergency rescue are usually unknown environments. Therefore, this paper proposes an unknown environment path exploration strategy

(UPES). The CBIACO algorithm will explore the path in the unknown environment through UPES (Figure 4).

In the unknown environment, the guidance point in the UPES is the intermediate path point selected by the mobile robot according to the prior information and local environment information detected by the sensor. It aims to more conveniently generate a high-quality guidance path in an unknown environment to improve the success rate of the mobile robot reaching the target point. The flow chart and specific steps of the UPES are as follows:

1) The target point  $G$  (virtual target point  $VG_i$ ) is vertically projected to the non-obstacle area in the known environment to generate the mapping point  $i_0$ .  $i_0 = G - h_1 * l - h_2 * l$ , where  $G$  is the grid number of the target point,  $h_1$  is the total row number of the unknown environment in the vertical direction,  $h_2$  is the total row number of the obstacle area connected with the unknown environment in the vertical direction, and  $l$  is the total grid number per row.

2) We expand  $n$  grid lengths on the left and right sides of point  $i_0$  to generate the preparatory area, so the preparatory area contains  $2n + 1$  squares. In the preparatory area, the squares beyond the map boundary and obstacle squares are removed, and the remaining squares are the set of preparatory guidance points, i.e., the  $i\_set$ .

3) The mobile robot uses the IACO algorithm to generate the preparatory paths from the current position  $c\_point$  to each point in the  $i\_set$ . Check whether the current  $i\_set$  is completely repeated with the historical  $i\_set$ . If so, go to step (4); otherwise, go to step (5).

4) The mobile robot sets the sensor detection radius to the maximum value and generates the virtual target point  $VG_i$  with an interval of  $n$  grid lengths on the left or right side of the point  $G(VG_i)$ .  $VG_i$  temporarily replaces point  $G$ . Then, go to step (1).

5) The path evaluation mechanism is used to evaluate the quality of each preparatory path, the best path among the preparatory paths is selected as the guidance path, and the guidance point corresponding to this guidance path is the current guidance point  $i$ . Check whether  $VG_i$  exists; if so, cancel  $VG_i$  and restore point  $G$ .

6) The mobile robot drives along the guidance path to point  $i$ . Meanwhile, the mobile robot uses sensors to update the map environment information in real time during the driving process.

7) Determine whether point  $G$  appears in the known environment. If point  $G$  appears in the known environment, the next guidance point is point  $G$ , and the process ends. If point  $G$  does not appear in the known environment, go to step (1).

#### 4.3.1. Path evaluation mechanism

In order to get a better exploration path for mobile robots in unknown environment, this paper proposes a path evaluation mechanism based on multi-performance indicators. The evaluation indexes of path quality include the path reachability, path safety, path length and path smoothness. The path quality is calculated as shown in Eq (14).

$$Q = \omega_1 P_r + \omega_2 P_s + \omega_3 P_l + \omega_4 P_m \quad (14)$$

where  $Q$  is the path quality value,  $P_r$  is the path reachability value,  $P_s$  is the path safety value,  $P_l$  is the path length value, and  $P_m$  is the path smoothness value.  $\omega_1$ ,  $\omega_2$ ,  $\omega_3$  and  $\omega_4$  are the weight coefficients of path reachability, path safety, path length and path smoothness, respectively, and  $\omega_1 + \omega_2 + \omega_3 + \omega_4 = 1$ .

### a. Path reachability

The path reachability value is defined as the difference between the distance from the preparatory guidance point  $z$  to target point  $G$  and the distance from mapping point  $i_0$  to target point  $G$ , multiplied by the adaptive expansion coefficient. The purpose is to make the guidance point as close to the target point as possible, and when the preparatory guidance point deviates more from the target point, it has a lower probability of being selected as the real guidance point.

Suppose that the coordinates of the preparatory guidance point  $z$  are  $(x_z, y_z)$ , the coordinates of the vertical mapping point  $i_0$  are  $(x_0, y_0)$ , and the coordinates of the target point  $G$  are  $(x_g, y_g)$ . Then, the path reachability is calculated as shown in Eqs (15)–(18).

$$dis\_1 = \left( (x_z - x_g)^2 + (y_z - y_g)^2 \right)^{0.5} \quad (15)$$

$$dis\_0 = \left( (x_0 - x_g)^2 + (y_0 - y_g)^2 \right)^{0.5} \quad (16)$$

$$P_r = coe1 * (dis\_1 - dis\_0) \quad (17)$$

$$coe1 = \begin{cases} init\_coe1 - num * step, & init\_coe1 - num * step > 1 \\ 1, & init\_coe1 - num * step \leq 1 \end{cases} \quad (18)$$

where  $dis\_1$  is the distance from  $z$  to  $G$ ,  $dis\_0$  is the distance from  $i_0$  to  $G$ ,  $P_r$  is the path reachability value,  $coe1$  is the expansion coefficient of path reachability,  $inti\_coe1$  is the initial expansion coefficient and is a fixed constant,  $num$  is the number of generated guidance points, and  $step$  is a fixed step size.

### b. Path length

The path length is defined as the Euclidean distance of a path. Suppose that a path consists of  $n$  path points, the coordinates of the  $i - 1$ th path point  $P_{i-1}$  are  $(x_{i-1}, y_{i-1})$ , the coordinates of the  $i$ th path point  $P_i$  are  $(x_i, y_i)$ , and the coordinates of the  $i + 1$ th path point  $P_{i+1}$  are  $(x_{i+1}, y_{i+1})$ . Then, the path length is calculated as shown in Eq (19).

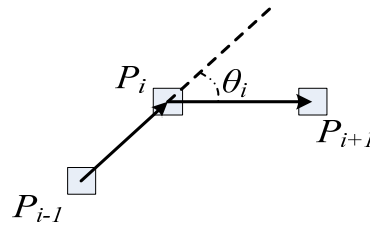
$$P_l = \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (19)$$

### c. Path smoothness

Path smoothness is defined as the cumulative sum of the required rotation angles of the mobile robot in a path [22], and its purpose is to make the guidance path as smooth as possible. Three consecutive path points can form two path segments  $P_{i-1}P_i$ ,  $P_iP_{i+1}$ . Let  $\theta_i$  be the rotation angle between two path segments  $P_{i-1}P_i$  and  $P_iP_{i+1}$ . Then, the path smoothness is calculated as shown in Eq (20), and the schematic diagram of rotation angle  $\theta_i$  is shown in Figure 5.

$$P_m = \sum_{i=2}^{n-1} \theta_i \quad (20)$$





**Figure 5.** Diagram of the rotation angle.

d. Path safety

The path safety is defined as the ratio of the safety penalty degree obtained by a path to the path length [23], multiplied by the fixed expansion coefficient. Assuming that a path consists of  $n$  path points, and the coordinates of the  $i$ th path point  $P_i$  are  $(x_i, y_i)$ , the path safety is calculated as shown in Eqs (21)–(23).

$$P_s = \text{coe2} * \frac{\sum_{i=1}^n S_i}{P_l} \quad (21)$$

$$S_i = \sum_{j=1}^8 \text{punish\_}w_j \quad (22)$$

$$\text{punish\_}w_j = \begin{cases} 1, & \text{If } \omega_j \text{ is an obstacle square and does not provide a security penalty} \\ 0, & \text{If } \omega_j \text{ is not an obstacle square} \\ 0, & \text{If } \omega_j \text{ is an obstacle square but has provided a security penalty} \end{cases} \quad (23)$$

In Eq (21),  $P_s$  is the path safety value,  $\text{coe2}$  is the expansion coefficient and is a fixed constant,  $S_i$  is the safety penalty degree obtained by the  $i$ th path point, and  $P_l$  is the path length of this path. In Eqs (22) and (23),  $\text{punish\_}w_j$  is the safety penalty degree provided by the  $j$ th square around the  $i$ th path point, and  $w_j$  is the  $j$ th square around the  $i$ th path point.

#### 4.3.2. Weight distribution

Aiming at the above four path quality evaluation indexes, this paper uses the Delphi weighting method to evaluate the weight coefficients of each path index to obtain a set of better weight coefficients. First, the order of importance of the four indexes is specified according to the actual requirements. Considering the specificity of the unknown environment, the order of importance of the four indexes is path reachability, path safety, path length and path smoothness in turn, i.e., the reachability and safety of the exploration path are ensured in priority, and then the path length and path smoothness are considered. Thus,  $\omega_1 > \omega_2 > \omega_3 > \omega_4$ .

The definition of the importance ratio is shown in Eq (24):

$$I_k = \frac{\omega_k}{\omega_{k+1}} \quad (24)$$

Then, there is  $I_k > 1$ . The larger  $I_k$  is, the more important the former is than the latter. The importance ratio table constructed in this paper is shown in Table 3. According to the defined  $I_k$ , the calculation process of  $\omega_4$  is shown in Eqs (25)–(27).

$$\sum_{k=1}^3 \left( \prod_{i=k}^3 I_i \right) = \frac{\sum_{k=1}^3 \omega_k}{\omega_4} \quad (25)$$

$$1 + \sum_{k=1}^3 \left( \prod_{i=k}^3 I_i \right) = \omega_4^{-1} \quad (26)$$

$$\omega_4 = \left( 1 + \sum_{k=1}^3 \left( \prod_{i=k}^3 I_i \right) \right)^{-1} \quad (27)$$

If we can specify the values of  $I_1, I_2, I_3$ , we will calculate the values of  $\omega_4, \omega_3, \omega_2, \omega_1$  in turn.

