IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 13, 2020, accepted April 23, 2020, date of publication April 28, 2020, date of current version May 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2991076

A Novel Path Planning Algorithm for Warehouse Robots Based on a Two-Dimensional Grid Model

BO YANG, WENTAO LI, JIANRONG WANG, JINGJIE YANG, TIANTIAN WANG, AND XIN LIU
School of Mathematics Sciences, Shanxi University, Taiyuan 030006, China

Corresponding author: Wentao Li (lwtcon@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant U1610116 and Grant 61403240, in part by the Natural Science Foundation of Shanxi under Grant 201801D221171, and in part by the Key Research and Development Program of Shanxi under Grant 201903D121145.

ABSTRACT In this paper, the path planning problem of goods transportation is formulated as a traveling salesman problem (TSP). A novel path planning algorithm for warehouse robots based on a two-dimensional (2D) grid model is proposed to solve this type of TSP. Firstly, we simplified the traditional pile type warehouse as a node-based 2D grid model. Then, a new concept called the largest convex polygon (LCP) is introduced to illustrate the shortest path to traverse all goods locations in an ideal condition. Next, the remaining locations are classified by their relationship with LCP and designed path planning rules separately. Finally, we merged the paths of different types of cargo locations to get the final path. The experimental results show that, compared with ant colony optimization (ACO) and genetic algorithm (GA), our proposed algorithm could effectively reduce the computation time and total path length.

INDEX TERMS Warehouse robots, path planning, TSP, 2D grid model, largest convex polygon.

I. INTRODUCTION

At present, the e-commerce industry has ushered in the best development period. Various e-commerce companies have sprung up, and the trend of people shopping online has continued to rise, which has created many challenges for all aspects of e-commerce [1]. Among them, the logistics industry is the foundation and guarantee of e-commerce [2]. With the increasing number of online orders, the logistics industry will face more difficult challenges, so it is necessary to ensure the efficiency of logistics. The efficiency of storage and retrieval of goods in the warehouse has the most direct impact on the efficiency of logistics operations. Many intelligent warehouses have used automated equipment instead of manpower, the most representative of which is the automated guided vehicles for cargo transportation. In this paper, we refer to such automated guided vehicles as warehouse transportation robots, and warehouse robots for short.

The warehouse robots can effectively reduce the labor cost in the logistics process, which does not require any manual control. It can receive the transportation task assigned by the central controller, and such transportation task mainly includes the coordinates of all the goods that the warehouse robot needs to pick up. Under the calculation of the warehouse robot's own CPU, the robot can obtain an order to traverse all the positions of the goods. Then it will automatically move from the parking position to the calculated first goods position to complete the loading task, and then go to the next goods position. After completing all the loading tasks, it will return to the parking position to deliver all the cargo, thus realizing the automatic goods transportation. Therefore, an excellent robot path planning algorithm can greatly improve the efficiency of cargo transportation and reduce manual errors. Moreover, The path planning process for transport robots picking up goods in turn can be considered as a type of TSP. So improving the path planning algorithm of the warehouse robot has a significant impact on improving its transportation efficiency.

The type of path planning for warehouse robots can be divided into two categories: conventional path planning methods and heuristic path planning methods. Many conventional methods such as A* algorithm [3] can obtain good results in warehouses with fewer goods locations. While the goods locations increase, however, the time-out problem of A* algorithm becomes serious. Based on the characteristics of A* algorithm, Zheng *et al.* [4] optimized the node search mode and search speed, and add the angle evaluation cost function to the cost function of A* algorithm to find the path with the least inflection point. But they did not suggest

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

any solution to optimize the long calculation time. Besides, Kusuma *et al.* [5] used A* algorithm to find the shortest path based on the coordinates of the current position and the coordinate of target position. And their research goal is that when the robot deviates from the target, the robot can approach the target object and change route. Because they did not perform any optimization on A* algorithm, the long calculation time of A* algorithm is still not solved. To reduce calculation time, many heuristic methods represented by evolutionary computation have been proposed. Evolutionary computation is a kind of adaptive optimization probabilistic search algorithm, mainly including GA [6]–[8], ACO [9]–[12], improved ACO [12]–[15], particle swarm optimization [16], [17], improved particle swarm optimization [18]–[20], and artificial bee colony [21], [22] etc.

Lamini *et al.* [23] proposed an improved crossover operator, for solving path planning problems using genetic algorithms (GA) in static environment. The proposed crossover operator avoids premature convergence and offers feasible paths with better fitness value than its parents, thus the algorithm converges more rapidly. Kumar and Kumar [24] design a robot path planning algorithm to avoid collision under four warehouse models but the complexity of the algorithm does not seem to be taken into account compared with other algorithms. YongBo *et al.* [25] studied the problem of UAV path planning in 3D space by minimizing the multi-objective cost function. They improved the traditional wolf pack search algorithm (WPS) by introducing the concepts of chromosome and gene in GA. Simulation results show that the trajectory generated by the improved WPS algorithm is far superior to traditional WPS algorithm, GA and random search methods. Harshal *et al.* construct a new objective function with the help of distance between robot to obstacle and goal in [26]. Then they demonstrate the exploration rate are more effective than conventional particle swarm optimization in their proposed APSO algorithm. Finally, it is verified that the robot can avoid the obstacle and follows the target path.

For most heuristic algorithms, the calculation time is significantly reduced, however, it is not stable enough to obtain the optimal solution. To solve the problem of too long computation time in the traditional algorithms and the problem that the optimal solution cannot be obtained steadily in these heuristic algorithms, a novel path planning algorithm for robots is proposed in this paper under a 2D grid model of logistics warehouses. After establishing a node-based 2D grid model represented by the center of the grids, we introduce a new concept called LCP to find the shortest path to traverse all goods locations in an ideal condition. Next, the remaining locations inside the LCP are classified according to their relationship with LCP and designed path planning rules. Finally, the paths of different types are merged to get the complete path. The experiments are performed to demonstrate that the proposed algorithm could effectively reduce the computation time and path length.

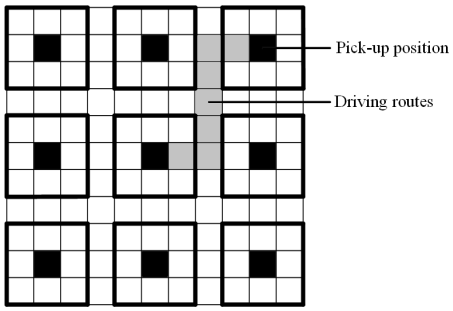


FIGURE 1. The aerial view of the pile type warehouse.

II. PROBLEM DESCRIPTION

As shown in Fig. 1, it is the aerial view of the pile type warehouse. The rough black rectangular box is the area occupied by a shelf, and the black solid square inside the rectangular box is the location where the robot picks up goods; the gray squares represent the roads that the robot can travel between two goods position. The rough black boxes are overhead so that the robot can freely pass under it.

Later, we consider simplifying this layout by using nodes to represent the robot pick-up positions, and straight lines to represent the roads, shown in Fig. 2.

