2nd International Conference on Autonomous Robots and Agents
December 13-15, 2004 Palmerston North, New Zealand

Priority based Dynamic Multiple Robot Path Planning

Taixiong Zheng
Department of Automation
Chongqing University of Post and Telecommunication, China
zhengtaixiong@163.net

D. K. Liu
ARC Centre of Excellence for Autonomous Systems (CAS), Faculty of Engineering
University of Technology, Sydney, Australia
dkliu@eng.uts.edu.au

Ping Wang
Department of Automation
Chongqing University of Post and Telecommunication, China
wangping@cqupt.edu.cn

Abstract

Path planning has been a topic of research in multiple robot coordination and control. The purpose of this paper is to investigate the path planning problem of a group of robots with communication between each other to perform a set of prioritized tasks in a dynamic environment. Unlike traditional path planning approaches that a path does not change once the path has been planned, the approach investigated in this paper plans each robot's path dynamically, i.e., the path will change according to the environment. When one robot starts to perform or finishes a task, it broadcasts the task's start position or goal position to other robots. Then other robots will change their paths if the broadcast position has influence on their planned paths.

Keywords: multiple robots, path planning, collaborative control

1 Introduction

Path planning is one of the fundamental problems in multiple mobile robotic systems. Various algorithms have been proposed and investigated to solve this problem. These algorithms can be roughly categorized as centralized and decentralized according to the information handling structure among robots. Centralized path planning algorithms normally assign priorities to robots in advance. They can guarantee optimal path if the environment is static and the number of robots is small. In a dynamic environment with large number of robots, decentralized path planning algorithms are more preferable. Decentralized path planning algorithms often apply traffic rules and thus are suitable for route network. They find the optimal paths and solve conflict problems by assigning priorities dynamically and negotiating among robots. They allow each robot to search its own path in advance without considering other robots. If there exits a potential collision, they can negotiate to solve the conflict problem. Solution of the negotiation could be that one robot stops and waits until the potential collision disappears.

In the case of a large number of tasks with different priority levels in a dynamic environment, each robot has to perform many tasks with different priority levels in the order of priority. The current algorithms, either centralized or decentralized, cannot guarantee that the planned path for each robot is always optimal because all the tasks are considered as static obstacles at the planning stage. For example, container handling in a container terminal, if one robot removes the container from a place before the other robots reaches that place, the container should not be considered as a static obstacle to other robots. Other robots need not move around the place to avoid collision. On the other hand, if a container is placed at another robot's path before it passes the position, the task will be an obstacle. This research will focus on these two cases and solve the problem by proposing an effective approach.

2 Literature Review

Different approaches for multiple mobile robot path planning have been proposed and studied by researchers. The approaches to avoid planning path in a high-dimensional composite configuration space are priority assignment and decoupling techniques.

Hu et al [1] proposed a tracking controller for mobile robots by incorporating neural dynamics equations into a conventional non-time based controller, which

resolves the collision problem for multi-robot systems by setting a set of control rules for every case. For different environment or situation, the control rules have to be set differently.

Bennewitz[2] proposed a path planning approach to multi-robot systems by optimizing priority schemes for prioritized and decoupled techniques. Every robot generates its path without considering the paths of others. Possible conflict in the trajectories of robots is then checked. The conflict between robots is resolved by introducing a priority scheme that determines the order of priority. The path of a robot is planned in its configuration time space based on the map of the environment and the paths of the other robots with higher priorities.

Aronov et al [3] studied the motion-planning problem for pairs and triples of robots operating in a shared workspace containing n obstacles. They showed that it is sufficient to consider a constant number of robot systems whose number of degree of freedom is at most d-1 for pairs of robots and d-2 for triples. They presented a $O(n^d)$ time algorithm to solve the motion-planning problem for a pair of robots, which is one order of magnitude faster than what the standard method would give. For a triple of robots the running time becomes $O(n^{d-1})$, which is two orders of magnitude faster than the standard method.

Guilherme [4] addressed the problem of planning the motion of a team of mobile robots subject to constraints imposed by sensors and the communication network. The goal was to develop a decentralized motion control system that leads each robot to their individual goals while keeping connectivity with its neighbours. Experimental results with a group of car-like robots equipped with omni directional vision systems were presented.

LaValle et al[5] introduced a form of optimality that is consistent with concepts from multi-objective optimization and game theory research by considering independent performance criteria. They implemented multi-robot motion planning algorithms derived from the principle of optimality for three cases: i) coordination along fixed and independent paths; ii) coordination along independent roadmaps; and iii) unconstrained motion planning.