**Table 3.** Importance ratio table.

| $I_k$ | Description  |
|-------|--|
| 1.3   | The former is slightly more important than the latter. |
| 1.6   | The former is more important than the latter.          |
| 1.9   | The former is far more important than the latter.      |

#### 4.3.3. Pseudo-code of UPES strategy

The pseudo-code of the UPES strategy is shown in Table 4.

**Table 4.** The pseudo-code of the IACO algorithm.

| Algorithm 3  |
|--|
| Initialization of algorithm parameters; //The current position of the mobile robot $(x_c, y_c)$ . Relevant //parameters of IACO algorithm.   |
| While the mobile robot does not reach the target point G   |
| Generate mapping point $i_0$ based on (virtual) target point information and vertical mapping method;  |
| Generate preparatory guidance points based on point $i_0$ and generate preparatory paths from the current position to the preparatory guidance points using IACO algorithm (Algorithm 2);            |
| If the current $i\_set$ is completely repeated with the historical $i\_set$ // $i\_set$ is the set of preparatory //guidance points.   |
| Set the sensor range to the maximum and generate the virtual target point $VG_i$ ;   |
| Else   |
| The guidance path is selected using the path evaluation mechanism, and its corresponding preparatory guidance point is the current guidance point, and then cancel the virtual target point $VG_i$ ; |
| The mobile robot drives along the guidance path to the current guidance point and updates the map environment information in real time;  |
| If the target point G appears in the known environment   |
| The next guidance point is the G point;  |

#### 4.4. Dynamic obstacle avoidance

[21] introduced several common potential collision modes, but its collision detection model lacks universality and does not consider collision modes such as pursuit collisions. Hence, the collision classification model is redesigned in this paper by using sequence pairs and sign functions. Finally, the corresponding obstacle avoidance strategy is given for each potential collision mode.

##### 4.4.1. Collision classification model

Define the row direction: -1 represents upward, 0 represents unchanged row direction, 1 represents downward. Define the column direction: -1 represents leftward, 0 represents unchanged column direction, 1 represents rightward. The direction of movement of the mobile robot can be represented by the direction sequence pair  $\langle x, y \rangle$ . For example, the sequence pair  $\langle 1, 0 \rangle$  indicates that the mobile robot moves downward, and the sequence pair  $\langle -1, -1 \rangle$  indicates that the mobile robot moves up and left.

Since a certain function transformation is required to transform the ranks vector of the robot or obstacle into the direction sequence pair, this paper uses the sign function as the direction transformation function, as shown in Eq (28).

$$\text{sign}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} \quad (28)$$

For example, if a dynamic obstacle moves from (3,3) to (5,3), the row and column vector of the dynamic obstacle can be calculated as (2,0), and the direction sequence pair  $\langle 1, 0 \rangle$  can be obtained after the sign function processing, so that the dynamic obstacle can be determined to move downward. For example, if the dynamic obstacle moves from (3,4) to (3,3), the row and column vector of the dynamic obstacle can be calculated as (0,-1), and the direction sequence pair  $\langle 0, -1 \rangle$  can be obtained after the sign function processing to determine that the dynamic obstacle moves to the left.

According to different moving directions of dynamic obstacles and robots, potential collisions are classified into four different types: forward collision, lateral collision, stagnation collision and pursuit collision. Among them, forward collision is further divided into forward surface collision and forward point collision depending on the collision location. The collision classification model is shown in Table 5.

Suppose that the coordinates of the current position  $R_c$  of the mobile robot are  $(x_c, y_c)$ , and the coordinates of the next position  $R_n$  are  $(x_n, y_n)$ . The coordinates of the current position  $O_c$  of the dynamic obstacle are  $(r_c, c_c)$ , and the coordinates of the next position  $O_n$  are  $(r_n, c_n)$ . Then, the collision type judgment formulas are as follows:

$$\begin{aligned} x_n &== r_n \&\&y_n == c_n, \\ \text{sign}(r_n - r_c) &== -\text{sign}(x_n - x_c) \&\& \text{sign}(c_n - c_c) == -\text{sign}(y_n - y_c) \end{aligned} \quad (29)$$





$$\begin{aligned} x_c &== r_n \&\&y_c == c_n \&\& x_n == r_c \&\&y_n == c_c, \\ \text{sign}(r_n - r_c) &== -\text{sign}(x_n - x_c) \&\& \text{sign}(c_n - c_c) == -\text{sign}(y_n - y_c) \end{aligned} \quad (30)$$

$$x_n == r_n \&\& y_n == c_n \quad (31)$$

$$x_n == r_n \&\& y_n == c_n, \quad r_n == r_c \&\& c_n == c_c \quad (32)$$

$$x_n == r_n \&\& y_n == c_n, \quad \text{sign}(r_n - r_c) == \text{sign}(x_n - x_c) \&\& \text{sign}(c_n - c_c) == \text{sign}(y_n - y_c) \quad (33)$$

**Table 5.** Collision classification model.

| Collision type            | Direction of motion             | Collision position         | Judgment formula | Collision characteristics   | Schematic diagram   |
|---------------------------|---------------------------------|----------------------------|------------------|---|---|
| Forward point collision   | Collinear and reverse direction | Common collision points    | (29)             | Mobile robot and dynamic obstacle drive into the same square.   |    |
| Forward surface collision | Collinear and reverse direction | No common collision points | (30)             | Face to face collision.   |   |
| Lateral collision         | noncollinear                    | Common collision points    | (31)             | At a certain moment, the two happen to have a common collision point.                                 |  |
| Stagnation collision      | --                              | Common collision points    | (32)             | The dynamic obstacle just stays on the planned path.  | --  |
| Pursuit collision         | Collinear and same direction    | Common collision points    | (33)             | At a certain moment, the common collision point is created due to the mismatch of the two velocities. |  |

#### 4.4.2. Obstacle avoidance strategy

This paper designs three obstacle avoidance strategies: the turning behavior strategy, in-situ waiting strategy and local path replanning strategy. The turning behavior strategy implies that when the mobile robot senses that it is about to have a forward surface collision, a forward point collision or a pursuit collision with a dynamic obstacle, the mobile robot turns left or right to avoid the dynamic obstacle.

In-situ waiting strategy implies that when the mobile robot senses that it is about to have a lateral collision with a dynamic obstacle, the mobile robot avoids the dynamic obstacle by stopping in-situ.

The local path replanning strategy implies that when the dynamic obstacle stagnates on the planned path, the robot uses a heuristic strategy to replan the local path between the current path point (the  $i$ th path point) and the  $i + 2$ th path point to avoid dynamic obstacles. After the mobile robot avoids the dynamic obstacle in this manner, the mobile robot will return to the planned path and continue to drive along the planned path.

Therefore, the turning behavior strategy is used to avoid the dynamic obstacle when a forward collision or pursuit collision is about to occur between the mobile robot and the dynamic obstacle. The in-situ waiting strategy is used to avoid the dynamic obstacle when a lateral collision is about to occur between the mobile robot and the dynamic obstacle. The local path replanning strategy is used to avoid the dynamic obstacle when a stagnation collision is about to occur between the mobile robot and the dynamic obstacle.

#### 4.4.3. Pseudo-code of obstacle avoidance strategy

The pseudo-code of the obstacle avoidance strategy is shown in Table 6.

**Table 6.** The pseudo-code of the IACO algorithm.

| Algorithm 4   |
|---|
| Initialize the algorithm parameters; //The current position of the mobile robot $(x_c, y_c)$ and the next //position $(x_n, y_n)$ . The current position of the dynamic obstacle $(r_c, c_c)$ and the next position // $(r_n, c_n)$ . |
| If their position relationship satisfies Eq (29)  |
| The mobile robot uses the turning behavior strategy to avoid the dynamic obstacle;  |
| If their position relationship satisfies Eq (30)  |
| The mobile robot uses the turning behavior strategy to avoid the dynamic obstacle;  |
| If their position relationship satisfies Eq (31)  |
| The mobile robot uses the in-situ waiting strategy to avoid the dynamic obstacle;   |
| If their position relationship satisfies Eq (32)  |
| The mobile robot uses the local path replanning strategy to avoid the dynamic obstacle;   |
| If their position relationship satisfies Eq (33)  |
| The mobile robot uses the turning behavior strategy to avoid the dynamic obstacle;  |

## 5. Experimental results and analysis

In order to verify the feasibility and effectiveness of the proposed method, this paper designs map environments of different scenes. The scenes 1 and 2 are the maps with a known global environment, and their purpose is to verify the performance of the CBIACO algorithm in a known environment. Scenes 3, 4 and 5 are representative maps of unknown environments, namely, unknown environment with irregular distribution of obstacles, unknown environment with special terrain and unknown environment with dynamic obstacles. At the same time, in the three kinds of unknown environments, the CBIACO algorithm is compared with the path planning method based on a greedy algorithm [24] (GRA) to verify the performance of the CBIACO algorithm in unknown environments.

