Combined priority and path planning through a double-layer structure for multiple robots

Haili Wang, Yunfan Li, Wenjun Jiang, Pengfei Wang, Qixin Cao

Abstract— **In this paper, we address the problem of both priority and path planning for multiple robots. Many relevant studies showed that priority assigned to robots has a significant impact on the planning results, such as whether a solution can be found or the global cost. We propose a double-layer structure that combines priority planning and path planning. A modified A* algorithm is introduced to get collision-free paths as path planner layer. A tree-like searching method that conducting in configuration space of priority is designed as priority planner layer, while the mode communicating with path planner layer is also defined to abstract heuristic information to guide searching direction. This new approach is implemented in simulation and represents good performance both in success rate and global paths cost.**

I. Introduction

For most situations, computing a collision-free path from one point to another point for a single robot is not a difficult task. But if referring to a multiple robot system, things will be much complex. Considering n robots sharing a common workspace and according to their task specification, the n planned path would be obtained independently. In order to avoid collisions with others, a robot may need to change its planned path, but what modification should be applied can give a better global solution is a tough job. This problem was early described as the multiple robot path coordination problem. Most existing methods to solve this problem can be classified into two categories, namely centralized and decoupled [1]. In a centralized approach, all robots usually would be regarded as an integral system using one common configuration space. Due to this, these methods have the ability to find the optimal global paths, though usually requiring a relevant extensive search and computation [2], [3]. On the contrary, decoupled approaches generate paths separately for individual robots and then resolve potential conflicts along with robots [4–6]. However, usually, these approaches only give an incomplete or sub-optimal solution. In many related works, the multiple robot system is regarded as a prioritized system [7–9]. In a general way, priority is one's attribute along in a group and a basis for the sequence of one's action in a group, which based on such standards as the importance and cost of one's specific task. The concept of priority often occurs in daily life. For example, private vehicles need to avoid emergency medical ambulances on urban roads. In some computer program scheduling methods, the temporary interrupt task can preempt the computing resources of the main program task at any time. In the logistics warehouse, depending on the importance of goods order, it is necessary to ensure that a forklift truck or an Automatic Guided Vehicle (AGV) carrying out specific order moves first. This paper focuses on the application and influence of priority in path planning of multi-mobile robots to obtain the optimal group execution cost or the shortest execution time. In stock, every robot will be given priority to identify how it or its task is essential, and robots with lower priority must avoid conflict to other robots [8], which have higher priority while traveling in some narrow environment. Therefore, most of situation priority configuration is a significant parameter, which plays a crucial role to the system's global planning result [9].

For a specific system and environment, there is an optimal priority configuration that leads to the optimal global solution. In order to obtain better priority configuration in this multi-robot path planning process, we borrowed some idea from the task and motion planning problem (TMP) in the research field of the robotic arm, which is often used to combine the declaration of a discrete task and continuous motion in one planning structure [11–15]. In this paper, we propose a TMP-like prioritized approach, of which the critical thought can be described as:

Multiple robot path planning problem is abstracted to a double-layer structure, which conducts both priority planning and path planning in two related searching space. One priority configuration would be regarded as a discrete state, which is obtained by a priority planner iteratively. While a path planner conducts path searching in the real map under each configuration. That is, if we have n robots, we would have n path planners within each priority planner. Every time after path planner comes out with a valid result, we estimate it and translate evaluation to the current state as feedback. The next iteration would try out the next planning from one state (with one priority configuration) in an order based on the value of the feedback. Via this iterative procedure, the double-layer planners not only ensure that each robot goes to its destination in the real map (if a valid plan exists) but also utilizes feedbacks such that an efficient solution can be found with the minimum time of searching.

Resrach supported by the National Key R&D Program, People's Republic of China (2017YFC0110502) and National Natural Science Foundation of China (61673261).

Haili Wang, Wenjun Jiang, Pengfei Wang and Qixin Cao are with State Key Laboratory of Mechanical System and Vibration, Shanghai Jiao Tong University, Shanghai, CO 200240 China (e-mail: sjtu_wanghaili@sjtu.edu.cn, wjjiang@sjtu.edu.cn, wpf790714@163.com, qxcao@sjtu.edu.cn).