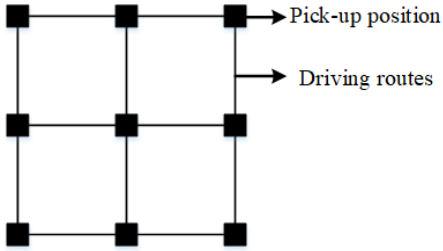


FIGURE 2. The simplified warehouse model.

$G(V, E, \omega)$ is used to express the grid model, where V is a set of all pick-up positions. E is a set of the driving routes, i.e. the grid edges in the 2D grid model. ω represents the set of weights determined by the distance between any two pick-up positions. V, E, and ω of the grid model G are generated according to real demands.

All shelves are located at the intersections of grid lines. P represents the parking location of a robot (i.e. the starting position when performing a task and the destination after completing the task). Note that, the path planning for warehouse robots in this paper is designed for a single robot and does not consider the collision avoidance and cooperation of multiple robots during driving, just optimize the performance of each robot in the task.

Specifically, a warehouse robot will start from P (starting position) when receives a task including the quantity and position of the shelves to be picked up, then go to each shelf position P_i according to the traversal order calculated by the robot CPU, where i denotes the shelves number,

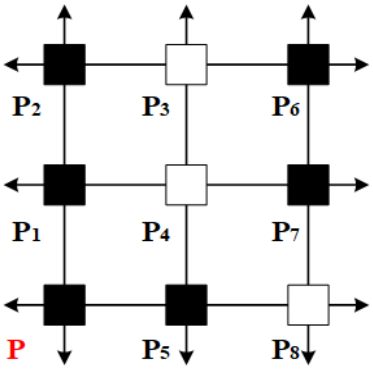


FIGURE 3. Task distribution.

$i = 1, 2, \cdots, n$. Finally, robots transport all goods back to P (destination). As Fig. 3 shows, solid nodes P_1, P_2, P_5, P_6 and P_7 indicate that there are goods to be picked up at these locations; hollow nodes P_3, P_4 and P_8 mean that the robot does not have to go to these locations.

According to different research priorities, this type of robot task can be divided into several parts. Among them, step "Task release" could be optimized through an excellent task distributing and scheduling, which can reduce the overall cost by assigning more suitable positions for each robot. Step "Starting moving" can be improved by more rapid and accurate data transmission, because there may be multiple robots working at the same time in real logistics warehouses, which may cause a lot of noise and information misplacement. If the robot have already picked up all goods, it will return to the parking position P.

In step "calculation", however, improving the path planning algorithm is prominent to reduce the cost of completing a task, which mainly includes the computational time of CPU and the length of the resulting path. This paper is mainly for designing the path planning algorithm, focusing on having the ability of obtaining better solutions with shorter computing time under a smaller number of nodes.

When the robot receives a task, it starts initialization to locate its current position. Then, according to the position coordinates of all nodes given by the task, the proposed algorithm is used to calculate the traversal order. Based on the obtained traversal order, the robot moves to the next position. The termination criterion T is that the robot has loaded all goods in the task. If it is satisfied, it returns to the parking position P, otherwise, it continues the task.

III. ALGORITHM DESIGN

In this section, under our grid model, an algorithm is created to solve this type of TSP. A partial solution S is represented as an ordered list, where $|S|$ expresses the number of nodes in S.

$$S = (v_1, v_2, \cdots, v_{|S|}) \tag{1}$$
$$\bar{S} = (r_1, r_2, \cdots, r_{|\bar{S}|}) \tag{2}$$

Note that, the nodes in S and $\bar{S}$ are unknown at this time. When the LCP is found for the first time, the nodes in S and $\bar{S}$ will be determined initially. Simultaneously, $\bar{S} = V - S$ denotes the remain nodes of V, where $|\bar{S}|$ expresses the number of nodes in $\bar{S}$. S is continuously filled and the initial S is actually the list of the vertexes on LCP mentioned earlier. Afterward, the remain nodes will be inserted continuously after a specified node v_i^* in S with the least cost. Then, the partial solution S will be extended as

$$S := (S, r_j), \tag{3}$$
$$v_i^* := \underset{v_i}{argminc}(S, G) \tag{4}$$

And (S, r_j) denotes appending r_j to a selected position in the ordered list S. v_i^* means the node in S where r_j will be inserted after. This step is repeated until the termination criterion T is satisfied. The total path cost is calculated by the cost function $c(S, G)$, which is written as

$$c(S, G) = \sum_{i=1}^{|S|-1} \omega(S(i), S(i+1)) + \omega(S(|S|), S(1)) \tag{5}$$

The termination criterion T is $S = V$.

For illustration, (M, N) is the shortest route between node M and N. Suppose the coordinates of M and N are (x_M, y_M) and (x_N, y_N), respectively. The shortest distance between two nodes can not be calculated by Euclidean distance under our grid model, but by Manhattan distance, which could be obtained by the following:

$$d_{MN} = |x_M - x_N| + |y_M - y_N| \tag{6}$$

In this paper, we use the calculated shortest Manhattan distance d_{MN} between M and N as the weight ω of the path (M, N).

$$\omega(M, N) = d_{MN} \tag{7}$$

The idea of our algorithm is essentially a insertion method. The algorithm will obtain a solution by inserting these nodes into a certain partial solution S, that can minimize the increment of cost function. The subpath is referred to the straight line linking two certain adjacent vertex on the LCP that all nodes can form. Below, we will propose a few concepts to design path planning algorithm.

A. BOUNDARY ATTRIBUTE

In this subsection, boundary attribute (BA) of nodes is proposed to design path planning rules for robots in transporting tasks. This section starts with the case of two pickup positions, then this problem can be considered as a three-point of TSP. In Fig. 4, P is the parking position, and P_1, P_2 are two shelves nodes. So there are only two orders to traverse the three nodes, that is $P \to P_1 \to P_2 \to P$ and $P \to P_2 \to P_1 \to P$.

Next, a rectangular coordinate system is established with the robot parking position P as the origin, and then assume that the coordinates of P, P_1, and P_2 are (0,0), (x_1, y_1)

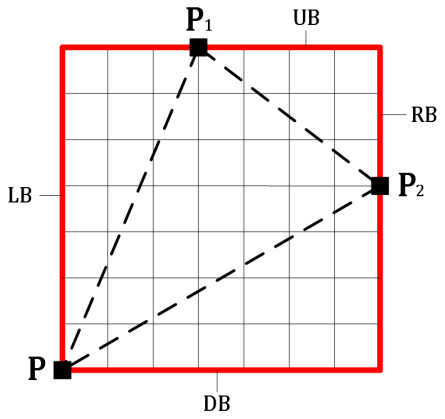


FIGURE 4. The shortest path of traversing these three nodes in the grid model.

and (x_2, y_2), respectively. Let these nodes with the maximum Y value have the ''up boundary attribute'' (UB). These nodes with the minimum Y value have the ''down boundary attribute'' (DB). The ones with the maximum X value have the ''right boundary attribute'' (RB) and the nodes with the minimum X value have the ''left boundary attribute'' (LB).