The hardware and software configurations of the experiments are shown in Table 7.

**Table 7.** Hardware and software configurations.

|          |                  |   |
|----------|------------------|---|
| Hardware | Processor        | Intel(R) Core(TM) i5-7300HQ CPU @ 2.50 GHz 2.50 GHz |
|          | RAM              | 8.00 GB (7.89 GB available)                         |
| Software | Operating System | Windows 10 (64-bit operating system)                |
|          | Simulation tool  | MATLAB R2018a                                       |

### 5.1. Weight coefficient selection

**Table 8.** Importance ratio and path evaluation value.

| $I_1$      | $I_2$      | $I_3$      | Path evaluation value |
|------------|------------|------------|-----------------------|
| 1.3        | 1.3        | 1.3        | 11.3912               |
| 1.3        | 1.3        | 1.6        | 11.5857               |
| 1.3        | 1.3        | 1.9        | 11.7792               |
| 1.3        | 1.6        | 1.3        | 11.5503               |
| 1.3        | 1.6        | 1.6        | 11.4031               |
| 1.3        | 1.6        | 1.9        | 11.3836               |
| 1.3        | 1.9        | 1.3        | 11.5194               |
| <b>1.3</b> | <b>1.9</b> | <b>1.6</b> | <b>11.3147</b>        |
| 1.3        | 1.9        | 1.9        | 11.4992               |
| 1.6        | 1.3        | 1.3        | 12.2246               |
| 1.6        | 1.3        | 1.6        | 11.6016               |
| 1.6        | 1.3        | 1.9        | 12.2307               |
| 1.6        | 1.6        | 1.3        | 12.0059               |
| 1.6        | 1.6        | 1.6        | 11.5781               |
| 1.6        | 1.6        | 1.9        | 11.7228               |
| 1.6        | 1.9        | 1.3        | 11.9375               |
| 1.6        | 1.9        | 1.6        | 12.2661               |
| 1.6        | 1.9        | 1.9        | 12.2719               |
| 1.9        | 1.3        | 1.3        | 12.9832               |
| 1.9        | 1.3        | 1.6        | 12.8568               |
| 1.9        | 1.3        | 1.9        | 12.8591               |
| 1.9        | 1.6        | 1.3        | 12.8352               |
| 1.9        | 1.6        | 1.6        | 12.9431               |
| 1.9        | 1.6        | 1.9        | 12.7933               |
| 1.9        | 1.9        | 1.3        | 12.7477               |
| 1.9        | 1.9        | 1.6        | 12.5693               |
| 1.9        | 1.9        | 1.9        | 12.6154               |

Because the weight coefficient is related to the path quality, this section will analyze and select the weight coefficient.

According to the importance ratio table (Table 3),  $I_k$  has three possible values, so theoretically there are 27 ways to combine  $I_1, I_2, I_3$ . In this paper, the scene shown in Figure 1 is taken as the test scene, and then the corresponding relationship between the importance ratio and the path evaluation value is shown in Table 8.

As can be seen from Table 8, when  $I_1 = 1.3, I_2 = 1.9,$  and  $I_3 = 1.6$ , the path evaluation value is the best. This shows that this importance ratio combination method can make the mobile robot cross the unknown environment to reach its destination at less cost. Therefore, we specify that  $I_1 = 1.3, I_2 = 1.9,$  and  $I_3 = 1.6$ . In other words, we think that in the unknown environment, the path reachability is slightly more important than path safety, path safety is far more important than the path length, and path length is more important than the path smoothness. According to Eqs (24)–(27), there are

$$\begin{cases} \omega_1 = 0.4117 \\ \omega_2 = 0.3171 \\ \omega_3 = 0.1669 \\ \omega_4 = 0.1043 \end{cases} \quad (34)$$

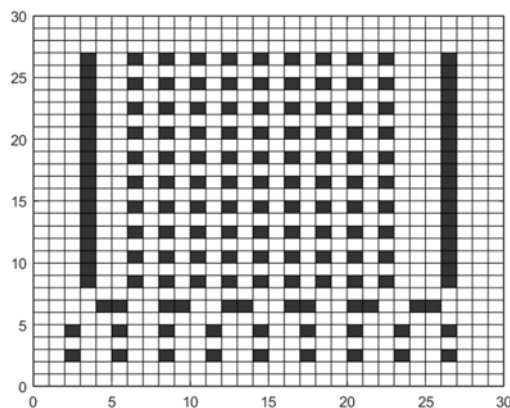
That is, the weight coefficients of path reachability, path safety, path length and path smoothness are 0.4117, 0.3171, 0.1669 and 0.1043, respectively. Compared with other weight coefficient combinations, the above weight coefficient combination has obvious superiority.

## 5.2. Sensor model

In this paper, based on the working mode of RPLIDAR A2 lidar [25], an analog sensor is designed for experimental application. The mobile robot can use sensors to sense local information in an unknown environment, and the detection radius of sensors can be adaptively adjusted according to the surrounding environment of the mobile robot's current position. When the sensor detects many obstacles in the surrounding environment, it will expand the detection radius to obtain more local environmental information to ensure the safety of the mobile robot. In contrast, when the sensor detects few obstacles in the surrounding environment, it will reduce the detection radius to reduce the energy loss of the sensor.

## 5.3. Scene 1

We verify the performance of the CBIACO algorithm and the quality of the generated global path. In this paper, the CBIACO algorithm is compared with the traditional ant colony algorithm (ACO) and adaptive ant colony algorithm [26] (AACO) in a  $30 \times 30$  simple grid map. The simulation environment is shown in Figure 6. In the grid, the white squares are the free squares, which represent the movable area of the mobile robot. The black squares are the obstacle squares, which represent the area where the mobile robot cannot move. The related parameters of the mobile robot path planning are set as follows: The starting point grid number is 1, the target point grid number is 829, the steering angle constraint of the mobile robot is 90, and the penalty information value cons is 1.



**Figure 6.**  $30 \times 30$  simple grid map.

The parameter design of the three algorithms is shown in Table 9.

**Table 9.** Parameters of ant colony algorithm.

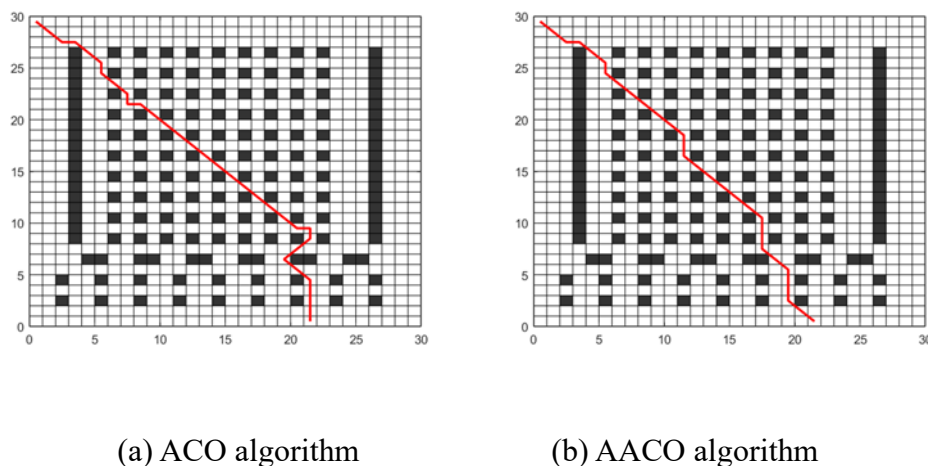
| Parameter Setting                           | ACO algorithm | AACO algorithm           | CBIACO algorithm         |
|---|---------------|--------------------------|--------------------------|
| Initial pheromone concentration             | 1             | 1                        | 1                        |
| Number of ants                              | 50            | 50                       | 50                       |
| Maximum number of iterations                | 300           | 300                      | 300                      |
| Pheromone factor                            | 1             | 1                        | Adaptive numerical value |
| Heuristic information factor                | 7             | 7                        | Adaptive numerical value |
| Pheromone volatile factor                   | 0.3           | Adaptive numerical value | 0.3                      |
| Pheromone constant                          | 1             | 1                        | 1                        |
| Pseudo-random probability adjustment factor | -             | Adaptive numerical value | 0.3                      |

### 5.3.1. Path quality

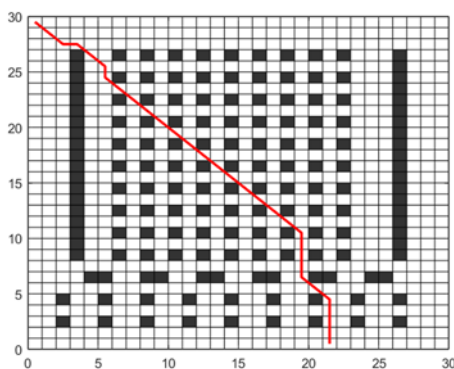
Figure 7(a) is the path trace diagram of the ACO algorithm. As can be seen from Figure 7(a), the quality of the path trajectory obtained by the ACO algorithm is obviously poor, as the path length is



long, the smoothness is poor, and there are many large turning angles. Figure 7(b) is the path trace of the AACO algorithm. As can be seen from Figure 7(b), compared with ACO algorithm, the quality of the path obtained by this algorithm has been improved to some extent in path length and path smoothness, but there is still room for further optimization. Figure 8 shows the path trajectory of the CBIACO algorithm. It can be seen from Figure 8 that the quality of the path obtained by the CBIACO algorithm is the best, and the path length and path smoothness are the best. Therefore, in terms of path quality, this paper shows that the CBIACO algorithm is superior to the ACO algorithm and AACO algorithm.