It can be seen that the method in [1] is to set different control rules for different environments or situations to solve collision problem but not deal with the path planning problem. The algorithm in [2] needs to compute the possible collision zones in advance, and it is obvious that the compute cost will increase rapidly with the increase of the number of robots. The algorithm in [3] will give a good performance only if it is applied in the situation with a small quantity of robots, such as the situation with two or three robots. The algorithm in [5] optimizes each robot's path by considering independent performance criteria. All the methods or algorithms mentioned in the literatures cannot guarantee an optimal path for each robot considering the dynamic change of the environment.

3 An approach for path planning

3.1 Problem description

Given n prioritized tasks are assigned to m robots and communications are existed among the robots. Each robot knows the environment in advance. For robot R_i, its tasks $r = \{r_i^1, r_i^2, r_i^3, \cdots r_i^k\}$, where r_i^j denotes the jth task of robot R_i, are arranged in the order of priority. The robot R_i executes the tasks from the one with highest priority to the one with lowest priority. For the task r_i^j of robot R_i, its start position and goal position are s_i^j and g_i^j respectively. To execute the task r_i^j, R_i need to move from the destination g_i^{j-1} of the predecessor task r_i^{j-1} to s_i^j, then from s_i^j to g_i^j. For the first task r_i^1, R_i is assumed to move from its initial position s_i^0 to s_i^1, then from s_i^1 to g_i^1.

For undertaking each task, the robot needs to plan a path from its current position to the start position of the next task, and to the destination based on the environment. For example, to execute task r_i^j, robot R_i will plans its path P^j in advance according to the static obstacles in the environment when it finishes the task r_i^{j-1}. When Robot R_i executes the task r_i^j along the planned path P^j, three cases might happen:

(1) The goal positions of other robots' tasks are on the path P^j and other robots put their objects on the way of P^j before robot R_i passes it. If robot R_i moves along its original path P^j, a collision will occur.

(2) The start positions of other robots are originally on the way of path P^j and are considered as static obstacle(s) when the robot plans its path. But these obstacles will be moved before the robot R_i moves around it. In these cases, robot R_i does not need to move around the obstacle. A straight path at that position should be generated.

(3) Robots' paths are crossed, collision among the robots may happen when they undertake their tasks.

3.2 Path planning

An environment is represented by occupancy grids, which separate the environment into a grid of equally spaced cells. Then the path planning for each robot becomes a graph search problem, shown in Figure 1.

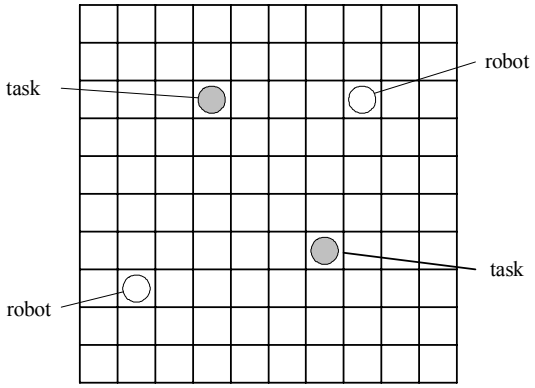


Figure 1: The occupancy grid expression of an environment

Given that robot R_i just finishes its task r_i^{j-1}, it plans its path P^j from g_i^{j-1} to s_i^j then to g_i^j base on the current environment. The path P^j satisfies the condition that the robot R_i does not collide with any object if all the objects keep static when R_i execute the task r_i^j. The path P^j will then be adjusted in the following cases:

- Case 1: R_i meets other robots at the intersection area of their paths. The robot performing the task with higher priority is endowed with higher priority. If the robots have the same priorities, the robot closer to its goal position of the current task will be assigned higher priority. That is to say, the robot with lower priority will stop and wait till the robot with higher priority passes the intersection.
- Case 2: The task's goal position of robot R_j is on the path of robot R_i and R_j has located the task at its goal position before R_i passes the position. In this case, R_i need to re-plan the path around the obstacle.
- Case 3: One task is removed by robot R_j from its position which is on the way of other robot R_i, this task will no longer be as obstacle to robot R_i. Then the path P^j of robot R_i is not the optimized path any more. The robot R_i will plan a new path from its current position to its goal position.

4 Simulation Results

In order to verify the method presented above, two simple examples, each of which consists of two robots and four tasks, are used. Each robot is assigned two tasks respectively, and each task is to move a box from a start position to a destination. The tasks' start positions and goal positions, and the robots' start positions in example one is shown in Figure 2. A* algorithm was applied as the path planning algorithm.