Yunfan Li is with the Data Science Department, Fudan University, Shanghai, CO 201105 China (e-mail: 16302010002@fudan.edu.cn).

II. PRELIMINARIES

This paper defines the prioritized multiple robot path planning as $\mathscr{M} = < R,\ Q,\ S,\ S_{init},\ S_{goal},\ A,\ F,\ P >$, where

- R denotes the system of multiple robots, which consist of a set of robots as $\{r_1,\ r_2,\ \dots,\ r_n\}$, where n is the number of robots in system.
- Q denotes the set of priority patterns. A priority pattern $q \in Q$ is a specific robot sequence, which can be represented as $q = [r_{k_1},\ r_{k_2},\ \dots\ r_{k_n}]$, where $k_1,\ \dots\ k_n \in 1,\ 2,\ \dots\ n$. Generally, for n robots system, there are A_n^n priority patterns.
- S denotes the state space, in which a state s of all robots' positions usually is expressed as a set $s = \{pos_1,\ pos_2,\ \dots\ pos_n\}$. Here pos_i is the i^{th} robot's position coordinates on the map.
- S_{init} and S_{goal} denote the start state and the target state of the path planning process.
- A denotes a set of feasible path planning algorithms for the specific environment. Not a few path planning algorithms have been developed in past related works, for example, A*, RRT, D* algorithms.
- F denotes the type of feedback from the planned path, which is designed for communication between path planner and priority planner. It will be discussed in detail in the following article.
- P denotes the result paths for all robots in the system, which is a set $P = \{p_1,\ p_2,\ \dots\ p_n\}$. A $p_i \in P$ is a collision-free path from the start position to the target position as $p_i\colon\ s_{init}^i \rightarrow s_{goal}^i$.

III. METHOD

To provide a collision-free and efficient path for all robots from their start position s_{init} to the target position s_{goal}, which is a mentioned prioritized path planning problem, a new approach will be discussed in this paper. The method would conduct searching both in priority configuration Q and state space S based on a double-layer planner structure.

In other words, a whole searching construct can be divided into two main layers, namely upper planner - priority planner and lower planner - path planner. For this two-layer searching construct, the critical point to achieve it is to realize the communication between priority planner and path planner. However, the problem is that priority planning is discrete, while path planning, unfortunately, is continuous. On the one hand, priority planner hard considers the underlying continuous, physical constraints while searching in the discrete priority configuration Q, for example, obstacles. On the other hand, the path planner also has difficulty using the upper layer's information to guide its lower-layer path searching, although that information theoretically can optimize the search result.

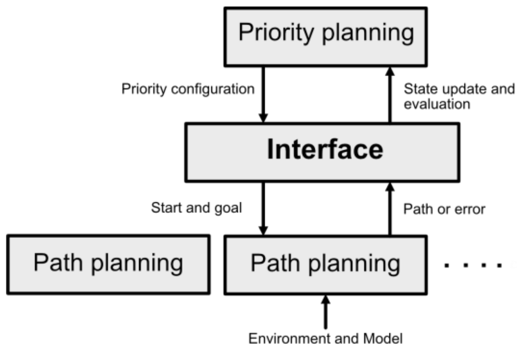


Figure 1. Interface in the proposed approach.

We design an interface (Fig. 1) between the upper priority planner and lower path planner, to enable communication between priority planning and path planning in the searching process. This interface extracts a task defined feedback, and periodically translates it to the upper priority planner, which would guide the search to the end iteratively. In this way, our approach can obtain not only optimal paths but also an optimized priority configuration.

Therefore, in state space S, we maintain the search data structure as a tree $\mathcal{T}(V,\ E)$. Each vertex $v \in \mathcal{T}(V)$ is associated with a state $s \in S$, written as $v.s$. An edge $e(v',\ v'') \in \mathcal{T}(E)$ indicates one iterative planning process conducted by our approach with a specific priority configuration. Throughout the planning process, the tree $\mathcal{T}$ explores the state space until reaching the goal state. The initial state s_{init} of robots is given as root of the tree $v_{root}.s_{init}$, and the goal state s_{goal} is the final state in state space that is needed to reach via tree's grow.

