HIGH TECHNOLOGY LETTERS | Vol. 29 No. 3 | Sep. 2023 | pp. 279-287

doi:10.3772/j.issn.1006-6748.2023.03.006

A greedy path planning algorithm based on pre-path-planning and real-time-conflict for multiple automated guided vehicles in large-scale outdoor scenarios①

WANG Tengda(王腾达)*, WU Wenjun*, YANG Feng*, SUN Teng**, GAO Qiang②*

(*Faculty of Information Technology, Beijing University of Technology, Beijing 100124, P. R. China)
(**The 54th Research Institute of CETC, Shijiazhuang 050081, P. R. China)

Abstract

With the wide application of automated guided vehicles (AGVs) in large scale outdoor scenarios with complex terrain, the collaborative work of a large number of AGVs becomes the main trend. The effective multi-agent path finding (MAPF) algorithm is urgently needed to ensure the efficiency and realizability of the whole system. The complex terrain of outdoor scenarios is fully considered by using different values of passage cost to quantify different terrain types. The objective of the MAPF problem is to minimize the cost of passage while the Manhattan distance of paths and the time of passage are also evaluated for a comprehensive comparison. The pre-path-planning and real-time-conflict based greedy (PRG) algorithm is proposed as the solution. Simulation is conducted and the proposed PRG algorithm is compared with waiting-stop A* and conflict based search (CBS) algorithms. Results show that the PRG algorithm outperforms the waiting-stop A* algorithm in all three performance indicators, and it is more applicable than the CBS algorithm when a large number of AGVs are working collaboratively with frequent collisions.

Key words: automated guided vehicle (AGV), multi-agent path finding (MAPF), complex terrain, greedy algorithm

0 Introduction

Automatic guided vehicle (AGV) has been widely developed and applied with the rapid development of the electronic technology and control theory. As a highly intelligent mobile robot, AGV can realize safe, reliable and efficient transportation. Hence it can replace traditional manual transportation when necessary [1]. It arises in many real-world applications, such as warehouses[2], outdoor hazardous environments and office robots[3]. Further, driven by the demand for flexibility and efficiency in those scenarios, it is expected that more and more mobile robots will be deployed and work collaboratively. Therefore, path planning for multi-AGV cooperation is a key technology to support the applications.

Multi-agent path finding (MAPF) is a problem that computes a set of collision-free paths for multiple agents connecting their respective starting points and destination while optimizing certain measures of paths. Finding an optimal solution for MAPF problem is NP-hard because the state space grows exponentially with the number of agents[4]. To implement MAPF in real-world applications, research on efficient algorithms is of great significance. At present, the algorithms of multi-agent path finding problem proposed by researchers can be broadly classified into two categories: search-based algorithms and learning-based algorithms.

Generally, the search-based MAPF algorithms for AGV applications are based on heuristic search algorithms. They can be divided into two main categories: optimal path planning and approximate optimal path planning.

In optimal approaches, the standard admissible algorithm is proposed based on the A* algorithm for single agent path planning, and the robot team is considered as a composite agent with a very high dimension, which needs to find a solution for all agents[5]. However, it suffers from an exponential growth in planning complexity with the increase of the number of robots. On this basis, the technology of operator decomposition

① Supported by the National Key Research and Development Program of China (No. 2020YFC1807904).
② To whom correspondence should be addressed. E-mail: gaoqiang@bjut.edu.cn.
Received on Nov. 4, 2022

and independent detection is introduced. Their combination can optimally solve the relatively large problem in milliseconds [6]. Then, the standard admissible algorithm is extended to the M* algorithm [7] and its variants. The path is planned for each agent in advance. When the robot collides and blocks during path planning, the dimension of the search space will locally increase to ensure that an alternative path can be found. Because agents will coordinate only when it is necessary, the computing cost is greatly reduced. ODrM* algorithm[8] further reduces the number of agents requiring joint planning by decomposing the agents into independent conflict sets, and the operator decomposition method is also used to plan the path for each agent.

Unlike the above A*-based methods that transform the problem into a single joint agent model, the conflict based search (CBS)[9-10] and its variants are a tree search method. It plans for a single agent and constructs a set of constraints on nodes when illegal actions are detected to find the optimal solution without exploring high-dimensional space. In many cases, this reformulation enables CBS to examine fewer states than A*-based methods while still maintaining optimality. However, CBS needs to plan a complete path for all agents in advance, which has poor real-time performance and scalability. Besides, an alternate iterative conflict-based search (AICBS) algorithm is proposed, it saves search time by checking whether there is a conflict at each step of the extension to avoid invalid plans. However, there are multiple suboptimal plans which occupy system resources[11].

