IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 19 February 2023, accepted 6 March 2023, date of publication 15 March 2023, date of current version 23 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3257570

RESEARCH ARTICLE

Robot Memorial Path Planning for Smart Access of Indoor Distributed Charging Piles

YANFANG DENG[1,2], TAIJUN LI[1,2], MINGSHAN XIE[3], AND WENHAN CHEN[3]
[1]School of Information and Communication Engineering, Hainan University, Haikou 570228, China
[2]Key Laboratory of Internet Information Retrieval of Hainan Province, Haikou 570228, China
[3]College of Big Data and Information Engineering, Guizhou University, Guiyang 550025, China

Corresponding author: Mingshan Xie (hnmingshanxie@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 62266010, in part by the Cultivation Project of Guizhou University under Grant [2019]57, and in part by the Research Project of Guizhou University for Talent Introduction under Grant [2019]31.

ABSTRACT Autonomous mobile service robots are used to complete many tasks, such as cleaning, transporting goods and monitoring. Such tasks usually require uninterrupted and continuous service. However, the battery of the robot is limited and must be charged frequently. For a large number of robots, it is essential to select a suitable charging pile. For this issue, we propose a path planning model for robots to intelligently access a limited number of charging piles distributed on the map. The traditional path planning model mainly considers the shortest path criterion to generate the path. Different from this, the path planning model in this paper not only considers the shortest path, but also the service position of the robot after charging, the remaining power of the robot, the state of the charging pile and the position of the robot in the map. Our path planning assigns the most suitable charging pile to the robot that needs charging. To solve the problem of high memory consumption and slow search speed when traditional A* algorithm is used for path planning, we propose local memorial path planning (LMPP) algorithm to quickly generate effective paths. The simulation results show that the proposed robot charging path planner can improve the robot service satisfaction and plan the effective path to the available charging piles.

INDEX TERMS Autonomous mobile robot, service robot, path planning, robot charging.

I. INTRODUCTION

Service robots usually work in an indoor environment with multiple charging piles, such as the handling robot [1] in the warehouse, the disinfection robot [2] in the hospital, the sales robot [3] in the shopping mall, and the patrol robot [4] in the airport. They use various path planning algorithms for navigation. The autonomous service robot can locate itself on the map and navigate to the charging pile for charging when the remaining power is lower than a certain threshold. However, charging piles are expensive, and the available charging piles are limited when a robot searches for charging piles. At the same time, the remaining power of the robot is limited. Under so many restrictions, how to improve the efficiency and robustness of the robot to find the charging pile has been a challenge. We propose a robot path planner to manage a limited number of available charging piles. The planner considers the remaining power of the robot and the service area of the robot after charging.

The traditional charging pile has no communication. The robot perceives the state of the charging pile through the visual sensor, so the robot needs to spend a large cost to obtain the state of the charging pile. This is a limitation, because inefficient planning may lead to waste of battery power and even service interruption. The charging piles in this paper are designed to communicate with each other, which is helpful for the robot to search for the globally optimal charging pile.

On the other hand, the traditional path planning mainly considers the shortest distance between the starting point and the target point for path planning [5], [6], [7], [8], [9], [10], [11], [12], [13], and completely ignores the remaining power of the robot and the service area after charging. This method

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

plans the most suitable charging pile according to the robot's position on the map, the remaining power and the service area after charging. The traditional charging path planning algorithm has a limit on the remaining power, but our algorithm has no limit on the power. There are corresponding path plans for different power levels. We give the simulation results of this algorithm in different scenarios and discuss the advantages of this algorithm compared with the traditional path planning algorithm.

At the same time, aiming at the problem of limited computing power and high delay requirement of path planning, we propose a local memorial path planning algorithm, which improves the speed of path planning and reduces the delay of path planning.

Our contributions are as follows:

1) We consider the shortest distance and the service frequency after charging, improve the service efficiency of the robot, and prevent the robot from planning to the area where it will not serve.

2) We propose a multistage charging model, which adapts to different energy levels, has a higher success rate, and improves the robustness of the robot in finding charging piles.

3) We propose the local memorial fast path planning, which improves the speed of robot path planning and reduces the delay of robot.

This paper is organized as follows. In Section II, the literature review is presented. Section III introduces preconditions and multistage selection model for charging pile selection. Section V presents the proposed LMPP algorithm to solve the path planning in the multistage charging. Section VI presents the simulation experiment and discussion about the proposed path planning model and LMPP algorithm. The conclusion is given in Section VI.

II. LITERATURE REVIEW

In the path planning of robot charging, the key problem to be considered is to build the objective function of path planning. Different application scenarios need to design the corresponding objective function. Gao et al. [14] regarded the unmanned aerial vehicle (UAV) delivery path planning as a multi-objective path planning problem, that is, to minimize the total cost while maximizing customer satisfaction, and proposed a new genetic method with constraints based on Non-dominant Sequencing Genetic Algorithm 2 (NSGA-II) to optimize the multi-UAV path planning problem. Flores Caballero et al. [15] regarded the total length of the path as a single objective function for UAV path planning in a continuous three-dimensional environment. Hou et al. [16] proposed a comprehensive framework for multi-objective electric vehicle path planning considering factors such as driving distance, total time consumption, energy consumption and charging cost, including various charging pricing strategies. The charging pricing strategy was designed based on the goal of maximizing the total revenue of the charging station and balancing the profit of the charging station. The method proposed in [16] was simulated on the street map of Shenzhen to verify the effectiveness of multi-objective charging path planning and the feasibility of charging pricing strategy. Ding et al. [17] aimed at the charging problem of electric vehicles as cleaning vehicles, combined with actual traffic information and map information, built a path network model based on time weight, queuing mechanism and charging calculation model. They adopted Dijkstra algorithm to solve the path planning problem of electric vehicle charging with the path network model, charging station queuing time and charging time as the parameters of the path planning objective function. Huang et al. [18] considered energy consumption and distance in the objective function of path planning and realized multi-objective path planning mainly through energy consumption estimation model (ECEM) and distance integrated estimation model (DIEM). For the path planning of connected and non-connected automated road vehicles on multi lane highways, Typalos et al. [19] regarded the objective function of minimization includes: the objective of vehicle propulsion, passenger comfort, and avoidance of collisions with other vehicles, of road departures and of negative speeds. Wang et al. [13] proposed an improved quantum particle swarm optimization algorithm for offline path planning of underwater vehicles, which aims to satisfy the three factors of path safety, path length and path point angle change, and constructs a fitness function to achieve multi-objective optimization. Yu et al. [8] established a multi-objective function aiming at the shortest path and the minimum number of cycles of automatic guided vehicle (AGV), proposed a parallel ant colony optimization algorithm for warehouse path planning, and obtained a path satisfying the objective function. Qiuyun et al. [9] studied the AGV path planning problem of the single production line in the intelligent manufacturing workshop for the materials required by the AGV transportation machine tool, established a mathematical model with the shortest transportation time as the objective function, and proposed an improved particle swarm optimization algorithm (IPSO) to solve the optimal path. Martinez et al. [10] proposed a genetic algorithm that can generate navigation waypoints, realize short distance and avoid collision with obstacles. This algorithm used multi-objective function to obtain waypoints, which considered the length of the path, the distance from the waypoints to the obstacles, and the probability of the final trajectory to cross an obstacle within a safe zone.

In conclusion, we can see that the objective functions of traditional path planning are designed as multi-objective functions. For the path planning of robot charging, due to the different charging limits of the robot, the idle time of each charging pile is inconsistent, resulting in the need to consider the objective function under different energy levels. The design of objective function under different energy levels is more and more urgent. We employed the issue and proposed a multistage objective function, which has strong adaptability.

The efficiency of robot charging path planning has been restricting the popularization of robots. When the power of the robot reaches the charging alarm threshold, the remaining power is not enough. If the efficiency of path planning is not high, the heavy calculation will consume most of the energy of the robot, which will make the robot unable to realize charging. Many scholars have been studying how to improve the efficiency of path planning. Ying et al. [11] improved the Rapidly-exploring Random Trees (RRT) algorithm when using the RRT algorithm for path planning of mobile robots, and adopted obstacle expansion to solve the limitation of long sampling time when the growing tree is close to the obstacle area. Aiming at the problem that the narrow channel robot is difficult to pass, the "bridge test" was introduced to guide the robot to walk. The simulation results showed that the improved RRT algorithm had shorter path, better time, and higher efficiency. Petavratzis et al. [20] applied a memory technology to solve the chaotic path planning problem, limiting the chaotic motion of the robot to the neighboring cells with the least number of visits, thereby comprehensively improving the performance. The numerical simulation results showed that the number of multiple visits to the previous cell can be significantly reduced. Zhao et al. [12] proposed an empirical memory Q-learning (EMQL) algorithm based on the continuous updating of the shortest distance from the current state node to the starting point in order to solve the problems of slow convergence and long planning path when the robot uses the Q-learning algorithm to plan the path in an unknown environment. The algorithm enhanced the autonomous learning ability of the robot through the different role assignment of the two tables. The EM table was a band (m * 1) dimension used to record distance information and reflect the learning process of the robot. Q-table was used as the assistant guidance of experience transfer strategy and experience reuse strategy, so that the robot can complete the task even when the target changes or the path was blocked. The comparison results in planning time, iteration times and path length showed that EMQL algorithm was superior to Q-learning algorithm in convergence speed and optimization ability. Zhang et al. [36] proposed a new multi-level humanoid model motion planning method based on the motion planning problem of mobile robots in complex indoor environment, which can ensure efficiency, smoothness, predictability and flexibility under motion and environmental constraints. Xie et al. [21] proposed a deep reinforcement learning method of three-dimensional path planning based on local information and relative distance for UAV path planning, constructed a recursive neural network with time memory to solve some observability problems by extracting key information from historical state action sequences, constructed two sample memory pools, and proposed an adaptive experience replay mechanism based on fault frequency. They had successfully planned a reasonable three-dimensional path with good obstacle avoidance in a large-scale complex environment. Bayat et al. [37] proposed an optimization to obtain the optimal and robust path planning solution for the environment with scattered obstacles based on the theory of charged particles' potential fields. They allocated potential functions for each individual obstacle, and obtained the best path by simultaneously achieving the trade-off between traversing the shortest path and avoiding collision. The scalar decision variable is used to make mathematical calculation very simple and the resulting path is non-oscillatory and collision-free.

For the current UAV path planning, the A* algorithm has a large memory overhead and a slow search speed in the planning process. Kong et al. [22] improved the A* algorithm. Firstly, combining an anytime repair search framework with a weighted A* algorithm to find a feasible path during the search process. Secondly, aiming at the disadvantage that the algorithm relies on low heuristic function weight, a double ranking criterion is proposed to improve the efficiency of the algorithm in approaching the optimal path in the iterative process. Thirdly, in order to reduce the expansion times of nodes in the planning process, the list storage constraint is improved. We find that many paths are used frequently in the path planning of indoor robots. Using this feature, we improve the A* algorithm and propose the local memorial path planning algorithm, which greatly improves the path planning efficiency of indoor service robots.

III. PROBLEM MODEL

Since the path planner of the robot considers the remaining power of the robot and the service area after charging in this paper, some parameters need to be configured, such as the probability of each robot near each charging pile. These parameters will be described below.

A. LOCALIZATION AND COMMUNICATION SETUP

Suppose all mobile robots have a map of indoor environment. It is also assumed that the robot can locate itself on the map and get the position of the charging pile. Map and localization of robot can be realized by using any SLAM (simultaneous localization and mapping) module that is already available in the literatures [23], [24], [25]. The navigation module is also considered available. The robot service area Ω in this paper uses grid map. The service region is decomposed into several non-overlapping subareas, such as rooms. The interface between the subareas is defined as the boundary-crossing, such as the room door.

The service area is provided with n charging piles, m subareas, r robots and u boundary-crossings. $\Omega = \bigcup_{i=1}^{m} \Omega_i$. The coordinates of the ith charging pile in the grid map are set as $s_i(x_{s_i}, y_{s_i})$. The set of charging pile positions is set as $S = \{s_1, s_2 \cdots s_n\}$. The coordinates of the ith boundary-crossing are set as $door_i(x_{door_i}, y_{door_i})$ in the grid map. The set of all boundary-crossings in the robot service area is set as $door_\Omega = \{door_1, door_2 \cdots door_u\}$.

Each charging pile is equipped with RFID electronic tag [26]. The charging pile can record the number of times that the robot passes by through RFID. The charging piles communicate with each other through Wi-Fi, ZigBee, and other modules. They can send broadcast instructions to the robot, so that the robot can update the state of the charging pile. The charging pile is equipped with a calculation module, which can sense and calculate the charging state of the robot.

B. CHARGING PILE STATE QUERY

The data stored by each robot includes the state Q of all charging piles, the frequency K^h of passing by each charging pile in each period, and the coordinates S of all charging piles.

The state of the charging piles is set as $Q = \{((q_{s_1}, q_{s_1}, \cdots, q_{s_1}), \psi)|0 \leq q_{s_i} \leq 1, q_{s_i} \in R\}$, where q_{s_i} represents the charging state value of the charging pile. $q_{s_i} = 0$ means that the charging pile s_i is idle and available. $q_{s_i} \neq 0$ indicates that there is a robot charging at the s_i charging pile. ψ is the update time stamp of the charging pile state.

If energy sensing is adopted for charging state of charging pile, energy sensing module shall be added. In order to control the cost of charging pile, the method of time allocation is adopted. When the robot is charging at the charging pile s_i, the calculation formula of the charging state value q_{s_i} is as formula (1).

$$q_{s_i} = \frac{t_{plan} - t_{fulfil}}{t_{plan}}, \tag{1}$$

where t_{fulfil} is the time that the charging pile has charged the robot. t_{plan} is the charging time allocated by the charging pile to the robot. The calculation of t_{plan} is as formula (2).

$$t_{plan} = \sum_{h=1}^{r} \frac{g^h_{full} - g^h_{real}}{g^h_{pertime}} + \eta, \tag{2}$$

where g^h_{full} is the full energy of battery of the hth robot, $g^h_{pertime}$ is the amount of electricity charged by the charging pile to the robot per unit-time, g^h_{real} means the remaining power, η is the adjustment factor.

The warning energy that the robot needs to be charged is set as g^h_{alert}. If $g^h_{real} \leq g^h_{alert}$, the robot sends a charging request to the charging pile. The current location of robot is set as $l^h_\sigma(x, y)$. The calculation formula for g^h_{alert} is calculated in formula (3).

$$g^h_{alert} = g^h_{perdis} \times d_\Omega + \delta, \tag{3}$$

where g^h_{perdis} is the energy consumed by the hth robot per unit distance, d_Ω is the diameter of the space serving the robot, δ is the adjustment factor.

The charging pile constantly monitors its charging state. The process of updating and sending charging state of charging pile is shown in Fig.1. When a robot is charging at the charging pile, the state value of the charging pile is calculated by formula (1). When no robot is charging at the charging pile or the robot leaves the charging pile after charging, the value of the charging state of the charging pile is 0. $q_{s_i}|_Q$ represents the charging state of the s_i charging pile recorded in set Q. When the value of charge state calculated by the monitoring charging module is equal to the value of charge state recorded in Q, that is, $q_{s_i}|_Q == q_{s_i}$, it indicates that the state of charge has not changed, and the charging pile continues to monitor its own state of charge. When $q_{s_i}|_Q \neq q_{s_i}$, it indicates that there is a change in the charging state, and the state value $q_{s_i}|_Q$ of the s_i charging pile in Q is updated to the calculated q_{s_i}, i.e., $q_{s_i}|_Q \leftarrow q_{s_i}$. $\hbar$ indicates the current time of the system. Update ψ of Q to the current time of the system, i.e $\psi|_Q \leftarrow \hbar$. After Q set is updated, the charging pile sends a copy of Q to other charging piles and robots through the wireless communication module.

The robot constantly listens all the charging states Q' sent by the charging pile. The flow of the robot receiving all the pile states is shown in Fig. 2.

Each charging pile also listens all the charging states Q' sent by other charging piles all the time. The flow of charging pile receiving all pile states is shown in Fig. 3.

The coordinates S of all charging piles stored by each robot are shown in Table 1.

TABLE 1. Charging pile specification.

Pile id.	X&Y Coord.	State Value
s_1	(x_{s_1}, y_{s_1})	q_{s_i}
s_2	(x_{s_2}, y_{s_2})	q_{s_2}
s_3	(x_{s_3}, y_{s_3})	q_{s_3}
$\vdots$	$\vdots$	$\vdots$
s_n	(x_{s_n}, y_{s_n})	q_{s_n}

C. FREQUENCY OF THE ROBOT'S ACTIVITIES NEAR THE CHARGING PILE AFTER CHARGING

Each charging pile is equipped with an RFID tag and an RFID reader is installed on the robot body. The frequency K of the robot passing by each charging pile in each time period is related to the setting of the time period and the electronic tag signal of the charging pile perceived by the robot.