Our approach explores among the state space iteratively. The function $SelectLeaf(\mathcal{T})$ selects a vertex $v_{check} \in \mathcal{T}(V)$, which is the next vertex to expand. The function $GrowTree(V_{check},\ q)$ generates new vertices V_{new} and adds them to the tree $\mathcal{T}$. (q here denotes a priority configuration.)

A pseudocode is given in Algo. 1. Details of the main sub-functions in our approach follow.

A. *Path planning*

The function $SearchingPath(r_i,\ P,\ \lambda)$ searches a collision-free path for robot r_i with a limited length. Hence usually, it may not reach the robot's goal position but be a pose on the whole path. In this way, our searching is discrete, and the entire path would be obtained as an iterative process.

Many path planning methods for single robot path planning were developed in related works. Therefore, our approach can take advantage of these researches and employ them as a plug-in our algorithm. However, the related position with other prior robots should be considered. More specifically, other prior robots should be regarded as a dynamic obstacle for other lower prior robots.

Algorithm 1 Priority and path planning algorithm

Input: problem information, such as robots' start and goal positions; searching time limit; step size in search

Output: a set of collision-free paths not only with obstacles but also with each others for all robots in system; NULL if no solution can be found

```
1: ♢ initialize data structures
2: $\mathcal{T} \leftarrow \emptyset$; $V_{init} \leftarrow \emptyset$
3: $S_{init} \leftarrow start\ state$; $S_{goal} \leftarrow goal\ state$
4: AppendNode($\mathcal{T}.root$, $V_{init}$)
5: ♢ core loop: searching in both priority set and state space alternately with a evaluate value as interactive imformation
6: while $ElapsedTime < t_{max}$ do
7:     $V_{check} \leftarrow SelectLeaf(\mathcal{T})$
8:     for each $q \in Q$ do
9:         $V_{new} = GrowTree(V_{check},\ q)$
10:        for each $r_i \in q$ do
11:            $p_i \leftarrow SearchingPath(r_i,\ P,\ \lambda)$
12:            $P.append(p_i)$
13:        end for
14:        if $\forall\ s^i_{goal} \in p_i$ then
15:            $V_{back} = V_{new}$
16:            goto GetPath;
17:        end if
18:        $V_{new}.evaluate \leftarrow Evaluate(P)$
19:    end for
20:    GetPath:
21:    while $V_{back}.parent \neq V_{init}$ do
22:        for each $p_i \in V_{back}.P$ do
23:            $p_i.append(V_{back}.parent.p_i)$
24:        end for
25:        $V_{back} = V_{back}.parent$
26:    end while
27:    return Path
28: end while
29: return NULL
```

In this paper, a modified A* algorithm that is similar to A* but considering conflict with other robots is used. The main modification of the new algorithm is:

- It has a limited length λ of result path.
- It regards adjacent point $MapPoint_{p \rightarrow s}(pathpoint)$ that is contained in the prior robot's path in one iteration as inviable.

These modifications allow the proposed algorithm can be conducted iteratively and, after every iteration, return a evaluation about the short, planned path. Its pseudocode is given in Algo. 2.

This method was tested on a very narrow map, in which just two passageway robots can pass through. Two robots were placed at both ends of the passageway, respectively. In addition, one has a start and goal position as the other's goal and start position on the contrary. The result of this test can be seen in Fig. 2.

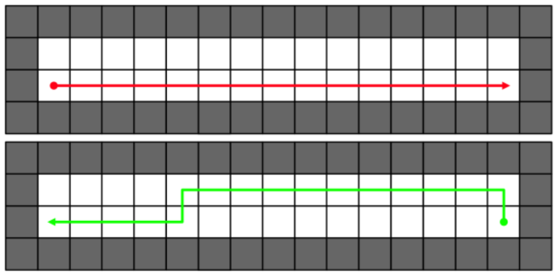


Figure 2. Result of path planning test.