In approximate optimal path planning approaches, many researchers have made improvements also based on the A* algorithm. Cooperative A* (CA*) searches space-time for a non-colliding route. Hierarchical cooperative A* (HCA*) uses an abstract heuristic to boost performance. Windowed hierarchical cooperative A* (WHCA*) limits the space-time search depth to a dynamic window, spreading computation over the duration of the route[12]. Flow annotation replanning(FAR) implements a flow restriction idea inspired by road networks. The movement along a given row (or column) is restricted to only one direction, avoiding head-to-head collisions. The movement direction alternates from one row (or column) to the next. After building the search graph, an A* search is independently run for each mobile unit[13]. Some also divide the map into subgraphs with known structures, and then search in a smaller subgraph configuration space [14].

In recent years, with the rapid development of deep reinforcement learning, the learning-based distributed multi-agent path planning algorithm has emerged. Agents make decisions by inputting local observations into neural networks. In pathfinding via reinforcement and imitation multi-agent learning (PRIMAL)[15] and its variants [16], deep reinforcement learning and imitation learning are combined to alleviate the problem of low sample efficiency and provide intensive rewards for agents. However, imitation learning may lead to overfitting problems. A distributed multi-agent routing method based on deep reinforcement learning is proposed in Ref. [17], which uses local and global guidance mechanisms and combines course learning to help agents plan feasible paths.

At present, there has been an in-depth study of path planning algorithms, but there is still a lack of research on the MAPF in the new large-scale outdoor hazardous application scenario, such as mining areas, coking contaminated sites and natural disaster rescue sites. Different from the dominating multi-AGV cooperation scenarios, these scenarios have complex terrain, which includes ramps or pits besides obstacles. By refinement modeling of complex terrain, the accuracy and effectiveness of the paths planned by the algorithm can be improved. There are also a large number of agents in large-scale scenarios. Most graph-based search algorithms will fail when the number of AGVs is large due to the increase in computational complexity. Therefore, an effective path planning method is still needed to ensure the efficiency of the whole system.

In this paper, the complex terrain and the cooperation among a large number of AGVs in outdoor hazardous scenarios are fully considered. The pre-path-planning and real-time-conflict based greedy (PRG) algorithm is proposed to solve the multi-agent path finding problem. The algorithm is evaluated from three aspects, which are the cost of passage, the time of passage and the distance of path. Experiments comparing the proposed PRG algorithm, waiting-stop A* algorithm and CBS algorithm are presented. The contribution can be further summarized as following three points.

(1) The complex terrain of outdoor hazardous scenarios is fully considered. In the scenario model, different terrains are quantified into three categories, which are the impassable obstacle, the passable slopes and pits and the flat area. Different values of passage cost are defined for different terrain types. This makes the modeling more relevant to the actual scenario.

(2) The objective of MAPF is designed as minimizing the cost of passage, the time of passage and the distance of the path considering the characteristic of the complex terrain. As the outdoor area is not flat, the energy consumption and control complexity of pass-through different terrain types are different. Therefore,

in addition to the time of passage and the distance of the path, the cost of passage becomes an important factor in the evaluation of the effectiveness of MAPF algorithm. This is considered in the work for a more comprehensive comparison of different algorithms.

(3) PRG algorithm is proposed to solve the complex MAPF problem for a large number of AGVs in large outdoor scenarios. As more AGVs work collaboratively in the scenario, more conflicts will occur. This dramatically increases the complexity of MAPF problem. Hence, the existing algorithms may face the problem of efficiency or effectiveness. Taking both aspects into account, the pre-path-planning information which reduces the calculation cost of implementation and the real-time collision information which improves the effectiveness are used to design PRG algorithm.

The rest of this paper is organized as follows. The model of MAPF problem in large-scale outdoor hazardous scenarios is given in Section 1. In Section 2, the PRG algorithm is proposed to generate the optimal path. Then, the simulation results and analysis are discussed in Section 3. Finally, conclusions are drawn, and future work is discussed in Section 4.

1 Scenarios and models

1.1 Map model of the working area

In this paper, the applications which require a large number of AGVs cooperatively working in outdoor hazardous scenarios are considered, and MAPF problem for those AGVs is the main concern.

Inspired by the existing MAPF model, the map of the working area is modeled by grids. As shown in Fig. 1, the map with $M \times M$ grids is used. Different terrains in the working area are quantified into three categories, which are the impassable obstacle, the passable slopes and pits and the flat area. These terrains are respectively colored in Fig. 1. As the energy consumption and control complexity of pass-through different terrain types are different, the cost of passage is defined to represent the terrain type of the grid. The cost of passage of the grid in the i-th row and the j-th column is denoted by s_{ij}, and the value of s_{ij} is selected from $\{C_O, C_P, C_F\}$ representing the passage cost of the impassable obstacle, the passable slopes and pits and the flat area, respectively. Thus, the grid map of the working area is modeled as an $M \times M$ matrix.

$$\boldsymbol{S} = \begin{pmatrix} s_{00} & \cdots & s_{0M} \\ \vdots & \ddots & \vdots \\ s_{M0} & \cdots & s_{MM} \end{pmatrix}_{M \times M} \tag{1}$$

As only the unchanged terrain is modeled, $\boldsymbol{S}$ is also called the static map. Define the density of obstacle grids and the density of slopes and pits grids in the static map are ρ_O and ρ_P, respectively.

