Research Article

Automatic Recharging Path Planning for Cleaning Robots

Bing Hao ¹, He Du ¹, Xuefeng Dai ¹ and Hong Liang ²

¹College of Computer and Control Engineering, Qiqihar University, Qiqihar, Heilongjiang Province, China

²College of Telecommunication and Electronic Engineering, Qiqihar University, Heilongjiang Province, China

Correspondence should be addressed to Bing Hao; haobing_learning@163.com

Received 10 February 2021; Revised 9 August 2021; Accepted 16 August 2021; Published 8 September 2021

Academic Editor: Hsu-Yang Kung

Copyright © 2021 Bing Hao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

To solve the problem of automatic recharging path planning for cleaning robots in complex industrial environments, this paper proposes two environmental path planning types based on designated charging location and multiple charging locations. First, we use the improved Maklink graph to plan the complex environment; then, we use the Dijkstra algorithm to plan the global path to reduce the complex two-dimensional path planning to one dimension; finally, we use the improved fruit fly optimization algorithm (IFOA) to adjust the path nodes for shorting the path length. Simulation experiments show that the effectiveness of using this path planning method in a complex industrial environment enables the cleaning robot to select a designated location or the nearest charging location to recharge when the power is limited. The proposed improved algorithm has the characteristics of a small amount of calculation, high precision, and fast convergence speed.

1. Introduction

Since the first autonomous mobile robot came out in the 1960s, mobile robot technology has developed rapidly. Mobile robot technology not only involves multiple disciplines such as artificial intelligence, pattern recognition, and automatic control but also involves many research topics such as navigation and positioning [1], sensor information fusion [2], path planning [3–5], and motion control [6]. In recent years, due to the advantages of saving labor cost, reusability, and intelligence, mobile robots have been applied to various fields: automatic assembly [7], sorting [8], and maintenance [9] of mobile robots in industrial field; mobile robots for surgery and rehabilitation in medical field [10]; and UAV [11] and AUV [12] in the military field. Among them, cleaning robot is the closest mobile robot to human life, which liberates people from tedious cleaning work coming out.

After decades of technological evolution, the development of cleaning robots is divided into three stages. The first phase started with the advent of the world's first cleaning robot. It was developed by the Swedish home appliance company Electrolux in 2001. The Trilobite sweeping robot produced in 2016 will automatically find the wall closest to

itself when it started to be used and scan along the wall to create a cleaning map. After the trilobite completes the scanning, the area of the map is used to calculate the cleaning time. When the battery is low, it will move to the charging stand for charging and then return to the original working position to continue cleaning. The second stage is to integrate positioning and navigation technology with SLAM to establish a map in real-time and design path planning to achieve. During the cleaning process, the Ecovacs series of cleaning robots combined with laser sensors can enable the robot to restore the room layout during the cleaning process, establish a real-time and accurate positioning system, and make the cleaning process more accurate and efficient. The third stage is to combine artificial intelligence with cleaning robots to realize the functions of human-computer interaction and recognition of complex obstacles and further realize the intelligence and efficiency of cleaning robots.

At any stage, the cleaning robot needs to automatically return to the charging location when the remaining power is limited, and, therefore, it is necessary to plan the return path. In [13], a new optimal path planning algorithm based on convolutional neural network (CNN) is proposed to quickly and efficiently explore the state space. However, it only applies to sampling based algorithms and needs to train the

parameters in advance. To solve the problem of convergence speed of ant colony algorithm, an improved ant colony optimization algorithm for mobile robot path planning based on grid method is proposed by Liu et al. [14]. The approach effectively generates good solutions quickly and lowers the risks of trapping in a local optimum. The disadvantage is that the convergence speed is too fast to get the optimal path. Ajeil et al. [15] propose a modification based on the standard ant colony optimization, which is implemented in association with grid-based modeling for the static and dynamic environments to solve the path planning problem. However, this algorithm has a large amount of computation and cannot quickly plan the path. Generally, the path planning algorithm is divided into two categories: heuristic algorithm [16] and artificial intelligence algorithm [3]. The traditional heuristic algorithm is determined according to the description situation of the established environment. It is usually based on the geometric mathematical model to complete the search of the robot path, and the solutions obtained are all definite values. This method increases exponentially with the complexity of the environment, resulting in very long computing time and inability to obtain a better solution, while the artificial intelligence algorithm applies the instinct of natural creatures to the robot path planning and uses the behavior of the group to find the global optimal solution in a complex space. The solution is not fixed. In complex industrial environments, robots often take a long time to calculate due to the complexity of the environment and cannot return to the charging place without collision and with a short path.

The traditional cleaning robot path planning algorithm is based on the two-dimensional path calculation of the environment map. The amount of calculation increases exponentially with the complexity of the environment, resulting in too long calculation time to reach the designated location. On the other hand, the cleaning robot has collided with the obstacle, causing the loss of the robot. Therefore, this paper combines the Dijkstra algorithm in the heuristic algorithm [17] and the improved fruit fly optimization algorithm (IFOA) in the artificial intelligence algorithm [18] to reduce the amount of calculation and avoid colliding with obstacles. First, the coordinate system of the cleaning robot is established, and the complex environment is modeled; secondly, to simplify the model and reduce the amount of calculation, the improved Maklink graph planning model is adopted; finally, the above-mentioned algorithm is used on this basis to realize the automatic recharging path planning of the cleaning robot.