Figure 3 shows different states of operation when the two robots are moving the boxes in the example one. In Figure 3(a) and 3(b), Robot 2 (white) waits at the intersection until Robot 1 (green) passes the intersection because Robot 1's task priority is higher than that of Robot 2. In Figure 3(c) and 3(d), As Robot 1 has moved the second box from its start position and the box is no longer the obstacle of Robot 2, Robot 2 re-planed its path to pass this start position. Figure 3(c) shows the moment just before Robot 1 reaches the start position of its second task. In Figure 3(c) the blue curve shows the current path of Robot 2 and the yellow line shows the changed path after Robot 1 take its second task away from the start position. Figure 3 (d) shows that Robot 2 goes directly through the original position of Robot 2's second task's start position. That is to say, Robot 1's second task is no longer an obstacle for Robot 1. Figure 3(e), 3(f) and 3(g) shows the Case 2 stated in Section 3.2, i.e., Robot 1 moved a box to its goal position, but the goal position of Robot 2's second task is on the way. Robot 1 re-planed its path to avoid collision with the box by taking a way around the box. Figure 3(e) shows the moment Robot 1 locates the box at the goal position and the box becomes an obstacle of Robot 2. Robot 2 re-plans its path at this moment. The blue line and yellow curve in Figure 3(e) shows the original path and the changed path of Robot 2 respectively. Figure 3(f) and 3(g) shows Robot 2 move around the obstacle. Figure 3(h) shows the final state of the two robots when they finish their tasks.

Figure 4 shows the tasks' start positions and goal positions and the robots' start position in example 2. Similar to Figure 3 in example 1, Figure 5 shows different states of operation when the two robots are moving the boxes in this example. Figure 5(a) and 5(b) shows the Case 1 in this example that Robot 1 and Robot 2 have the same priority. But Robot 2 must waits at the intersection until Robot 1 passes the intersection because the distance from Robot 1's current position to its goal position is shorter than that of Robot 2, therefore higher priority is assigned to Robot 1. Figure 5(c) shows the moment that before Robot 2 reaches its first task goal position Robot 1 has reached its second task start position. The blue line in Figure 5(c) indicates the original path of Robot 1 at this moment. That is to say, Robot 1 should not go around any static obstacle. Figure 5(d) shows the

moment that Robot 2 reaches its first task's goal position and Robot 1 is on the way to its goal position. Figure 5(e) shows the moment that Robot 2 moves its first task to its goal position and Robot 1 re-plans its path according to state of the static obstacle. The yellow curve in Figure 5(e) indicates the changed path of Robot 1. That is to say, Robot 1 will move along the yellow curve to its goal position. Figure 5(f) shows that Robot 1 goes around the obstacle. Figure 5(g) shows the final state of the two robots when they finish their tasks.

5 Conclusion

The simulation results have shown that the proposed method in this paper can dynamically planning the paths of a team of robots under the conditions of the three cases. Particularly, when several robots perform distributed prioritized tasks, the method can guarantee collision-free paths for each robot. This method will be further investigated and implemented for simultaneous task allocation and path planning problems and applied to a number of robots in a highly dynamic environment with many tasks. Experiments will be conducted.

6 References

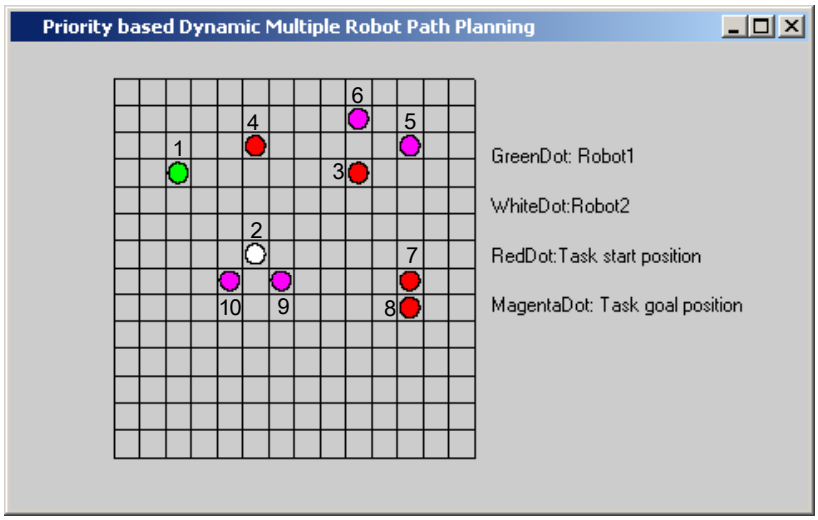
[1] Hu, E., Yang, S.Y., Chou, D., and Smith, W.R., "Real time tracking control obstacle with obstacle of multiple mobile robots". Proceedings of IEEE International Symposium on Intelligent Control, pages 87-92, Vancouver, Canada, 2002