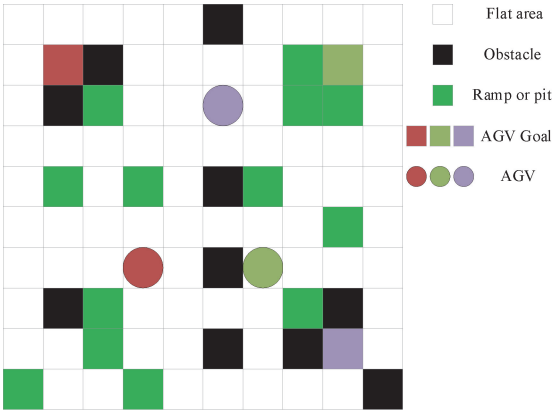

Fig. 1 Map of working area

Assume that there are N AGVs $U = \{U_1, \cdots, U_n, \cdots, U_N\}$ cooperatively working in the area, and the N pairs of start grids and goal grids $(P_n, Q_n)_{n=1,\cdots,N}$ are randomly selected from the girds representing flat area. It makes sure that each goal grid is reachable from its start grid in the static map, and there is no overlap among the $2N$ selected grids. As shown in Fig. 1, the circles are the starting grids of AGVs and the grids of the same color are the corresponding goals.

For ease of description, the system time is discretized into time steps. At each time step, AGVs move simultaneously to neighboring grids or wait at their current grids. If a grid is occupied by an AGV, it can be considered as an impassable obstacle for other AGVs. Therefore, for the grid in the i-th row and the j-th column at time step t, $a_{ij,t}$ is used to denote the occupancy. If the grid is occupied, $a_{ij,t} = C_O$. Otherwise, $a_{ij,t} = s_{ij}$. Thus, the real-time map of the working area is denoted by

$$\boldsymbol{A}_t = \begin{pmatrix} a_{00,t} & \cdots & a_{0M,t} \\ \vdots & \ddots & \vdots \\ a_{M0,t} & \cdots & a_{MM,t} \end{pmatrix}_{M \times M} \tag{2}$$

In practice, the environmental information can be continuously updated through sensors and broadcast to all the AGVs in the area. Hence, assume that each AGV can obtain both the static map and the real-time map.

1.2 Definition of MAPF problem

In this paper, the classical MAPF problem is considered. As the outdoor area is not flat, the energy consumption and control complexity of pass-through different terrain types are different. Therefore, in addition to

conventional path planning metrics which are the time of passage and the distance of path, the cost of passage becomes an important factor to evaluate the selected path. Hence, the optimization objective of the MAPF problem is defined as minimizing the cost of passage, the time of passage and the distance of the path.

At time step t, the AGV U_n can obtain a real-time graph $\boldsymbol{G}_{n,t} = (V_{n,t}, E_{n,t})$, in which $V_{n,t}$ is the set of vertexes indicating the grids, and $E_{n,t}$ is the set of weighted edges indicating the comprehensive cost between any two directly connected vertexes. According to the map model, there are four types of grids in $V_{n,t}$, which are free vertexes, obstacle or occupied vertexes, ramps or pits vertexes and goal vertex. Obviously, the goal vertexes for different AGVs are different. Hence, the real-time graphs obtained by different AGVs are different. Like most of the classic MAPF problems, four adjacent vertexes of each vertex are defined to have direct connections to the current vertex (except for vertexes at the edge of the area). The weights of edges are defined by different path-finding algorithms.

Based on $\boldsymbol{G}_{n,t}$, U_n selects to move to one of the adjacent vertexes or stay at the current vertex to avoid collision. Therefore, the MAPF problem can be mathematically defined as

$$d^*_{n,t} = F(\boldsymbol{G}_{n,t}) \tag{3}$$

where F is the path finding function, $d^*_{n,t} \in \{U, D, L, R, \varnothing\}$ is the action of U_n at time step t. If U_n arrives at it goal vertex, it will stay there until all the other AGVs arrive at their goal vertexes.

Further, to define the collision clearly, assume all the AGVs make routing decisions successively. A collision between AGVs is either a vertex collision or an edge collision. The vertex collision is represented by a tuple $\langle U_i, U_j, v, t\rangle$ which means AGV U_i decides to move to the vertex v but AGV U_j has already reached the vertex v at time step t. The edge collision is represented by a tuple $\langle U_i, U_j, u, v, t\rangle$ which means AGVs U_i and U_j traverse the same edge (u, v) in opposite directions at time step t. A solution to MAPF problem is a set of collision-free paths, each of which for each AGV.