Algorithm 2 SearchingPath

Input: a robots and its current positon, goal positon; a path set that contains all paths that other prior robot have planned; a specific path planning algorithm for single robot

Output: a new path for the current robot, which points to robot's goal but has a limit length.

```
1: ♢ initialize
2: $s_{init} \leftarrow current\ positon$; $s_{goal} \leftarrow goal\ position$
3: $P \leftarrow path\ set$; $\lambda \leftarrow limit\ length$
4: $A \leftarrow algorithm$
5: ♢ core procedure
6: for each $p \in P$ do
7:     for each $pathpoint\ \in p$ do
8:         $MapPoint_{p \rightarrow s}(pathpoint).viable = false$
9:     end for
10: end for
11: $result \leftarrow A(s_{init},\ s_{goal},\ \lambda)$
12: return result
```

B. Result evaluation

The evaluation of one iteration is designed to represent, from which state it will have more possibility to reach the goal position, and it better can work efficiently in practice. Concretely, this evaluation can be the total length of all planned path for every robot, and also can be determined by the time when the last robot completes its task. In short, it should be described by the specific environment and task request. In our approach $Evaluate(P)$ is obtained as

$$Evaluate(P) = \frac{1}{\sum_{i=1}^{n} |p_i.pathend - p_i.goal|},$$

where P denotes the set that contains all planned paths for robots and *pathend* denotes the last point of short path in one iteration. Apparently, this evaluation describes the sum of all robots' remaining distances to their goals. The function would be invoked every time after an iterative search to provide feedback(evaluation). Robots may find a set of paths to their targets more quickly if choosing to expand from a priority state with a minimal evaluation.

C. Priority planning

In our approach, priority planning is a tree search. Throughout the planning process, a tree-like data structure $\mathcal{T}$

is maintained to form the search. The initial state S_{init} of robots is given as root of the tree V_{root}, and the goal state S_{goal} is the final state in state space that is needed to reach via tree's grow.

The priority planner explores the state space. The function $SelectLeaf(\mathcal{T})$ selects a vertex $V_{check} \in \mathcal{T}$, the next vertex to expand. The function $GrowTree(V_{check}, q)$ generates new vertices V_{new} and adds them to the tree $\mathcal{T}$. (q here denotes a priority configuration.)

Every vertex $V \in \mathcal{T}$ has a member variable $Evaluate(P)$, expressing how good a vertex is as described hereinbefore. During involving function $SelectLeaf(\mathcal{T})$, a vertex V_{check} is selected if it has a minimal $Evaluate(P)$ along all $V \in \mathcal{T}$. Or using a more probabilistic method a vertex V_{check} is then chosen with probability proportional to its evaluation,

$$Prob(V) = \frac{V.evaluate}{\sum_{V' \in \mathcal{T}} V'.evaluate}.$$

In this way, vertex V associated with high evaluation has a greater chance of being selected.

Every vertex V_{new} is a path planner that has been described hereinbefore. From the perspective of breadth grow, in one iteration, the variety of priority configurations determines the number of vertices. Generally, for n robots system, there are A_n^n priority configurations, that is, in every iteration, $A_n^n \times V_{new}$ would be inserted as children to V_{check}. Then from the perspective of depth grow, new vertex V_{new} would inherit its parent vertex's state. V_{new} would take the current position of V_{check} as its starting point and generate a path accordingly. The function $GrowTree(V_{check}, q)$ is represented in Fig. 3 to make it easier to understand.

Till now, our searching tree $\mathcal{T}$ can grow from state space S until a new vertex V_{end} that satisfies the goal state S_{goal}, before an integrated path is generated by connecting the segmental path from the last vertex V_{end} to the root vertex V_{root}.

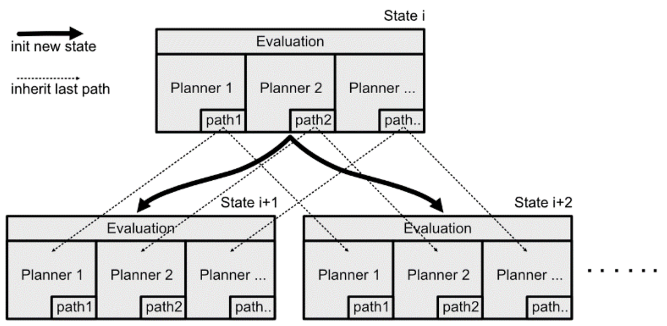


Figure 3. Grow of tree.