**Figure 7.** Path diagram of ACO algorithm and AACO algorithm for scene 1.

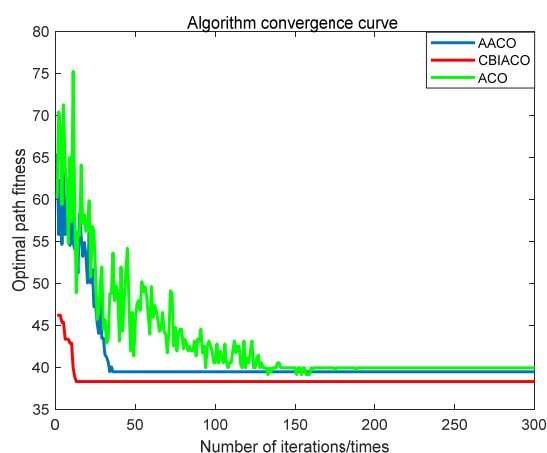


**Figure 8.** Path diagram of CBIACO algorithm for scene 1.

### 5.3.2. Convergence curve analysis

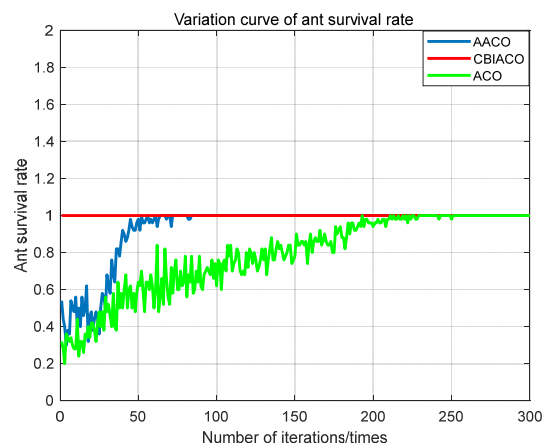
Figure 9 shows the convergence curve of the algorithm. As can be seen from Figure 9, the ACO algorithm converges in the 189th generation, the convergence speed is very slow, the path quality after convergence is not high, and the optimal path fitness is 39.94. It can also be found from the figure that the ACO algorithm had a better path individual in the iterative process (in the 155th generation, the optimal path fitness was 39.11), but the path was not retained in the iterative process of the algorithm.

The AACO algorithm converges in 38th generation, and the optimal path fitness after convergence is 39.46. Compared with the ACO algorithm, the AACO algorithm has improved in optimal path fitness and convergence iteration times. The CBIACO algorithm has converged in the 13th generation, and the optimal path fitness after convergence is 38.28. This shows that the CBIACO algorithm is superior to the ACO algorithm and AACO algorithm in terms of optimal path fitness and convergence iteration times. This is because the CBIACO algorithm can accelerate the convergence speed and improve the convergence quality of the algorithm to the greatest extent by improving the probability transfer function, introducing the adaptive pheromone factor and adaptive heuristic factor and the adding path optimization strategy. In addition, aiming at the data fallback phenomenon in the ACO algorithm and AACO algorithm, the CBIACO algorithm effectively avoids this problem by adding an elite preservation strategy.



**Figure 9.** Convergence curves of the algorithms.

### 5.3.3. Analysis of ant survival rate



**Figure 10.** Variation curves of ant survival rate.

The variation curves of ant survival rate of the three algorithms in the  $30 \times 30$  grid map are shown in Figure 10. It can be seen from Figure 10 that before 251 generations, the ant survival rate of the ACO algorithm gradually increased with the increase of the number of iterations. After 251 generations, the ant survival rate reached 100% and remained stable. The AACO algorithm proposed in [26] achieved a 100% ant survival rate after 82 generations. This shows that the deadlock penalty factor of the AACO algorithm plays a certain role. However, at the initial stage of algorithm iteration, the ant survival rate is still low. The CBIACO algorithm proposed in this paper ensures the survival of ants through the backtracking mechanism, and the path length zeroing mechanism can guide ants to adjust the direction of advance and jump out of the deadlock. The CBIACO algorithm ensures that the ant survival rate remains 100% from beginning to end, thus effectively solving the deadlock problem.

#### 5.3.4. Comprehensive comparative analysis

In scene 1, each of the three algorithms is simulated 20 times and averaged to get the data shown in Table 10. It can be seen from Table 10 that the CBIACO algorithm is superior to the ACO algorithm and AACO algorithm in average path fitness, average convergence iteration times and average convergence time. The formula for calculating the convergence time of the algorithm is shown in Eq (35).

$$t = \frac{N_{c1}}{N_{cmax}} * T \quad (35)$$

T represents the running time of the whole program,  $N_{c1}$  represents the iteration times when the algorithm converges,  $N_{cmax}$  represents the total iteration times of the algorithm, and t represents the running time of the program when the algorithm converges (that is, the convergence time of the algorithm).

**Table 10.** Simulation results of the three algorithms.

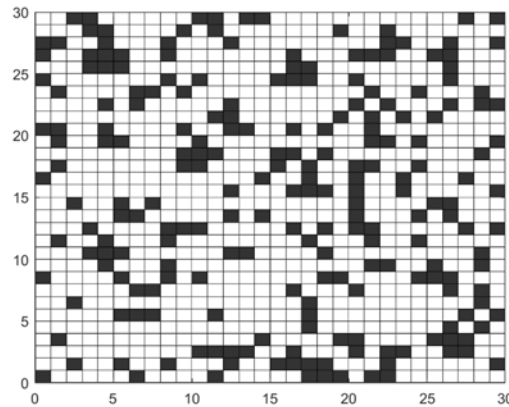
| Parameter Setting | Average path fitness (/) | Average number of convergence iterations (times) | Average convergence time of algorithm (s) |
|-------------------|--------------------------|--|---|
| CBIACO algorithm  | 38.61                    | 18   | 2.4908                                    |
| AACO algorithm    | 39.73                    | 48.45  | 6.9167                                    |
| ACO algorithm     | 40.5                     | 204.8  | 27.8559                                   |

It can be seen from Table 10 that the CBIACO algorithm is superior to the ACO algorithm and AACO algorithm in average path fitness, average convergence iteration times and average convergence time. Therefore, the performance superiority of the CBIACO algorithm can be proved.

#### 5.4. Scene 2

Scene 1 verifies the performance of the CBIACO algorithm in a  $30 \times 30$  simple grid map. In order to further verify the performance of the CBIACO algorithm, in this paper, the CBIACO algorithm is compared with the beetle antennae search [27] (BAS) algorithm, the biased min-consensus [28] (BMC) algorithm and the distributed biased min-consensus [29] (DBMC) algorithm in a  $30 \times 30$  complex grid

map. The simulation environment is shown in Figure 11. In the grid, the white squares are the free squares, which represent the movable area of the mobile robot. The black squares are the obstacle squares, which represent the area where the mobile robot cannot move. The related parameters of the mobile robot path planning are set as follows: The starting point grid number is 1, the target point grid number is 900, the steering angle constraint of the mobile robot is 90, and the penalty information value cons is 1.



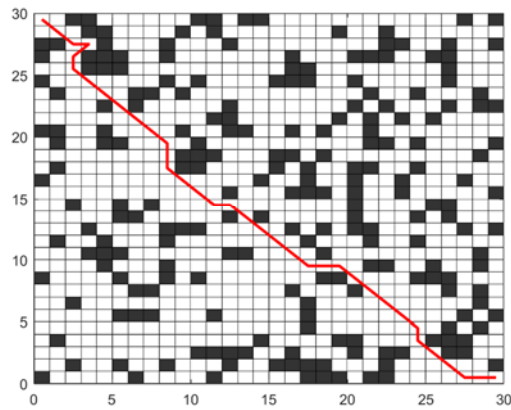
**Figure 11.**  $30 \times 30$  complex grid map.

The parameter design of the CBIACO algorithm is shown in Table 9. The BAS algorithm, BMC algorithm and DBMC algorithm respectively adopt the default parameters in [27–29].