2 PRG algorithm for MAPF problem

Finding an optimal solution for the MAPF problem is a typical NP-hard problem, which is difficult to be solved by traditional mathematical programming algorithms. Moreover, AGVs in large outdoor complex terrain scenarios need to make decisions in real-time, which requires the path planning algorithm to be implemented efficiently. Therefore, PRG algorithm is proposed in this paper.

In MAPF problem, the state of the area is changed at every time step due to the movement of AGVs. Hence, if the path from the current vertex to the goal vertex is calculated at each decision step, only the current step is optimal, and the rest of the path will not be optimal due to the change of state at each time step. Taking both the optimality and efficiency into account and inspired by the Bellman equation in reinforce learning, PRG algorithm combines the real-time-conflict-based accurate passage cost of the current step and the static-map-based estimation of the passage cost of the rest of the path to form the decision metric. Then, the current action is selected based on this decision metric using the greedy algorithm.

In the following, the three stages of PRG algorithm are described in detail. The first stage is estimating the static passage cost from each vertex to the goal vertex. The second stage is calculating the accurate passage cost of the current step using the real-time dynamic map. The third stage is conducting the decision metric and making the greedy-based decision.

2.1 Preprocess to estimate the static passage cost

As the static map $\boldsymbol{S}$ remains unchanged in the process of path planning, the static passage cost is pre-estimated based on $\boldsymbol{S}$ using Dijkstra algorithm. For AGV U_n, its goal vertex is Q_n. The static passage cost from all the vertexes in $\boldsymbol{S}$ to Q_n is recorded in a static matrix denoted by

$$\boldsymbol{B}_n = \begin{pmatrix} b_{00,n} & \cdots & b_{0M,n} \\ \vdots & \ddots & \vdots \\ b_{M0,n} & \cdots & b_{MM,n} \end{pmatrix}_{M\times M} \tag{4}$$

where $b_{ij,n}$ is the static passage cost from the vertex in the i-th row and the j-th column to the goal vertex Q_n. If the vertex in the i-th row and the j-th column is an obstacle, $b_{ij,n}$ is designed to be a very large value.

If U_n is located in i-th row and the j-th column at time step t, the estimated static passage cost is calculated as

$$C_{S,n,t}(d_{n,t}) = \begin{cases} b_{i-1,j,n} & d_{n,t} = U \\ b_{i+1,j,n} & d_{n,t} = D \\ b_{i,j-1,n} & d_{n,t} = L \\ b_{i,j+1,n} & d_{n,t} = R \\ b_{i,j,n} & d_{n,t} = \varnothing \end{cases} \tag{5}$$

2.2 Calculate the accurate passage cost of the current step

As all AGVs make routing decisions successively, the order is defined as $\{U_1, \cdots, U_n, \cdots, U_N\}$. Since the

decision of each AGV will change the state of the map, further define

$$\boldsymbol{A}_{t,n}=\begin{pmatrix} a_{00,t,n} & \cdots & a_{0M,t,n} \\ \vdots & \ddots & \vdots \\ a_{M0,t,n} & \cdots & a_{MM,t,n} \end{pmatrix}_{M\times M} \tag{6}$$

as the real-time map observed by U_n at time step t.

If U_n is located in i-th row and the j-th column at time step t, the accurate passage cost of the current step is calculated as

$$C_{\mathrm{R},n,t}(d_{n,t})=\begin{cases} a_{i-1,j,n,t} & d_{n,t}=U \\ a_{i+1,j,n,t} & d_{n,t}=D \\ a_{i,j-1,n,t} & d_{n,t}=L \\ a_{i,j+1,n,t} & d_{n,t}=R \\ C_{\mathrm{W}} & d_{n,t}=\varnothing \end{cases} \tag{7}$$

where, $d_{n,t}=\varnothing$ means U_n stay at the current location, and C_{W} is the corresponding cost.

2.3 Greedy-based decision

Combining the two kinds of cost together, the decision metric for U_n located in i-th row and the j-th column at time step t can be calculated as

$$C_{n,t}(d_{n,t})=w_{\mathrm{S}}C_{\mathrm{S},n,t}(d_{n,t})+w_{\mathrm{R}}C_{\mathrm{R},n,t}(d_{n,t}) \tag{8}$$

where w_{S} and w_{R} are the constant coefficients.

Based on the decision metric, the action for U_n at time step t is selected by

$$d_{n,t}^{*}=\underset{d_{n,t}\in\{\mathrm{U},\mathrm{D},\mathrm{L},\mathrm{R},\varnothing\}}{\operatorname{argmin}}\{C_{n,t}(d_{n,t})\} \tag{9}$$

2.4 Algorithm procedure

Based on the above analysis, the specific algorithm procedure is given in Algorithm 1.

Algorithm 1 The procedure of PRG algorithm

Input: the grid map with impassable obstacle, the passable slopes and pits and the flat area, and assign start and goal vertices to agents.