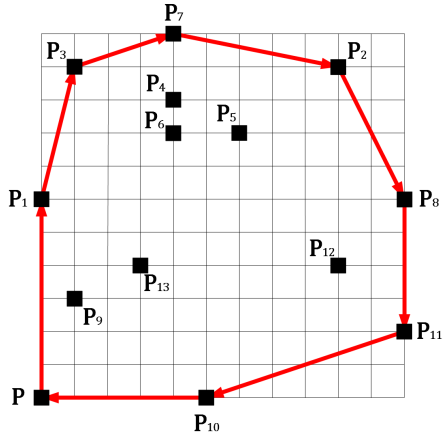


FIGURE 5. The search process of LCP.

Since the robot can only travel horizontally or vertically, the shortest path length is the sum of four boundary lengths, as the red lines shown in Fig. 5. The shortest length is calculated by

$$d_{min} = 2(|x_{max} - x_{min}| + |y_{max} - y_{min}|) \tag{8}$$

Next, we can generalize this problem into four nodes. This situation can be viewed as placing a new node inside or outside of the triangle formed by the first three nodes, which is represented as the dashed black lines shown in Fig. 5. These two different distributions will make the path connecting the four nodes form a convex or concave quadrilateral. Both distributions will be considered as follows.

B. THE LARGEST CONVEX POLYGON

This subsection proposes a method of finding the LCP with any number of nodes, explained in algorithm 1 and shown in Fig. 5.

Algorithm 1 Search Process of LCP Which Can be Formed With n Nodes

Input: All nodes positions

Output: The vertexes set N of the LCP that all nodes can form

1: Set the node with the minimum value Y on the left boundary as the initial node P.
2: A Cartesian coordinate is established with P as the origin and makes a directed line L_d overlapped with the y-axis.
3: Termination criterion T: L_d repasses through the initial node P.
4: Initializes node subset $N = \{P\}$.
5: **While Not** T **Do**
6: **if** other nodes just above P **then**
7: According to the distance between these positions and P, add them into N from near to far, $N := (N, P_i)$, where $i = 1, 2, \cdots, n-1$.
8: **end if**
9: Move the rotation center to P_i newly added to N.
10: Rotate L_d clockwise until intersects any node. If L_d passes through more than 1 node at a certain angle, then append these nodes to S from near to far.

Note that the LCP here is the polygon with the largest size. Besides, the position of P can also be arbitrary. In this paper, we place P at the down-left corner of the grid model.

C. SUBPATH COVERAGE AREA

A new concept called subpath coverage area (SCA) is proposed to solve the problem to minimize cost increments. Each subpath refers to the path connected by two adjacent nodes of the LCP. The shortest length of a subpath is calculated by Manhattan distance, but there may be several actual routes with the same shortest length. For example, the shortest distance between P and P_1 is 4, but more than one path can be selected, as displayed in Fig. 6(a-e). Note that, we assume that the length of each segment of the grid line is 1 in this paper. All the shortest routes locate in the rectangle formed by the two nodes. Then, if a new node P_2 locates in the rectangle formed by P and P_1, the length of the newly generated path $P \rightarrow P_2 \rightarrow P_1$ will not increase after inserting P_2, as the gray area shown in Fig. 6(f). So we call the rectangle as the coverage area of this subpath. Meanwhile, some subpaths can only cover a grid edge, such as $P \rightarrow P_1$, $P_8 \rightarrow P_{11}$ and $P_{10} \rightarrow P$ in Fig. 5. So we can draw a conclusion: if a new node locates in a SCA, the new path length will not increase after inserting it into the subpath.

IV. ATTRIBUTION OF THE INTERIOR NODES

If a node to be inserted does not locate in any SCA, it can be called as an interior node, such as P_4, P_5, P_6, P_9, P_{12} and P_{13}

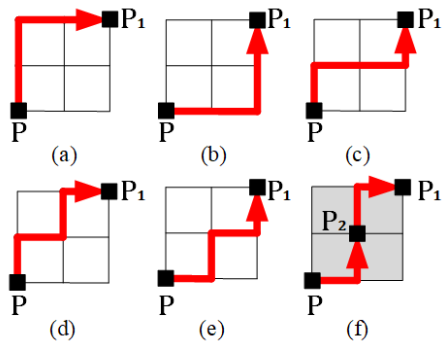


FIGURE 6. Several actual routes with the same shortest length between two nodes.

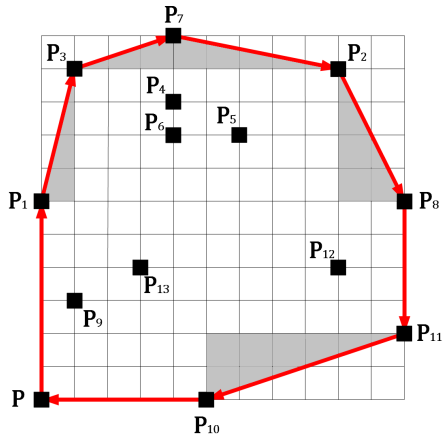


FIGURE 7. Subpath coverage areas.

Algorithm 2 Calculation Method of Point-to-Line Segment Distance Based on the Grid Model

Input: The coordinate of node N (x_N, y_N). The coordinates of the two vertices U and V of the line segment (x_U, y_U) and (x_V, y_V).

Output: The closest distance between N and the line segment.

if $\frac{y_V - y_U}{x_V - x_U} = 0$ **then**
  if $x_U \le x_N \le x_V$ or $x_V \le x_N \le x_U$ **then**
    $d_{min} = |y_N - y_U| = |y_N - y_V|$
  else $d_{min} = min(d_{NU}, d_{NV})$
  end if
else if $y_U \le y_N \le y_V$ or $y_V \le y_N \le y_U$ **then**
    $d_{min} = |x_N - x_U| = |x_N - x_V|$
else $d_{min} = min(d_{NU}, d_{NV})$
end if

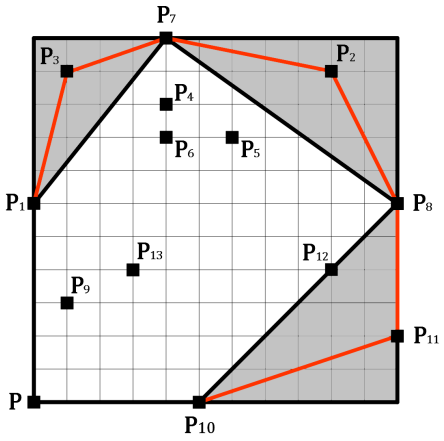


FIGURE 8. Region between boundaries (RBB).

in Fig. 7. Next, the problem can be translated into: find the shortest path between each interior node and a certain SCA.

When searching the SCA closest to P_4, only the distances between P_4 and four boundaries of each SCA needs to be determined. And each boundary is a line segment, not a straight line. The calculation method of the distance of point-to-line segment is different from the distance of point-to-straight line extremely, as explained in algorithm 2.

The problem is viewed as finding out the shortest distance of point-to-line segments, but there are still many boundaries to be considered.