The set of time periods is set as $T = \{T_1, T_2, \cdots T_m\}$. The time of a day is divided into some periods by interval λ which is the period interval time. For example, a day can be divided into time periods at intervals of 2 hours, where $\lambda = 2$. T is a set of cycle periods, as shown in Figure 4.

The robot samples once every τ time and accumulates the times of obtaining radio frequency signal from the charging pile s_i. The total times of passing the charging pile s_i in time period T_j is recorded as $k^t_{s_i,T_j}$, and the calculation formulation is as formula (4).

$$k^t_{s_i,T_j} \underset{t \in T_j}{=} k^{t'}_{s_i,T_j} + 1, \tag{4}$$

where $k^{t'}_{s_i,T_j}$ represents the total number of times that the robot passed the charging pile s_i last time. The quotient of t' and t divided by λ is T_j. The number of times the robot encounters the charging pile in the same period is accumulated. The frequency of the hth robot passing s_i in T_j period is set as

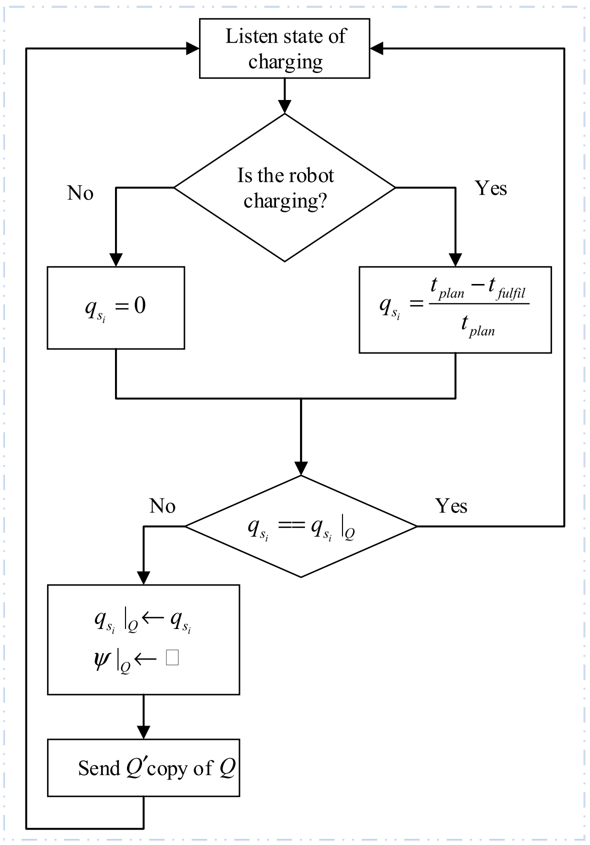


FIGURE 1. Flow diagram of updating and sending charging state of charging pile.

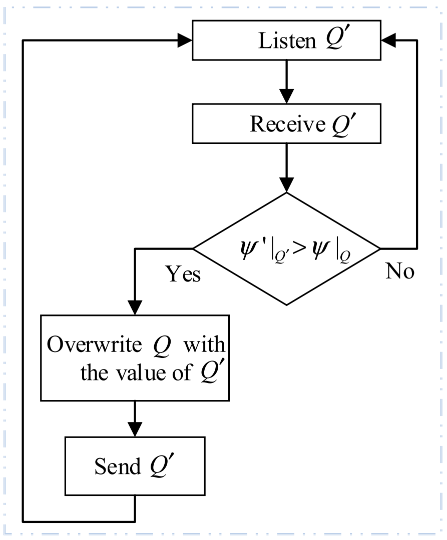


FIGURE 2. Flow diagram of robot receiving and updating all charging states.

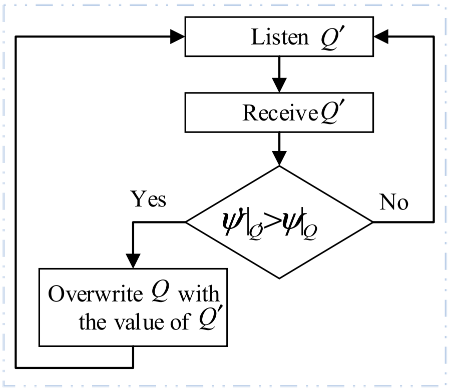


FIGURE 3. Flow diagram of charging pile receiving and updating all charging states.

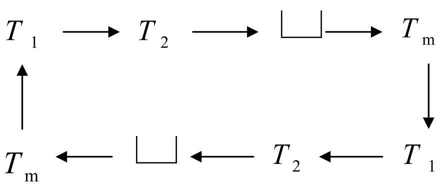


FIGURE 4. Cycle diagram of *T*.

$p^h_{s_i,T_j}$, and the calculation method is shown in formula (5).

$$p^h_{s_i,T_j} = \frac{k^t_{s_i,T_j}}{\sum_{i=1}^{n} k^t_{s_i,T_j}}. \tag{5}$$

The frequency K of all charging piles stored by r robots in each period is a third-order tensor, as shown in Fig. 5.

K is expressed as shown at the bottom of the page.

D. MULTISTAGE CHARGING MODEL

Selection of the charging pile and path planning to charging pile are completed by the mobile service robot, and each robot completes the two tasks independently. In the working process of robot, if the energy reaches the warning value g^h_{alert}, but the robot is performing a very important task, it must be charged after the task is completed. If the task is not performed or the task level is low, the charging calculation may be performed.

$$K = \begin{bmatrix} \begin{vmatrix} p^{h_1}_{s_1,T_1} & p^{h_1}_{s_2,T_1} & \cdots & p^{h_1}_{s_n,T_1} \\ p^{h_1}_{s_1,T_2} & p^{h_1}_{s_2,T_2} & \cdots & p^{h_1}_{s_n,T_2} \\ \vdots & \vdots & \ddots & \vdots \\ p^{h_1}_{s_1,T_m} & p^{h_1}_{s_2,T_m} & \cdots & p^{h_1}_{s_n,T_m} \end{vmatrix} \begin{matrix} p^{h_2}_{s_1,T_1} & p^{h_2}_{s_2,T_1} & \cdots & p^{h_2}_{s_n,T_1} \\ p^{h_2}_{s_1,T_2} & p^{h_2}_{s_2,T_2} & \cdots & p^{h_2}_{s_n,T_2} \\ \vdots & \vdots & \ddots & \vdots \\ p^{h_2}_{s_1,T_m} & p^{h_2}_{s_2,T_m} & \cdots & p^{h_2}_{s_n,T_m} \end{matrix} \Bigg| \cdots \Bigg| \begin{matrix} p^{h_r}_{s_1,T_1} & p^{h_r}_{s_2,T_1} & \cdots & p^{h_r}_{s_n,T_1} \\ p^{h_r}_{s_1,T_2} & p^{h_r}_{s_2,T_2} & \cdots & p^{h_r}_{s_n,T_2} \\ \vdots & \vdots & \ddots & \vdots \\ p^{h_r}_{s_1,T_m} & p^{h_r}_{s_2,T_m} & \cdots & p^{h_r}_{s_n,T_m} \end{matrix} \Bigg| \end{bmatrix}$$

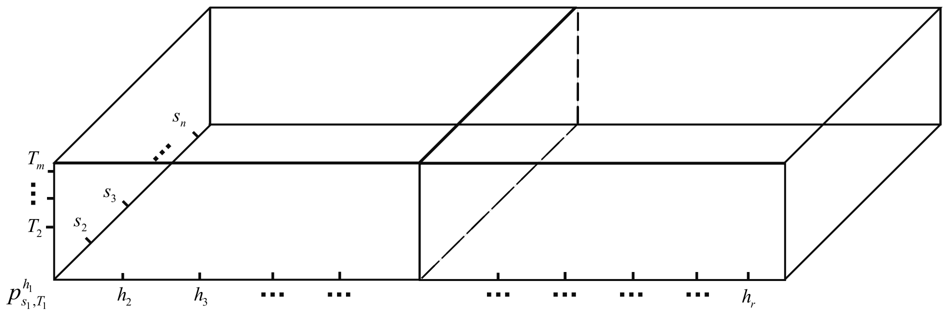


FIGURE 5. ***K*** **Model.**

Let the distance that the hth robot can travel with the remaining power be d_l^h, and the calculation method is as shown in formula (6).

$$d_l^h = d_{perpower}^h(m/J) \times g_{real}^h(J), \tag{6}$$

where $d_{perpower}^h$ represents the distance travelled per unit energy.

The robot uses radar, visual sensors [27], [28] and other sensing devices to locate its own location l_σ^h, and uses the state Q of all charging piles, the frequency set K of passing through each charging pile in each period, and the coordinates of all charging piles S to find a suitable charging pile.

In this paper, the symbol '$\cdot \rightarrow \cdot$' is used to represent the path, which is composed of many waypoints. The robot navigates according to the '$\cdot \rightarrow \cdot$' path. The path can be obtained by path planning algorithms such as particle swarm optimization [29] and A* algorithm [30]. After obtaining the path, the distance between each waypoint is summed to obtain the length $d^h_{l_\sigma \to s_i}$ of the path of the robot from the current location $l_\sigma^h(x, y)$ to the charging pile s_i. $d^h_{l_\sigma \to s_i}$ is normalized by formula (7).

$$d^{norm^h}_{l_\sigma \to s_i} = \frac{d^h_{l_\sigma \to s_i} - d^{\min^h}_{l_\sigma \to s_i}}{d^{\max^h}_{l_\sigma \to s_i} - d^{\min^h}_{l_\sigma \to s_i}}, \tag{7}$$

where $d^{\min^h}_{l_\sigma \to s_i}$ and $d^{\max^h}_{l_\sigma \to s_i}$ represent the maximum and minimum distance from the robot's current position to all charging stations respectively.

The robot first searches for the most suitable charging pile in the set Q_1 of idle and available charging piles, $Q_1 = \left\{s_i \left| q_{s_i} = 0\right.\right\}$. When the most suitable charging pile is not found in the Q_1 set, the search range is expanded to the set of charging piles Q_2 that the robot is almost fully charged. $Q_2 = \left\{s_i \left| q_{s_i} \geq q_{th}\right.\right\}$, where q_{th} is the threshold, $0 < q_{th} < 1$. The normalized path set from the robot to each charging pile in the Q_1 set is: $D_{Q_1} = \{d^{norm}_{l_\sigma \to s_i} | s_i \in Q_1\}$. The normalized path set from the robot to each charging pile in the Q_2 set is: $D_{Q_2} = \{d^{norm}_{l_\sigma \to s_i} | s_i \in Q_2\}$.

Some charging piles are deployed in areas where some robots have never been able to go. If the robot is planned to be charged at such a charging pile deployed in the area that is not its service scope, it will take a certain amount of electricity and time to return to its service area after the robot is charged. It is necessary to select the charging pile close to the location where the task is performed after the robot is charged. In this way we can improve the response speed of the robot, reduce redundant path planning, and improve the service satisfaction of the robot. The robot evaluates which period the charging time belongs to, and the calculation method is as shown in formula (8).

$$T_j^h = \left[\frac{t_0^h + t_{plan}}{\lambda}\right], \tag{8}$$

where t_0^h is the time when the robot sends a charging request, and the operator $[\cdot]$ indicates rounding.

The robot charging model in this paper comprehensively considers the distance from robot to charging pile and the nearest charging pile after the robot is charged. The multistage charging optimization model applicable to various remaining power is defined in formula (9), as shown at the bottom of the page, where $l_\sigma^h \rightarrow s_i^*$ represents the path of the robot from the current location to the most suitable charging pile. If the distance d_l^h can make the robot reach all the charging piles in the set Q_1, i.e. $d_l^h \geq \max(D_{Q_1})$, the

$$l_\sigma^h \rightarrow s_i^* = \begin{cases} \underset{s_i \in Q_1}{\operatorname{argmax}}\, p_{s_i,T_j}, \; d_l^h \geq \max(D_{Q_1}) \\ \underset{s_i \in Q_1}{\operatorname{argmax}}[(1 - \frac{1}{|Q_1|}) \times p_{s_i,T_j} + \frac{1}{|Q_1|} \times (1 - d^{norm}_{L \to s_i})], \; \min(D_{Q_1}) \leq d_l^h < \max(D_{Q_1}) \\ \underset{s_i \in Q_2}{\operatorname{argmax}}[\frac{1}{|Q_2|} \times p_{s_i,T_j} + (1 - \frac{1}{|Q_2|}) \times (1 - d^{norm}_{L \to s_i})], \; \min(D_{Q_2}) \leq d_l^h < \min(D_{Q_1}) \\ \emptyset, \; others, \end{cases} \tag{9}$$

robot gives priority to the charging piles that are close to the location where the task is performed after being fully charged.

If the distance d_l^h can make the robot reach the charging pile in the Q_1 set, that is, $\min(D_{Q_1}) \leq d_l^h < \max(D_{Q_1})$, the model selects the charging pile that is closer to the robot and has a higher frequency p_{s_i,T_j}. In formula (9), $|Q_1|$ is the number of all items in set Q_1. It is shown in formula (9) that the more the number of items, the greater the proportion of frequency p_{s_i,T_j} in path selection.

If the distance d_l^h cannot reach the charging pile in the set Q_1, the search range is expanded to select from the charging pile set Q_2. The charging piles in set Q_2 are the ones on which the robot is about to be full of electricity.

If $\min(D_{Q_2}) \leq d_L < \min(D_{Q_1})$ exists, the charging pile with short distance should be preferentially selected. The greater the number of charging piles in the set Q_2, the greater the proportion of the distance between the robot and the charging pile in the robot selection.

After extending the search range, if $\min(D_{Q_1}) \leq \min(D_{Q_2})$ appears, or if the charging pile is not selected in the set Q_2, the robot cannot search for the charging pile that can be charged. The path from the robot to the charging pile is represented by $\emptyset$, and the robot needs to wait or send a mobile charging request or a request to be towed for charging. If the battery of the robot is completely exhausted during the process, the robot will stop in the middle of the tunnel. In this case, we must manually or call the charging trailer to move the robot to the charging pile or conduct mobile charging process.

After selecting the appropriate charging pile through our model, how can we get the path from the robot to the charging pile? In this paper, a fast path planning algorithm is proposed to solve the path planning from the robot to the charging pile.

IV. LOCAL MEMORIAL PATH PLANNING

In order to solve the delay problem of path planning from the robot to the charging pile and realize real-time fast path planning, local memorial path planning (LMPP) algorithm is proposed according to the characteristics of multiple rooms in the indoor environment. The idea of this algorithm comes from the fact that people always walk on the familiar road without thinking when walking on the familiar road, because the path planning of the familiar road has been kept in mind. The flowchart of the LMPP algorithm is shown in Fig.6.

A. PRE-CONFIGURATION OF ALGORITHM

The preparation work of our algorithm is to build the local path memorial banks, plan some paths with certain laws in advance and store these paths in the path memorial banks. Since the location of the charging pile is fixed, the path from the boundary-crossing to the charging pile can be planned in advance.

Firstly, we must determine the subarea where the charging pile is located. The charging pile is a point in the map. Determining the subarea of the charging pile is transformed into determining which subarea a position X is in. In this

Algorithm 1 $\Theta_X = \text{Findsubreg}(\Omega, X)$

```
// Search the subarea $\Theta_X$ where position $X$ is located.
Initialize $\Theta_X$
For all $\Omega_j$ in $\Omega$
        If $X \in \Omega_j$ then
            Return $\Theta_X \leftarrow \Omega_j$
        Endif
Endfor
```

Algorithm 2 $door^{\Theta_X} = \text{Finddoor}(\Theta_X, door_\Omega)$

```
//search the boundary-crossing set $door^{\Theta_X}$ of the subarea $\Theta_X$.
Initialize $k$
For all $(x_i, y_i)$ in $\Theta_X$
    For all $door_j$ in $door_\Omega$
        If $(x_{door_j}, y_{door_j}) = (x_i, y_i)$ then
            $door_k^{\Theta_X} \leftarrow door_j$
            $k = k + 1$
            $door^{\Theta_X} \leftarrow door_k^{\Theta_X}$
        Endif
    Endfor
Endfor
```

paper, the robot can use Algorithm 1 to determine the subarea where the charging pile is located.

Next, the boundary-crossing set of the subarea where the charging pile is located is determined. In this paper, Algorithm 2 can be used to determine the boundary-crossing set of the subarea where the charging pile is located.