Initializes the static map $\boldsymbol{S}$ and the real-time map $\boldsymbol{A}_{0,1}$.
for $n\in[1,N]$, **do**
  Calculate static passage cost matrix $\boldsymbol{B}_n$ using Dijkstra algorithm
end for
for $t\in[0,T]$, **do**
  for $n\in[1,N]$, **do**
    if U_n reaches the goal vertex Q_n, **then**
      $d_{n,t}=\varnothing$.
    else
      Observe $\boldsymbol{A}_{t,n}$.
      Calculate $\{C_{n,t}(d_{n,t})\}_{d_{n,t}\in\{\mathrm{U},\mathrm{D},\mathrm{L},\mathrm{R},\varnothing\}}$ for all the possible actions using Eqs(7) and (8).
      Select the action $d_{n,t}^{*}$ using Eq. (9).
      Execute the action and update the real-time map $\boldsymbol{A}_{t,n+1}$.
      Calculate and record all the performance indicators of U_n at time step t.
    end if
  end for
  $\boldsymbol{A}_{t+1,1}=\boldsymbol{A}_{t,n+1}$.
  if all agents reach their goal vertexes
    break
  end if
end for

3 Results and analysis

To evaluate the performance of PRG algorithm, a simulation environment is conducted, and the waiting-stop A* algorithm[18] and CBS algorithm[9] are also implemented for comparison. Three performance indicators which are the time of passage, the distance of path and the cost of passage are compared. The detailed simulation setting and the results are given in the following.

3.1 Simulation setting

The simulation environment is Python 3.8. The configuration parameter of the PC is as follows. The processor is Intel(R) Core (TM) i5-9300H CPU @ 2.40 GHz 2.40 GHz. The memory is RAM 8.00 GB. The system type is a 64 bit operating system based on the X64 processor, and the operating system version is Windows 10. Simulation parameters are shown in Table 1.

Table 1 Simulation parameters

Parameter	Value
Map sizes M	$\{50,80\}$
Density of obstacles ρ_{O}	$\{0.05,0.10,0.15,0.20,0.25\}$
Density of slopes and pits ρ_{P}	0.1
Number of AGVs N	$\{16,32,\cdots,144\}$
Cost of passage $\{C_{\mathrm{O}},C_{\mathrm{P}},C_{\mathrm{F}},C_{\mathrm{W}}\}$	$\{160,3,1,3\}$
Coefficients in decision metric $\{w_{\mathrm{S}},w_{\mathrm{R}}\}$	$\{1,1\}$

A sample in the performance evaluation is considered as a map with randomly generated obstacles, ramps, pits and AGVs. The simulation parameters are given in Table 1. For each group of simulation parameters, 50 samples are generated and simulated to obtain convincing results.

3.2 Comparison algorithms

The proposed PRG algorithm is compared with the waiting-stop A* algorithm [18] and CBS algorithm [9] to reveal its advantages. The waiting-stop A* algorithm uses the A* algorithm to plan the shortest path for all AGVs in advance. When two AGVs are about to collide, the conflict is resolved by waiting or giving way. Specifically, when a vertex conflict occurs, that is, the next position of the current AGV has been occupied, AGV will choose to wait and then enter the next position. When an edge conflict occurs, that is, two AGVs are moving towards each other, then an AGV will randomly select an accessible adjacent vertex and give way to another AGV. CBS algorithm[9] consists of two layers of the search process, and the low-level searches an effective path for each AGV. The high-level search is responsible for checking path collisions and selecting the least costly branch to re-search the low-level path until the high-level search finds a valid path.

To facilitate the interpretation of simulation results, the complexity of the three algorithms is also analyzed in advance. The complexity indicator used here is the total number of floating-point operations (FLOPs) required to carry it out. As for the waiting-stop A* algorithm, its computational complexity is mainly determined by the A* algorithm. The time cost for the A* algorithm is $O(p^3)$, where $p = M^2$ is the number of vertexes in the map. Since it is necessary to plan a path for each AGV in advance, and the number of AGVs and p are of the same order of magnitude, the computational complexity of the waiting-stop A* algorithm is $O(p^4)$. The computational complexity of the proposed PRG algorithm is similar to the waiting-stop A* algorithm. It is mainly determined by the Dijkstra algorithm, which is a variation of the A* algorithm. Hence, the computational complexity is also $O(p^4)$. However, in CBS algorithm, the computational complexity of the high-level search process is $O(2^q)$, where q is the number of collisions encountered during the solving process of high-level search tree of CBS. The low-level search algorithm is A* algorithm, so the computational complexity of CBS is $O(2^q p^4)$, which means it is more difficult for CBS to compute results in real time especially when the number of AGVs is large.