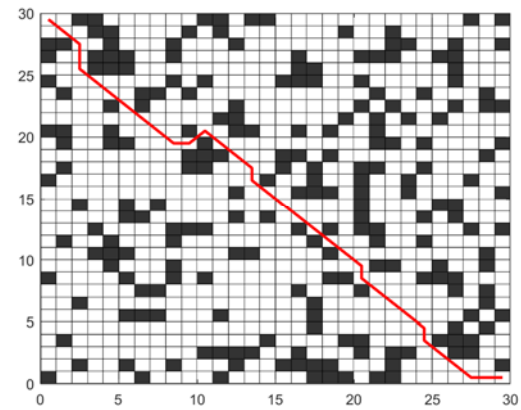
Figure 12(a) is the path trace diagram of the DBMC algorithm. As can be seen from Figure 12(a), the quality of the path trajectory obtained by the DBMC algorithm is very good. Unfortunately, there is a large turning angle at the starting point, which greatly affects the smoothness of the path. This situation may be related to the special terrain at the starting point. Figure 12(b) is the path trace of the BMC algorithm. As can be seen from Figure 12(b), the quality of the path trajectory obtained by the BMC algorithm is also very good. However, this path crosses diagonal obstacles, which will greatly affect the safety of the path. Obviously, the BMC algorithm does not consider the existence of diagonal obstacles. Figure 12(c) is the path trace diagram of the BAS algorithm. As can be seen from Figure 12(c), the quality of the path trajectory obtained by the BAS algorithm is obviously the worst, as the path length is long, the smoothness is poor, and there are many large turning angles. Figure 12(d) shows the path trajectory of the CBIACO algorithm. It can be seen from Figure 12(d) that the quality of the path obtained by the CBIACO algorithm is the best, and its path length and path smoothness are the best. Therefore, in terms of path quality, this paper shows that the CBIACO algorithm is superior to the BAS algorithm, BMC algorithm and DBMC algorithm.

For scene 2, each of the four algorithms is simulated 20 times and averaged to get the data shown in Table 11. It can be seen from Table 11 that the CBIACO algorithm is superior to BAS algorithm, BMC algorithm and DBMC algorithm in average path fitness. Specifically, the BAS algorithm has the worst average path fitness, because the path length is long, the smoothness is poor, and there are many large turning angles. The average path fitness of the DBMC algorithm is slightly better than that of the BMC algorithm. This may be due to the distributed nature of the DBMC algorithm. However, in general, the average path fitness values of the BMC algorithm and DBMC algorithm are close. Both

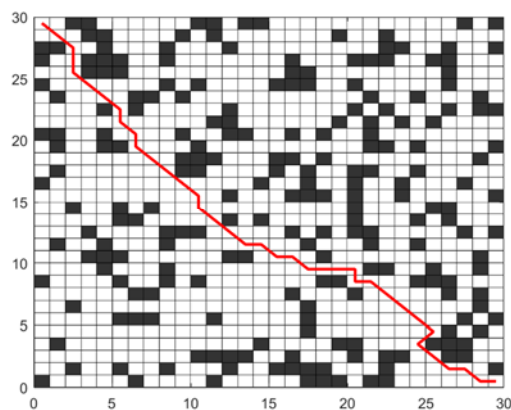
algorithms' paths could be better. The quality of the path obtained by the CBIACO algorithm is the best, and its path length and path smoothness are the best. Therefore, the performance superiority of the CBIACO algorithm can be proved.



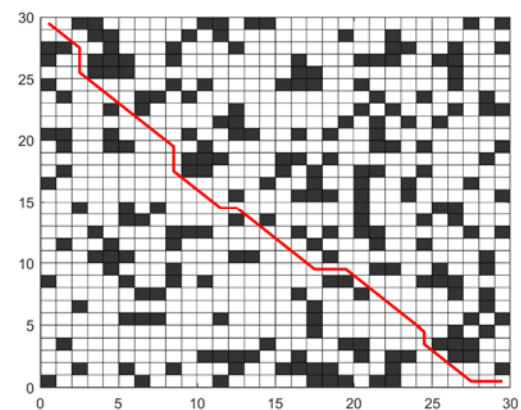
(a) DBMC algorithm



(b) BMC algorithm



(c) BAS algorithm



(d) CBIACO algorithm

**Figure 12.** Path diagram of the four algorithms for scene 2.

**Table 11.** Simulation results of the four algorithms.

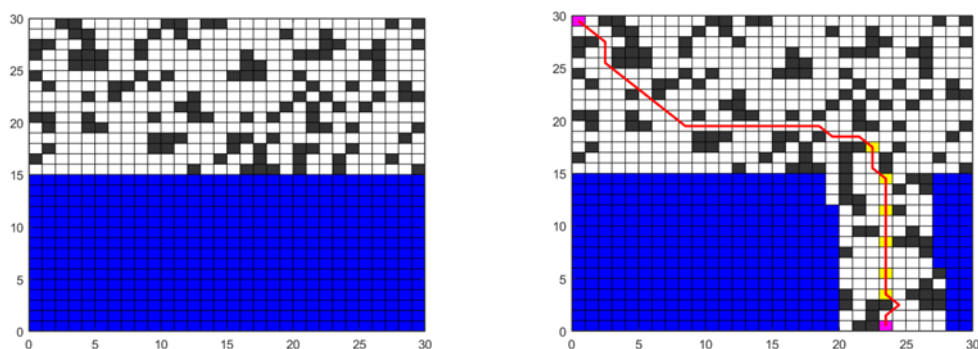
| Algorithm            | CBIACO algorithm | BAS algorithm | BMC algorithm | DBMC algorithm |
|----------------------|------------------|---------------|---------------|----------------|
| Average path fitness | 43.94            | 52.71         | 47.12         | 46.35          |

### 5.5. Scene 3

This scene is an unknown environment with irregular obstacle distribution. The motion trajectory of the mobile robot based on the CBIACO algorithm is shown in Figure 13. The blue grid area represents the unknown environment, the pink squares represent the starting point and target point,

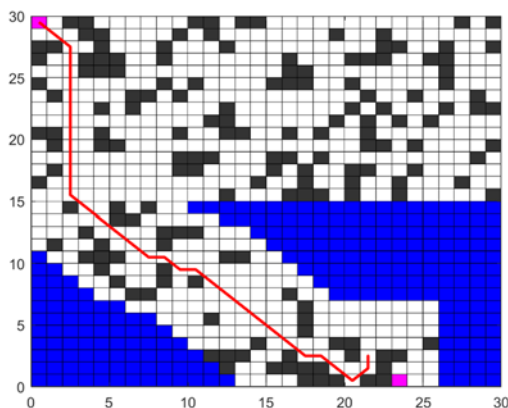
and the yellow squares represent the guidance points. Figure 13(a) shows the initial map. Figure 13(b) shows the complete exploration trajectory.

It can be seen from Figure 13(b) that the mobile robot can cross the complex environment with irregular distribution of obstacles without falling into oscillation or stagnation. This is because UPES makes mobile robots make full use of prior information such as current position information, local environment information and target point information, and it reduces the probability of being misled by terrain information. At the same time, the CBIACO algorithm can effectively deal with the environment with irregular distribution of obstacles, generate a feasible exploration path and protect the mobile robot from oscillation or stagnation.



(a) Initial map of the complex environment (b) Complete exploration trajectory

**Figure 13.** Path exploration trajectory diagram of mobile robot based on CBIACO algorithm in Scene 3.



**Figure 14.** Path exploration trajectory diagram of mobile robot based on GRA in Scene 3.

The exploration trajectory of the mobile robot based on the GRA is shown in Figure 14. From Figure 14, the random entrance is chosen as grid number 423. The random entrance is far from target point  $G$ , which implies that the theoretical cost of the mobile robot detecting an unknown environment is higher. When the mobile robot uses the GRA algorithm to drive to grid numbers 862 and 832, the mobile robot will fall into the state of repeated oscillation on these two grids; thus, the mobile robot cannot reach the target point. There are two main reasons: First, the local information generated by the

complex terrain causes misleading information for the GRA algorithm, which eventually makes the mobile robot make incorrect decisions. Second, the greedy strategy adopted by the GRA algorithm pays too much attention to the selection of the best neighbor path point and ignores the feasibility and effectiveness of the exploration path.

For Scene 3, the CBIACO algorithm and GRA are simulated 10 times, and the average and standard deviation of the corresponding path evaluation index are taken. The results are shown in Table 12. The CBIACO algorithm has a much higher success rate (100%) than the GRA (40%), which indicates that the CBIACO algorithm has a higher success rate and is more suitable for Scene 3. In addition, the CBIACO algorithm is superior to the GRA in terms of reachability for Scene 3. The CBIACO algorithm has obviously better average path length, average path smoothness and average path evaluation value than the GRA. This shows that the UPES adopted by the CBIACO algorithm can indeed play an important role. In terms of the average path safety, the CBIACO algorithm is slightly inferior to the GRA. That is because the path safety value defined in this paper is the ratio of the safety penalty obtained by a path to its length, and a longer path may increase the safety of the path.