To reduce automatically the number of boundaries to be considered, the following concept called ''region between boundary'' (RBB) is introduced to remove unused boundary. RBB is composed of the rectangular frame and the straight lines connected by the nodes with boundary attributes. In Fig. 8, the gray areas are the RBBs, which represent the active range of the remaining vertices of the LCP. Then these subpaths in the RBBs can be generally classified as $k > 0$ and $k < 0$ in terms of slope k. But it is not concluded that the boundary selection of SCAs is on the basis of k. Such as $P_1 \rightarrow P_3$ and $P_{11} \rightarrow P_{10}$, the the bottom and right side of SCA of $P_1 \rightarrow P_3$ is obviously closer to the interior nodes, while $P_{11} \rightarrow P_{10}$ is the left and top side. Even if the slope k of both paths is greater than 0, the selected boundaries are different. Therefore, if the slopes of several paths are all greater or less than 0, it cannot be relied on this principle to classify subpaths roughly.

As shown in Fig. 8, we use the boundary attributes of nodes to determine the four boundaries of the LCP, namely PP_1, P_7, P_8P_{11} and $P_{10}P$, respectively. Although P_7 is only a node, it also represents a boundary. These nodes can form a list of boundary nodes, $B = \{P, P_1, P_7, P_8, P_{11}, P_{10}, P\}$. After extending these boundaries separately, the obtained rectangular frame will generate RBBs with the polygon connected by the boundary nodes in sequence. Then all the remaining vertices of the LCP are in the gray area.

The selection rules of SCA boundaries are given in table. 1. For the subpaths in the RBB formed by the left and top boundary, the bottom and right side of the SCA is selected. For the subpaths in the RBB formed by the top and right boundary, we choose the bottom and left side of the SCA. For the subpaths in the RBB formed by the right and down

TABLE 1. The selection rules of SCA boundaries in different RBBs.

The formed boundaries of a RBB	The selected sides of the SCA
left+top	right+bottom
top+right	bottom+left
right+bottom	left+top
bottom+left	top+right

boundary, we select the top and left side of the SCA. For the subpaths in the RBB formed by the left and bottom boundary, the top and right side of the SCA will be picked.

As can be seen from table. 1, the selected sides of a SCA and the formed boundaries of a RBB are complementary. Such as the selected sides of the SCA of $P_1 \rightarrow P_3$ and $P_3 \rightarrow P_7$ are bottom and right side by the proposed selection rules, while the RBB is formed by the top and left boundaries. When unnecessary boundaries are eliminated, these selected boundaries will make up the minimum range of SCA, as the black thick lines shown in Fig. 9. The distance between a interior node and the selected boundaries of a SCA will be calculated as algorithm 3. After calculating the nearest SCA boundary, the interior node will belong to the subpath that form this SCA.

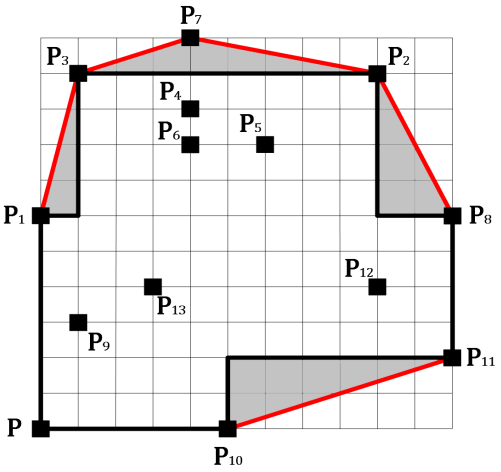


FIGURE 9. Minimum range of SCAs.

Algorithm 3 Calculate the Nearest Boundary of a SCA to a Interior Node

1: Use algorithm 1 to find the LCP.
2: Determine the mathematical representation of RBBs.
3: Determine the RBB which the remaining vertices locate in, respectively.
4: According to the path selection rules in table 1, obtain the minimum range boundary of all SCAs.
5: Algorithm 2 is used to calculate the distance from the node to each boundary $d_i (i = 1, 2, \cdots, n)$.
6: The nearest boundary of SCA to a node is calculated by $d_{min} = min\{d_1, d_2, \cdots, d_n\}$.

V. PATH PLANNING IN COMPLEX SITUATIONS

After formulating the simple path planning rules, three situations are considered in this section as follows: (1) multiple interior nodes locate in the same SCA; (2) multiple interior nodes do not locate in any SCA; (3) path fusion of the two types of nodes above.

A. PATH PLANNING FOR MULTIPLE NODES IN A SAME SCA

This subsection mainly introduces the path planning rules for multiple nodes in the same SCA. Following the LCP is determined, the SCAs will be created by two adjacent vertexes. As shown in Fig. 10(a), there are six nodes in the SCA of $P \rightarrow A$. It is considered that inserting all the six nodes into subpath $P \rightarrow A$ with a minimum cost. As shown in Fig. 10(b), the thick black lines are the edges of the LCP that all interior nodes can form. The edges of the LCP obtained by algorithm 1 are $P \rightarrow C$, $C \rightarrow F$, $F \rightarrow G$ and $G \rightarrow A$ beside $P \rightarrow A$. There are several new SCAs and three interior nodes B, D and E in the newly formed convex polygon. The newly produced SCAs is given as the gray areas in Fig. 10(b).

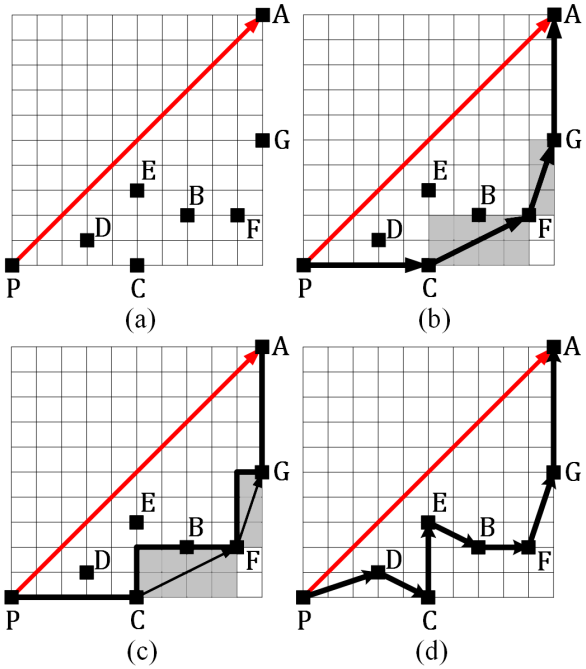


FIGURE 10. Path generation process of the nodes in the same SCA.

The interior nodes D and E does not locate in any new SCAs. So find the nearest SCAs to the two interior nodes is necessary. First, table 1 is used to find out the minimum range of the SCAs, as the thick black lines in Fig. 10(c). Then algorithm 2 and 3 are used to find the nearest SCA boundary to D and E, respectively. The final path is $P \rightarrow D \rightarrow C \rightarrow E \rightarrow B \rightarrow G \rightarrow A$, as shown in Fig.10(d).

B. PATH PLANNING FOR MULTIPLE INTERIOR NODES NOT IN ANY SCA

This subsection will mainly design the rules for path planning under the background of multiple interior nodes not in any SCA, as shown in Fig.11.