3.3 Simulation results

First, the performance of the three algorithms is tested when the number of AGVs is fixed as $N = 25$ in the map of size 50×50. To test the adaptability of the algorithms to changes in the static environment, the density of obstacles varies as $\rho_0 = \{0.05, 0.1, 0.15, 0.2, 0.25\}$, which accordingly means the numbers of obstacles grids in the map are $\{125, 250, 375, 500, 625\}$. Under these situations, the paths between two connected vertexes in the static map are different. Besides, the probability of dynamic collisions increases with the increase in the density of obstacles due to the reduction in the passable vertexes although the number of AGVs is fixed. Results are shown in Figs 2, 3 and 4.

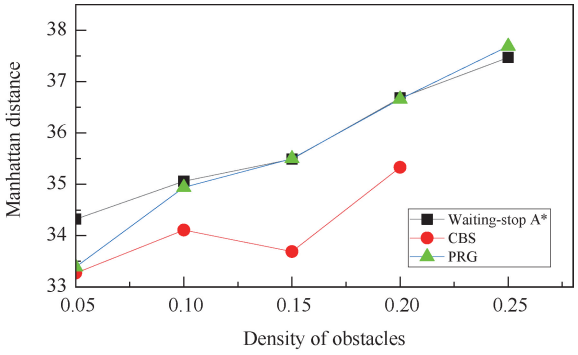

Fig. 2 Average distance of paths with various ρ_0

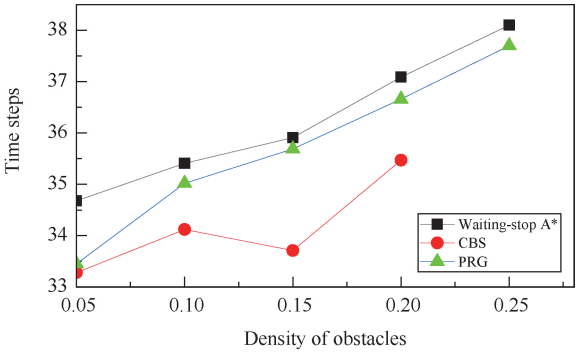

Fig. 3 Average time of passage with various ρ_0

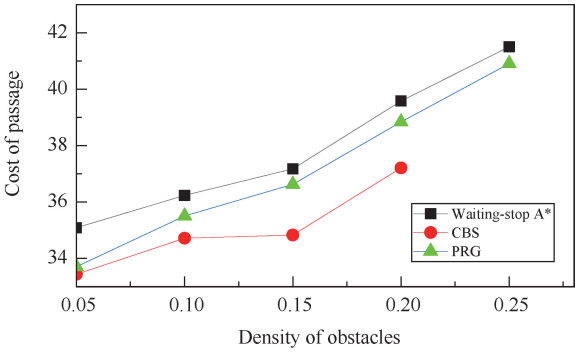

Fig. 4 Average cost of passage with various ρ_0

Obviously, as an optimal centralized path planning algorithm, CBS algorithm performs the best in all the three performance indicators evaluated here. However,

with the increased density of obstacles, the number of collisions encountered increased. When the density of obstacles reaches 0.25, CBS algorithm is no longer applicable in the simulation environment due to the extremely high computational complexity caused by the frequently triggered high-level search processes.

In contrast, PRG algorithm and the waiting-stop A* algorithm are less complex and easier to implement even when there are a large number of collisions. The two algorithms obtain a similar performance of the average distance of paths measured in Manhattan distance as both are based on the pre-searched path. However, they handle dynamic collisions differently. Results show that PRG algorithms can solve collisions more effectively and perform better in terms of the average time of passage and the average cost of passage. The performance gain obtained by PRG algorithm is between 1.0% and 4.0% when the value of ρ_0 varies.

To further evaluate the ability of PRG algorithm in handling dynamic collisions, the simulations with different numbers of AGVs in a map are conducted. The map size is 50×50 and $\rho_0 = 0.05$. The number of AGVs varies as $N = \{16, 32, 48, 64, 80, 96, 112\}$. Results are shown in Figs 5, 6 and 7.

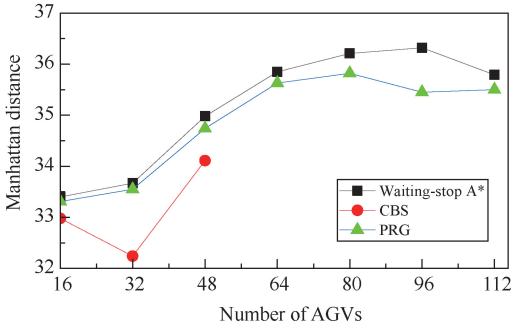

Fig. 5 Average distance of paths with different number of AGVs ($M = 50$)

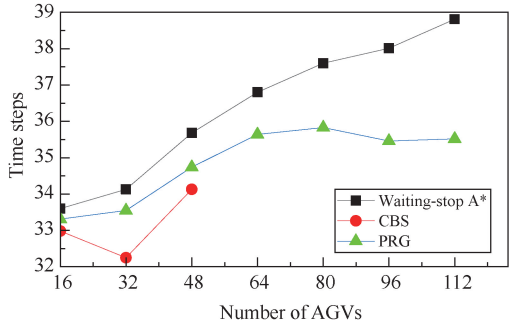

Fig. 6 Average time of passage with different number of AGVs ($M = 50$)