**Table 12.** Evaluation results in a complex unknown environment.

| Algorithm | Success rate | Measurement index  | Path length | Path safety | Path smoothness | Path evaluation value |
|-----------|--------------|--------------------|-------------|-------------|-----------------|-----------------------|
| CBIACO    | 100%         | Average value      | 46.7162     | 5.7633      | 11.6239         | 10.8368               |
|           |              | Standard deviation | 0.5792      | 0.461       | 1.7287          | 0.128                 |
| GRA       | 40%          | Average value      | 50.5417     | 5.5943      | 15.3153         | 11.8067               |
|           |              | Standard deviation | 2.8168      | 0.3246      | 3.2383          | 0.6343                |

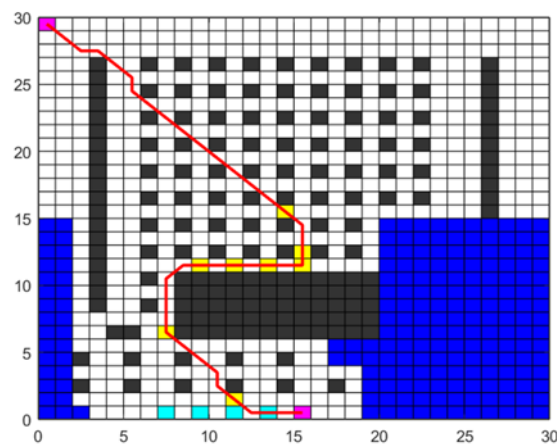
In terms of the stability of the two path exploration algorithms, the CBIACO algorithm and GRA are unstable in Scene 3. The instability of the GRA comes from the randomness of the entrance of the unknown environment. The instability of the CBIACO algorithm comes from the instability of the IACO algorithm in a complex environment (as a probabilistic intelligent search algorithm, the IACO algorithm is unstable in complex maps). However, the standard deviation data in Table 12 show that although both are unstable, the CBIACO algorithm is far more stable than the GRA algorithm. Thus, the CBIACO algorithm is superior to the GRA algorithm in stability.

Based on the above analysis, the proposed CBIACO algorithm in this paper is superior to the GRA algorithm in many performance indices, which verifies the feasibility and superiority of the CBIACO algorithm for Scene 3.

#### 5.6. Scene 4

Scene 4 is an unknown environment with special terrain. The exploration trajectory of the mobile robot based on the CBIACO algorithm is shown in Figure 15. In the grid, the blue squares represent

the unknown environment, the pink squares represent the starting point and target point, the yellow squares represent the guidance points, and the sky blue squares represent the virtual target points. As shown in Figure 15, the mobile robot travels along the guidance path. When the mobile robot travels to the third guidance point (grid number 556), the new  $i\_set$  completely repeats with the previous  $i\_set$ . At this time, the mobile robot sets the detection range of the sensor to the maximum and generates virtual target point  $VG_1$  (grid number 884) at two squares to the left side of point  $G$ . Under the joint action of virtual target point  $VG_1$ , prior information and local environment information, the fourth guidance point (grid number 554) is generated, and the mobile robot moves to the fourth guidance point. Afterwards, virtual target point  $VG_2$  (grid number 882), the fifth guidance point (grid number 552), virtual target point  $VG_3$  (grid number 880), the sixth guidance point (grid number 550), virtual target point  $VG_4$  (grid number 878) and the seventh guidance point (grid number 698) are successively generated, and the mobile robot travels along the guidance path to the seventh guidance point.



**Figure 15.** Path exploration trajectory diagram of mobile robot based on CBIACO algorithm in Scene 4.

At this time, the mobile robot did not completely get rid of the special terrain. Under the joint action of virtual target point  $VG_2$ , prior information and local environmental information, the eighth guidance point (grid number 852) and the corresponding guidance path are generated. When the mobile robot reaches the eighth guidance point, it completely bypasses the special terrain, and the target point also appears in the known environment. Finally, the mobile robot uses the IACO algorithm to generate the guidance path to point  $G$  and subsequently drives along the guidance path to point  $G$ .

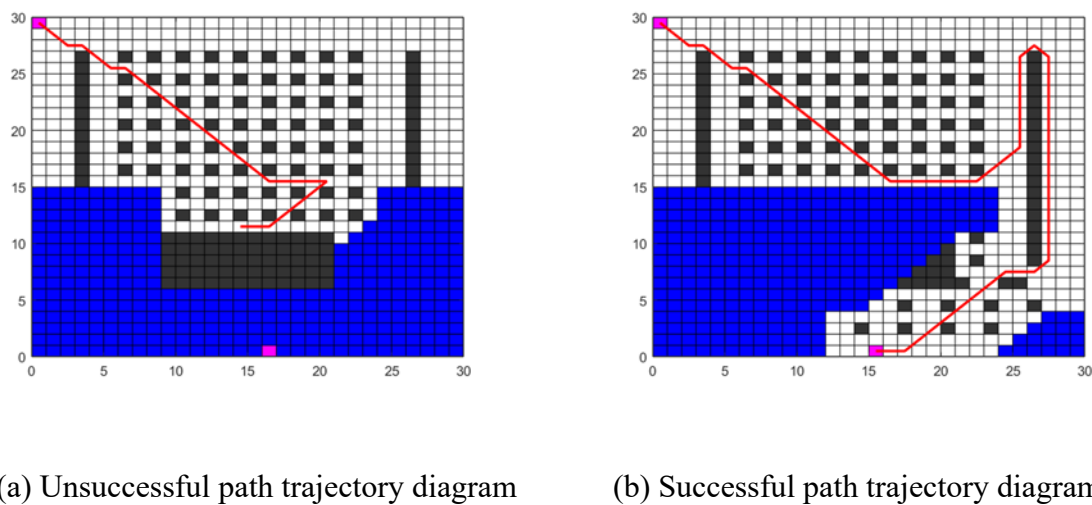
As shown in Figure 15, the mobile robot can bypass the special terrain to reach its destination without falling into the state of oscillation or stagnation. That is because the CBIACO algorithm can generate virtual target points when the mobile robot encounters special terrain. Under the joint action of current position information, local environment information and virtual target point information, the mobile robot can move along the boundary of special terrain until the mobile robot bypasses the special terrain.

The exploration trajectory of the mobile robot based on the GRA is shown in Figure 16. Figure 16(a),(b) show the path exploration trajectories of the mobile robots that did not successfully reach the target point and that successfully reached the target point, respectively. In Figure 16(a), the



random entrance is selected as grid number 441. When the mobile robot adopts the GRA to travel to grid number 555 and grid number 556, the mobile robot encounters the special terrain and falls into the repeated oscillation state, so it cannot reach the target point. The main reasons are described in Section 5.5.

In Figure 16(b), the random entrance is selected as grid number 448. The mobile robot first uses the IACO algorithm to reach grid number 448, subsequently uses the sensor and GRA to plan the path and finally reaches the target point. Although the mobile robot successfully reaches the target point, the quality of the entire exploration trajectory is very poor. Figure 16 shows that when the mobile robot adopts the GRA algorithm, whether it can reach the target point depends on the location of the random entrance and construction of the map environment. In many cases, even if the mobile robot can successfully reach the target point, the exploration path has very poor quality.



**Figure 16.** Path exploration trajectory diagram of mobile robot based on GRA in Scene 4.

For scene 4, the CBIACO algorithm and GRA are simulated 10 times, and the average and standard deviation of the corresponding path evaluation index are taken. The results are shown in Table 13. In Table 13, the CBIACO algorithm has a much higher success rate (100%) than the GRA (30%), which indicates that the CBIACO algorithm has better reachability than the GRA and is more suitable for scene 4. In terms of average path length, average path smoothness and average path evaluation value, the CBIACO algorithm is significantly better than the GRA. Thus, the UPES in the CBIACO algorithm can indeed play an important role. In terms of the average path safety value, the CBIACO algorithm is slightly inferior to the GRA. The main reasons are described in Section 5.5.

In terms of the stability of the two path exploration algorithms, the CBIACO algorithm is very stable for scene 4, and its standard deviation is 0. The GRA is unstable for scene 4, and its standard deviation is larger. The instability of the GRA comes from the randomness of the entrance to the unknown environment.

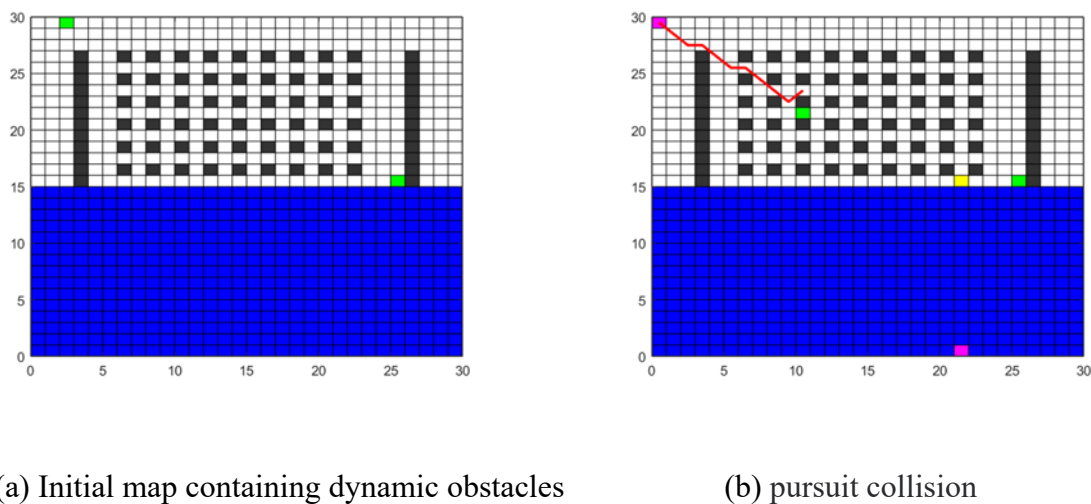
Based on the above analysis, the proposed CBIACO algorithm in this paper is superior to the GRA in many performance indices, which verifies the feasibility and superiority of the CBIACO algorithm for scene 4.