Then, we need to build a local memory path bank M_1 from the boundary-crossing to the charging pile. Since the location of the charging pile is fixed, the path from the charging pile to each boundary-crossing of the subarea can be used as a local memory. The path from the charging pile to the boundary-crossing of its subarea can be generated by using the traditional path planning algorithm, and it can be determined before the robot performs path planning. Set the path memorial bank from the charging pile to the boundary-crossing as $M_1 = \{door_k^{\Theta_{s_i}} \rightarrow s_i | i = 1, 2 \cdots n, k = 1, 2 \cdots u\}$, where $door_k^{\Theta_{s_i}} \rightarrow s_i$ indicates the path from the kth boundary-crossing of the subarea Θ_{s_i} where the charging pile s_i is located to the charging pile s_i. There may be many waypoints in the middle of this path. The algorithm for building M_1 is shown in Algorithm 3.

In Algorithm 3, Sizeof $(\cdot)$ is to get the number of items. Sizeof $\left(door^{\Theta_{s_i}}\right)$ indicates computing the number of items of the boundary-crossing set of the subarea Θ_{s_i}. The Function of Pathplan(,) is the path planning by using A* and other algorithms [31], [32], [33], [34] to generate a path between two positions.

Finally, we need to build a memory bank of paths between boundary-crossings. The boundary-crossing of each subarea usually does not change any more, and the paths between

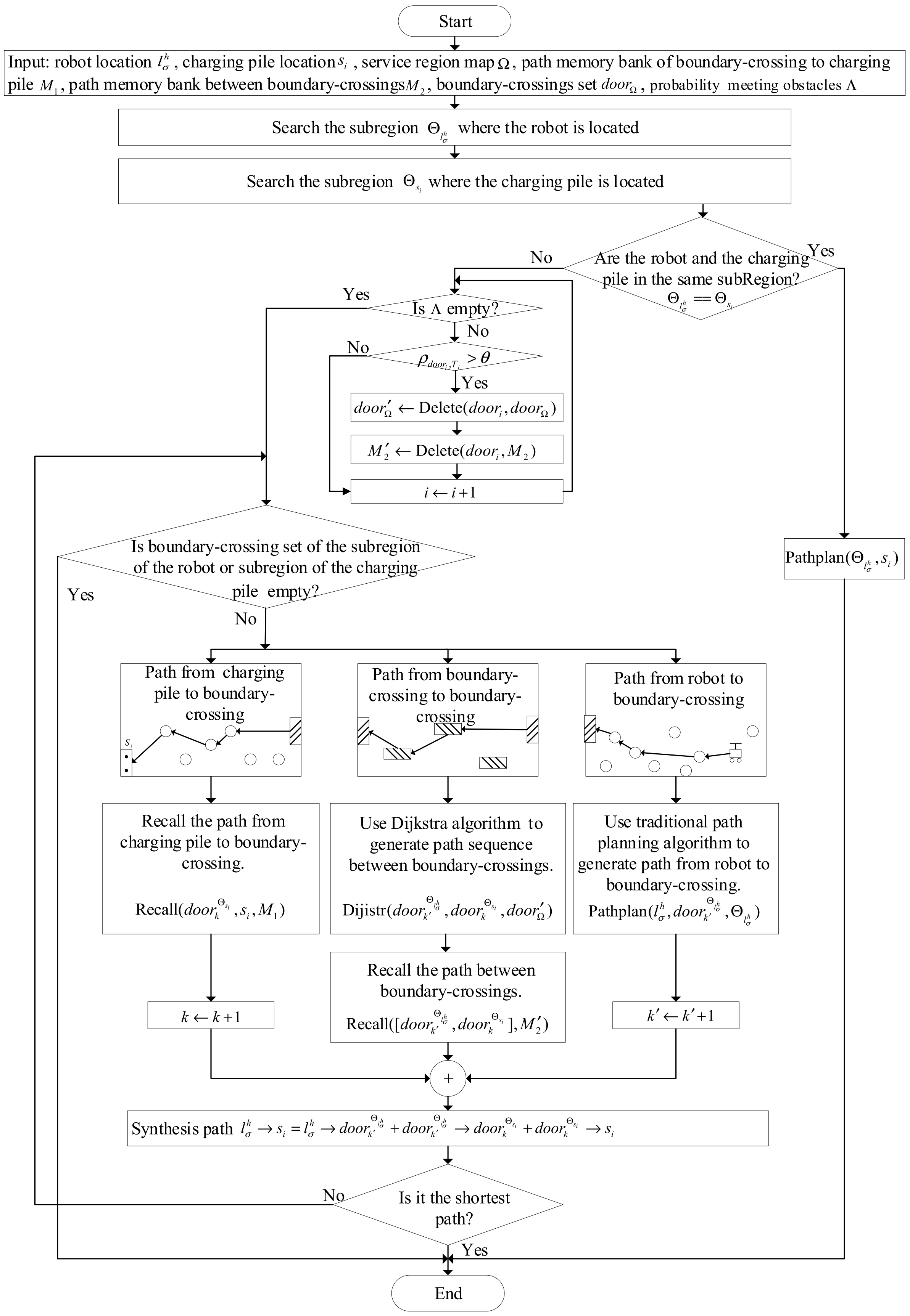

Start
Input: robot location l_σ^h, charging pile location s_i, service region map Ω, path memory bank of boundary-crossing to charging pile M_1, path memory bank between boundary-crossings M_2, boundary-crossings set $door_\Omega$, probability meeting obstacles Λ
Search the subregion $\Theta_{l_\sigma^h}$ where the robot is located
Search the subregion Θ_{s_i} where the charging pile is located
Are the robot and the charging pile in the same subRegion? $\Theta_{l_\sigma^h} == \Theta_{s_i}$
Yes
No
Is Λ empty?
Yes
No
$\rho_{door_i,T_i} > \theta$
No
Yes
$door'_\Omega \leftarrow \text{Delete}(door_i, door_\Omega)$
$M'_2 \leftarrow \text{Delete}(door_i, M_2)$
$i \leftarrow i+1$
Pathplan$(\Theta_{l_\sigma^h}, s_i)$
Is boundary-crossing set of the subregion of the robot or subregion of the charging pile empty?
Yes
No
Path from charging pile to boundary-crossing
s_i
Path from boundary-crossing to boundary-crossing
Path from robot to boundary-crossing
Recall the path from charging pile to boundary-crossing.
Recall$(door_k^{\Theta_{s_i}}, s_i, M_1)$
Use Dijkstra algorithm to generate path sequence between boundary-crossings.
Dijistr$(door_{k'}^{\Theta_{l_\sigma^h}}, door_k^{\Theta_{s_i}}, door'_\Omega)$
Use traditional path planning algorithm to generate path from robot to boundary-crossing.
Pathplan$(l_\sigma^h, door_{k'}^{\Theta_{l_\sigma^h}}, \Theta_{l_\sigma^h})$
Recall the path between boundary-crossings.
Recall$([door_{k'}^{\Theta_{l_\sigma^h}}, door_k^{\Theta_{s_i}}], M'_2)$
$k \leftarrow k+1$
$k' \leftarrow k'+1$
+
Synthesis path $l_\sigma^h \to s_i = l_\sigma^h \to door_{k'}^{\Theta_{l_\sigma^h}} + door_{k'}^{\Theta_{l_\sigma^h}} \to door_k^{\Theta_{s_i}} + door_k^{\Theta_{s_i}} \to s_i$
Is it the shortest path?
No
Yes
End

FIGURE 6. Flow chart of local memorial path planning.

Algorithm 3 Construction of M_1 Local Path Memory Bank From Charging Pile to Boundary

```
Input $\Omega, S$
  Output $M_1$
  Initialize $M_1$
  For all $s_i$ in $S$
    $\Theta_{s_i} = \text{Findsubreg}(\Omega, s_i(x_{s_i}, y_{s_i}))$// Find the subarea where the charge pile is located.
    $door^{\Theta_{s_i}} = \text{Finddoor}(\Theta_{s_i}, door_\Omega)$// Search the boundary-crossing set $door^{\Theta_{s_i}}$ of the subarea $\Theta_{s_i}$.
    For $k = 1 : \text{Sizeof}(door^{\Theta_{s_i}})$
      $door_k^{\Theta_{s_i}} \to s_i = \text{Pathplan}(s_i(x_{s_i}, y_{s_i}), door_k^{\Theta_{s_i}})$// Call the path planning algorithm to generate a path.
      $M_1 \leftarrow (door_k^{\Theta_{s_i}} \to s_i)$// Store the path into bank.
  endfor
endfor
```

the boundary-crossings can be calculated and stored in the boundary-crossing path bank. A reachable path can be connected between two boundary-crossings in the same subarea, and it can be solved by Pathplan(,). Assuming that the path of two boundary-crossings in the same subarea is $door_k^\Theta \to door_{k+j}^\Theta$, the path memorial bank between boundary-crossings is set as $M_2 = \{door_k^\Theta \to door_{k+j}^\Theta | k = 1, 2 \cdots u, j = 1, 2 \cdots u\}$. Adjacency linked list or adjacency matrix can be used to store M_2. The generation algorithm of M_2 is shown in Algorithm 4.

The recall algorithm for recalling the navigation path from the X position to the Y position in the memory bank M is shown in Algorithm 5, where $[X \cdots Y]$ represents a sequence of boundary-crossings between the two boundary-crossings.

Because the width of each boundary-crossing is often very small, the robot encounters obstacles at some boundary-crossing more frequently at some time. When the robot passes through these boundary-crossings, it is easy to be blocked, which will cause the robot to re-plan the path, thus wasting more time. A camera or infrared detection calculation module can be installed at the boundary to count the number of people or other objects passing through the boundary at each time period. We record the total number of times of passing pedestrians or other objects at the boundary in the period T_j as $\vartheta^t_{door_i,T_j}$, and the calculation method is shown in formula (10).

$$\vartheta^t_{door_i,T_j} \underset{t \in T_j}{=} \vartheta^{t'}_{door_i,T_j} + 1, \tag{10}$$

where $\vartheta^{t'}_{door_i,T_j}$ represents the total number of times that people or other objects passed through the boundary-crossing $door_i$ at the last moment. The quotient value of t' and t divided by λ is T_j. the accumulation of the number of people or objects passing through the boundary-crossing in the same period. The frequency of people or other objects passing through the boundary-crossing $door_i$ is set as ρ_{door_i,T_j} in T_j period, and the calculation method is shown in formula (11).

$$\rho_{door_i,T_j} = \frac{\vartheta^t_{door_i,T_j}}{\sum_{i=1}^{\mu} \vartheta^t_{door_i,T_j}}, \tag{11}$$

where $0 \leq \rho_{door_i,T_j} < 1$. Λ represents the set of probability of robot encountering obstacles at each boundary:

$$\Lambda = \begin{bmatrix} \rho_{door_1,T_1} & \rho_{door_2,T_1} & \cdots & \rho_{door_\mu,T_1} \\ \rho_{door_1,T_2} & \rho_{door_2,T_2} & \cdots & \rho_{door_\mu,T_2} \\ \vdots & \vdots & \vdots & \vdots \\ \rho_{door_1,T_m} & \rho_{door_2,T_m} & \cdots & \rho_{door_\mu,T_m} \end{bmatrix}.$$

Set the threshold value as θ. When ρ_{door_i,T_j} exceeds the threshold value θ, it indicates that the robot has the high probability of encountering obstacles when crossing the boundary, which will make the robot stay or be damaged. Before path planning, the robot receives Λ. When the range of θ is between (0, 1), the LMPP algorithm performs obstacle avoidance setting. Delete the boundary-crossing where ρ_{door_i,T_j} exceeds the threshold θ from $door_\Omega$, M_1 and M_2, so as to reduce the probability that the robot will encounter obstacles at the boundary-crossing, and thus increase the overall time for the robot to reach the charging station. When θ is set to more than 1, the LMPP algorithm cancel the obstacle avoidance.

B. MAIN PROCESS OF THE ALGORITHM

When the robot uses the LMPP algorithm proposed in this paper for path planning, the robot first determines the location l_σ^h using the locating algorithm, the location of the charging pile s_i, the map of the service region Ω, the path memory bank M_2 between the boundary-crossings, the local path memory bank M_1 from the charging pile to the boundary-crossing, the boundary-crossing set $door_\Omega$, and the probability meeting obstacles set Λ. These are inputs to the LMPP algorithm.

Firstly, Findsubreg(Ω, X) algorithm is used to search the subarea where the robot and the charging pile are located respectively.

If the robot and the charging pile are in the same subarea, the path from the current location l_σ^h of the robot to the charging pile s_i is calculated by Pathplan(,) whose function is to use A* or other path planning algorithms to generate a path between two positions.

If the robot and the charging pile are not in the same subarea, the path from the robot to the charging pile consists of three parts. The first part is the path from the robot to the boundary-crossing. The second part is the path from the boundary-crossing to the boundary-crossing, and the third part is the path from the boundary-crossing to the charging pile.

Part I: plan the path $l_\sigma^h \to door_{k'}^{\Theta_{l_\sigma^h}}$ from robot to boundary-crossing. Since the location of the robot l_σ^h is random, the traditional path planning algorithm is used to solve the path of the robot to the boundary-crossing.

Algorithm 4 Generation Algorithm for M_2

```
Input $\Omega, S$
Output $M_2$
Initialize $M_2$
  For all $\Omega_j$ in $\Omega$// Search all the subareas.
      If Sizeof($door^{\Omega_j}$) $\geq 2$ then // If there are more than two boundary-crossings in a subarea, there will be a path connecting tow boundary-crossings. Then, plan the path between the boundary-crossings in the subarea.
        For $k = 1$: Sizeof($door^{\Omega_j}$)-1 // Find a path between all the boundary-crossings of a subarea.
          For $k' = k+1$: Sizeof($door^{\Omega_j}$)
            $door_k^{\Omega_j} \to door_{k'}^{\Omega_j} =$ Pathplan($door_k^{\Omega_j}, door_{k'}^{\Omega_j}$)
            //Call the path planning algorithm to generate a path.
            $M_2 \leftarrow (door_k^{\Omega_j} \to door_{k'}^{\Omega_j})$// Save the path into the bank.
          Endfor
        Endfor
      Endif
    Endfor
```

Part II: plan the path $door_{k'}^{\Theta_{l_\sigma^h}} \to door_k^{\Theta_{s_i}}$ between the boundary-crossings, that is, the path between the boundary-crossing of the area where the robot is located and the boundary-crossing of the area where the charging pile is located. The path between the boundary-crossings can form a graph. We can use Dijkstra algorithm to obtain the boundary-crossing sequence of the shortest path between the two boundary-crossings, and then recall the path from the path memory bank M_2 by using Algorithm 5.

Part III: plan path $door_k^{\Theta_{s_i}} \to s_i$ from the boundary-crossing to the charging pile, that is, the path from the boundary-crossing of the subarea where the charging pile is located to the charging pile. To solve the path from the charging pile to the boundary-crossing, the method of recalling the path from the memory bank M_1 can be used. The path from $door_k^{\Theta_{s_i}}$ to s_i can be recalled from the memory bank by using Algorithm 5.

Algorithm 5 $X \to Y =$ Recall($[X \cdots Y], M$)

```
// Find the path from $X$ to $Y$ in $M$.
For i = 1: Sizeof($[X \cdots Y]$)-1
    For j = 1: Sizeof($M$)
        If $X|_M == X|_{[X \cdots Y]}$ and $Y|_M == Y|_{[X \cdots Y]}$ then
      $X \to Y + = X \to Y|_M$// The paths of several boundary-crossings are connected in pairs.
    Endif
  Endfor
Endfor
```

Finally, it is judged whether the boundary-crossing set of the subarea where the robot is located and the boundary-crossing set of the subarea where the charging pile is located are empty. If they are not empty, the recall step will be continued. If they are empty or the path has been solved, it will be returned.

The LMPP algorithm proposed in this paper is shown in algorithm 6.

TABLE 2. Coordinates of the door.

Door id.	Coord. X (unit-grid)	Coord. Y (unit-grid)
'1'	24	70
'2'	26	70
'3'	70	55
'4'	75	70
'5'	33	30
'6'	27	28
'7'	70	70

TABLE 3. Charging station specification for the experiment.

Charging pile id.	Coord. X (unit-grid)	Coord. Y (unit-grid)	State Value
s_1	0	100	0
s_2	50	70	0
s_3	70	100	0.9
s_4	100	100	0
s_5	0	70	0.7
s_6	100	70	0.8
s_7	0	50	0
s_8	40	50	0
s_9	100	29	0.6
s_{10}	0	0	0
s_{11}	20	0	0.3
s_{12}	100	0	0
s_{13}	70	20	0

In Algorithm 6, function of Dist( ) is to calculate the length of the path. Many paths will be planned for each iteration, and the path with the shortest distance will be selected.