The main contributions of this paper can be summarized as follows:

- (1) The improved Maklink graph further simplifies the complexity of industrial environment on the basis of Maklink graph. It prevents the cleaning robot from colliding with obstacles and speeds up the computation for later charging path planning.
- (2) It hybridizes the Dijkstra and IFOA to convert two-dimensional path planning into one-dimensional path planning and further reduces the computation.

The IFOA improves the searching precision and avoids too fast convergence rate on the basis of FOA.

- (3) Fixed and multiple charging locations are carried out by considering various complex environments to verify the effectiveness of IFOA. By comparing IFOA with FOA, ACA, and RRT in the same environments, it is proved that the proposed algorithm can make the cleaning robot return to the charging site without collision in a relatively short time.

2. Cleaning Robot Coordinate System

The cleaning robot studied in this paper adopts a moving structure of two standard wheels, each of which is equipped with an independent drive motor, and the purpose of changing the direction of movement of the robot is achieved by controlling the speed difference between the two wheels [19]. In the analysis process, the robot is modeled as a rigid body on wheels [20]. Take the ground as the global reference coordinate system: X and Y , robot local reference coordinate system: X_r and Y_r . The X_r is the longitudinal axis of the robot; Y_r is the lateral axis of the robot. The angle difference between the global reference system and the local reference system is determined by θ .

The global reference system and the local reference system are shown by

$$\dot{P} = (x, y, \theta)^T, \quad (1)$$

$$\dot{P}_r = (x_r, y_r, \theta)^T. \quad (2)$$

The mapping relationship between the two reference systems can be represented by orthogonal rotation transformation [21], as shown by

$$R(\theta) = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (3)$$

$$\dot{P} = R(\theta)\dot{P}_r, \quad (4)$$

$$\dot{P} = \begin{bmatrix} x_r \cos \theta + y_r \sin \theta \\ -x_r \sin \theta + y_r \cos \theta \\ \theta \end{bmatrix}. \quad (5)$$

Establishing the robot coordinate system is the primary condition for studying its specific position in the environmental area, judging the relative position of obstacles and the ending point, and laying the foundation for the next step of environmental modeling and path planning.

3. Environmental Modeling Method

Before the autonomous charging path of the cleaning robot is planned, it is necessary to know the specific locations of the cleaning robot, obstacles, starting point, and ending point in the environmental area. The robot is able to return to the charging place without collision with the obstacle [22]

FIGURE 1: Return charging environment. (a) Map 1. (b) Map 2. (c) Map 3.

when the power is limited. Therefore, modeling the environment area has become the basic premise of the autonomy charging path planning of the cleaning robot. An appropriate environment model will help the robot to understand and recognize the environment and reduce the computational complexity of path planning.

3.1. Typical Environmental Modeling Methods. Environmental modeling is to transform the external environment of the robot into a form that can be recognized by the computer. It is usually divided into three typical methods: Cell Decomposition Approach, Visibility Graph, and Topological Method [23]. Cell Decomposition Approach [24] is divided into precise unit decomposition and approximate unit decomposition. The main idea is to decrease the working environment of the robot into regular base areas, and each area is usually a grid. The description of the grid is used to realize the environment modeling. The advantage is that the model is constructed according to the actual environment, so it is easy to expand and maintain. The description of the environment depends on the fineness of the grid. The finer the grid is, the more the robot can recognize the information in the environment; the

disadvantage is the calculation of path planning based on this method. The quantity mainly depends on the unit decomposition and connected graph search operation. When the object density is high, and the environment is complex, the calculation quantity increases exponentially.

Visibility Graph [25] abstracts the obstacles in the working environment as polygons and connects the mutually visible nodes of the starting point and the target point with the vertices of the obstacle polygon. The unobstructed path between these vertices is the shortest distance between them. The main advantage is that it is simple to implement and is suitable for working areas with simple environments and sparse obstacles, so the path planning in the two-dimensional space of robots is more common. There are two problems in path planning using visual views: one is that the number of nodes and obstacle edges will increase with the number of obstacle polygons; the other is that when used in a complex environment, invalid path efficiency may be significantly reduced. It is not suitable for high-dimensional space above three dimensions.

The Topological Method [26] is a method of dimensionality reduction, in which path planning transforms high-dimensional geometric space into a connectivity problem. The topological graph is composed of nodes and arcs. The

FIGURE 2: Environmental modeling of designated charging locations. (a) Map 1. (b) Map 2. (c) Map 3.

nodes represent a characteristic state or location in the environment. The arcs directly connected between the nodes are equivalent to the paths in the environment. The advantage of the topology method is that it simplifies the map's requirements for precise location, deletes the details of the map, and changes it to a more compact expression, so it reduces the modeling time and storage space and can quickly realize path planning. The disadvantages are also obvious. This method is only suitable for environments with space obstacles; otherwise, it is difficult to perform stable navigation control. On the other hand, when the number of obstacles increases, the environmental information in the topology map is difficult to modify, and the topology is complicated.

3.2. Environmental Modeling Based on Improved Maklink Graph. When the above three typical environment modeling methods are used in a complex environment with dense obstacles, they will cause a large amount of calculation and the robot to collide with obstacles. To solve this problem,

this paper proposes an improved Maklink graph modeling method, which transforms the complex environment into a simple graph theory algorithm to search path, effectively reduces the search, and provides a prerequisite for the next path planning.