[2] Bennewitz, M., Burgard, W., and Thrum, S., "Optimizing schedules for propertied path planning of multi-robot systems", Proceedings of IEEE International Conference on Robotics and Automation, volumn 1, P.271-276, Seoul, South Korea, May 2001.

[3] Aronov, B., M. de Berg, F. van der Stappen, Svestka, P., and Vleugels, J., "Motion planning for multiple robots," Discrete and Computational Geometry, 22:505-525 (1999).

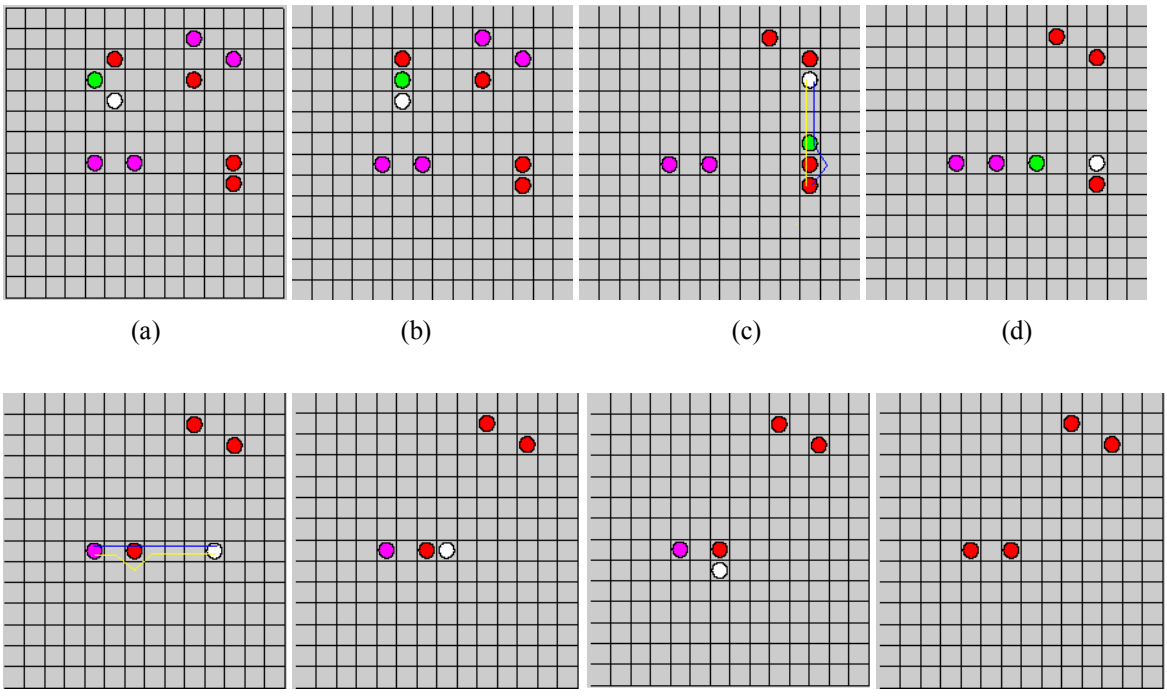
[4] Pereira, Guilherme A. S., Das, Aveek K., Vijay Kumar; Campos, Mario F. M., "Decentralized motion planning for multiple robots subject to sensing andcommunication constraints", Second International Workshop on Multi-Robot Systems Mar, 2003 Washington DC,USA, P.267-278.

[5] LaValle, S.M.; Hutchinson, S.A, "Optimal motion planning for multiple robots having independent goals", IEEE International Conference on Robotics and Automation . P.2847-2852, 1996.