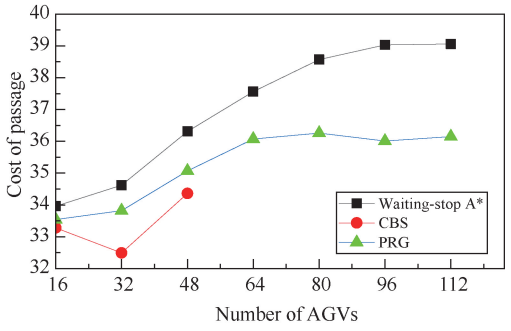

Fig. 7 Average cost of passage with different number of AGVs ($M = 50$)

Similar to the previous group of simulations, CBS algorithm performs the best in all the three performance indicators evaluated here. Unfortunately, when the number of AGVs is larger than 48, CBS algorithm is no longer applicable in the simulation environment due to the extremely high computational complexity. Existing research also confirmed that using CBS algorithm to compute an optimal solution when the number of AGVs is larger than 50 is often intractable[19]. This means CBS algorithm is not suitable for the scenario in which a large number of AGVs are working collaboratively.

Compared with the waiting-stop A* algorithm, PRG algorithm performs better in terms of the average distance of paths, the average time of passage and the average cost of passage, and the performance gains peaked at 2.4%, 8.5% and 7.5%, respectively. When collisions occur, PRG algorithm can make decisions based on the real-time map of the working area, which can reach the goal vertex more flexibly. Meanwhile, the waiting-stop A* algorithm resolves collisions only by two pre-defined rules which are waiting or giving way. Therefore, with the increase in the number of AGVs, collisions occur more frequently, and the performance gain of PRG algorithm is more obvious. This means PRG algorithm can well adapt to dynamic environments with frequent collisions and maintain a low implementation complexity.

The performance of PRG algorithm in the large working area is also evaluated. The map size is set as 80×80 with $\rho_0 = 0.05$. The number of AGVs varies as $N = \{16, 32, 48, 64, 80, 96, 112, 128, 144\}$. Results given in Figs 8, 9 and 10 show the same trends as that shown in Figs 5, 6 and 7. Thus, the effectiveness and efficiency of PRG algorithm are confirmed again.

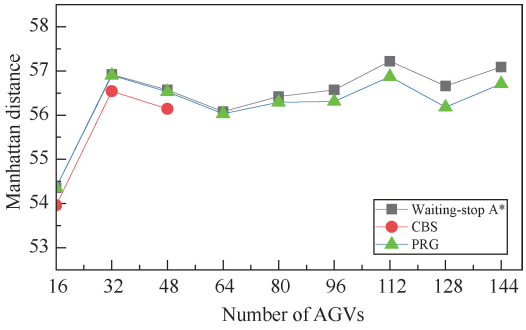

Fig. 8 Average distance of paths with different number of AGVs ($M = 80$)

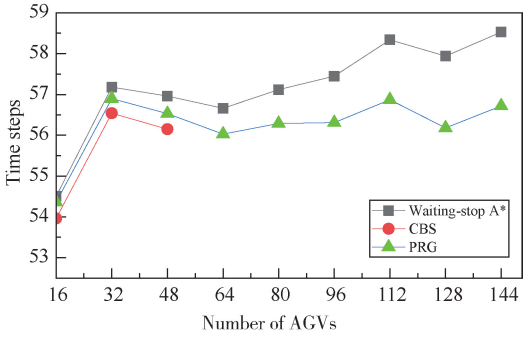

Fig. 9 Average time of passage with different number of AGVs ($M = 80$)

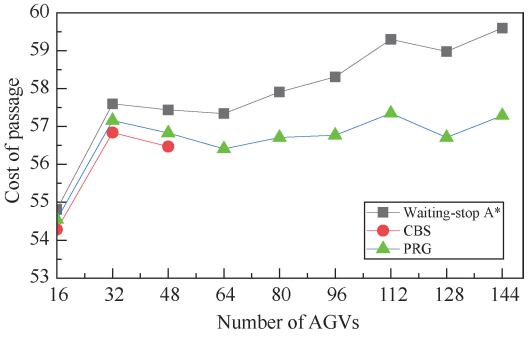

Fig. 10 Average cost of passage with different number of AGVs ($M = 80$)

4 Conclusions

This paper considers the MAPF problem for a large number of AGVs in outdoor hazardous scenarios with complex terrains and proposes the PRG algorithm combining the real-time-conflict-based accurate passage cost of the current step and the static-map-based estimation of the passage cost of the rest of the path to form the decision metric. The objective of MAPF problem is to minimize the cost of passage while the Manhattan distance of paths and the time of passage are also evaluated for comprehensive comparison. Simulation results show that PRG algorithm performs better than the waiting-stop A* algorithm in all three performance indicators evaluated in this paper. Compared with CBS algorithm, the performance of PRG algorithm is similar when the number of AGVs is small. When the number of AGVs is moderate, CBS algorithm performs the best. However, CBS algorithm is not applicable to the scenario in which a large number of AGVs are working collaboratively with frequent collisions. In all, PRG algorithm is a practical solution for MAPF problem in large-scale scenarios with a large number of AGVs. It obtains a good trade-off between complexity and performance. Its effectiveness and efficiency are confirmed through simulation.