Algorithm 6 LMPP Algorithm

```
Input: $\Omega, s_i, l_\sigma^h, door_\Omega, \Lambda, M_1, M_2$
Output: $l_\sigma^h \to s_i, d^h_{l_\sigma \to s_i}$// Generate the path from the current location of the robot to the charging pile.
$\Theta_{l_\sigma^h} \leftarrow$ Findsubreg$(\Omega, l_\sigma^h)$//Determine the subarea where the robot is located.
$\Theta_{s_i} \leftarrow$ Findsubreg$(\Omega, s_i)$//Determine the subarea where the charging pile is located.
        If $\Theta_{l_\sigma^h} == \Theta_{s_i}$ then// Judge whether the robot and the charging pile are in the same subarea.
        Pathplan$(l_\sigma^h, s_i)$// Traditional path planning algorithm is used for path planning.
            Else
            Initialize min
            $door^{\Theta_{l_\sigma^h}} \leftarrow$ Finddoor$(door_\Omega, l_\sigma^h)$//Calculate the boundary-crossing set of the subarea where the robot is located.
            $door^{\Theta_{s_i}} \leftarrow$ Finddoor$(door_\Omega, s_i)$//Calculate the boundary-crossing set of the subarea where the charging pile is located.
            for all $door_{k'}^{\Theta_{l_\sigma^h}}$ in $door^{\Theta_{l_\sigma^h}}$
              for all $door_k^{\Theta_{s_i}}$ in $door^{\Theta_{s_i}}$
              $l_\sigma^h \to door_{k'}^{\Theta_{l_\sigma^h}} \leftarrow$ Pathplan$(l_\sigma^h, door_{k'}^{\Theta_{l_\sigma^h}}, \Theta_{l_\sigma^h})$
              // Path from the current location of robot to boundary-crossing.
              For all $\rho_{door_i,T_j}$ in $\Lambda$// The function of obstacle-avoiding.
                If $\rho_{door_i,T_j} > \theta$ then
              $door'_\Omega =$ Delete$(door_i, door_\Omega)$//Remove the boundary-crossing exceeding the threshold from $door_\Omega$.
              $M'_2 =$ Delete$(door_i, M_2)$//Remove the boundary-crossing exceeding the threshold from $M_2$.
              Endif
            Endfor
            $[door_{k'}^{\Theta_{l_\sigma^h}} \cdots door_k^{\Theta_{s_i}}] \leftarrow$ Dijkstra $(door_{k'}^{\Theta_{l_\sigma^h}}, door_k^{\Theta_{s_i}}, door'_\Omega)$
            // Boundary-crossing sequence from boundary-crossing to boundary-crossing.
            $door_{k'}^{\Theta_{l_\sigma^h}} \to door_k^{\Theta_{s_i}} \leftarrow$ Recall$([door_{k'}^{\Theta_{l_\sigma^h}}, door_k^{\Theta_{s_i}}], M'_2)$
            // Path from boundary-crossing to boundary-crossing.
            $door_k^{\Theta_{s_i}} \to s_i \leftarrow$ Recall $(door_k^{\Theta_{s_i}}, s_i, M_1)$
            // Path from boundary-crossing to charging pile.
            $l_\sigma^h \to s'_i \leftarrow l_\sigma^h \to door_k^{\prime\Theta_{l_\sigma^h}} + door_{k'}^{\Theta_{l_\sigma^h}} \to door_k^{\Theta_{s_i}} + door_k^{\Theta_{s_i}} \to s_i$
          // Temporary path from boundary-crossing to charging pile.
          If min < Dist$(l_\sigma^h \to s'_i)$ then
              min $\leftarrow$ Dist$(l_\sigma^h \to s'_i)$
              $l_\sigma^h \to s_i \leftarrow l_\sigma^h \to s'_i$// Output the temporary path with the smallest distance.
          endif
        endfor
      endfor
      endif
      $d^h_{l_\sigma \to s_i} =$ Dist$(l_\sigma^h \to s_i)$ // Calculate the length of the shortest path.
      Return $l_\sigma^h \to s_i, d^h_{l_\sigma \to s_i}$
```

V. SIMULATION EXPERIMENT AND DISCUSSION

The simulation experiment was conducted in 100 ×100 grid environment, as shown in Fig. 7. The lower left corner is the coordinate origin. Black squares are obstacles, including 7 doors. 13 charging piles s_{1-13} are deployed. The number and coordinates of each door are shown in Table 2.

Suppose that the state set Q and the coordinates of all the charging piles are as shown in Table 3 at the time when a robot sends a charging request. The moving cost of the robot in the vertical direction and the diagonal direction is set as 1 unit-grid and $\sqrt{2}$ unit-grid respectively. Our computing device CPU is Intel ®CoreTM i5-9300H @ 2.40GHZ, memory is 8GB (2667 MHZ). In the experiment, the charging piles and the robot can communicate with each other. In the real-world scenario, such a configuration can be easily established using ROS [35] (robot operating system), which provides the communication between robots. We tested the model and algorithm in five different tests.

Test 1: The first test is to verify the running time of LMPP algorithm proposed in this paper. The proposed local memory path planning algorithm is compared with the traditional A* algorithm [33]. We randomly selected the ten charging

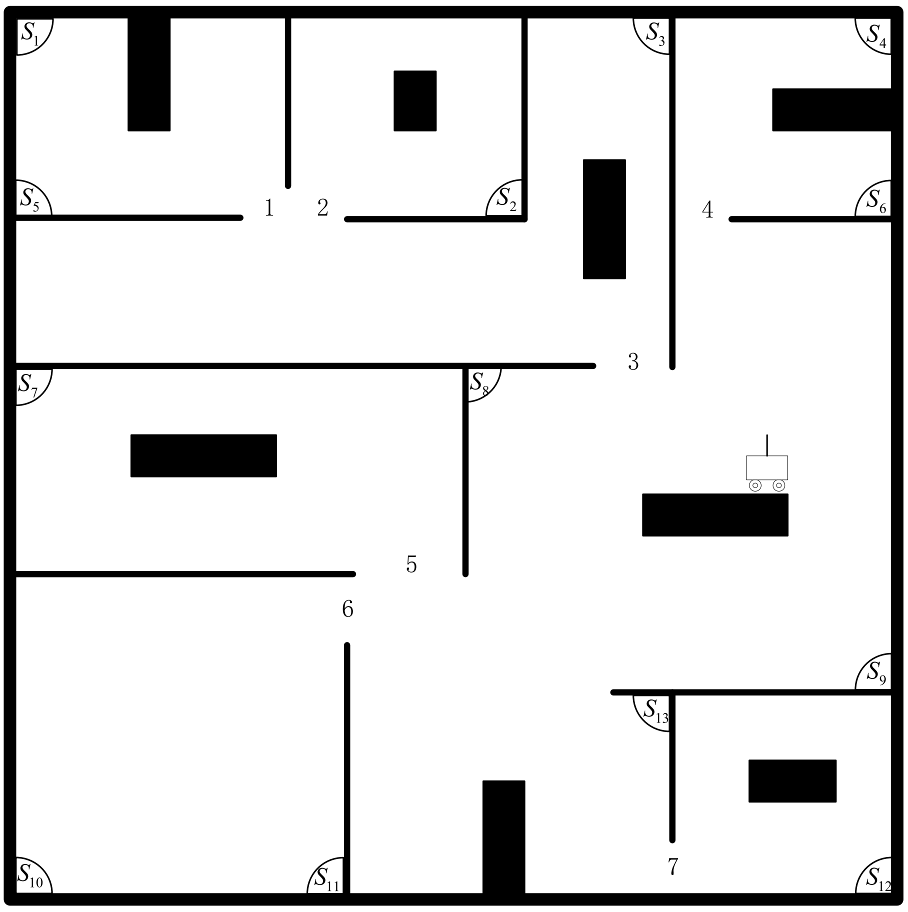


FIGURE 7. Experimental environment.

TABLE 4. The locations where robot sends recharging request.

Location No.	Coord. X(unit-grid)	Coord. Y(unit-grid)
1	92	13
2	10	64
3	55	28
4	97	96
5	98	16
6	49	96
7	15	81
8	92	43
9	96	80
10	4	66

TABLE 5. Time value comparison of robot path planning in each random locaition.

Location No.	LMPP	Traditional A*
1	60.9105s	17367s
2	828.6s	11770s
3	743.6s	7691s
4	953.7s	8569s
5	906s	13010s
6	182.5s	14990s
7	558.2s	12850s
8	593.2s	6503s
9	623.2s	10680s
10	981.5s	12040s

request positions, as shown in Table 4. These 10 locations can be used as the 10 possible locations for the robot to send out charging requests, or as the locations for the robot path planning when 10 robots send out recharging requests at the same time, so that the robot can select charging piles and perform path planning.

The time consumed by the traditional A* algorithm and LMPP algorithm is shown in Fig. 8. The time consumed by the robot at each random location is shown in Table 5. In the 10 randomly selected locations, the time variation of robot calculation is also relatively large by traditional A* algorithm. The path obtained by the traditional A* algorithm is long and short. However, the calculation time of LMPP algorithm is relatively slow, because our algorithm has stored some fixed

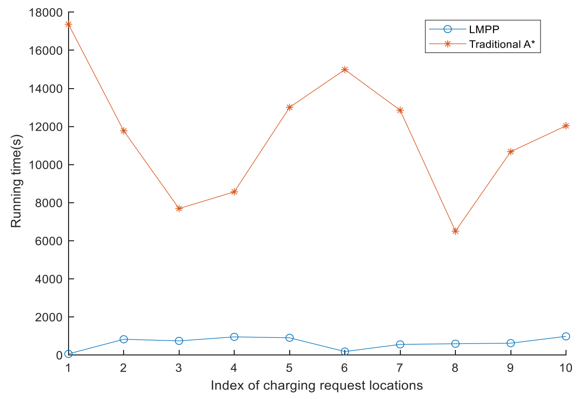


FIGURE 8. Comparison of planning time of algorithms.

TABLE 6. Length of the planned path at 10 random locations.

Location No.	Path length by traditional A* (unit-grid)	Path length by LMPP (unit-grid)
1	14.6011261594915	14.6011261594915
2	39.4721359549996	40.7082039324994
3	14.6011261594915	14.6011261594915
4	5.06449510224598	5.06449510224598
5	15.4721359549996	15.4721359549996
6	26.2360679774998	26.2360679774998
7	18.0219124314841	18.0219124314841
8	16.2448349897449	16.2448349897449
9	10.9442719099992	10.9442719099992
10	45	46.2360679774998

paths, and only a few unknown paths need to call the A* algorithm, so the overall time is basically the same.

As shown in Fig. 8 and Table 5, the time consumption of the LMPP algorithm proposed in this paper is obviously superior to that of the traditional algorithm.

The comparison of the paths planned by the two algorithms is shown in Fig. 9, and (1) - (10) is the location No. of the robot requesting recharging. the path is indicated by a green line. Fig. 9(1)a - (10)a are the paths planned by the traditional A* algorithm starting from 10 locations, and Figs. 9(1)b - (10)b are the paths planned by the LMPP algorithm proposed in this paper starting from 10 locations. The length of each path is shown in Table 6, whose unit is unit-grid.

It can be seen from Fig. 9 and Table 6 that all path lengths planned by the algorithm proposed in this paper are almost the same as that planned by the traditional algorithm. However, the speeds of the LMPP algorithm are much less than that of the traditional A* algorithm. LMPP has certain advantages and is applicable to the scenario of fast path planning.

Test 2: The purpose of this test is to verify the superiority of the multistage model proposed in this paper. The number of charging piles searched by the robot varies with the change of threshold, as shown in Table 7. There are 13 charging piles in this test. 8 of these charging piles are

TABLE 7. Specification of search scope of charging pile.

Threshold	Search Scope	
q_{th}	Q_1	$Q_1 + Q_2$
0.9	8	9
0.8	8	10
0.7	8	11
0.6	8	12
0.5	8	12
0.4	8	12
0.3	8	13
0.2	8	13
0.1	8	13

idle, i.e, $|Q_1| = 8$. The charging multistage model designed in this paper expands the search range of the optimal charging pile and increases the probability of the robot obtaining the optimal charging pile. As the set threshold q_{th} becomes smaller, more and more charging piles can be selected. When the threshold is 0.9, the model expands the search range to 9. When the threshold is 0.1, the model expands the search range to 13. The search range is getting larger and larger, and the probability of obtaining the optimal charging pile is getting higher and higher.

Test 3: The purpose of this test is to verify that when the robot starts from the same starting location and under different post charging activity probabilities $p^h_{s_i,T_j}$, whether the model and algorithm proposed in this paper can find a suitable charging pile. At the time when the robot sends recharging requests, the initial location of the robot planning path to find the charging pile is set as $l^h_\sigma(80, 50)$, and the robot plans a suitable path to the charging pile from this location. q_{th} is set as 0.1. $Q_1 = \{s_1, s_2, s_4, s_7, s_8, s_{10}, s_{12}, s_{13}\}$ $Q_2 = \{s_3, s_5, s_6, s_9, s_{11}\}$, $\max D(Q_1) = 103.5848$, $\min D(Q_1) = 38.88854$, $\max D(Q_2) = 88.67878$, and $\max D(Q_2) = 29.56512$ are obtained according to Table 3.

Six conditions were conducted in this test. Table 8 lists the probability of the robot moving near each charging pile. Table 8 can be regarded as the probability of six robots moving near each charging pile during their respective charging period. The condition serial number (1)-(6) is the corresponding the robot number. The serial number of the condition can be regarded as the serial number of the robot. Table 8 can also be regarded as the probability of a robot moving near each charging pile in six different charging periods, and the condition serial number corresponds to six different time periods.

Three values are set for the distance d^h_l that the remaining electric energy can travel, and the corresponding generated path is shown in Fig. 10.

In (1) - (6) conditions, when $d^h_l = 35$, then $\min(D_{Q_2}) \leq d^h_l < \min(D_{Q_1})$. The distance that the robot can travel is limited, the robot expands the search range, and the robot searches in Q_1 and Q_2 sets. Among all the charging piles, the distance between the robot and s_9 is the smallest, $s_9 \in$