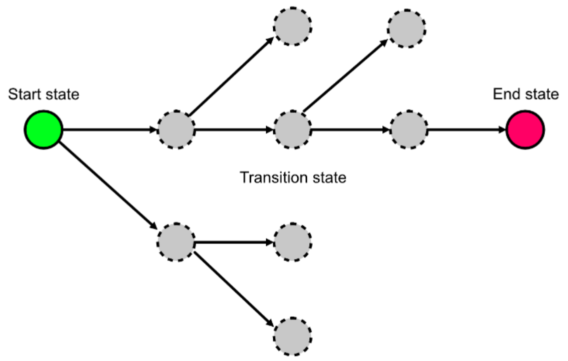


Figure 4. Heuristic searching process.

However, the most important thing is that thanks to the evaluation feedbacks from every vertex V, which plays as heuristic information, we no longer need to search the whole state space S. This heuristic searching process is illustrated in Fig. 4.

IV. Simulations and Results

A. Grid map configuration

The proposed approach is tested in a grid map environment to search for a global path solution for a multi-robot system. An illustration is provided in Fig. 5 for describing grid map configuration, which with n row m column and is configured, including free position, robot, start, goal, and obstacle.

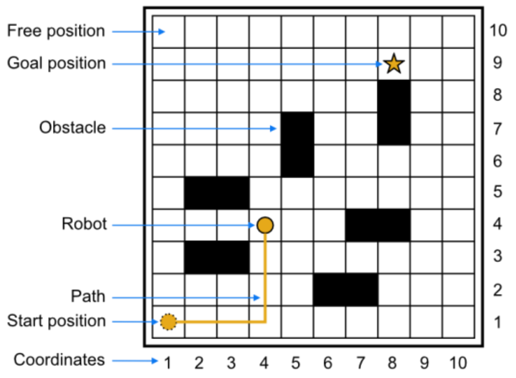


Figure 5. Configuration of grid map with 10 $\times$ 10 in scale.

B. Simulation 1: Priority's influence on the existence of a solution

In some case, the priority of robots in a system can have a severe impact on whether a global solution can be found or not. For instance, Fig. 6 shows a case in which robot 1 starts from position (1,1) to its goal position (9,1) while another robot 2 from position (5,4) aims to goal position (5,1). Since goal position of robot 2 locates in robot 1's path, which is easily obtained and unique as the yellow line way, if robot 2 has a higher priority, it would get in the yellow way without considering robot 1 in advance and lead to failure result. Robot 1 would treat robot 2 as a dynamic obstacle and then be stuck in position (4,1). On the contrary, robot 1 passes

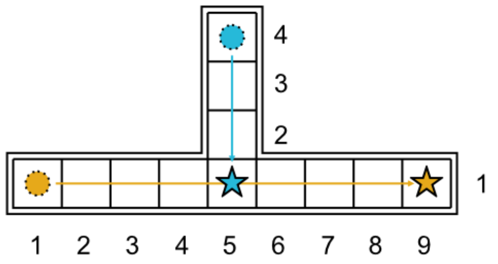


Figure 6. The environment of simulations 1.

through robot 2's goal position in advance with a higher priority and robot 2 can wait at neighboring position (5,2), then still arrive at its goal position after robot 1.

Our approach described earlier has been tested in this situation. And the result in Table I proves that feasible paths of two robots can be obtained without specifying any priority information via our approach.