The cleaning robot needs to return to the charging location as soon as the remaining power is limited. To solve this problem, this article studies two environments: one is how the robot chooses the shortest path to return to the charging place when a charging location is specified; the other is when there are multiple charging locations, how does the robot judge which is the closest place and return to the charging place with the shortest path?

Therefore, the establishment of a complex environment is shown in Figure 1, a square area with a side length of 30 meters, in which the black area is the wall or obstacle in the environment, and the white area is the area to be cleaned by the cleaning robot. To increase the complexity of the environment, a number of return-type "dead zones" are established in the figure to detect whether the robot can return to the charging place smoothly.

FIGURE 3: Environmental modeling of multiple charging locations. (a) Map 1. (b) Map 2. (c) Map 3.

3.2.1. Improved Maklink Graph Environment Modeling Method. The Maklink graph [27, 28] requires the following preconditions when modeling the working area of a cleaning robot:

- (1) The robot only performs motion planning in a two-dimensional plane environment, without considering the height of the robot and the environment;
- (2) Environmental boundaries and obstacles are approximated by convex polygons. The algorithm is based on accurate environment model and location.
- (3) The robot is approximated by “points,” which can appropriately expand the range of obstacles, so that the robot does not collide with obstacles when it moves.

The basic Maklink graph method divides the space to plan the free path that the robot does not collide with obstacles in the environment. It has the characteristics of simplicity and speed [29]. But there are two main disadvantages: one is that it is only suitable for convex polygonal

obstacles; the other is that when dealing with the environment model of complex obstacles, the planned link lines are cumbersome and nonintrusive [30].

Aiming at the above shortcomings, this paper proposes an improved Maklink graph modeling method, which is also suitable for concave polygons and complex obstacle environments, and reduces the number of link lines, thereby reducing the calculation and iteration of the path planning search space frequency. For concave polygons that can be split into two convex polygons that share vertices at the vertices, the purpose of taking the midpoint of the link line in the Maklink graph method is to plan a collision-free path with obstacles, so it is necessary to ensure that the adjacent midpoint is connected by a solid line. After connection, no matter how the robot moves at both ends of the dotted line at the midpoint, the connected solid line does not pass through the obstacle.

The specific steps of the improved Maklink graph method for environmental modeling are as follows:

- (1) Connect the vertices of the nearest obstacles in the environment with dotted lines one by one.

FIGURE 4: Preliminary path of charging location designated by Dijkstra algorithm. (a) Map 1. (b) Map 2. (c) Map 3.

- (2) Take the midpoint of the dotted line to connect with a solid line. When the two ends of the solid line move on the dotted line and cross the obstacle, increase the line from the obstacle to the vertical line of the boundary or the apex of the obstacle.
- (3) For a concave polygon, split into two convex polygons sharing the vertex at the vertex.

3.2.2. Environmental Modeling for One or More Charging Locations. When specifying a charging location, the improved Maklink map planning space is shown in Figure 2, and the midpoint sequence of the link line is identified as $p_0, p_1, p_2, \dots, p_n, p(n+1)$. When the robot has low power when $p_0 = S$ (S coordinate is (0,0)), it needs to return to $p(n+1) = T$ (T coordinate is (30,30)) for charging, and connect S and T to the nearest midpoint to form the robot's free movement path.

When there are multiple charging locations, the improved Maklink map planning space is shown in Figure 3, and the midpoint sequence of the link line is identified as $p_0, p_1, p_2, \dots, p_n, p(n+4)$. Suppose that the cleaning robot needs to be charged when it is located at points S_1, S_2, S_3 , and the available charging locations in the environment are located at the four corners of the map: $p(n+1) = T_1, p(n+2) = T_2,$

$p(n+3) = T_3, p(n+4) = T_4$. Connect S_1, S_2, S_3 and T_1, T_2, T_3, T_4 to the nearest midpoint to form a free movement path of the robot.

4. Cleaning Robot Charging Path Planning

After the improved Maklink graph environment modeling, the cleaning robot's charging path planning problem is transformed into the shortest path to solve the graph, and the path is two-dimensional path planning [31, 32]. To further reduce the solution space and calculation amount of the algorithm, this paper uses Dijkstra algorithm to simply plan the shortest path in the improved Maklink graph, which converts two-dimensional to one-dimensional path planning, but the resulting path is composed of points in the improved Maklink graph, not the shortest path. Therefore, this paper uses the improved fruit fly optimization algorithm to finely plan the path points, so that the cleaning robot returns to the charging point by the shortest path when the remaining power is limited.

4.1. Preliminary Planning Based on Dijkstra Algorithm. The Dijkstra algorithm was proposed by the Dutch mathematician E.W. Dijkstra in 1959. It is a typical shortest path algorithm for solving the shortest path of a directed graph

FIGURE 5: Dijkstra algorithm path when the starting point is in S1. (a) Map 1. (b) Map 2. (c) Map 3.

TABLE 1: Distance from S1, S2, and S3 to the four charging locations.