**Table 13.** Evaluation results in an unknown environment containing special terrain.

| Algorithm | Success rate | Measurement index  | Path length | Path safety | Path smoothness | Path evaluation value |
|-----------|--------------|--------------------|-------------|-------------|-----------------|-----------------------|
| CBIACO    | 100%         | Average value      | 52.2843     | 6.3116      | 10.2102         | 11.7926               |
|           |              | Standard deviation | 0           | 0           | 0               | 0                     |
| GRA       | 30%          | Average value      | 58.1027     | 6.09        | 15.603          | 13.2559               |
|           |              | Standard deviation | 12.9312     | 0.1017      | 1.1785          | 2.2622                |

### 5.7. Scene 5

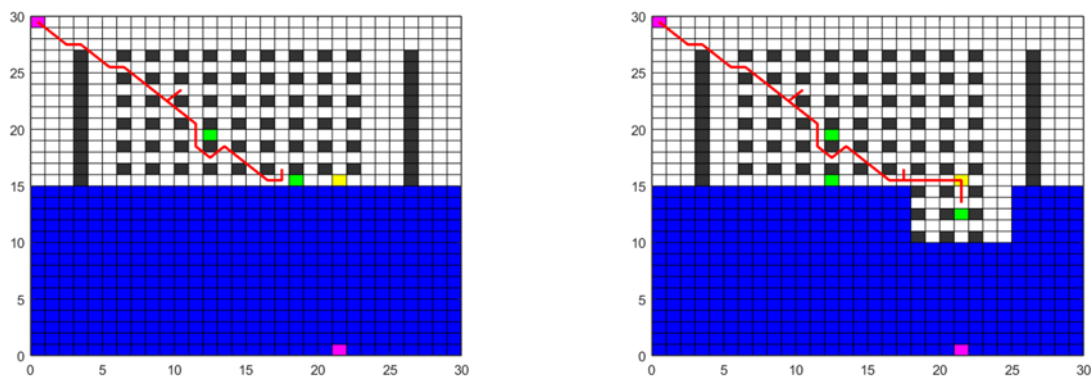
Scene 5 is an unknown environment with dynamic obstacles. The exploration trajectory of the mobile robot based on the CBIACO algorithm is shown in Figure 17. In the grid, the blue squares represent the unknown environment, the pink squares represent the start point and target point, the yellow square represents the guidance point, and the green squares represent the dynamic obstacles. Figure 17(a) shows the initial map that contains the unknown environment and dynamic obstacles. In this map, the mobile robot travels along the guidance path and guidance points.

**Figure 17.** Initial map and track diagram of pursuit collision.

Dynamic obstacle 1 starts to move after the mobile robot has traveled for 6 steps and travels from grid number 3 to the lower right direction at a speed of 2 unit steps. When the mobile robot travels along the guidance path, it is predicted that it will successively encounter pursuit collision and stagnation collision with dynamic obstacle 1, and the potential collision points are grid number 251 and grid number 313, respectively. For the pursuit collision, the mobile robot performs steering

behavior. In other words, when the mobile robot predicts the collision, the mobile robot turns left to avoid the dynamic obstacle, waits for dynamic obstacle 1 to pass, returns to the position before obstacle avoidance and continues to move along the guidance path (as shown in Figure 17(b)). The decision time of obstacle avoidance is 0.005732 s. For the stagnation collision, the mobile robot implements the local path replanning strategy to avoid dynamic obstacles (as shown in Figure 18(a)). Then, the mobile robot continues to move along the guidance path, and the decision time of obstacle avoidance is 0.006879 s. The decision time of obstacle avoidance mentioned in this paper refers to the time taken by the mobile robot from detecting dynamic obstacles to formulating obstacle avoidance strategies, rather than the time taken by the mobile robot to actually avoid dynamic obstacles. In fact, the time taken by the mobile robot to actually avoid dynamic obstacles depends on its own performance.

Dynamic obstacle 2 starts to move after the mobile robot has traveled for 15 steps and travels from grid number 446 to the left direction at a speed of 1 unit step. When the mobile robot is driving, it is predicted that it will have a forward point collision with dynamic obstacle 2 moving to the left, and the potential collision point is grid number 439, so the mobile robot performs steering behavior. In other words, when the mobile robot predicts the collision, the mobile robot turns left to avoid the dynamic obstacle, waits for dynamic obstacle 2 to pass, returns to the position before obstacle avoidance (as shown in Figure 18(a)) and continues to move along the guidance path until it reaches the initial guidance point. The decision time of obstacle avoidance is 0.006063 s.



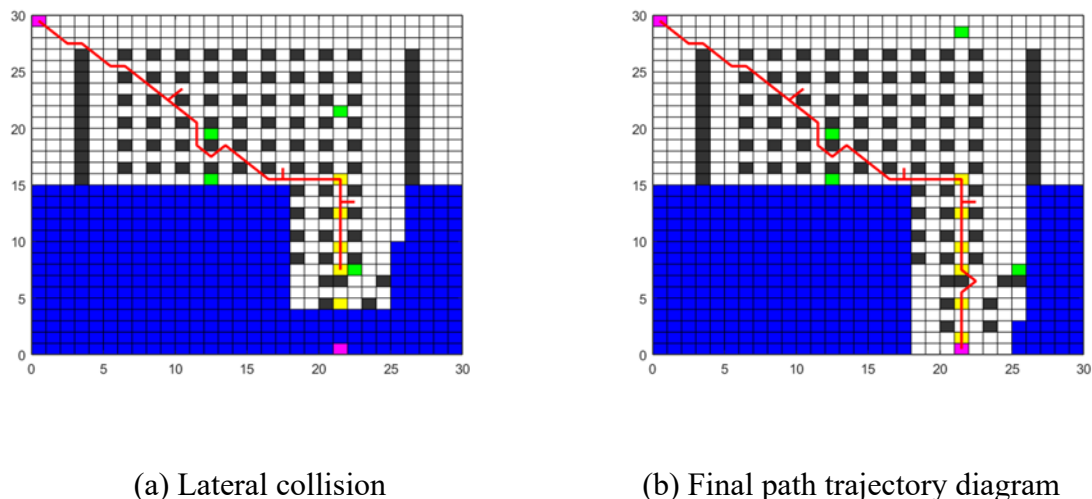
(a) Stagnation collision and forward point collision

(b) Forward surface collision

**Figure 18.** Exploration trajectory diagram with multiple collision types.

Then, the mobile robot travels along the guidance path and guidance points. In the process of sensor detection, the mobile robot suddenly finds dynamic obstacle 3 moving upward from grid number 562 with the speed of 1 unit step. When the mobile robot travels to grid number 502, it only meets dynamic obstacle 3. According to the proposed collision classification model in this paper, there will be a forward surface collision between them, and the mobile robot will perform the steering behavior, i.e., when the mobile robot predicts the collision, the mobile robot will turn left to avoid the dynamic obstacle, wait for dynamic obstacle 3 to pass, return to the position before obstacle avoidance and continue to move along the guidance path (as shown in Figure 18(b)). The decision time of obstacle avoidance is 0.00067 s.

In the process of continuing to drive, the mobile robot encounters dynamic obstacle 4, which is moving to the right from grid number 679 with the speed of 1 unit step. The mobile robot predicts in the traveling process that it will have a lateral collision with dynamic obstacle 4 that moves to the right, and the potential collision point is grid number 682, so the mobile robot performs waiting behavior, i.e., when the mobile robot predicts the collision, the mobile robot temporarily stops traveling and waits for dynamic obstacle 4 to pass. Then, the mobile robot continues to move along the guidance path, and the obstacle avoidance decision time is 0.000221 s (as shown in Figure 19(a)). The final path trajectory diagram of the mobile robot is shown in Figure 19(b).



**Figure 19.** Lateral collision diagram and full motion trajectory diagram.

Figures 17–19 show that the collision classification model and corresponding dynamic obstacle avoidance strategy adopted by the CBIACO algorithm are applicable to the unknown environment containing dynamic obstacles. Thus, the potential collision types considered by the CBIACO algorithm are comprehensive, and the CBIACO algorithm has a good dynamic obstacle avoidance effect and strong real-time performance.

The average path evaluation values of the CBIACO algorithm in scene 3, scene 4 and scene 5 are approximately 89.74%, 91.79% and 88.96% of those of the GRA, respectively. The experimental results in three different unknown environments show that the proposed CBIACO algorithm in this paper is more applicable and general and more comprehensively considers the situation. However, the GRA may make the mobile robot stagnate or fluctuate, so the mobile robot cannot reach its destination. Additionally, the quality of the exploration path generated by the GRA is low, and the length, smoothness and accessibility of the exploration path are obviously poor.