TABLE I. PLANNED PATHS OF SIMULATION 1

Start and Goal	Planned paths
R1: (1,1)-(9,1) R2: (5,4)-(5,1)	(1,1), (2,1), (3,1), (4,1), (5,1), (6,1), (7,1), (8,1), (9,1) (5,4), (5,3), (5,2), (5,2), (5,2), (5,1), (5,1), (5,1), (5,1)
R1: (5,4)-(5,1) R2: (1,1)-(9,1)	(5,4), (5,3), (5,2), (5,2), (5,2), (5,1), (5,1), (5,1), (5,1) (1,1), (2,1), (3,1), (4,1), (5,1), (6,1), (7,1), (8,1), (9,1)

C. Simulation 2: Path planning for multiple robots with optimal priority

Our approach has also been tested and implemented in a further complicated situation, and the results obtained below show an excellent performance. To evaluate the general applicability of our approach, the tested environment is developed as a small warehouse. The warehouse is modeled as a grid map with 10 rows 10 columns in scale as Fig. 7, in which 5 robots move in total. 4 of them need to swap their position in pairs, and then another one needs to arrive at the left-up corner, while usually would pass through other's planned paths.

Fig. 8(a), 7(b), 7(c) represent one set of planned paths for all robots in each searching iteration (limited length is 7). Some features can be observed to analyze the proposed

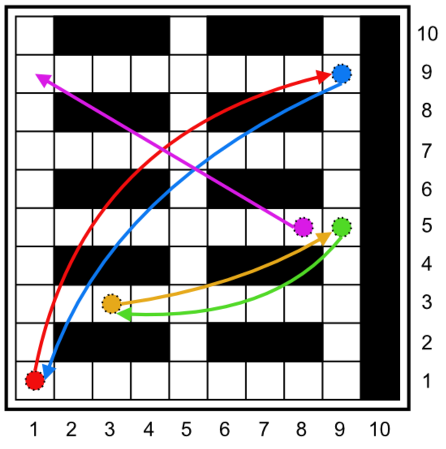
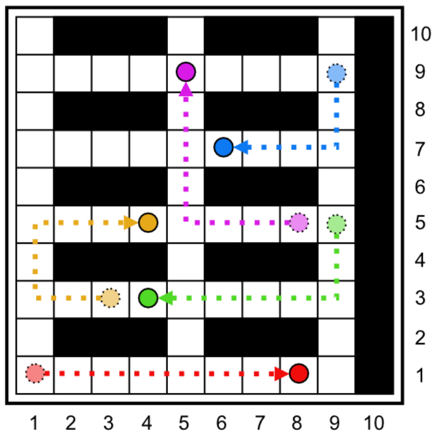
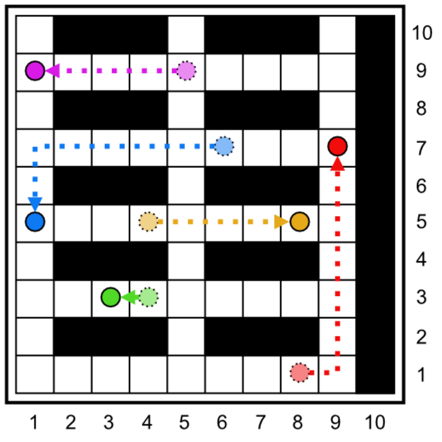


Figure 7. Evironment of simulation 2.


(a) Planned paths in 1st iteration.


(b) Planned paths in 2nd iteration.

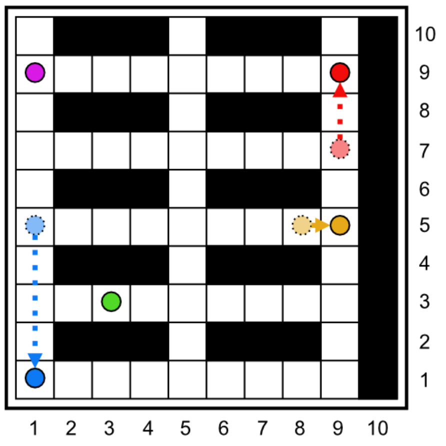

(c) Planned paths in 3rd iteration.

Figure 8. Results of simulation 2.

approach. In 1st iteration, yellow robot chooses a sub-optimal path to avoid collisions with others while achieved a lower global cost. On the other hand, sometimes, robots pause at the place to ensure lead to searching failure. All of these were realized without any given priority information, and any deadlock occurred. Every robot arrives in its goal position within 20 steps, and searching time is also a short cause of our heuristic priority search method.