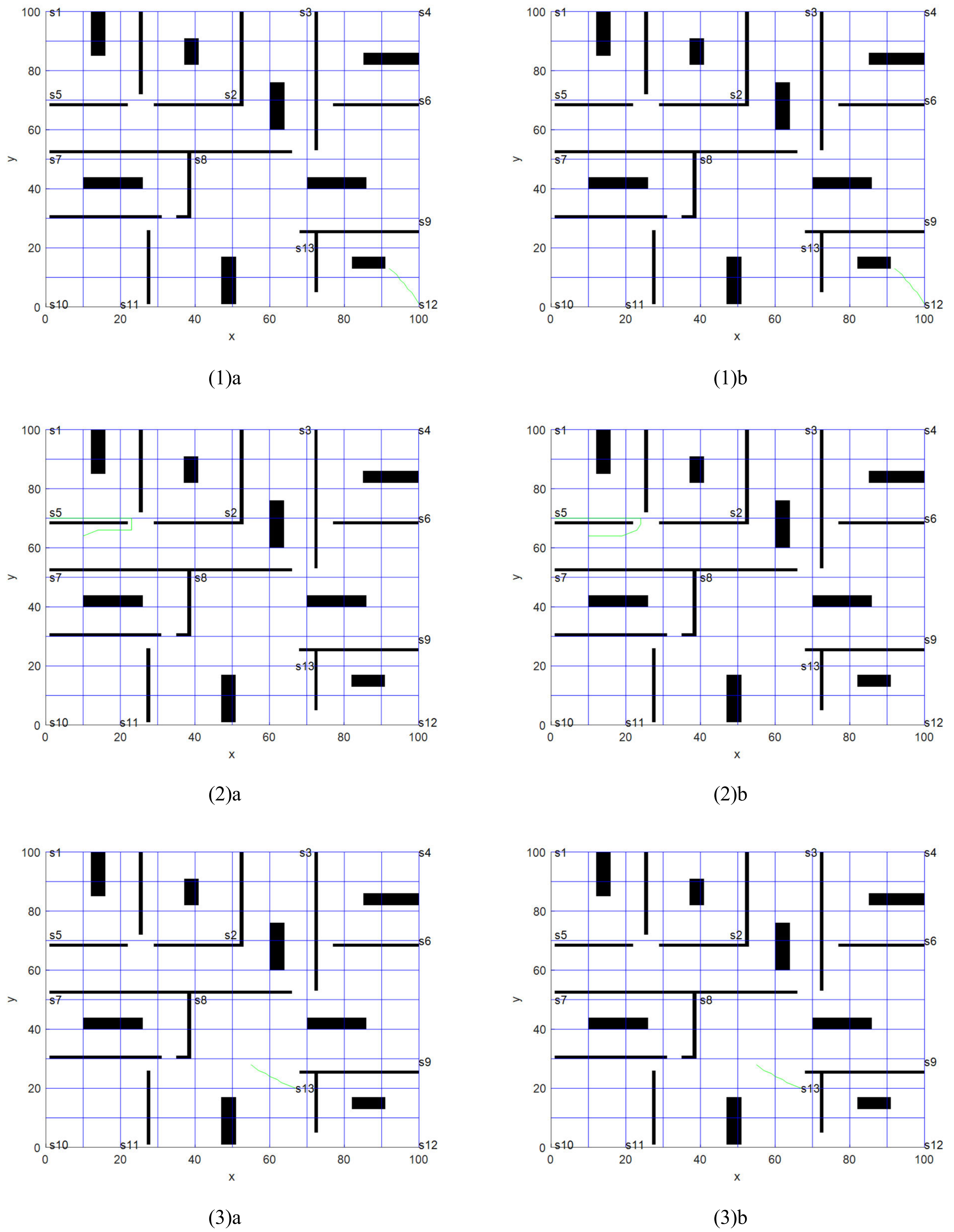
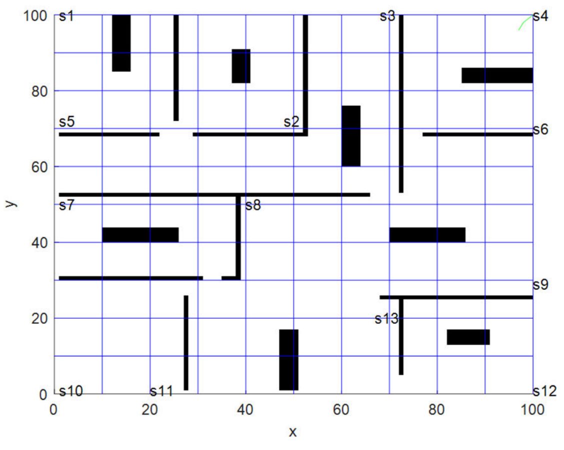
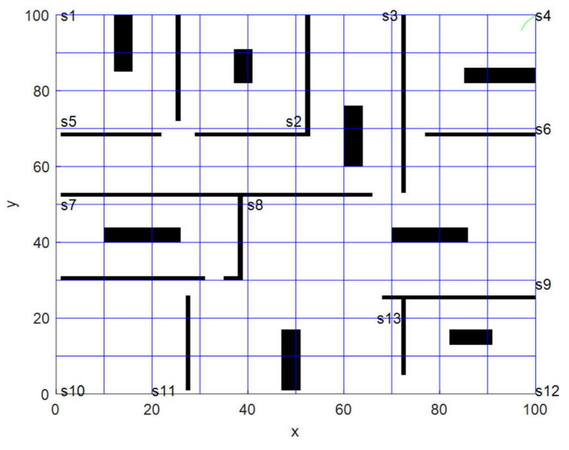


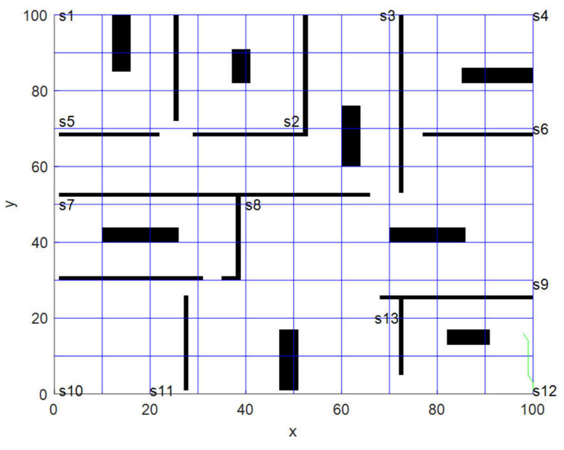
FIGURE 9. Comparison of planned paths at 10 random locations.



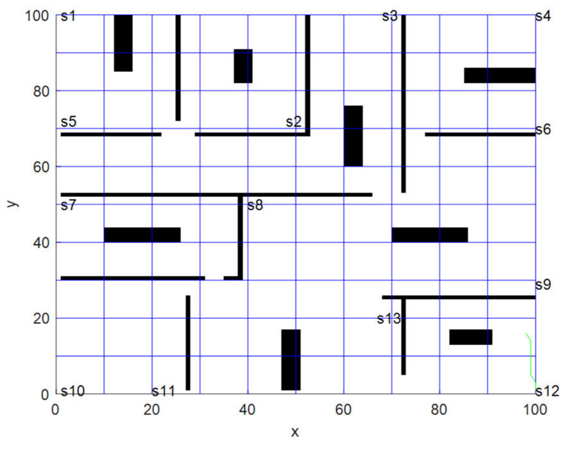
(4)a



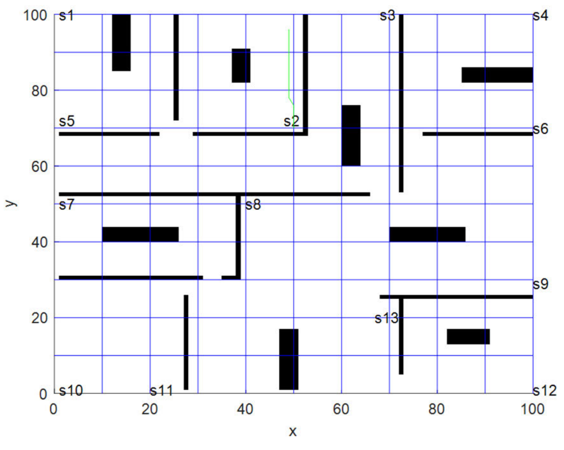
(4)b



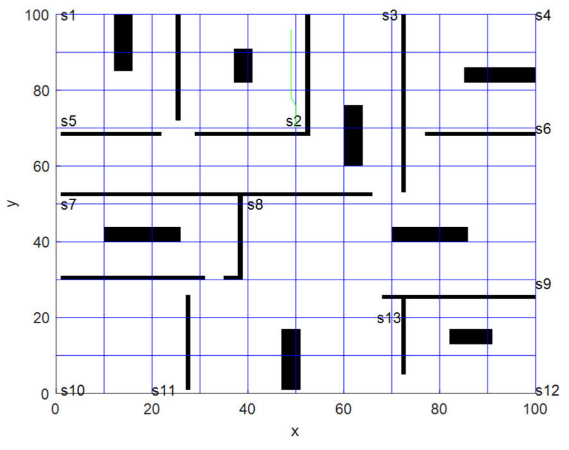
(5)a



(5)b

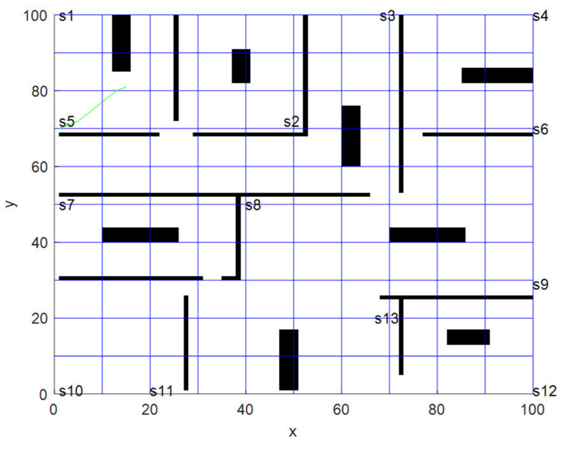


(6)a

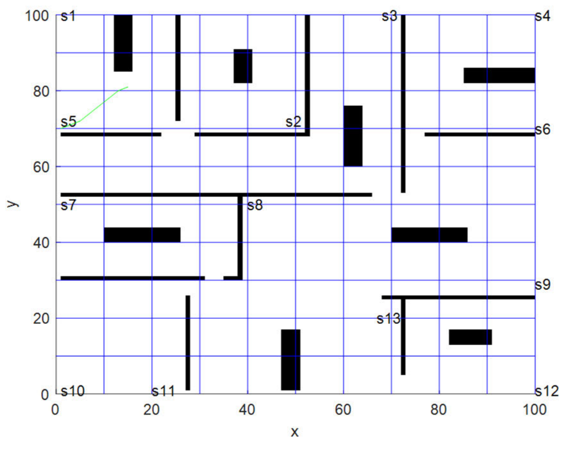


(6)b

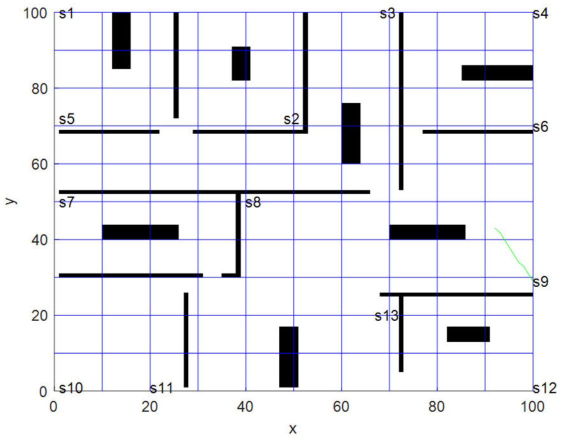
FIGURE 9. ***(Continued.)*** **Comparison of planned paths at 10 random locations.**



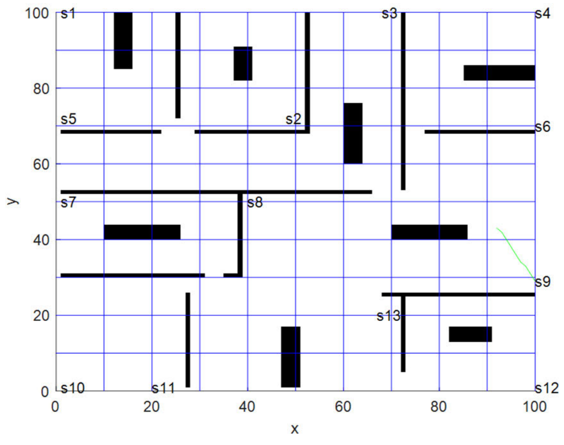
(7)a



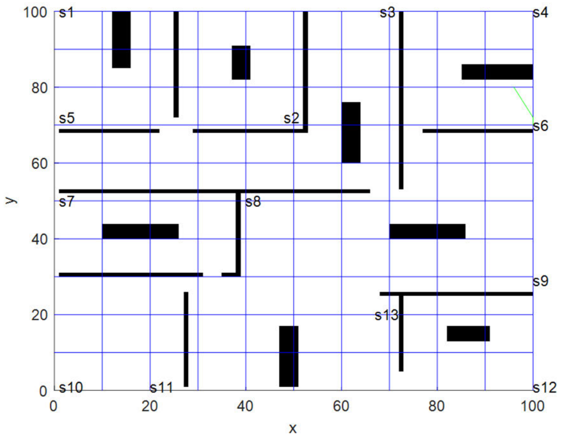
(7)b



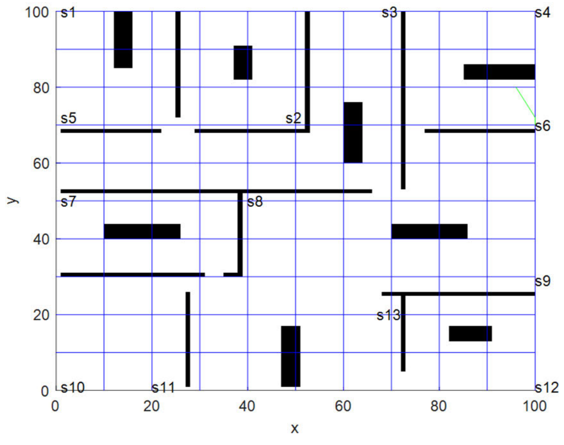
(8)a



(8)b

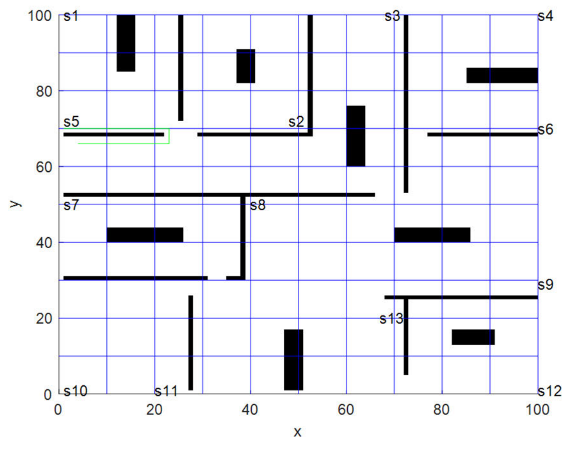


(9)a

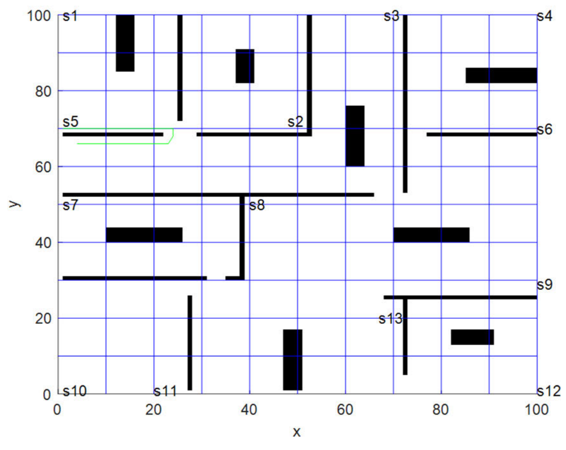


(9)b

FIGURE 9. ***(Continued.)*** **Comparison of planned paths at 10 random locations.**



(10)a



(10)b

FIGURE 9. ***(Continued.)*** **Comparison of planned paths at 10 random locations.**

TABLE 8. Probability of the robot moving near each charging pile.

Condition No.	p_{s_1}	p_{s_2}	p_{s_3}	p_{s_4}	p_{s_5}	p_{s_6}	p_{s_7}	p_{s_8}	p_{s_9}	$p_{s_{10}}$	$p_{s_{11}}$	$p_{s_{12}}$	$p_{s_{13}}$
(1)	0.003	0.01	0.12	0	0.001	0	0	0.04	0.13	0.09	0.11	0.376	0.12
(2)	0	0	0.11	0.08	0	0.001	0.075	0.06	0.004	0.34	0.27	0	0.06
(3)	0.10	0.10	0.10	0.10	0.10	0.10	0.00	0.10	0.10	0	0.10	0	0.10
(4)	0.080	0.026	0.113	0.015	0.071	0.120	0.016	0.036	0.057	0.145	0.028	0.143	0.150
(5)	0.124	0.094	0.107	0.039	0.118	0.113	0.003	0.059	0.106	0.094	0.068	0.005	0.07
(6)	0.066	0.087	0.073	0.001	0.141	0.069	0.021	0.053	0.064	0.054	0.185	0.007	0.179

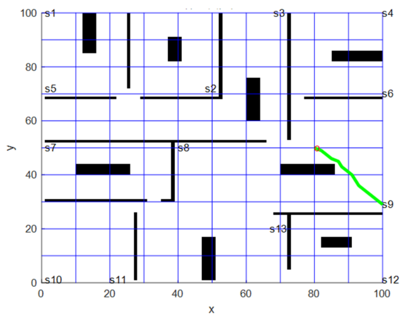
Q_2. At this time, the proportion of distance is too large. For 6 experiments, our algorithm selected the nearest s_9. When the distance that the robot can travel is very small, the robot expands the search range and preferentially selects the nearest charging pile.