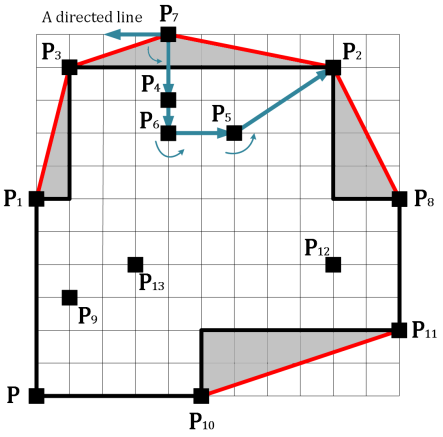


FIGURE 11. Path planning for multiple nodes classified to a same subpath.

If multiple interior nodes get the shortest distance from a same SCA boundary, then these nodes will be classified into a group. As Fig.11 shows, P_4 obtains the shortest distance from the minimum range of the SCA of $P_3 \rightarrow P_7$ and $P_7 \rightarrow P_2$, so it will be classified to these two subpath randomly. The rest nodes P_5, P_6, P_9, P_{12} and P_{13} are divided to their corresponding category.

Such as subpath $P_7 \rightarrow P_2$, a directed line is set from P_7. Then, the line is rotated as the way similar to algorithm 1 to find the LCP, until it interacts with the destination P_2. Note that, the LCP generated at this time is called the second-generation LCP and the LCP determined by all nodes for the first time is called the first-generation LCP. If there is any new interior node in this new polygon, algorithm 2 and 3 are used to find out the closest new SCA.

C. PATH FUSION OF THE TWO TYPES OF NODES ABOVE

In this subsection, we consider fusing the paths of the nodes in the same SCA and the nodes classified into this SCA. As shown in Fig.12(a), the red triangles are the first-generation SCAs and the gray areas represent the second-generation SCAs.

As Fig.12(b) shows, the gray lines represent the path planned for the nodes in the same SCA and the blue lines represent the path designed for the nodes classified into a same subpath. Note that, these straight lines between two nodes does not indicate the actual routes, just highlight the order of traversing nodes.

Next, two types of paths are considered to merge into one path. Firstly, we divide the plane into four parts by extending two right-angled sides of the triangle of SCA, as shown in Fig.13(a), where part $A_1 \supseteq \{X_1, X_2, X_3, Y_1, Y_2, Y_3, Y_4\}$,

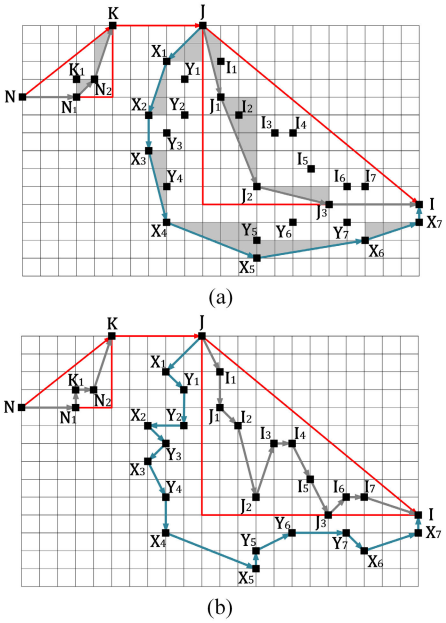


FIGURE 12. Generation of two types of paths.

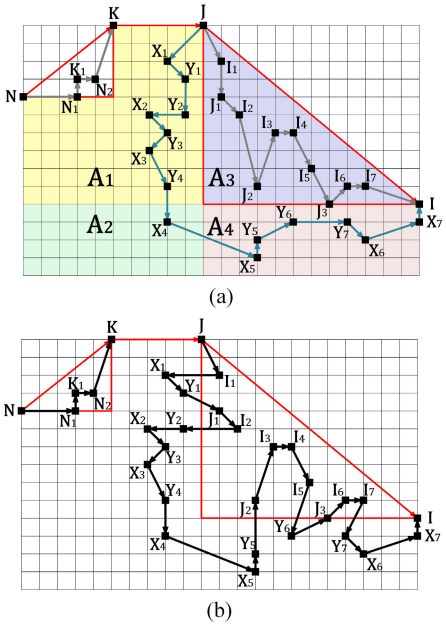


FIGURE 13. Path fusion process.

part $A_2 \supseteq \{X_4\}$, part $A_4 \supseteq \{X_5, X_6, X_7, Y_5, Y_6, Y_7, J_3\}$ and part $A_3 \supseteq \{I_1, I_2, I_3, I_4, I_5, I_6, I_7, J_1, J_2, J_3\}$.

Then the path starts from J and finds the nearest node I_1. Meanwhile, determine whether there are other nodes in region A_3 with the same y value as node I_1. If so, go to the node next step; If not, go to X_1 with the same y value as I_1. When selecting the next node, the node to be selected that is consistent with the region of the current node has a higher priority. Otherwise, continue searching the closest position

along with Y axis. The rest may be deduced by analogy and the complete path as shown in Fig.13(b). Notice a special case appears in the subpath $I_4 \rightarrow I_5 \rightarrow Y_6$. I_4 choose I_5 to be the next node rather than Y_6 that have the same x value with it. That is because that I_5 has the biggest x value than the remain nodes (i.e. I_5 is the farthest node from the bottom of the SCA, the path will never arrive at this level) and the length of $I_4 \rightarrow I_5 \rightarrow Y_6$ is shorter than $I_4 \rightarrow Y_6 \rightarrow I_5$. Having both conditions at the same time will go forward in this order.

VI. ALGORITHM COMPLEXITY ANALYSIS AND EXPERIMENTAL EVALUATION

ACO and GA are constantly considered classic algorithms to solve this problem, so they are chose to compare with our proposed algorithm on the performance of calculation time and path length, respectively. The performance of algorithms depends on the configuration of the computer and the programming language used. During the experiments, the configurations of the computer are Microsoft Windows 10, the CPU of intel(R) Core(TM) i7-8550U CPU @1.80GHz, the graphics card of NVIDIA GeForce MX150 and 8G RAM for simulation. All algorithms are implemented by python 3.7 programming.

Moreover, to verify the performance of the three algorithms on the number of small-scale positions, the experiments will be executed separately with 20, 30, 40 and 50 random nodes. Our test instances in the experiments are the random nodes on Cartesian coordinates rather than the instances on TSPLIB, because we want to observe the performances of our algorithm in this logistics warehouse grid model rather than the effect in a particular environment.

A. ALGORITHM COMPLEXITY ANALYSIS

Suppose the scale of the current TSP problem is n, i.e. there are n nodes in total. The proposed algorithm starts with finding the LCP. And the LCP search algorithm described in Algorithm 1 can be understood as every time find a vertex of LCP, all n nodes need to be traversed, so the time complexity of the proposed LCP search algorithm is $O(nh)$, where h is the number of the vertexes of LCP. If all nodes are vertexes of LCP, the time complexity of LCP searching algorithm is also the time complexity of the proposed algorithm, i.e. $O(n^2)$. In the meanwhile, the opposite of this situation is that the number of the vertices of LCP are found to be the smallest each time, that is, 4 vertexes.