In the future, multi-agent cooperative path planning methods based on deep reinforcement learning should be explored to further improve performance.

References

[1] LIAO X, WANG Y, XUAN Y, et al. AGV path planning model based on reinforcement learning[C]//2020 Chinese Automation Congress. Shanghai: CAC, 2020: 6722-6726.

[2] PINKAM N, BONNET F, CHONG N. Robot collaboration in warehouse[C]//2016 16th International Conference on Control, Automation and Systems. Gyeongju: IEEE, 2016: 269-272.

[3] WANG C, MAO J. Summary of AGV path planning[C]//2019 3rd International Conference on Electronic Information Technology and Computer Engineering. Xiamen: IEEE, 2019: 332-335.

[4] YIU Y, MAHAPATRA R. Multi-agent pathfinding with hierarchical evolutionary hueristic A[C]//2020 IEEE 3rd International Conference on Artificial Intelligence and Knowledge Engineering. Irvine: IEEE, 2020: 9-16.

[5] LAVALLE S, HUTCHINSON S. Optimal motion planning for multiple robots having independent goals[C]//Proceedings of IEEE International Conference on Robotics and Automation. Minneapolis: IEEE, 1996: 2847-2852.

[6] STANDLEY T. Finding optimal solutions to cooperative pathfinding problems[C]//Proceedings of the 24th AAAI Conference on Artificial Intelligence. Atlanta: AAAI Press, 2010: 173-178.

[7] WAGNER G, CHOSET H. Subdimensional expansion for multirobot path planning[J]. Artificial Intelligence, 2015, 219(2): 1-24.

[8] FERNER C, WAGNER G. CHOSET H. ODrM * optimal multirobot path planning in low dimensional search spaces [C]//2013 IEEE International Conference on Robotics and Automation. Karlsruhe: IEEE, 2013: 3854-3859.

[9] SHARON G, STERN R, FELNER A, et al. Conflict-based search for optimal multi-agent pathfinding[J]. Artificial Intelligence, 2015, 219(2): 40-66.

[10] SHARON G, STERN R, FELNER A, et al. Conflict-based search for optimal multi-agent path finding[C]//Proceedings of the 26th AAAI Conference on Artificial Intelligence. Toronto: AAAI Press, 2012: 563-569.

[11] TAO L, ZHANG S, CHEN S, et al. Multi-AGV pathfinding

for automatic warehouse applications[C]//2021 China Automation Congress. Beijing: CAC, 2021: 7194-7199.
[12] SILVER D. Cooperative pathfinding[C]// Proceedings of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference. Marina del Rey: AAAI Press, 2005: 117-122.
[13] WANG K, BOTEA A. Fast and memory-efficient multi-agent path finding[C]//The 18th International Conference on Automated Planning and Scheduling. Sydney: ICAPS, 2008: 380-387.
[14] RYAN M. Exploiting subgraph structure in multi-robot path planning[J]. The Journal of Artificial Intelligence Research, 2008, 31: 497-542.
[15] SARTORETTI G, KERR J, SHI Y, et al. PRIMAL: path-finding via reinforcement and imitation multi-agent learning [J]. IEEE Robotics and Automation Letters, 2019, 4(3): 2378-2385.
[16] DAMANI M, LUO Z, WENZEL E, et al. PRIMAL2: path-finding via reinforcement and imitation multi-agent learning-lifelong[J]. IEEE Robotics and Automation Letters, 2021, 6(2): 2666-2673.
[17] XU Y, LI Y, LIU Q, et al. Multi-agent pathfinding with local and global guidance[J]. 2021 IEEE International Conference on Networking, Sensing and Control, 2021, 1(1): 1-7.
[18] VAN DEN BERG J, OVERMARS M. Prioritized motion planning for multiple robots[C]// 2005 Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. Edmonton: IEEE, 2005: 430-435.
[19] WANG B, LIU Z, LI Q, et al. Mobile robot path planning in dynamic environments through globally guided reinforcement learning[J]. IEEE Robotics and Automation Letters, 2020, 5(4): 6932-6939.

WANG Tengda, born in 1999. He received the B. S. degree in Faculty of Information Technology from North China University of Technology in 2021, China. He is currently pursuing the M. S. degree in Faculty of Information Technology from Beijing University of Technology. His current research interests include path planning and deep reinforcement learning.