Map 1	T1 (0, 0)	T2 (0, 30)	T3 (30, 30)	T4 (30, 0)
S1 (12, 3.5)	76.2913 m	50.4909 m	69.9310 m	53.9134 m
S2 (23, 10)	92.2263 m	61.8426 m	80.2386 m	64.2238 m
S3 (6, 12)	41.6665 m	59.5753 m	96.7111 m	76.6636 m
Map 2	T1 (0, 0)	T2 (0, 30)	T3 (30, 30)	T4 (30, 0)
S1 (4.5, 15)	84.0328 m	74.8810 m	78.8113 m	87.6712 m
S2 (12.5, 2.5)	37.8648 m	55.7663 m	59.6965 m	68.5564 m
S3 (25.5, 20)	124.1846 m	109.0067 m	45.8574 m	107.9382 m
Map 3	T1 (0, 0)	T2 (0, 30)	T3 (30, 30)	T4 (30, 0)
S1 (13, 3)	73.3489 m	43.4753 m	40.0499 m	73.5339 m
S2 (23, 28.5)	72.6284 m	29.0777 m	21.3851 m	65.5631 m
S3 (22, 17)	132.6538 m	94.6245 m	130.8422 m	79.2115 m

[33]. Its main feature is the breadth-first preliminary traversal search algorithm, with the starting point at the center and being extended to the ending point. It can solve the shortest path of the node from the starting point to the ending point in the figure. The starting point S is specified

when calculating, and divides the points into two sets: H and U , H is to record the vertex of the calculated shortest path and the corresponding shortest path length. U is to record the vertex of the shortest path that is also solved and the distance from the vertex to the starting point S [34].

The specific steps of Dijkstra algorithm are as follows:

- (1) In initialization, specify the starting point S ; H only includes the starting point S , and U includes other vertices except for S
- (2) Select the vertex P with the shortest distance from U , add the vertex P to H , and remove the vertex P from U
- (3) At this time, update the distance of each vertex in U to the starting point S , and use P to update the distance of other vertices
- (4) Repeat steps (2) to (3) until all vertices are traversed

Dijkstra algorithm is used to generate a suboptimal path, which successively passes through the path nodes S , $P_1, P_2, \dots, P_d$ and T . The Maklink lines corresponding to

FIGURE 6: Dijkstra algorithm path when the starting point is in S2. (a) Map 1. (b) Map 2. (c) Map 3.

nodes are $L_i (i = 1, 2, \dots, d)$. Let A and B be the two ends of C , and the mathematical model of other points on the chain route is

$$P_i(h_i) = P_i^{(0)} + (P_i^{(1)} - P_i^{(0)}) * h_i, \quad h_i \in [0, 1], i = 1, 2, \dots, d, \quad (6)$$

where h_i is the proportional parameter and d is the Maklink's path node.

It can be seen from equation (6) that when the path obtained by the Dijkstra algorithm passes through the Maklink lines, as long as a set of parameters $(h_1, h_2, \dots, h_d)$ is given, a new path from the starting point to the end point can be obtained, and the solution of the fruit fly algorithm is expressed as $(h_1, h_2, \dots, h_d)$.

The preliminary path planning by Dijkstra algorithm of the designated charging location in Map 1 is $S \rightarrow p1 \rightarrow p2 \rightarrow p3 \rightarrow p4 \rightarrow p5 \rightarrow p6 \rightarrow p10 \rightarrow p11 \rightarrow p13 \rightarrow p19 \rightarrow p20 \rightarrow p30 \rightarrow p31 \rightarrow p32 \rightarrow p33 \rightarrow p34 \rightarrow$

$p35 \rightarrow p36 \rightarrow T$. In Map 2, it is $S \rightarrow p1 \rightarrow p2 \rightarrow p3 \rightarrow p4 \rightarrow p5 \rightarrow p6 \rightarrow p7 \rightarrow p10 \rightarrow p11 \rightarrow p12 \rightarrow p23 \rightarrow p37 \rightarrow T$. In Map 3 is: $S \rightarrow p1 \rightarrow p3 \rightarrow p6 \rightarrow p7 \rightarrow p8 \rightarrow p9 \rightarrow p10 \rightarrow p12 \rightarrow p26 \rightarrow p24 \rightarrow T$. As shown in Figure 4, the path experienced by the black dot is the preliminary path planned by the Dijkstra algorithm, with a total length of 116.2120 m in Map 1, 81.4383 m in Map 2, and 80.0383 m in Map 3.

The method of path planning for multiple charging locations is to use Dijkstra algorithm to first determine the distance from the starting point to the four charging locations and then select the shortest preliminary path based on the improved Maklink graph after sorting. The distance between the starting point and the four charging locations is shown in Table 1.

When the cleaning robot is in S1, the Dijkstra algorithm path of the four charging locations is shown in Figure 5. According to the relative distance, select the closest charging location T2 in Map 1: $S1 \rightarrow p18 \rightarrow p17 \rightarrow p16 \rightarrow p14 \rightarrow p12 \rightarrow p13 \rightarrow p37 \rightarrow p43 \rightarrow p42 \rightarrow p41 \rightarrow p40 \rightarrow p$

FIGURE 7: Dijkstra algorithm path when the starting point is in S3. (a) Map 1. (b) Map 2. (c) Map 3.

$39 \rightarrow T2$, the path length is 50.4909 m. Select the closest charging location $T2$ in Map 2: $S1 \rightarrow p17 \rightarrow p16 \rightarrow p15 \rightarrow p14 \rightarrow p13 \rightarrow p12 \rightarrow p18 \rightarrow p19 \rightarrow p20 \rightarrow p21 \rightarrow p22 \rightarrow T2$, the path length is 74.8810 m. Select the closest charging location $T3$ in Map 3: $S1 \rightarrow p11 \rightarrow p12 \rightarrow p20 \rightarrow p26 \rightarrow p24 \rightarrow T3$, the path length is 40.0499 m.