As shown in Fig. 14, if the number of the vertexes of LCP is 4, then the plane can be divided into 5 parts, which are the SCAs composed of the four subpaths and an area between them. Suppose the remain $(n-4)$ nodes are evenly distributed in these 5 parts, so there are $(n-4)/5$ node in each part. Next, the process of finding new LCPs for the nodes in the first to fourth parts and the division of the nodes in the fifth part will be performed. At the same time, assume that the number of vertexes of the new LCP is also the smallest, i.e. 3 vertexes, so the time complexity so far is $O(4n)+O(((n-4)/5+2)\times 4)+O((n-4)/5)$. We treat the calculation so far as a loop. Next,

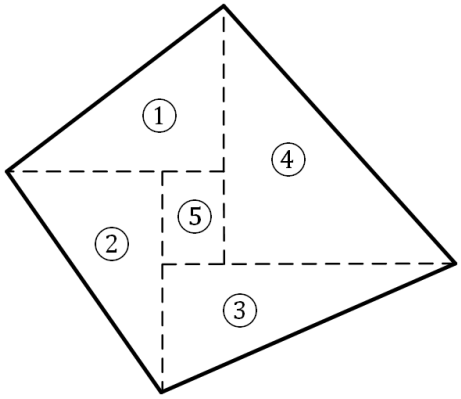


FIGURE 14. The general distribution of 4 LCP vertexes.

after adding some nodes to the partial solution S, the remain nodes will continue the process of finding LCP or division. Assuming the number of loops is L and L is a constant. After add all nodes are into S, the overall time complexity is

$$T(proposed) = L \times (O(4n) + O((\frac{n-4}{5} + 2) \times 4) + O(\frac{n-4}{5}) \approx O(n) \quad (9)$$

So the upper bound of the time complexity of the proposed algorithm is $O(n^2)$ and the lower bound is $\Omega(n)$.

We also analyzed the space complexity of the proposed algorithm. Firstly, we use S to store the nodes have been identified, and the initial S contains the vertexes of LCP, so the space complexity is $O(h)$. And if all the nodes are the vertexes of LCP, the overall space complexity of the proposed algorithm in this special case is $O(n)$. Next, the remain $(n-h)$ nodes will continue to search for LCP or be assigned to a subpath according to their location, and the space complexity of this step is $O(n-h)$. Simultaneously, several new subpaths will be generated. If there are no unprocessed nodes in the coverage area of a certain subpath, the subpath will not continue to be stored. Later, the next loop will start, and L also means the number of loops which is a constant. And as mentioned above, if all nodes are the vertexes of LCP, the overall space complexity is $O(n)$, so the space complexity of the proposed algorithm is

$$S(proposed) = L \times (O(h) + O(n-h)) \approx O(n) \quad (10)$$

B. LENGTH PERFORMANCE COMPARISON

From Fig.15, it can be seen that the proposed algorithm achieves a better performance than ACO and GA. GA has the worst performance in shortest path length and stability ratio. The reason is that each step of GA requires local optimization, which cannot guarantee global optimization. Besides, the optimization degree is also related to starting position. Note that, all length values are calculated from the nodes coordinate without any units. And the length of each segment of the grid line is 1.

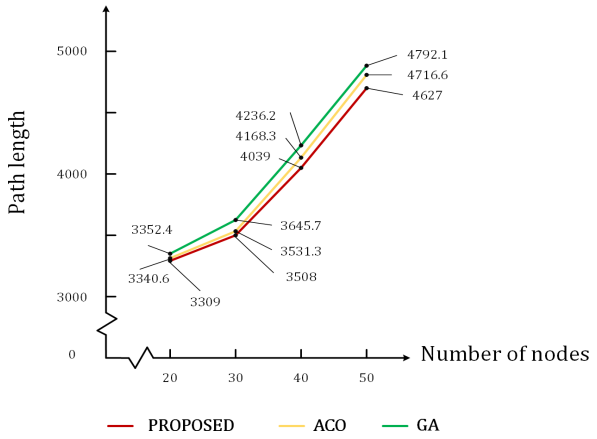


FIGURE 15. Length performance comparison.

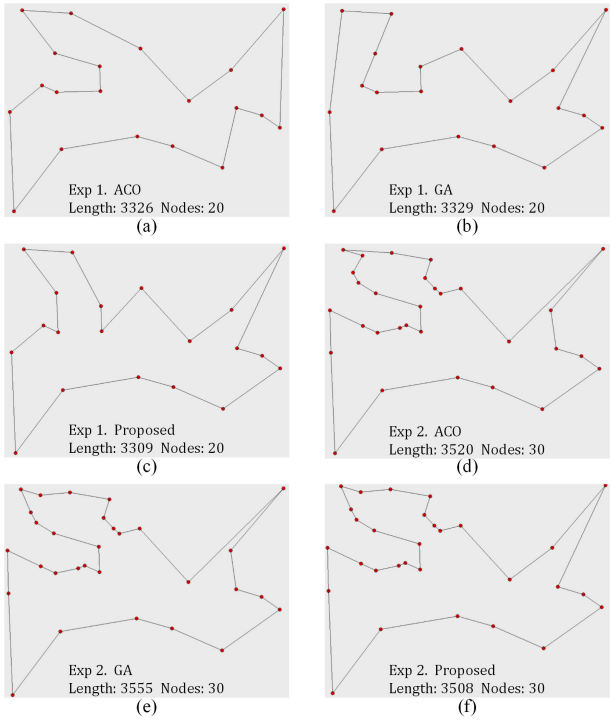


FIGURE 16. Length performance of the algorithms in Exp 1. and Exp 2.: (a) The most obtained solution with 3326 by ACO in Exp 1. (b) The most obtained solution with 3329 by GA in Exp 1. (c) The most obtained path with 3309 by our proposed algorithm in Exp 1. (d) The most obtained path with 3520 by ACO in Exp 2. (e) The most obtained path with 3555 by GA in Exp 2. (f) The most obtained path with 3508 by the proposed method in Exp 2.

Meanwhile, the local search ability of GA is poor, which results in consuming time. The search efficiency is lower in later stages of evolution, and it is easy to cause premature convergence problems. With a large number of positions, obtaining a satisfactory solution often requires an unbearably quantity of iterations.

However, ACO performs much better than GA on convergence, because it is not affected by the starting position and the ending position. When the number of positions is large, ACO is easy to fall into the non-optimal solution, and even the detours and deadlocks may appear. Moreover, if the parameters α and β are not set properly, the calculating efficiency is very slow, and the quality of obtained solution is particularly poor. In contrast, the proposed algorithm can maintain good length performance in small-scale number of random positions. The total traversal length is basically linear with the number of positions n.

Note that the paths in the simulation pictures show that the path between two positions is directly connected by a straight line instead of the grid lines, which is to observe the traversal order more intuitively. In fact, in the program we use the Manhattan distance to calculate the two-position distance.