When $d_l^h = 80$, then $\min(D_{Q_1}) \leq d_l^h < \max(D_{Q_1})$. The robot searches for charging piles in set Q_1. At this time, the proportion of probability is too large. In condition (1), the probability of s_{12} is the largest, which is 0.376, and $s_{12} \in Q_1$; In condition (2), the probability of s_{12} is the largest, which is 0.34, and $s_{10} \in Q_1$; In condition (3), The probability of all charging stations is the same, which is 0.10. Our algorithm selects the closer s_{13} because $s_{13} \in Q_1$, but not s_9 because $s_9 \notin Q_1$. Our algorithm selects the nearest charging pile under the same probability, which shows the rationality of our algorithm; In condition (4), our algorithm selects s_{13}, $s_{13} \in Q_1$. The probability of s_{13} is the largest among all charging piles and it can be seen from Table 8 that $p_{s_{13}} = 0.150$; In condition (5), our algorithm selects s_{13}, $s_{13} \in Q_1$. The probabilities of s_1, s_2 and s_{10} are greater than those of s_{13}. The distance from the starting point of the robot to s_1, s_2 and s_{10} is greater than that to s_{13}. Our algorithm selects s_{13} with large probability and small distance; In condition (6), Our algorithm selects s_{13}, $s_{13} \in Q_1$. Where, the probability of s_{11} is the largest, but $s_{11} \notin Q_1$, then our algorithm does not select s_{11}. In the Q_1 set, the probability of s_{13} is the largest.

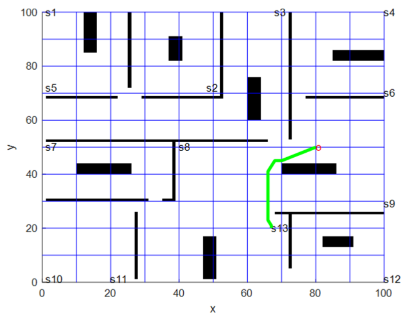
When $d_l^h = 120$, $d_l^h \geq \max(D_{Q_1})$, the robot can travel a long distance, and the charging pile with the highest activity probability after charging is selected from Q_1. s_{12}, s_{10}, s_{13}, s_1 and s_{11} are selected respectively in condition (1), (2), (4), (5), (6). It is known from Table 8 that $p_{s_{12}}$ is the largest in condition (1), $p_{s_{10}}$ is the largest in condition (2), P_{13} is the largest in condition (4), p_{s_1} is the largest in condition (5), $p_{s_{11}}$ is the largest in condition (6). In condition (3), the probability of charging piles is the same, and our algorithm selects the nearest s_9. It can be seen that our model and algorithm can plan the most suitable charging pile.

Test 4: The purpose of this test is to verify whether the robot can find a suitable charging pile using the model and algorithm proposed in this paper from different starting locations under the same activity probability after charging. Assume that the activity probability after charging is shown in Table 9.

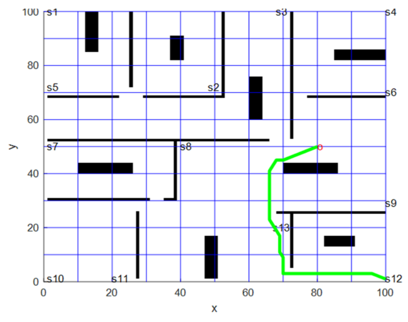
We conducted 6 times and set six starting locations, whose coordinates are shown in Table 10. The distance d_l^h that the remaining electricity can travel is assumed to be 50, 80 and 120 respectively, and the corresponding generated paths are shown in Fig. 11.



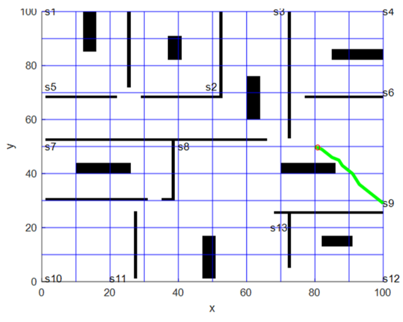
(1) $d_l^h = 35$



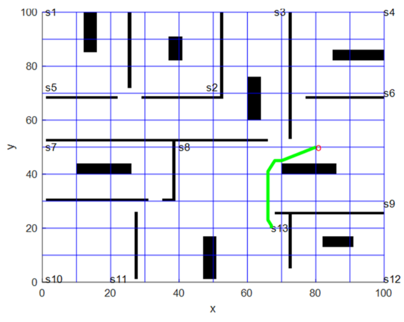
(1) $d_l^h = 80$



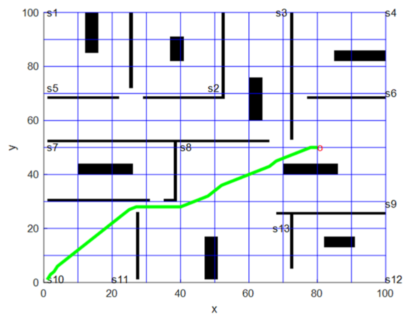
(1) $d_l^h = 120$



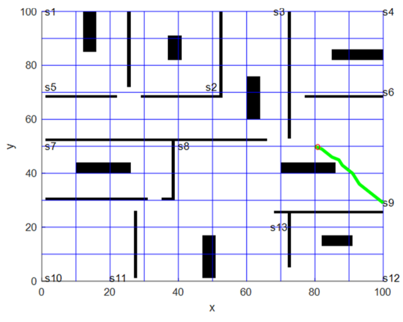
(2) $d_l^h = 35$



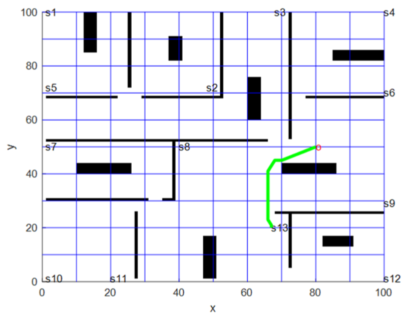
(2) $d_l^h = 80$



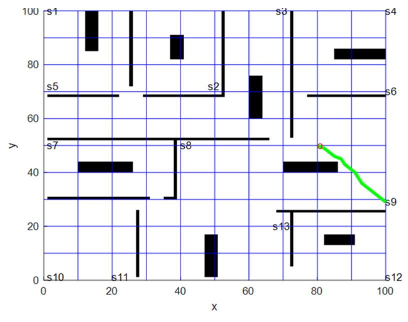
(2) $d_l^h = 120$



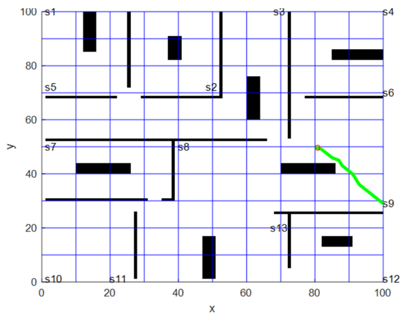
(3) $d_l^h = 35$



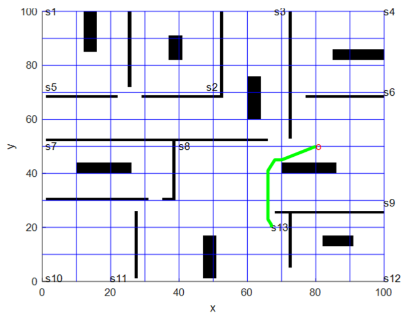
(3) $d_l^h = 80$



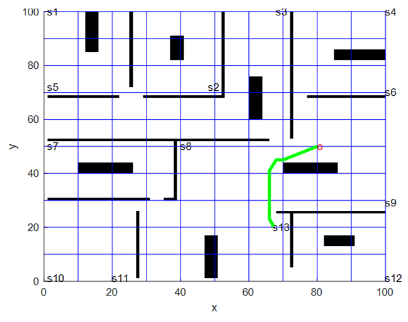
(3) $d_l^h = 120$



(4) $d_l^h = 35$

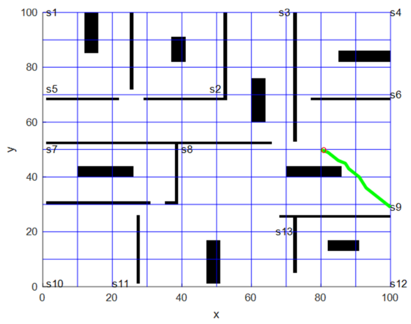


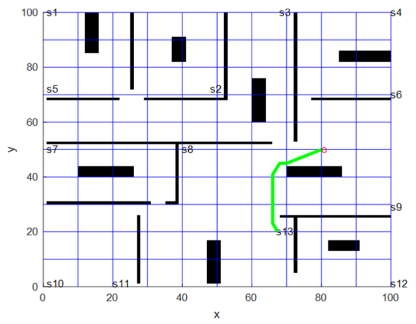
(4) $d_l^h = 80$

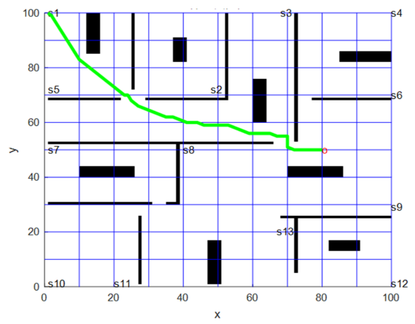


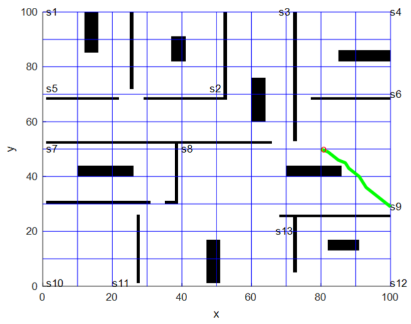
(4) $d_l^h = 120$

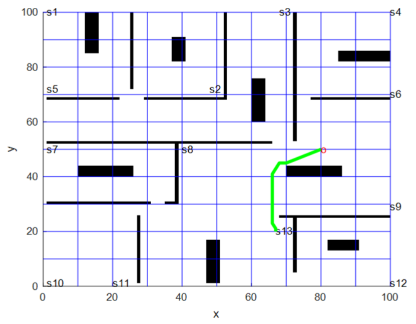
FIGURE 10. **Path comparison under the same starting point and different probabilities.**


(5) $d_l^h = 35$


(5) $d_l^h = 80$


(5) $d_l^h = 120$


(6) $d_l^h = 35$


(6) $d_l^h = 80$

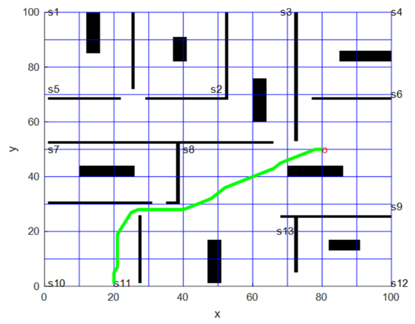

(6) $d_l^h = 120$

FIGURE 10. ***(Continued.)*** **Path comparison under the same starting point and different probabilities.**

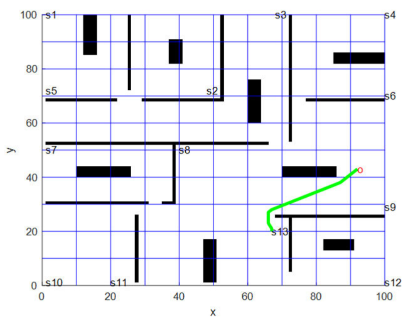
TABLE 9. **Probability of robot activity near each charging pile corresponding to different start location.**

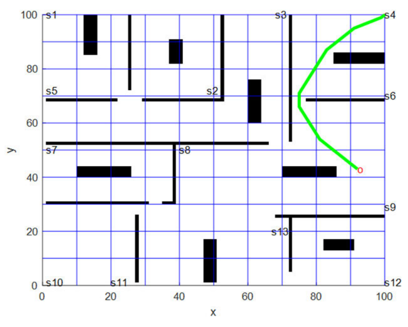
p_{s_1}	p_{s_2}	p_{s_3}	p_{s_4}	p_{s_5}	p_{s_6}	p_{s_7}	p_{s_8}	p_{s_9}	$p_{s_{10}}$	$p_{s_{11}}$	$p_{s_{12}}$	$p_{s_{13}}$
0.01	0.02	0.09	0.22	0.09	0.01	0.05	0.03	0.13	0.07	0.25	0.02	0.01

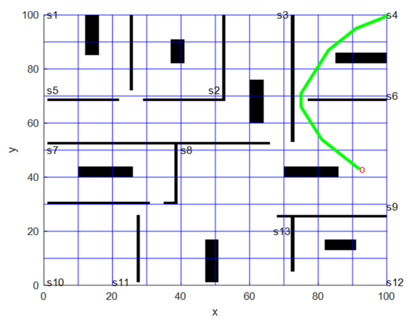
TABLE 10. **Coordinates of starting locations corresponding to different starting location.**

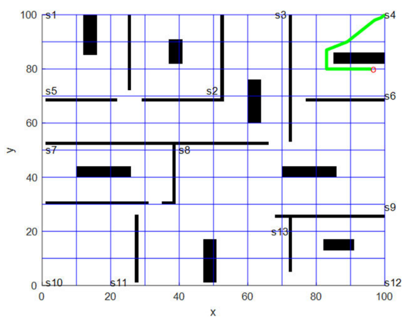
Starting location No.	Coord. X	Coord. Y	max $D(Q_1)$	min $D(Q_1)$	max $D(Q_2)$	min $D(Q_2)$
(1)	92	43	117.2373	38.08203	101.5498	16.24483
(2)	96	80	139.9455	41.67219	126.5805	10.94427
(3)	4	66	160.526	101.4831	152.1215	87.30495
(4)	40	75	156.9424	11.18034	148.5379	40.18034
(5)	18	66	146.526	87.4831	138.1215	73.30495
(6)	4	71	162.8265	29.7082	154.422	3.236068

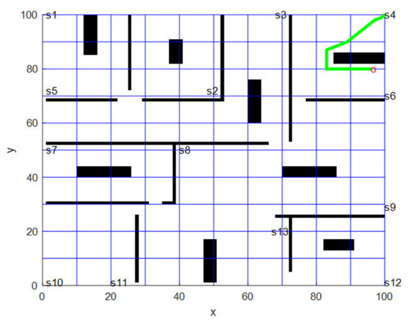
In test of starting location No. (1), the coordinates of the starting location are (92,43). When $d_l^h = 50$, then $\min(D_{Q_1}) \le d_l^h < \max(D_{Q_1})$, our model selects $s_{13}, s_{13} \in Q_1$; When $d_l^h = 80$, then $\min(D_{Q_1}) \le d_l^h < \max(D_{Q_1})$, our model selects $s_4, s_4 \in Q_1$; When $d_l^h = 120$, $d_l^h \ge \max(D_{Q_1})$, our model selects s_4, In Q_1, the probability of s_4 is the largest. In test of starting location No. (2), the coordinates of the starting location are (96,80). When $d_l^h = 50$, then $\min(D_{Q_1}) \le d_l^h < \max(D_{Q_1})$, our model selects $s_4, s_4 \in Q_1$; When $d_l^h = 80$, then $\min(D_{Q_1}) \le d_l^h < \max(D_{Q_1})$, our model selects $s_4, s_4 \in Q_1$; When $d_l^h = 120$, $\min(D_{Q_1}) \le d_l^h < \max(D_{Q_1})$, our model selects $s_4, s_4 \in Q_1$. s_4 has the largest probability after charging. Under the condition that the distance can be reached is small, our model selects s_4

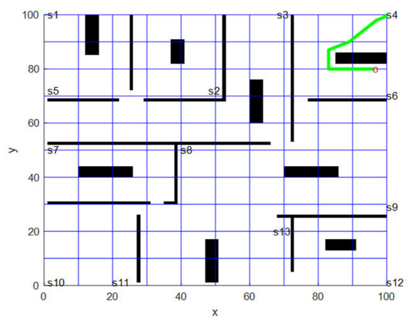

(1) $d_l^h = 50$

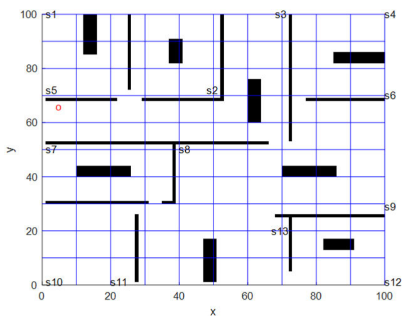

(1) $d_l^h = 80$

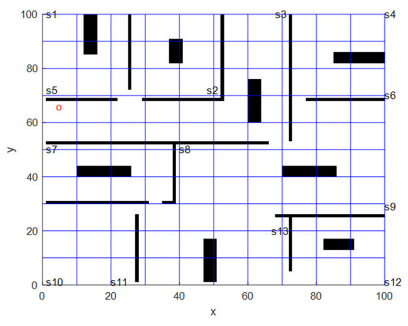

(1) $d_l^h = 120$

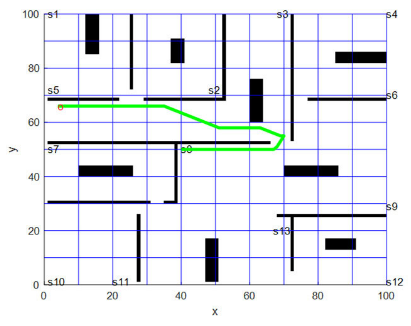

(2) $d_l^h = 50$

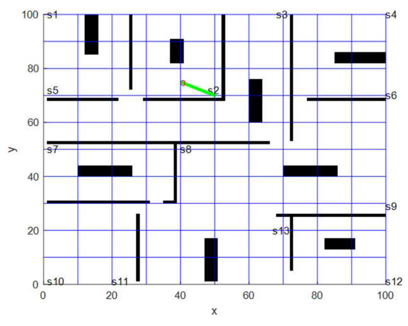

(2) $d_l^h = 80$

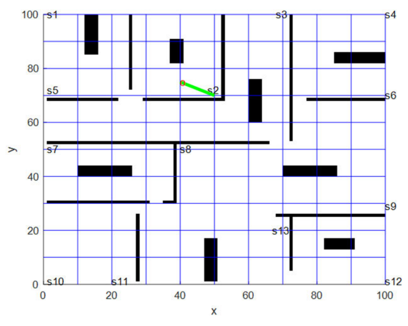

(2) $d_l^h = 120$


(3) $d_l^h = 50$


(3) $d_l^h = 80$


(3) $d_l^h = 120$


(4) $d_l^h = 50$


(4) $d_l^h = 80$

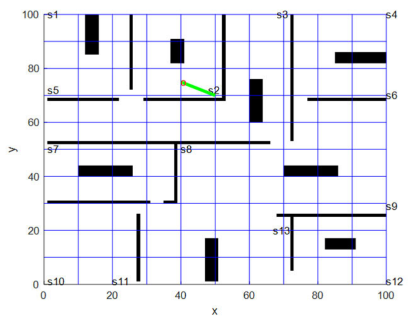

(4) $d_l^h = 120$

FIGURE 11. Comparison of planned paths with different starting locations and the same probability.