When the cleaning robot is in S2, the Dijkstra algorithm path of the four charging locations is shown in Figure 6. According to the relative distance, select the closest charging location $T2$ through the improved Maklink graph for preliminary path planning in Map 1: $S2 \rightarrow p26 \rightarrow p25 \rightarrow p24 \rightarrow p21 \rightarrow p20 \rightarrow p19 \rightarrow p37 \rightarrow p43 \rightarrow p42 \rightarrow p41 \rightarrow p40 \rightarrow p39 \rightarrow T2$, the path length is 61.8426 m. Select the closest charging location $T1$ in Map 2: $S2 \rightarrow p9 \rightarrow p8 \rightarrow p7 \rightarrow p6 \rightarrow p5 \rightarrow p4 \rightarrow p3 \rightarrow p2 \rightarrow p1 \rightarrow T1$, the path length is 37.8648 m. Select the closest charging location $T3$ in Map 3: $S2 \rightarrow p22 \rightarrow p26 \rightarrow p24 \rightarrow T3$, the path length is 21.3851 m.

When the cleaning robot is in S3, the Dijkstra algorithm path of the four charging locations is shown in Figure 7. According to the relative distance, select the closest charging location $T1$ through the improved Maklink graph for preliminary path planning in Map 1: $S3 \rightarrow p9 \rightarrow p8 \rightarrow p7 \rightarrow p6 \rightarrow p5 \rightarrow p4 \rightarrow p3 \rightarrow p2 \rightarrow p1 \rightarrow T1$, the path length is 41.6665 m. Select the closest charging location $T3$ in Map 2: $S3 \rightarrow p46 \rightarrow p45 \rightarrow p44 \rightarrow p43 \rightarrow p42 \rightarrow p41 \rightarrow p40 \rightarrow p39 \rightarrow p38 \rightarrow T3$, the path length is 45.8574 m. Select the closest charging location $T3$ in Map 2: $S3 \rightarrow p35 \rightarrow p34 \rightarrow p33 \rightarrow p32 \rightarrow p31 \rightarrow p30 \rightarrow p29 \rightarrow p28 \rightarrow p36 \rightarrow p37 \rightarrow p38 \rightarrow T4$, the path length is 79.2115 m.

4.2. Fine Planning Based on IFOA. The path planned by the Dijkstra algorithm is only a preliminary plan based on the improved Maklink graph. The path can be further optimized by changing the position of the dotted line at the midpoint of the passing. In order to solve this problem, this paper

FIGURE 11: Selection of initialization parameters.

FIGURE 12: Comparison of optimal paths on Map 1. (a) S-T. (b) S1-T2. (c) S2-T2. (d) S3-T1.

FIGURE 13: Comparison of evaluation indicators on Map 1. (a) The best path length comparison. (b) The worst path length comparison. (c) Average path length comparison. (d) Average computing time comparison. (e) Average iterative convergence time comparison. (f) Standard deviation comparison.

FIGURE 14: Comparison of optimal paths on Map 2. (a) S-T. (b) S1-T2. (c) S2-T1. (d) S3-T3.

where Len is a constant; gen is the current number of iterations; and $Maxgen$ is the maximum number of iterations. When $Len=1$ and $Maxgen=50$, the function image is shown in Figure 9.

It can be seen from the image that the larger search distance at the beginning of the iteration facilitates the movement of the optimal solution, and the smaller search distance in the later stage performs a fine search to improve the accuracy of the optimal solution.

The specific steps of the IFOA are as follows:

- (1) Initialize the parameters of the fruit fly optimization algorithm: population size: $popsiz$, the maximum number of iterations: $Maxgen$. Initialize the position of the fruit fly population on the path of Dijkstra algorithm: X_axis and Y_axis .

- (2) At the beginning of the fruit fly search, the individual fruit fly uses the sense of smell to find food:

$$\begin{aligned} X_p &= X_axis + Dist(2 * rand() - 1), \\ Y_p &= Y_axis + Dist(2 * rand() - 1), \end{aligned} \quad (8)$$

where $rand()$ is an arbitrary random number in the interval $[0,1]$, and $Dist$ indicates the distance of a single flight of the individual fruit fly as shown in equation (7).

- (3) Determine the taste concentration determination function of individual fruit fly. Since the fruit fly population needs to find the location of the food source, but the specific location of the food is unknown, first calculate the distance D from the origin

FIGURE 15: Continued.

FIGURE 15: Comparison of evaluation indicators on Map 2. (a) The best path length comparison. (b) The worst path length comparison. (c) Average path length comparison. (d) Average computing time comparison. (e) Average iterative convergence time comparison. (f) Standard deviation comparison.

(0,0) and calculate the reciprocal of D as a judgment function of taste concentration:

$$D(i, j) = \sqrt[3]{(X_p^2 + (Y_p^2))}, \quad (9)$$

$$S(i, j) = \frac{1}{D(i, j)}.$$

- (4) Using the taste concentration judgment function to determine the taste concentration value of individual fruit fly,

$$\text{Smell}_p = \text{Function}(S(i, j)). \quad (10)$$

- (5) According to the taste concentration value of each individual fruit fly, keep the individual with the highest taste concentration value:

$$[\text{bestsmell}, \text{bestindex}] = \min(\text{Smell}), \quad (11)$$

where bestsmell is the optimal taste concentration value in this iteration and bestindex is the position coordinates corresponding to the optimal taste concentration value in the iteration.