1) EXP 1 (WITH 20 RANDOM NODES)

From the experiments of 20 random positions, it is difficult for ACO and GA to obtain the optimal solution. For ACO, the most obtained solution of path length that can be achieved with unlimited number of iterations is 3326 in Fig. 16(a). The path lengths obtained in many tests are even longer. Then for GA, the most obtained path length in Fig. 16(b) with 3329 is not as good as ACO, and it needs more iterations than ACO. The proposed method does not require iterative calculations, and it can obtain the shorter path length with 3309 in each test as shown in Fig. 16(c).

2) EXP 2 (WITH 30 RANDOM NODES)

Since the second experiment, the gap between the three algorithms has expanded. Not only ACO and GA is difficult to get the same length as the path obtained by the proposed method, but also many detours will appear in the path obtained as shown in Fig. 16(d)-(f).

3) EXP 3. (WITH 40 RANDOM NODES)

In exp 3, deadlocks often occur in the paths obtained by ACO and GA, that is, these two algorithms often fall into a local optimum, and regard the current suboptimal solution as the optimal solution. This is also the biggest drawback of these two algorithms as shown in Fig. 17.

4) EXP 4. (WITH 50 RANDOM NODES)

In Fig. 17, ACO can still not avoid deadlock and detour with 2000 iterations and even more, which is enough to see the disadvantage of ACO. With fewer iterations, the resulting path will be worse in the face of 50 random nodes. GA achieved the most obtained length performance similar to ACO with 4773 by 2000 iterations. At this time, GA resolved the detour that ACO had not resolved sometimes, but another larger detour appeared. In some cases, as the number of iterations increases, the performance of the GA even deteriorates.

Through the results of the experiments above, it can be seen that the proposed method can solve the problem of local optimum and detour in ACO and GA, respectively.

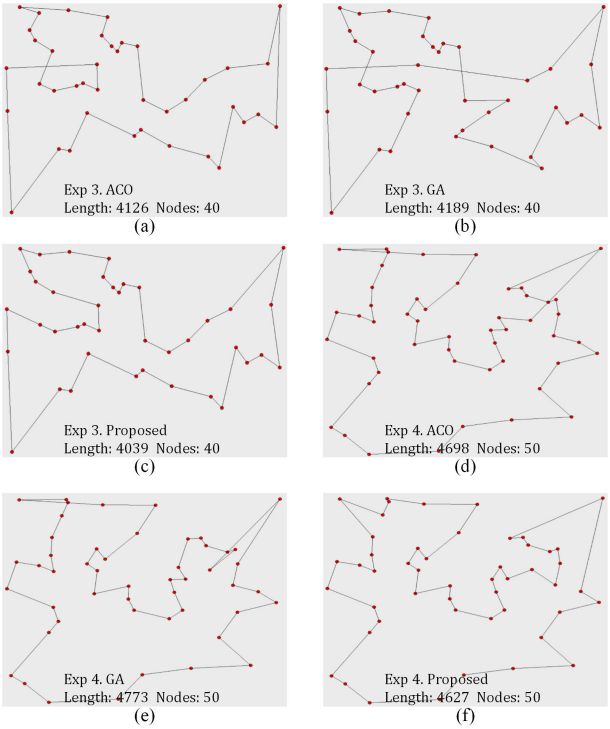


FIGURE 17. Length performance of the algorithms in Exp 3 and Exp 4): (a) The most obtained solution with 4126 by ACO in Exp 3. (b) The most obtained path with 4189 by GA in Exp 3. (c) The most obtained solution with 4039 by our proposed algorithm in Exp 3. (d) The most obtained solution with 4698 by ACO in Exp 4. (e) The most obtained path with 4773 by GA in Exp 4. (f) The most obtained solution with 4627 by our proposed algorithm in Exp 4.

C. TIME PERFORMANCE COMPARISON

The time optimization performance of each algorithm is shown in Fig. 18. The data is the average value of 100 experiments.

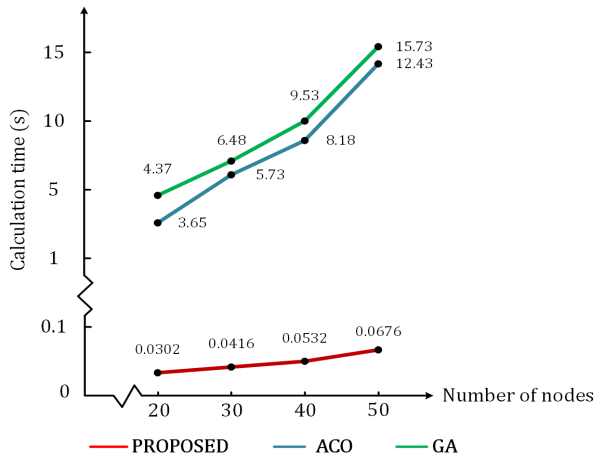


FIGURE 18. Time performance comparison.

It can be seen from Fig.18 that the ACO has a great advantage on the stability of operation results. However, the performance of ACO is determined by the pheromone. When the initial pheromone is lacking, it is easy to cause the algorithm to solve too slowly or fall into local optimum. In addition, if α and β are set improperly. The calculation speed is also slow, and the quality of resulting solution is particularly poor. In order to obtain a satisfactory result, ACO often requires a large number of iterations, but it is difficult to obtain the optimal solution similar to GA due to a limited number of iterations. The running time of the proposed algorithm is linear with n and has the shortest running time compared with others.

Note that, the execution time of an algorithm refers to the time it spends converging to the optimal solution at the first time. The data of all algorithms execution time is also the average of 100 experiments.

As can be seen from Fig.18, the execution time of the proposed method has been significantly optimized, which has been reduced by 120.86, 137.74, 153.76 and 183.87 times in 4 experiments compared with ACO and reduced by 144.70, 155.77, 179.14 and 232.69 times compared with GA.

VII. CONCLUSION

The paper is concerned with the path planning problem for robots under a given logistics warehouse environment. Two issues focused on in this paper are the calculation time of the algorithm and the driving path length. We propose a novel path planning algorithm for solving this type of robot path planning. Through extensive experimental evaluation, the results demonstrate the superiority of the proposed method compared with ACO and GA. In the future, we will focus on the path planning problem in 3D space, then the cooperation of multiple warehouse robots will also be investigated.