V. Conclusion

In this paper, we presented an approach to combine both priority configuration planning and paths planning for groups of mobile robots. The approach has a double-layer structure, which ensures the communication between the two different planners. It can assign a better priority to robots and also obtain a more optimal solution without any given priority information. Additionally, with feedback from the path planner, which acts as heuristic information, the priority planner would be guided to reduce the computation need. The approach has been implemented and tested in simulations for both two and several robots. The simulation results show that this technique can decrease the rate of failure and shorten the total length of paths.

One idea for future work is improving the path planning algorithm to adapt to iteratively searching, for instance, using some modified RRT. Another optimizing direction is utilizing the specific priority information of a particular situation. In fact, sometimes robots have original priority features, like urgency. This information should be considered in the planning algorithm in future work.

References

[1] J. Barraquand and J.-C. Latombe, "Robot motion planning: a distributed representation approach," *Int. J. Robot. Res.*, vol. 10, no. 6, pp. 628–49, Dec. 1991, doi: 10.1177/027836499101000604.

[2] M. Cirillo, F. Pecora, H. Andreasson, T. Uras, and S. Koenig, "Integrated Motion Planning and Coordination for Industrial Vehicles," in *Twenty-Fourth International Conference on Automated Planning and Scheduling*, Palo Alto: Assoc Advancement Artificial Intelligence, 2014, pp. 463–471.

[3] H. Andreasson *et al.*, "Autonomous Transport Vehicles Where We Are and What Is Missing," *Ieee Robot. Autom. Mag.*, vol. 22, no. 1, pp. 64–75, Mar. 2015, doi: 10.1109/MRA.2014.2381357.

[4] R. Olmi, C. Secchi, and C. Fantuzzi, "An Efficient Control Strategy for the Traffic Coordination of AGVs," in *2011 Ieee/Rsj International Conference on Intelligent Robots and Systems*, New York: Ieee, 2011, pp. 4615–4620.

[5] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Hierarchical Traffic Control for Partially Decentralized Coordination of Multi AGV Systems in Industrial Environments," in *2014 Ieee International Conference on Robotics and Automation (icra)*, New York: Ieee, 2014, pp. 6144–6149.

[6] I. Draganjac, D. Miklic, Z. Kovaci, G. Vasiljevic, and S. Bogdan, "Decentralized Control of Multi-AGV Systems in Autonomous Warehousing Applications," *Ieee Trans. Autom. Sci. Eng.*, vol. 13, no. 4, pp. 1433–1447, Oct. 2016, doi: 10.1109/TASE.2016.2603781.

[7] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *robot. Auton. Syst.*, vol. 41, no. 2–3, pp. 89–99, Nov. 2002, doi: 10.1016/S0921-8890(02)00256-7.

[8] J. P. van den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 430–435, doi: 10.1109/IROS.2005.1545306.

[9] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, 2001, vol. 1, pp. 271–276.

[10] L. P. Kaelbling and T. Lozano-Perez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 1470–1477, doi: 10.1109/ICRA.2011.5980391.

[11] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014, pp. 639–646, doi: 10.1109/ICRA.2014.6906922.

[12] E. Plaku and G. D. Hager, "Sampling-Based Motion and Symbolic Action Planning with geometric and differential constraints," in *2010 IEEE International Conference on Robotics and Automation*, Anchorage, AK, 2010, pp. 5002–5008, doi: 10.1109/ROBOT.2010.5509563.

[13] A. Akbari, Muhayyuddin, and J. Rosell, "Knowledge-oriented task and motion planning for multiple mobile robots," *J. Exp. Theor. Artif. Intell.*, vol. 31, no. 1, pp. 137–162, Jan. 2019, doi: 10.1080/0952813X.2018.1544280.

[14] M. Gorner, R. Haschke, H. Ritter, and J. Zhang, "MoveIt! Task Constructor for Task-Level Motion Planning," in *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada, 2019, pp. 190–196, doi: 10.1109/ICRA.2019.8793898.