- (6) According to the position of the fruit fly with a strong taste and the best taste in step (5), the fruit fly colony moves to the best position by vision.

$$\begin{aligned} &\text{if } \text{bestsmell} < \text{SmellBest}, \\ &\quad X_axis = X(\text{bestindex}, :), \\ &\quad Y_axis = Y(\text{bestindex}, :), \\ &\quad \text{SmellBest} = \text{bestsmell}. \end{aligned} \quad (12)$$

- (7) Iterative optimization to the maximum number of iterations Maxgen , repeat steps (2) to (5), according

to whether the taste concentration value of the current iteration is greater than the optimal taste concentration value that has been retained; if yes, proceed to step (6); otherwise, proceed to the next iteration until the maximum number of iterations is reached, and output the optimal result.

IFOA, compared with ant colony algorithm, genetic algorithm, and particle swarm algorithm, is simple and easy to implement, has fast convergence, and can get the optimal solution quickly, but the disadvantage of the algorithm is that it is only suitable for low dimensional solution and may fall into local minimum.

The flowchart of the IFOA is shown in Figure 10.

5. Simulation

To select the optimal FOA initialization parameters, this paper tests the connection between iteration, popsize, time, and path length. Figure 11 is obtained by repeating the test 30 times from S to T on map 1. Through Figure 11, it can be seen that the distance gradually decreases with the increase of iteration and population, but the time increases exponentially. Taking into account the time cost, the optimal FOA parameters are set as $\text{popsize} = 400$, $\text{Maxgen} = 40$.

To evaluate the performance of path planning based on the IFOA, this paper conducts a comparison test of the basic FOA, the IFOA, Rapidly Exploring Random Tree (RRT), and ant colony algorithm (ACA) under the environment of designated one or more charging locations. The FOA and IFOA parameters are set as $\text{popsize} = 400$, $\text{Maxgen} = 40$, $\text{Len} = 0.5$, the ACA parameters are set as $\text{popsize} = 400$, $\text{Maxgen} = 40$, Pheromone calculation parameters: 2, Pheromone selection threshold: 0.8, Pheromone update

FIGURE 16: Comparison of optimal paths on Map 3. (a) S-T. (b) S1-T3. (c) S2-T3. (d) S3-T4.

parameters: 0.1, the RRT parameters are set as Step=2, MaxAttempts=1000, and the experiment is repeated 30 times on the same map with different starting and ending points.

The comparison of the optimal path and comparison of evaluation indicators on Map 1 is shown in Figures 12 and 13.

The comparison of the optimal path and comparison of evaluation indicators on Map 2 is shown in Figures 14 and 15.

The comparison of the optimal path and comparison of evaluation indicators on Map 3 is shown in Figures 16 and 17.

It can be observed from average path length, computing time and iterative convergence time in Figures 13 and 15 that IFOA is superior in the test environments. To check whether the algorithms are statistically different or not, the t -test is performed. The level 0.05 of significance is considered and shown in Table 2. The results indicate that IFOA

outperforms all other methods in all test environments, because the p values are smaller than the level 0.05 of significance.

The solid line is the IFOA, the dotted dashed line is the FOA, the dotted line is ACA, and the chain line is RRT.

From Figures 12 to 15, it can be seen that the path planned by IFOA is significantly better than FOA, and the smoothness of the path is also greatly improved. The path planned by IFOA is better than FOA planning. Although the path of ACA is short, the computing time is twice that of IFOA, and it converges after 40 times. Compared with RRT, IFOA has significant advantages in the path length and path smooth computing time. It can be seen from the eight sets of simulation experiments that, based on the improved fruit fly optimization algorithm (IFOA), the relative path has the fastest convergence speed, and the path obtained is better and smoother than FOA. In addition, it can be seen from the comparative test that the computing speed and convergence times of IFOA are better

FIGURE 17: Comparison of evaluation indicators on Map 3. (a) The best path length comparison. (b) The worst path length comparison. (c) Average path length comparison. (d) Average computing time comparison. (e) Average iterative convergence time comparison. (f) Standard deviation comparison.

TABLE 2: Comparison of t -test between IFOA, FOA, and ACA (significant difference is represented by “+”).

Map 1	IFOA	FOA	ACA	RRT
S-T	—	$7.50e-05(+)$	$3.51e-07(+)$	$3.51e-09(+)$
S1-T2	—	$1.14e-03(+)$	$1.04e-06(+)$	$5.18e-11(+)$
S2-T2	—	$1.68e-05(+)$	$8.17e-07(+)$	$6.93e-06(+)$
S3-T1	—	$7.90e-04(+)$	$1.43e-05(+)$	$1.12e-13(+)$
Map 2	IFOA	FOA	ACA	RRT
S-T	—	$3.68e-04(+)$	$4.56e-06(+)$	$9.42e-11(+)$
S1-T2	—	$6.12e-05(+)$	$2.92e-06(+)$	$1.19e-06(+)$
S2-T1	—	$8.74e-05(+)$	$5.56e-07(+)$	$7.96e-09(+)$
S3-T3	—	$4.89e-04(+)$	$8.87e-07(+)$	$6.33e-12(+)$
Map 3	IFOA	FOA	ACA	RRT
S-T	—	$6.17e-05(+)$	$5.32e-03(+)$	$4.50e-09(+)$
S1-T3	—	$1.87e-03(+)$	$1.62e-04(+)$	$1.93e-12(+)$
S2-T3	—	$3.68e-07(+)$	$4.68e-05(+)$	$7.53e-08(+)$
S3-T4	—	$5.41e-05(+)$	$3.43e-03(+)$	$5.69e-05(+)$