REFERENCES

[1] R. Bogue, "Growth in e-commerce boosts innovation in the warehouse robot market," *Ind. Robot, Int. J.*, vol. 43, no. 6, pp. 583–587, Oct. 2016.
[2] L. Zhou, J. Liu, X. Fan, D. Zhu, P. Wu, and N. Cao, "Design of V-Type warehouse layout and picking path model based on Internet of Things," *IEEE Access*, vol. 7, pp. 58419–58428, 2019.
[3] F. Duchoå, A. Babinec, M. Kajan, P. Beåo, M. Florek, T. Fico, and L. Juriäica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Eng.*, vol. 96, pp. 59–69, 2014.
[4] T. Zheng, Y. Xu, and D. Zheng, "AGV path planning based on improved A-star algorithm," in *Proc. IEEE 3rd Adv. Inf. Manage., Communicates, Electron. Autom. Control Conf. (IMCEC)*, Oct. 2019, pp. 1534–1538.
[5] M. Kusuma, Riyanto, and C. Machbub, "Humanoid robot path planning and rerouting using A-Star search algorithm," in *Proc. IEEE Int. Conf. Signals Syst. (ICSigSys)*, Jul. 2019, pp. 110–115.
[6] M. Nazarahari, E. Khanmirza, and S. Doostie, "Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm," *Expert Syst. Appl.*, vol. 115, pp. 106–120, Jan. 2019.
[7] J. Lee and D.-W. Kim, "An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph," *Inf. Sci.*, vol. 332, pp. 1–18, Mar. 2016.
[8] T. Oral and F. Polat, "MOD lite: An incremental path planning algorithm taking care of multiple objectives," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 245–257, Jan. 2016.
[9] C. Zhang, C. Hu, J. Feng, Z. Liu, Y. Zhou, and Z. Zhang, "A self-heuristic ant-based method for path planning of unmanned aerial vehicle in complex 3-D space with dense U-type obstacles," *IEEE Access*, vol. 7, pp. 150775–150791, 2019.

[10] X. Jiang, Z. Lin, T. He, X. Ma, S. Ma, and S. Li, "Optimal path finding with beetle antennae search algorithm by using ant colony optimization initialization and different searching strategies," *IEEE Access*, vol. 8, pp. 15459–15471, 2020.
[11] X. Yu, W. Chen, T. Gu, H. Yuan, H. Zhang, and J. Zhang, "Aco-a: Ant colony optimization plus a* for 3-d traveling in environments with dense obstacles," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 617–631, Aug. 2019.
[12] H. Yang, J. Qi, Y. Miao, H. Sun, and J. Li, "A new robot navigation algorithm based on a double-layer ant algorithm and trajectory optimization," *IEEE Trans. Ind. Electron.*, vol. 66, no. 11, pp. 8557–8566, Nov. 2019.
[13] H. Shuyun, T. Shoufeng, S. Bin, T. Minming, and J. Mingyu, "Robot path planning based on improved ant colony optimization," in *Proc. Int. Conf. Robots Intell. Syst. (ICRIS)*, May 2018, pp. 25–28.
[14] J. Liu, J. Yang, H. Liu, X. Tian, and M. Gao, "An improved ant colony algorithm for robot path planning," *Soft Comput.*, vol. 21, pp. 5829–5839, Oct. 2017.
[15] H. Wang, F. Guo, H. Yao, S. He, and X. Xu, "Collision avoidance planning method of USV based on improved ant colony optimization algorithm," *IEEE Access*, vol. 7, pp. 52964–52975, 2019.
[16] 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.
[17] G. Li and W. Chou, "Path planning for mobile robot using self-adaptive learning particle swarm optimization," *Sci. China Inf. Sci.*, vol. 61, no. 5, May 2018, Art. no. 052204.
[18] J.-J. Kim and J.-J. Lee, "Trajectory optimization with particle swarm optimization for manipulator motion planning," *IEEE Trans. Ind. Informat.*, vol. 11, no. 3, pp. 620–631, Jun. 2015.
[19] X. Yao, F. Wang, J. Wang, and X. Wang, "Bilevel optimization-based time-optimal path planning for AUVs," *Sensors*, vol. 18, no. 12, p. 4167, 2018.
[20] R.-J. Wai and A. S. Prasetia, "Adaptive neural network control and optimal path planning of UAV surveillance system with energy consumption prediction," *IEEE Access*, vol. 7, pp. 126137–126153, 2019.
[21] F. Xu, H. Li, C.-M. Pun, H. Hu, Y. Li, Y. Song, and H. Gao, "A new global best guided artificial bee colony algorithm with application in robot path planning," *Appl. Soft Comput.*, vol. 88, Mar. 2020, Art. no. 106037.
[22] M. A. Contreras-Cruz, V. Ayala-Ramirez, and U. H. Hernandez-Belmonte, "Mobile robot path planning using artificial bee colony and evolutionary programming," *Appl. Soft Comput.*, vol. 30, pp. 319–328, May 2015.
[23] C. Lamini, S. Benhlima, and A. Elbekri, "Genetic algorithm based approach for autonomous mobile robot path planning," *Procedia Comput. Sci.*, vol. 127, pp. 180–189, 2018.
[24] N. V. Kumar and C. S. Kumar, "Development of collision free path planning algorithm for warehouse mobile robot," *Procedia Comput. Sci.*, vol. 133, pp. 456–463, 2018.
[25] C. YongBo, M. YueSong, Y. JianQiao, S. XiaoLong, and X. Nuo, "Three-dimensional unmanned aerial vehicle path planning using modified wolf pack search algorithm," *Neurocomputing*, vol. 266, pp. 445–457, Nov. 2017.
[26] H. S. Dewang, P. K. Mohanty, and S. Kundu, "A robust path planning for mobile robot using smart particle swarm optimization," *Procedia Comput. Sci.*, vol. 133, pp. 290–297, 2018.

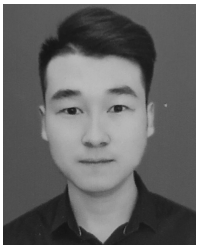
BO YANG received the M.Sc. degree from the College of Science and the Ph.D. degree from the Institute of Electrical Engineering, Yanshan University, Qinhuangdao, China, in 2009 and 2013, respectively. From 2018 to 2019, he was a Visiting Scholar with the Griffith School of Engineering, Griffith University, Gold Coast, QLD, Australia. He is currently a Lecturer with the School of Mathematics Sciences, Shanxi University, Taiyuan, China. His current research interests include distributed filter and visual servoing, indoor localization and their applications, and path-planning of mobile robot.

WENTAO LI was born in Datong, Shanxi, China, in 1996. He received the B.E. degree in automation from the University of South China, in 2017. He is currently pursuing the M.Sc. degree in control engineering with Shanxi University, Taiyuan, China. His research interests include robots path planning, big data, and computer vision.

JIANRONG WANG was born in Shanxi, China, in 1986. He received the M.S. degree from the North University of China, in 2013, and the Ph.D. degree from the University of Science and Technology Beijing, in 2017. Since 2017, he has been a Teacher with the School of Mathematics Sciences, Shanxi University. His research interests include complex system modeling and analysis and artificial intelligence.

JINGJIE YANG received the B.S. degree in information and computing science from Shanxi University, Taiyuan, China, in 2017, where she is currently pursuing the M.Sc. degree in control engineering. Her research interests include intelligent transportation systems, machine learning, and data mining.

TIANTIAN WANG received the B.E. degree in automation from Huaqiao University, Xiamen, China, in 2017. She is currently pursuing the M.Sc. degree in pattern recognition and intelligent systems from Shanxi University, Taiyuan, China. Her current research interests include set-membership filtering and indoor localization.



XIN LIU received the B.E. degree from the Henan University of Urban Construction, China, in 2018. He is currently pursuing the M.Sc. degree in control engineering with Shanxi University, China. His research interests include machine learning, deep learning, and image processing.