1: Robot 1 start position
2: Robot 2 start position
3: Robot 1 first task start position
4: Robot 2 first task start position
5: Robot 1 first task goal position
6: Robot 2 first task goal position
7: Robot 1 second task start position
8: Robot 2 second task start position
9: Robot 1 second task goal position
10: Robot 2 second task goal position

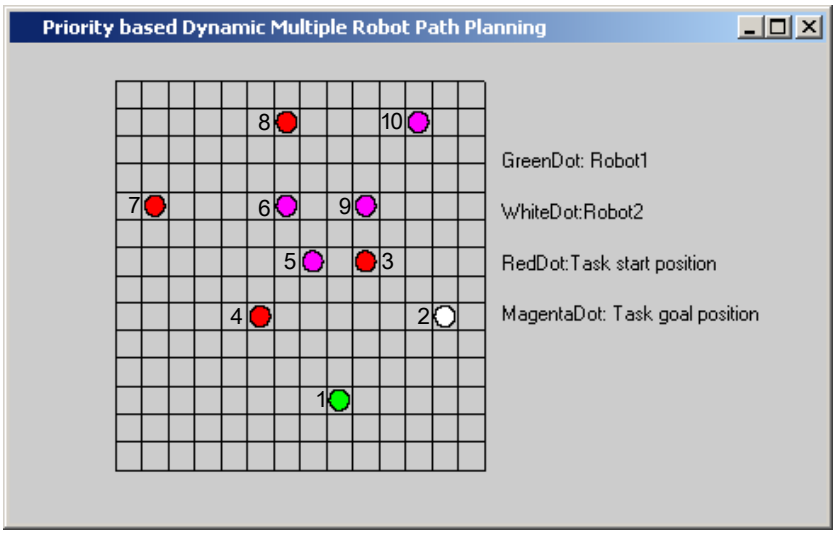
Figure 2: The initial positions of the robots and their tasks in example one



(a) (b) (c) (d)

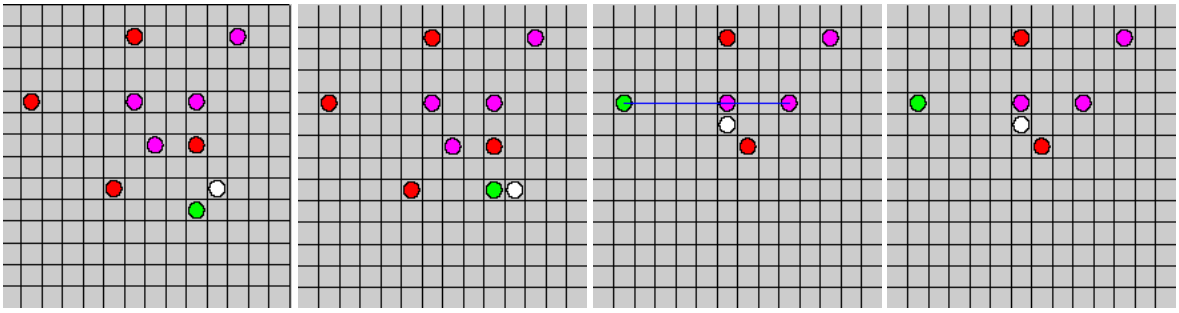
(e) (f) (g) (h)

Figure 3: The state of robot1 and robot2 at different time example one

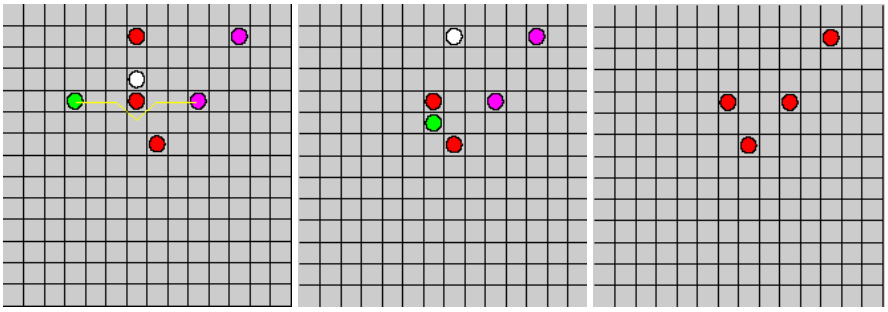


1: Robot 1 start position
2: Robot 2 start position
3: Robot 1 first task start position
4: Robot 2 first task start position
5: Robot 1 first task goal position
6: Robot 2 first task goal position
7: Robot 1 second task start position
8: Robot 2 second task start position
9: Robot 1 second task goal position
10: Robot 2 second task goal position

Figure 4: The initial positions of the robots and their tasks in example two



(a) (b) (c) (d)



(e) (f) (g)

Figure 5: The state of robot1 and robot2 at different time in example two