than those of ACA. The algorithm converges to the best in the number of iterations before 25 times. Compared with an enhanced genetic algorithm [37], the improved Maklink graph is more suitable for complex environment. Compared with the improved particle swarm optimization [38], the convergence times of IFOA are the third of it; however, the convergence speed of IFOA is too fast to find the optimal solution. In [39], the authors proposed an improved version of fruit fly optimization (FOA) to solve the path planning problem. It solves the problem that the convergence speed of FOA is too fast; however, the amount of calculation is too high.

It proves the feasibility of using the IFOA to plan the return charging path of the cleaning robot, which can make the path without collision with obstacles and the shortest. It is guaranteed that the robot should choose the shortest path to the charging location under the condition of limited power.

6. Conclusions

Through eight sets of comparative experiments and analysis, the effectiveness and practicability of the proposed IFOA method are proved. The cleaning robot can return to the charging location as soon as possible without colliding with obstacles in a complex industrial environment. Through comparative experiments, it can be seen that the original algorithm has been greatly improved. The path obtained is not the shortest path, but the calculation time is shorter than other algorithms.

There are also many possible avenues for future work. First, we will study how to plan a path for dynamic obstacle avoidance when the environment is unknown [40]. Second, we will combine IFOA with other algorithms to shorten the path length. Furthermore, we will consider the path planning problem when multiple cleaning robots charge [41].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partially supported by the Basic Scientific Research Business Cost Scientific Research Project of Heilongjiang Provincial University (135509114) and the Joint Guiding Project of Natural Science Foundation of Heilongjiang Province under Grant LH2019F038.

References

- [1] Y. Yuan, W. Su, and Z. Li, “Brain-computer interface-based stochastic navigation and control of a semiautonomous mobile robot in indoor environments,” *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 1, pp. 129–141, 2018.
- [2] R. Voges and B. Wagner, “Interval-based visual-LiDAR sensor fusion,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1304–1311, 2021.
- [3] H.-y. Zhang, W.-m. Lin, and A.-x. Chen, “Path planning for the mobile robot: a review,” *Symmetry*, vol. 10, no. 10, p. 450, 2018.
- [4] U. Orozco-Rosas, K. Picos, and O. Montiel, “Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots,” *IEEE Access*, vol. 7, pp. 156787–156803, 2019.
- [5] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, “Mobile robot path planning using membrane evolutionary artificial potential field,” *Applied Soft Computing*, vol. 77, pp. 236–251, 2019.
- [6] J. Fu, F. Tian, and T. Chai, “Motion tracking control design for a class of nonholonomic mobile robot systems,” *IEEE transactions on systems, man, and cybernetics: Systems*, vol. 50, no. 6, pp. 2150–2156, 2018.
- [7] D. A. Duque, F. A. Prieto, and J. G. Hoyos, “Trajectory generation for robotic assembly operations using learning by demonstration,” *Robotics and Computer-Integrated Manufacturing*, vol. 57, pp. 292–302, 2019.
- [8] H. Yoshitake, R. Kamoshida, and Y. Nagashima, “New automated guided vehicle system using real-time holonic scheduling for warehouse picking,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1045–1052, 2019.
- [9] K. Khateri, M. Pourgholi, M. Montazeri, and L. Sabattini, “A comparison between decentralized local and global methods for connectivity maintenance of multi-robot networks,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 633–640, 2019.
- [10] J. Troccaz, G. Dagnino, and G.-Z. Yang, “Frontiers of medical robotics: from concept to systems to clinical translation,” *Annual Review of Biomedical Engineering*, vol. 21, no. 1, pp. 193–218, 2019.
- [11] J. Tang, G. Chen, and J. P. Coon, “Secrecy performance analysis of wireless communications in the presence of UAV jammer and randomly located UAV eavesdroppers,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 11, pp. 3026–3041, 2019.
- [12] A. Sahoo, S. K. Dwivedy, and P. S. Robi, “Advancements in the field of autonomous underwater vehicle,” *Ocean Engineering*, vol. 181, pp. 145–160, 2019.