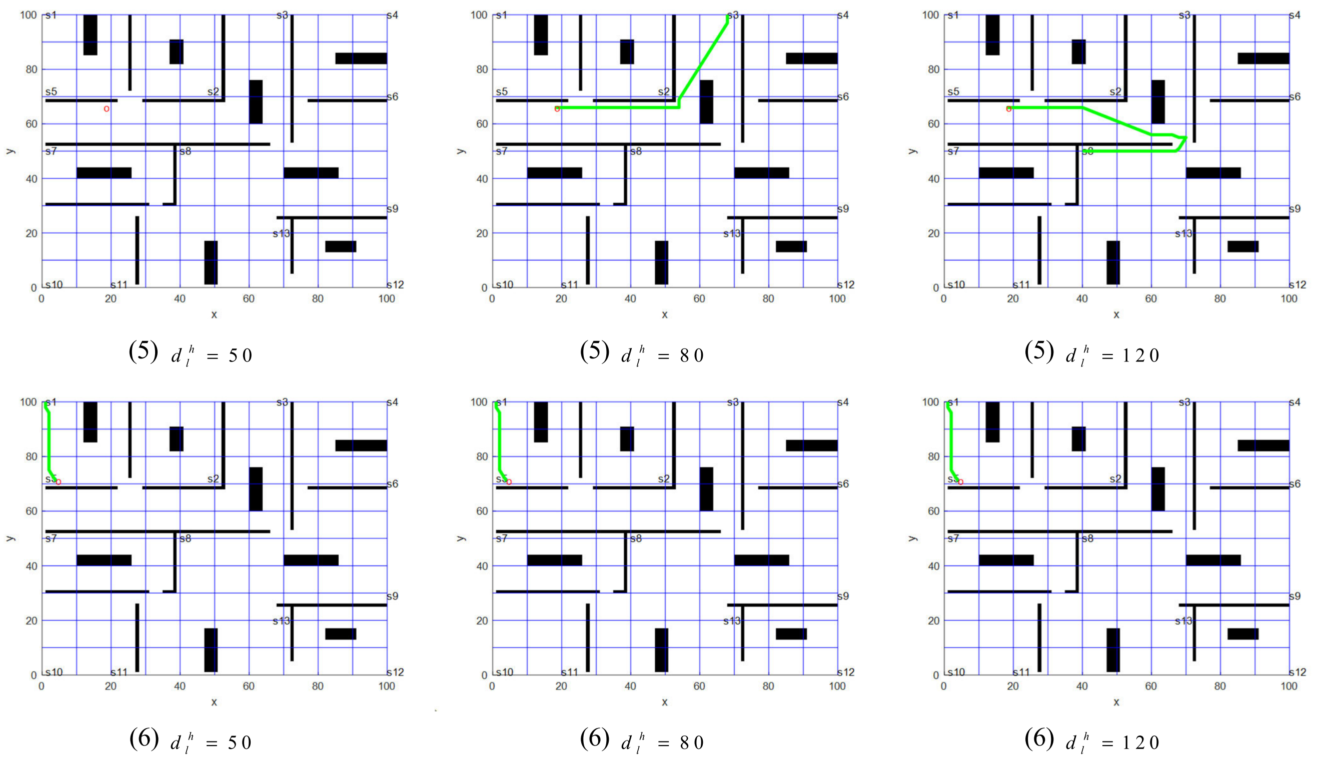


FIGURE 11. *(Continued.)* **Comparison of planned paths with different starting locations and the same probability.**

with the largest probability after charging. In test of starting location No. (3), the coordinates of the starting location are (4, 66). When $d_l^h = 50$, then $d_l^h < \min D(Q_1)$. Since $d_l^h < \min D(Q_2)$, the robot does not have a path that can be reached, and no path is planned; When $d_l^h = 80$, still $d_l^h < \min D(Q_1)$, Since $d_l^h < \min D(Q_2)$, the robot does not have a path that can be reached, and no path is planned; When $d_l^h = 120$, then $\min(D_{Q_1}) \le d_l^h < \max(D_{Q_1})$, our model selects s_8, $s_8 \in Q_1$. In test of starting location No. (4), the coordinates of the starting point are (40, 75). When $d_l^h = 50$, $\min(D_{Q_1}) \le d_l^h < \max(D_{Q_1})$, our model selects s_2, $s_2 \in Q_1$; When $d_l^h = 80$, then $\min(D_{Q_1}) \le d_l^h < \max(D_{Q_1})$, our model selects s_2; When $d_l^h = 120$, then $\min(D_{Q_1}) \le d_l^h < \max(D_{Q_1})$, our model selects s_2. In test of starting point No. (5), the coordinates of the starting point are (18, 66). When $d_l^h = 50$, then $d_l^h < \min D(Q_1)$, Since $d_l^h < \min D(Q_2)$, the robot does not have a path that can be reached, and no path is planned; When $d_l^h = 80$, then $\min(D_{Q_2}) \le d_l^h < \max(D_{Q_1})$, Our model extends the robot search range to Q_2. our model selects s_3, $s_3 \in Q_2$. The probability of s_3 is not the largest, but its distance from the robot is relatively small. In Q_2, the probability of s_{11} is the largest, but the robot is far from s_{11}; When $d_l^h = 120$, then $\min(D_{Q_1}) \le d_l^h < \max(D_{Q_1})$, our model selects s_8, $s_8 \in Q_1$. In test of starting point No. (6), the coordinates of the starting location are (4, 71), When $d_l^h = 50$, $d_l^h = 80$ and $d_l^h = 120$, all $\min(D_{Q_1}) \le d_l^h < \max(D_{Q_1})$, our model selects s_1, The distance from the starting location to s_1 is the minimum.

TABLE 11. **Locations of charging request in obstacle avoidance test.**

Location No.	Coord. X (unit-grid)	Coord. Y (unit-grid)
1	70	13
2	40	64
3	55	28
4	50	96
5	67	32
6	49	60
7	15	81
8	92	43
9	96	80
10	44	66

TABLE 12. **Probability of encountering obstacles at each boundary-crossing (door).**

$P_{'1'}$	$P_{'2'}$	$P_{'3'}$	$P_{'4'}$	$P_{'5'}$	$P_{'6'}$	$P_{'7'}$
0.45	0.12	0.09	0.77	0.09	0.35	0.05

From the results of Test 3 and Test 4, we can see that our path planning model has strong flexibility and can adapt to different scenarios. Our charging path planning model is

TABLE 13. **Comparison of the time value from each random position to the charging pile in the obstacle avoidance experiment.**

Location No.	$0<\theta<1$ （s）	$\theta>1$ （s）
1	15.4164078649987	15.4164078649987
2	**83.4164078649988**	**283.073262114491**
3	29.2022523189831	29.2022523189831
4	52	52
5	24.9442719099992	24.9442719099992
6	90.3957478239784	90.3957478239784
7	36.0438248629683	36.043824862968
8	32.4896699794899	32.4896699794899
9	21.8885438199983	21.888543819998
10	**90.4721359549996**	**290.128990204492**
Average time	47.62692624	87.55829709

TABLE 14. **The length of the planned path at 10 random locations in the obstacle avoidance experiment.**

Location No.	$\theta\geq1$ (unit-grid)	$0<\theta<1$ (unit-grid)
1	7.70820393249937	7.70820393249937
2	**41.5366310572456**	**41.7082039324994**
3	14.6011261594915	14.6011261594915
4	26	26
5	12.4721359549996	12.4721359549996
6	45.1978739119892	45.1978739119892
7	18.0219124314841	18.0219124314841
8	16.2448349897449	16.2448349897449
9	10.9442719099992	10.9442719099992
10	**45.0644951022460**	**45.2360679774998**

robust. The robot can plan an appropriate path under various remaining power conditions.

Test 5: This experiment is to verify the adaptability of obstacle avoidance of LMPP algorithm proposed in this paper. Since the LMPP in this paper considers the characteristics of the boundary-crossing, the obstacle-avoiding function selection is added. We have verified the necessity of adding obstacle avoidance function to LMPP algorithm. When $0 < \theta < 1$, LMPP algorithm has obstacle avoidance function. When $\theta > 1$, LMPP algorithm cancels obstacle avoidance function. We compared the time of LMPP algorithm to complete the charging task in these two cases. We assume that the running speed of the robot is 0.5 unit-grid/s. We randomly selected ten charging request locations, as shown in Table 11. These ten locations can be used as ten possible locations for robots to send charging requests, and also as locations for robot path planning when 10 robots send charging requests at the same time. When the robot completes the charging task, it is necessary to select the charging pile and plan the path. Due to the focus on obstacle avoidance, we set the probability of activity after charging to be the same, with a value of 0.1. At the same time, it is assumed that the robot has sufficient residual power.

The probability of encountering obstacles at each boundary-crossing (door) is set as shown in Table 12. When the robot encounters obstacles at the boundary-crossing, it needs to stop and re-plan, or make a detour, which will delay the time. We assume that the delay time when encountering obstacles is 200s.

For the case of $0 < \theta < 1$, we assume $\theta = 0.3$. The time consumed by LMPP algorithm with obstacle avoidance ($0 < \theta < 1$) and without obstacle avoidance ($\theta \geq 1$) at each random location is shown in Table 13. In both cases, the average time consumed by the robot to reach the charging pile is 47.62692624s and 87.55829709s respectively. In the second and tenth locations of the 10 randomly selected locations, the robot uses the LMPP algorithm with obstacle avoidance function to avoid the boundary-crossing with high probability of encountering obstacles, so that the time for the robot to reach the charging pile does not increase. However, the

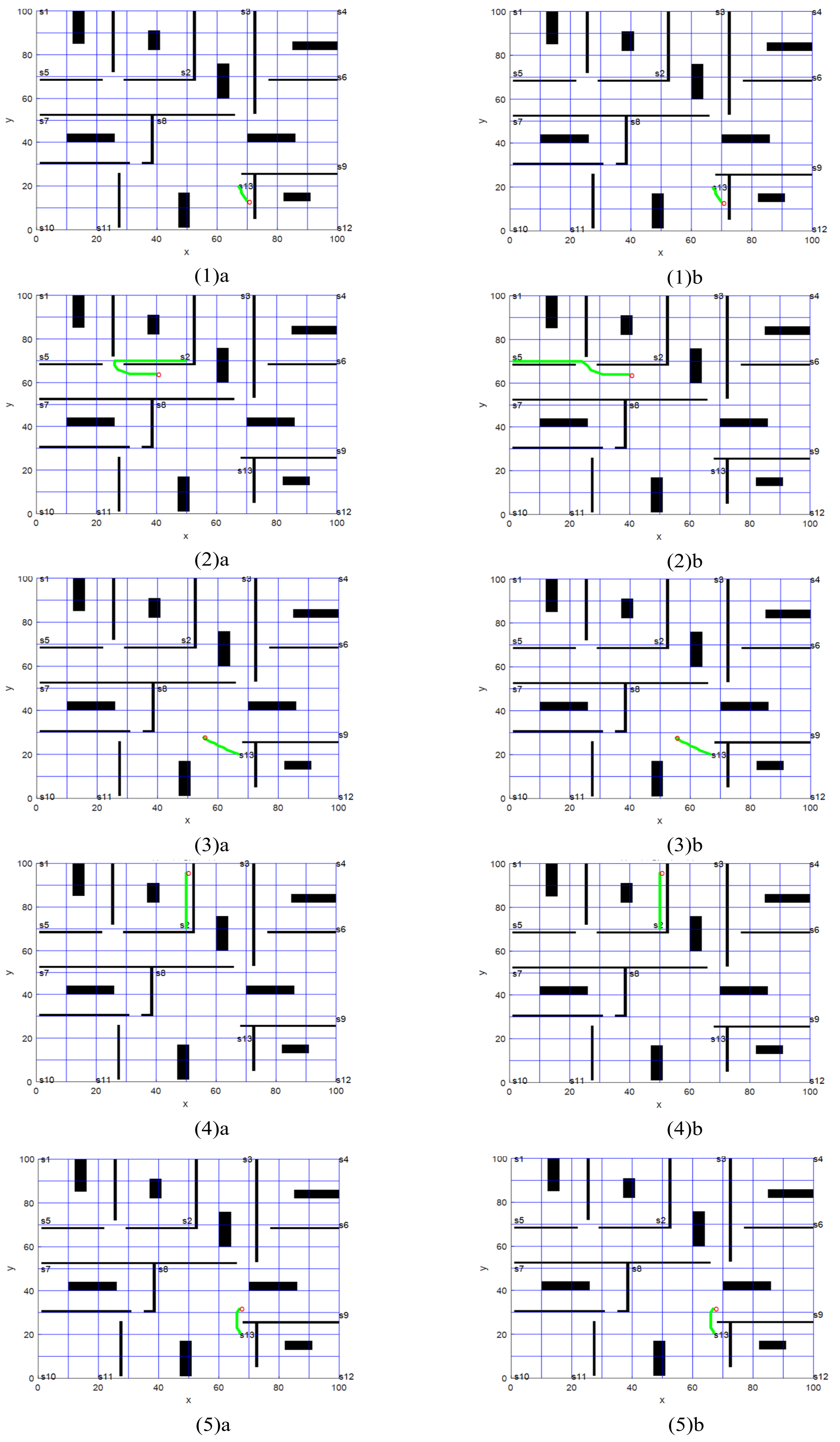
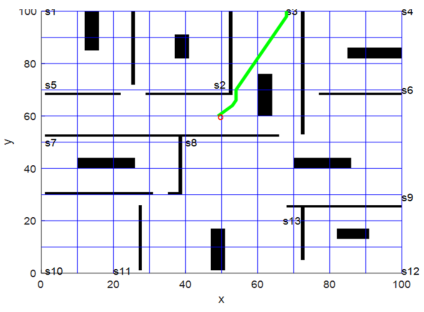
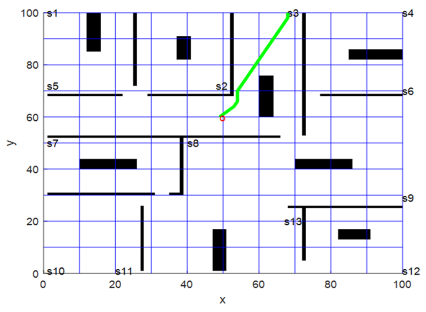
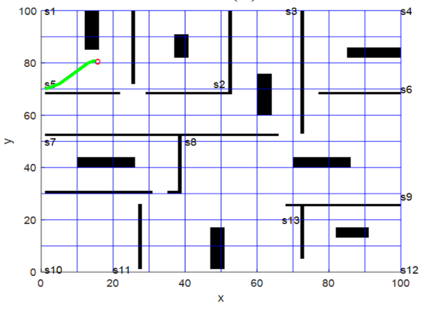
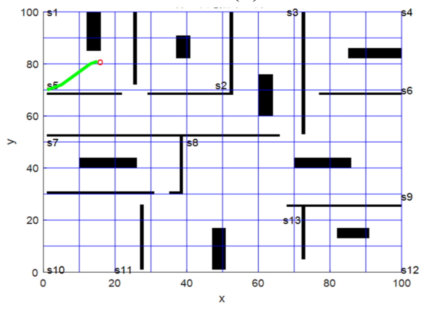


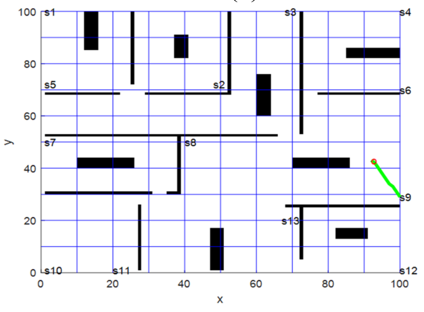
FIGURE 12. Planned paths with different locations in the obstacle avoidance experiment.