## 6. Summary

In this paper, a path planning method for mobile robots in complex environments based on an improved ant colony algorithm (CBIACO) is proposed. First, by optimizing the traditional ant colony algorithm, the generation time of the global path is reduced, and the generation quality of the global path is improved by improving the probability transfer function and designing adaptive component

weights. Second, a global path optimization strategy based on diagonal obstacle detection and optimization mechanism is proposed to solve the problem that the global path crosses diagonal obstacles, realize the re-optimization of the global path and further improve the quality of the global path. Then, UPES is proposed, which makes the mobile robot reach the destination safely, efficiently and quickly by using the real-time local information obtained by sensors and the corresponding path exploration mechanism. Finally, a collision classification model is proposed, and the corresponding dynamic obstacle avoidance strategy is given. A more comprehensive potential collision situation is considered, and the mobile robot can effectively avoid various obstacles through behavioral obstacle avoidance and local path replanning. The experimental results show that the CBIACO algorithm can quickly generate high-quality global paths in known environments; the CBIACO algorithm can make the mobile robot safely and quickly cross the unknown environment to reach the designated target point in the unknown environment. The new dynamic obstacle avoidance strategy can make mobile robots avoid dynamic obstacles in different directions at a lower cost.

However, the proposed CBIACO algorithm in this paper has some shortcomings. The collision between a mobile robot and dynamic obstacles with irregular motion is not considered. In addition, the mechanical performance constraints of mobile robots are not considered. In the future, we will focus on the collision between mobile robots and dynamic obstacles with irregular motion and increase the mechanical performance constraints of robots to further improve the algorithm, so that the CBIACO algorithm can be more applicable in complex environments.

### Use of AI tools declaration

The authors declare they have not used artificial intelligence (AI) tools in the creation of this article.

### Acknowledgments

This research was funded by the Chinese National Natural Science Foundation, grant number 61902273.

### Conflict of interest

The authors declare there is no conflict of interest.

### Reference

1. T. Ort, I. Gilitschenski, D. Rus, Autonomous navigation in inclement weather based on a localizing ground penetrating radar, *IEEE Rob. Autom. Lett.*, **5** (2020), 3267–3274. <https://doi.org/10.1109/LRA.2020.2976310>
2. H. Nam, Data-gathering protocol-based AUV path-planning for long-duration cooperation in underwater acoustic sensor networks, *IEEE Sens. J.*, **18** (2018), 8902–8912. <https://doi.org/10.1109/JSEN.2018.2866837>
3. Z. Huang, C. Chen, M. Pan, Multiobjective UAV path planning for emergency information collection and transmission, *IEEE Internet Things J.*, **7** (2020), 6993–7009. <https://doi.org/10.1109/JIOT.2020.2979521>

4. J. Han, Y. Seo, Mobile robot path planning with surrounding point set and path improvement, *Appl. Soft Comput.*, **57** (2017), 35–47. <https://doi.org/10.1016/j.asoc.2017.03.035>
5. P. Sudhakara, V. Ganapathy, B. Priyadharshini, K. Sundaran, obstacle avoidance and navigation planning of a wheeled mobile robot using amended artificial potential field method, *Procedia Comput. Sci.*, **133** (2018), 998–1004. <https://doi.org/10.1016/j.procs.2018.07.076>
6. R. Fareh, M. Baziyad, T. Rabie, M. Bettayeb, Enhancing path quality of real-time path planning algorithms for mobile robots: A sequential linear paths approach, *IEEE Access*, **8** (2020), 167090–167104. <https://doi.org/10.1109/ACCESS.2020.3016525>
7. J. Song, C. Hao, J. Su, Path planning for unmanned surface vehicle based on predictive artificial potential field, *Int. J. Adv. Rob. Syst.*, **17** (2020), 1–13. <https://doi.org/10.1177/1729881420918461>
8. S. Katoch, S. S. Chauhan, V. Kumar, A review on genetic algorithm: past, present, and future, *Multimedia Tools Appl.*, **80** (2021), 8091–8126. <https://doi.org/10.1007/s11042-020-10139-6>
9. F. Wang, H. Zhang, A. Zhou, A particle swarm optimization algorithm for mixed-variable optimization problems, *Swarm Evol. Comput.*, **60** (2021), 1–36. <https://doi.org/10.1016/j.swevo.2020.100808>
10. S. Gao, Y. Ding, B. M. Chen, A frontier-based coverage path planning algorithm for robot exploration in unknown environment, in *2020 39th Chinese Control Conference (CCC)*, IEEE, (2020), 3920–3925. <https://doi.org/10.23919/CCC50068.2020.9188784>
11. N. Yu, S. Wang, C. Xu, RGB-D based autonomous exploration and mapping of a mobile robot in unknown indoor environment, *Robot.*, **39** (2017), 860–871. <https://doi.org/10.13973/j.cnki.robot.2017.0860>
12. X. Lan, X. Lv, W. Liu, Y. He, X. Zhang, Research on robot global path planning based on improved A-star ant colony algorithm, in *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, IEEE, (2021), 613–617. <https://doi.org/10.1109/IAEAC50856.2021.9391099>
13. L. Meng, X. You, S. Liu, Multi-colony collaborative ant optimization algorithm based on cooperative game mechanism, *IEEE Access*, **8** (2020), 154153–154165. <https://doi.org/10.1109/ACCESS.2020.3011936>
14. S. Biswas, S. G. Anavatti, M. A. Garratt, A particle swarm optimization based path planning method for autonomous systems in unknown terrain, in *2019 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, IEEE, (2019), 57–63. <https://doi.org/10.1109/ICIAICT.2019.8784851>
15. K. Wu, H. Wang, M. A. Esfahani, S. Yuan, Achieving real-time path planning in unknown environments through deep neural networks, *IEEE Trans. Intell. Transp. Syst.*, **23** (2022), 2093–2102. <https://doi.org/10.1109/TITS.2020.3031962>
16. J. S. Willners, D. Gonzalez-Adell, J. D. Hernández, È. Pairet, Y. Petillot, Online 3-dimensional path planning with kinematic constraints in unknown environments using hybrid A\* with tree pruning, *Sensors*, **21** (2021), 1–20. <https://doi.org/10.3390/s21041152>
17. Z. Jia, P. Lin, J. Liu, L. Liang, Online cooperative path planning for multi-quadrotors in an unknown dynamic environment, *Proc. Inst. Mech. Eng., Part G: J. Aerosp. Eng.*, **236** (2022), 567–582. <https://doi.org/10.1177/09544100211016615>
18. A. Q. Faridi, S. Sharma, A. Shukla, R. Tiwari, J. Dhar, Multi-robot multi-target dynamic path planning using artificial bee colony and evolutionary programming in unknown environment, *Intell. Serv. Rob.*, **11** (2018), 171–186. <https://doi.org/10.1007/s11370-017-0244-7>

19. H. Huang, G. Tan, L. Jiang, Robot path planning using improved ant colony algorithm in the environment of internet of things, *J. Rob.*, **2022** (2022), 1–8. <https://doi.org/10.1155/2022/1739884>
20. Q. Jin, C. Tang, W. Cai, Research on dynamic path planning based on the fusion algorithm of improved ant colony optimization and rolling window method, *IEEE Access*, **10** (2020), 28322–28332. <https://doi.org/10.1109/ACCESS.2021.3064831>
21. K. Hao, H. Zhang, Z. Li, Y. Liu, Path planning of mobile robot based on improved obstacle avoidance strategy and double optimization ant colony algorithm, *Trans. Chin. Soc. Agric. Mach.*, **53** (2022), 303–312, 422.
22. K. Hao, J. Zhao, K. Yu, C. Li, C. Wang, Path planning of mobile robots based on a multi-population migration genetic algorithm, *Sensors*, **20** (2020), 1–23. <https://doi.org/10.3390/s20205873>
23. K. Hao, J. Zhao, B. Wang, Y. Liu, C. Wang, The application of an adaptive genetic algorithm based on collision detection in path planning of mobile robots, *Comput. Intell. Neurosci.*, **2021** (2021), 1–20. <https://doi.org/10.1155/2021/5536574>
24. M. Kulich, J. J. Miranda-Bront, L. Preucil, A meta-heuristic based goal-selection strategy for mobile robot search in an unknown environment, *Comput. Oper. Res.*, **84** (2017), 178–187. <https://doi.org/10.1016/j.cor.2016.04.029>
25. W. Yue, *Design of Independent System for Map Construction of Robots*, Master thesis, Shanghai Jiao Tong University, 2020. <https://doi.org/10.27307/d.cnki.gsjtu.2020.001039>
26. 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), 1–10. <https://doi.org/10.1016/j.cie.2021.107230>
27. H. Zhang, X. Gan, S. Li, Z. Chen, UAV safe route planning based on PSO-BAS algorithm, *J. Syst. Eng. Electron.*, **33** (2022), 1151–1160. <https://doi.org/10.23919/JSEE.2022.000111>
28. S. Ma, K. Feng, J. Li, H. Wang, G. Cong, J. Huai, Proxies for shortest path and distance queries, *IEEE Trans. Knowl. Data Eng.*, **28** (2016), 1835–1850. <https://doi.org/10.1109/TKDE.2016.2531667>
29. Y. Zhang, S. Li, Distributed biased min-consensus with applications to shortest path planning, *IEEE Trans. Autom. Control*, **62** (2017), 5429–5436. <https://doi.org/10.1109/TAC.2017.2694547>



AIMS Press

©2023 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0>).