- [13] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural RRT*: learning-based optimal path planning," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020.
- [14] J. Liu, J. Yang, H. Liu, X. Tian, and M. Gao, "An improved ant colony algorithm for robot path planning," *Soft Computing*, vol. 21, no. 19, pp. 5829–5839, 2017.
- [15] F. H. Ajeil, I. K. Ibraheem, A. T. Azar, and A. J. Humaidi, "Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments," *Sensors*, vol. 20, no. 7, p. 1880, 2020.
- [16] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: a survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.
- [17] B. K. Patle, G. Babu L, A. Pandey, D. R. K. Parhi, and A. Jagadeesh, "A review: on path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019.
- [18] H. Gao, X. Song, L. Ding, K. Xia, N. Li, and Z. Deng, "Adaptive motion control of wheeled mobile robot with unknown slippage," *International Journal of Control*, vol. 87, no. 8, pp. 1513–1522, 2014.
- [19] T. Cabreira, L. Brisolara, and P. R. Ferreira Jr., "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, p. 4, 2019.
- [20] M. Radmanesh, M. Kumar, P. H. Guentert, and M. Sarim, "Overview of path-planning and obstacle avoidance algorithms for UAVs: a comparative study," *Unmanned Systems*, vol. 6, no. 2, pp. 95–118, 2018.
- [21] J. Delmerico, E. Mueggler, J. Nitsch, and D. Scaramuzza, "Active autonomous aerial exploration for ground robot path planning," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 664–671, 2017.
- [22] H. Zhang, Y. Wang, J. Zheng, and J. Yu, "Path planning of industrial robot based on improved RRT algorithm in complex environments," *IEEE Access*, vol. 6, pp. 53296–53306, 2018.
- [23] L. C. Santos, F. N. Santos, and E. J. S. Pires, "Path planning for ground robots in agriculture: a short review," in *Proceedings of the 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 61–66, IEEE, Ponta Delgada, Portugal, April 2020.
- [24] A. Khan, I. Noreen, and Z. Habib, "On complete coverage path planning algorithms for non-holonomic mobile robots: survey and challenges," *Journal of Information Science & Engineering*, vol. 33, no. 1, 2017.
- [25] Z. Li and W. A. Xu, "Path decision modelling for passengers in the urban rail transit hub under the guidance of traffic signs," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 1, pp. 365–372, 2019.
- [26] Y. Yang, M. Jiang, and W. Li, "2D path planning by lion swarm optimization," in *Proceedings of the 2020 4th International Conference on Robotics and Automation Sciences (ICRAS)*, pp. 117–121, IEEE, Chengdu, China, June 2020.
- [27] T. Lei, C. Luo, and J. E. Ball, "A graph-based ant-like approach to optimal path planning," in *Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–6, IEEE, Rio de Janeiro, Brazil, July 2020.
- [28] J. Guo, X. Huo, and S. Guo, "Path planning of the spherical robot based on improved ant colony algorithm," in *Proceedings of the 2020 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 273–278, IEEE, Beijing, China, August 2020.
- [29] G. Qing, Z. Zheng, and X. Yue, "Path-planning of automated guided vehicle based on improved Dijkstra algorithm," in *Proceedings of the 2017 29th Chinese Control and Decision Conference (CCDC)*, pp. 7138–7143, IEEE, Chongqing, China, May 2017.
- [30] M. Akram, A. Habib, and J. C. R. Alcantud, "An optimization study based on Dijkstra algorithm for a network with trapezoidal picture fuzzy numbers," *Neural Computing and Applications*, vol. 33, pp. 1–14, 2021.
- [31] A. Darvish and A. Ebrahimzadeh, "Improved fruit-fly optimization algorithm and its applications in antenna arrays synthesis," *IEEE Transactions on Antennas and Propagation*, vol. 66, no. 4, pp. 1756–1766, 2018.
- [32] R. Hu, S. Wen, Z. Zeng, and T. Huang, "A short-term power load forecasting model based on the generalized regression neural network with decreasing step fruit fly optimization algorithm," *Neurocomputing*, vol. 221, pp. 24–31, 2017.
- [33] S. Kanarachos, J. Griffin, and M. E. Fitzpatrick, "Efficient truss optimization using the contrast-based fruit fly optimization algorithm," *Computers & Structures*, vol. 182, pp. 137–148, 2017.
- [34] A. Babalik, H. İscan, and İ. Babaoğlu, "An improvement in fruit fly optimization algorithm by using sign parameters," *Soft Computing*, vol. 22, no. 22, pp. 7587–7603, 2018.
- [35] G. Wang, L. Ma, and J. Chen, "A bilevel improved fruit fly optimization algorithm for the nonlinear bilevel programming problem," *Knowledge-Based Systems*, vol. 138, pp. 113–123, 2017.
- [36] L. Wang, Y. Xiong, S. Li, and Y.-R. Zeng, "New fruit fly optimization algorithm with joint search strategies for function optimization problems," *Knowledge-Based Systems*, vol. 176, pp. 77–96, 2019.
- [37] M. Nazarahari, E. Khanmirza, and S. Doostie, "Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm," *Expert Systems with Applications*, vol. 115, pp. 106–120, 2019.
- [38] S. Shao, Y. Peng, C. He, and Y. Du, "Efficient path planning for UAV formation via comprehensively improved particle swarm optimization," *ISA Transactions*, vol. 97, pp. 415–430, 2020.
- [39] X. Zhang, X. Lu, S. Jia, and X. Li, "A novel phase angle-encoded fruit fly optimization algorithm with mutation adaptation mechanism applied to UAV path planning," *Applied Soft Computing*, vol. 70, pp. 371–388, 2018.
- [40] H. Lyu and Y. Yin, "COLREGS-constrained real-time path planning for autonomous ships using modified artificial potential fields," *Journal of Navigation*, vol. 72, no. 3, pp. 588–608, 2019.
- [41] X. Wang, H. Zhao, T. Han, H. Zhou, and C. Li, "A grey wolf optimizer using Gaussian estimation of distribution and its application in the multi-UAV multi-target urban tracking problem," *Applied Soft Computing*, vol. 78, pp. 240–260, 2019.