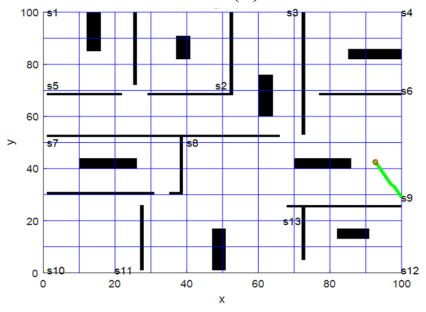

(6)a

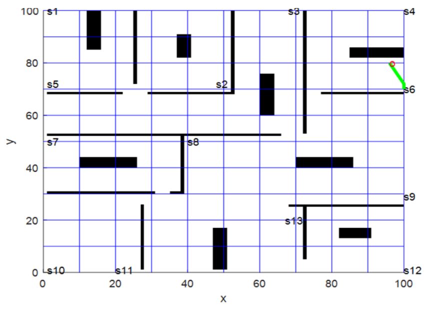

(6)b

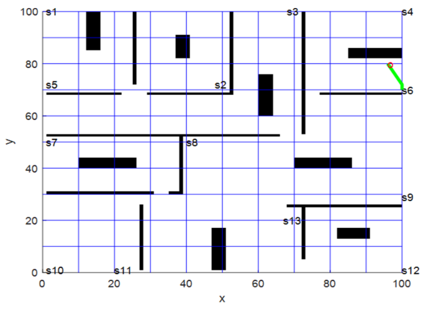

(7)a

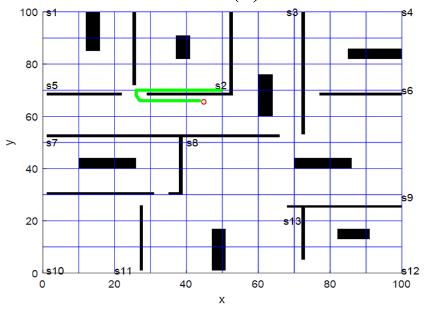

(7)b


(8)a


(8)b


(9)a


(9)b


(10)a

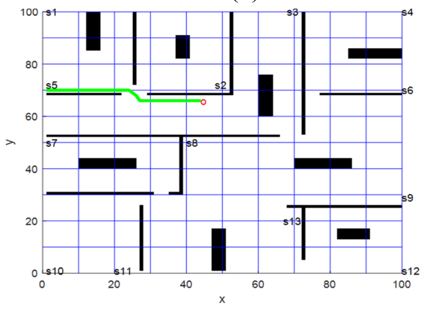

(10)b

FIGURE 12. ***(Continued.)*** **Planned paths with different locations in the obstacle avoidance experiment.**

path planned by LMPP algorithm without obstacle avoidance function meets the boundary-crossing with high probability of encountering obstacles, which increases the time to reach the charging pile.

The paths planned by LMPP algorithm with obstacle avoidance ($0 < \theta < 1$) and without obstacle avoidance ($\theta \geq 1$) are shown in Fig. 12.

(1) - (10) are the number of locations where the robot requests charging, and the paths are represented by green bold lines in Fig.12. Fig. 12(1)a - (10)a show the paths obtained by LMPP algorithm with obstacle avoidance function. Fig. 12(1)b - (10)b show the paths obtained by LMPP algorithm without obstacle avoidance function. The length of each path is shown in Table 14, whose unit is unit-grid.

As can be seen from Table 14, when the robot completes the charging task, the LMPP algorithm spends less time when the obstacle avoidance function is turned on than the average time when the obstacle avoidance function is not turned on. However, according to Fig. 12 and Table 14, we find that the path planned by LMPP algorithm with the obstacle avoidance function is not the worst. The path length planned with the obstacle avoidance function turned on is only slightly different from that planned without the obstacle avoidance function, but the average time to reach the charging pile is quite different. Therefore, it is necessary to turn on the obstacle avoidance function of the LMPP algorithm to effectively avoid the boundary-crossing with high probability of encountering obstacles, so that the robot can complete charging in less time.

VI. CONCLUSION

Robots need to be charged frequently to provide continuous service. We propose an intelligent multistage path planner model for accessing distributed charging piles on indoor map. Based on the communication between charging piles, the model considers the distance that the robot can travel with the remaining power and the range of the robot's activities after charging. When the robot can reach the charging pile, we plan the robot to the area with frequent activities. The multistage path planning model in this paper can make the robot choose the charging pile according to the distance of the remaining power and adopt the step planning, which has strong adaptability. The experiment and simulation show that our model can select the appropriate charging pile.

In the process of robot path planning, we designed a local memorial path planning algorithm to effectively improve the efficiency of robot path planning. The experimental results show that the algorithm is superior to the traditional algorithms.

REFERENCES

[1] M. P. Li, P. Sankaran, M. E. Kuhl, R. Ptucha, A. Ganguly, and A. Kwasinski, "Task selection by autonomous mobile robots in a warehouse using deep reinforcement learning," in *Proc. Winter Simulation Conf. (WSC)*, Dec. 2019, pp. 680–689.

[2] H. Hong, W. Shin, J. Oh, S. Lee, T. Kim, W. Lee, J. Choi, S. Suh, and K. Kim, "Standard for the quantification of a sterilization effect using an artificial intelligence disinfection robot," *Sensors*, vol. 21, no. 23, p. 7776, Nov. 2021.

[3] S. Song, J. Baba, J. Nakanishi, Y. Yoshikawa, and H. Ishiguro, "Teleoperated robot sells toothbrush in a shopping mall: A field study," in *Proc. Extended Abstr. CHI Conf. Hum. Factors Comput. Syst.*, May 2021, pp. 1–6.

[4] Z. Gui and H. Li, "Automated defect detection and visualization for the robotic airport runway inspection," *IEEE Access*, vol. 8, pp. 76100–76107, 2020.

[5] W. Zhao, R. Lin, S. Dong, W. Zhao, and Y. Cheng, "Dynamic node allocation-based multirobot path planning," *IEEE Access*, vol. 9, pp. 106399–106411, 2021.

[6] M. Baziyad, M. Saad, R. Fareh, T. Rabie, and I. Kamel, "Addressing real-time demands for robotic path planning systems: A routing protocol approach," *IEEE Access*, vol. 9, pp. 38132–38143, 2021.

[7] J. Wu, Y. Xue, and E. Qiu, "Research on unmanned surface vehicle path planning based on improved intelligent water drops algorithm," in *Proc. 4th Int. Conf. Electron. Inf. Technol. Comput. Eng.*, Nov. 2020, pp. 637–641.

[8] J. Yu, R. Li, Z. Feng, A. Zhao, Z. Yu, Z. Ye, and J. Wang, "A novel parallel ant colony optimization algorithm for warehouse path planning," *J. Control Sci. Eng.*, vol. 2020, pp. 1–12, Aug. 2020.

[9] T. Qiuyun, S. Hongyan, G. Hengwei, and W. Ping, "Improved particle swarm optimization algorithm for AGV path planning," *IEEE Access*, vol. 9, pp. 33522–33531, 2021.

[10] M. A. Gutierrez-Martinez, E. G. Rojo-Rodriguez, L. E. Cabriales-Ramirez, L. A. Reyes-Osorio, P. Castillo, and O. Garcia-Salazar, "Collision-free path planning based on a genetic algorithm for quadrotor UAVs," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Sep. 2020, pp. 948–957.

[11] Y. Ying, Z. Li, G. Ruihong, H. Yisa, T. Haiyan, and M. Junxi, "Path planning of mobile robot based on improved RRT algorithm," in *Proc. Chin. Autom. Congr. (CAC)*, Nov. 2019, pp. 4741–4746.

[12] M. Zhao, H. Lu, S. Yang, and F. Guo, "The experience-memory Q-learning algorithm for robot path planning in unknown environment," *IEEE Access*, vol. 8, pp. 47824–47844, 2020.

[13] L. Wang, L. Liu, J. Qi, and W. Peng, "Improved quantum particle swarm optimization algorithm for offline path planning in AUVs," *IEEE Access*, vol. 8, pp. 143397–143411, 2020.

[14] G. Changjiang, C. Yinan, and T. Xiaohai, "Research on distribution route planning model and algorithm of unmanned aerial vehicle (UAV) based on improved multi-objective genetic algorithm," in *Proc. 2nd Int. Conf. Artif. Intell. Inf. Syst.*, May 2021, pp. 1–7.

[15] G. Flores-Caballero, A. Rodriguez-Molina, M. Aldape-Perez, and M. G. Villarreal-Cervantes, "Optimized path-planning in continuous spaces for unmanned aerial vehicles using meta-heuristics," *IEEE Access*, vol. 8, pp. 176774–176788, 2020.

[16] W. Hou, Q. Luo, X. Wu, Y. Zhou, and G. Si, "Multiobjective optimization of large-scale EVs charging path planning and charging pricing strategy for charging station," *Complexity*, vol. 2021, pp. 1–17, Feb. 2021.

[17] D. Ding, J. Li, P. Tu, H. Wang, T. Cao, and F. Zhang, "Electric vehicle charging warning and path planning method based on spark," *IEEE Access*, vol. 8, pp. 8543–8553, 2020.

[18] G. Huang, X. Yuan, K. Shi, Z. Liu, and X. Wu, "A 3-D multi-object path planning method for electric vehicle considering the energy consumption and distance," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 7508–7520, Jul. 2022.

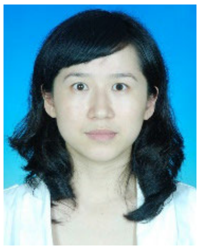
[19] P. Typaldos, M. Papageorgiou, and I. Papamichail, "Optimization-based path-planning for connected and non-connected automated vehicles," *Transp. Res. C, Emerg. Technol.*, vol. 134, Jan. 2022, Art. no. 103487.

[20] E. Petavratzis, L. Moysis, C. Volos, I. Stouboulos, H. Nistazakis, and K. Valavanis, "A chaotic path planning generator enhanced by a memory technique," *Robot. Auto. Syst.*, vol. 143, Sep. 2021, Art. no. 103826.

[21] R. Xie, Z. Meng, L. Wang, H. Li, K. Wang, and Z. Wu, "Unmanned aerial vehicle path planning algorithm based on deep reinforcement learning in large-scale and dynamic environments," *IEEE Access*, vol. 9, pp. 24884–24900, 2021.

[22] X. Kong, B. Pan, E. Cherkashin, X. Zhang, L. Liu, and J. Hou, "Multi-constraint UAV fast path planning based on improved A* algorithm," *J. Phys., Conf. Ser.*, vol. 1624, no. 4, Oct. 2020, Art. no. 042009.

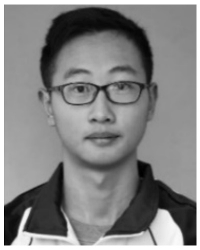
[23] D. Li, S. Liu, W. Xiang, Q. Tan, K. Yuan, Z. Zhang, and Y. Hu, "A SLAM system based on RGBD image and point-line feature," *IEEE Access*, vol. 9, pp. 9012–9025, 2021.
[24] J. Liu, W. Gao, and Z. Hu, "Bidirectional trajectory computation for odometer-aided visual-inertial SLAM," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1670–1677, Apr. 2021.
[25] A. Steenbeek and F. Nex, "CNN-based dense monocular visual SLAM for real-time UAV exploration in emergency conditions," *Drones*, vol. 6, no. 3, p. 79, Mar. 2022.
[26] C. Panatarani, D. Murtaddo, D. W. Maulana, S. Irawan, and I. M. Joni, "Design and development of electric vehicle charging station equipped with RFID," in *Proc. AIP Conf.*, Feb. 2016, Art. no. 030007.
[27] J. Yu, W. Jiang, Z. Luo, and L. Yang, "Application of a vision-based single target on robot positioning system," *Sensors*, vol. 21, no. 5, p. 1829, Mar. 2021.
[28] A. Baikovitz, P. Sodhi, M. Dille, and M. Kaess, "CMU-GPR dataset: Ground penetrating radar dataset for robot localization and mapping," 2021, *arXiv:2107.07606*.
[29] Y. Wang, P. Bai, X. Liang, W. Wang, J. Zhang, and Q. Fu, "Reconnaissance mission conducted by UAV swarms based on distributed PSO path planning algorithms," *IEEE Access*, vol. 7, pp. 105086–105099, 2019.
[30] L. Zhang and Y. Li, "Mobile robot path planning algorithm based on improved a star," *J. Phys., Conf. Ser.*, vol. 1848, no. 1, 2021, Art. no. 012013.
[31] J. Gibson, T. Schuler, L. McGuire, D. M. Lofaro, and D. Sofge, "Swarm and multi-agent time-based A* path planning for lighter-than-air systems," *Unmanned Syst.*, vol. 8, no. 3, pp. 253–260, Jul. 2020.
[32] L. Zhang and Y. Li, "Mobile robot path planning algorithm based on improved a star," *J. Phys., Conf. Ser.*, vol. 1848, no. 1, Apr. 2021, Art. no. 012013.
[33] L. J. Zhou, W. Cui, and H. Y. Fu, "AGV path planning combining A* and ant colony algorithm," *J. Phys., Conf. Ser.*, vol. 1948, no. 1, Jun. 2021, Art. no. 012006.
[34] A. Wang, P. Zhi, W. Zhu, H. Qiu, H. Wang, and W. Wang, "Path planning of unmanned surface vehicle based on A* algorithm optimization considering the influence of risk factors," in *Proc. 4th IEEE Int. Conf. Ind. Cyber-Phys. Syst. (ICPS)*, May 2021, pp. 810–815.
[35] A. F. Olalekan, J. A. Sagor, M. H. Hasan, and A. S. Oluwatobi, "Comparison of two SLAM algorithms provided by ROS (robot operating system)," in *Proc. 2nd Int. Conf. Emerg. Technol. (INCET)*, May 2021, pp. 1–5.
[36] X. Zhang, J. Wang, Y. Fang, and J. Yuan, "Multilevel humanlike motion planning for mobile robots in complex indoor environments," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 3, pp. 1244–1258, Jul. 2019.
[37] F. Bayat, S. Najafinia, and M. Aliyari, "Mobile robots path planning: Electrostatic potential field approach," *Expert Syst. Appl.*, vol. 100, pp. 68–78, Jun. 2018.



YANFANG DENG received the B.S. degree in information and computing science and the M.S. degree in applied mathematics from Hainan University, China, in 2007 and 2010, respectively, where she is currently pursuing the Ph.D. degree with the School of Information and Communication. Her current research interests include big data and robot charging.

TAIJUN LI received the B.S. degree in radio engineering and the M.S. degree in electronic materials and devices from the South China University of Technology, China, in 1984 and 1987, respectively. Currently, he is with the School of Information and Communication Engineering, Hainan University. His current research interests include internet network multimedia information processing, internet information retrieval, microcomputer interface, and digital signal processing.

MINGSHAN XIE received the B.S. degree in educational technology from Southwest University, Chongqing, China, in 2004, the M.S. degree in computer application technology from Hainan University, Haikou, China, in 2013, and the Ph.D. degree in information and communications, in 2019. Currently, he is with the Department of College of Big Data and Information Engineering, Guizhou University. His current research interests include the Internet of Things, smart robot, and artificial intelligence.



WENHAN CHEN was born in Quanzhou, Fujian, China, in 2001. He is currently pursuing the B.E. degree in communication engineering with Guizhou University. Since 2020, he has been doing his research with his tutor. His main research interests include artificial intelligence, image processing, and task scheduling, especially in study of path planning.