Proceedings of 2015 IEEE 12th International Conference on Networking, Sensing and Control
Howard Civil Service International House, Taipei, Taiwan, April 9-11, 2015

Multi-Robot Path Planning Based on Improved D* Lite Algorithm

Jung-Hao Peng
Department of Electrical Engineering
National Taiwan Normal University
Taipei, Taiwan, R.O.C.
pgh87564231@gmail.com

I-Hsum Li
Department of Information Technology
Lee-Ming Institute of Technology
Taipei, Taiwan, R.O.C.
ihsumlee@gmail.com

Yi-Hsing Chien, Chen-Chien Hsu, and Wei-Yen Wang
Department of Electrical Engineering
National Taiwan Normal University
Taipei, Taiwan, R.O.C.
wywang@ntnu.edu.tw

Abstract—**This paper proposes an improved multi-robot path planning algorithm for finding the path via interacting with multiple robots. The task is to find the path with a minimum amount of computation time by using fast re-planning algorithm. To solve multi-robot path planning problem which cannot be executed in real-time, we regard other robots, exclusive the origin robot, as obstacles. Therefore, the robot uploads location information to the MySQL server to plan a safe distance between robots.**

Keywords—multi-robot; path planning; D*lite; fast re-planning.

I. INTRODUCTION

For any mobile device, the ability to navigate in its environment is important. Navigation prevents the robot from getting into dangerous areas and environment such as swamps or radiation contaminated regions. Robotic navigation includes three categories, self-localization, path planning, and map-building [1-3]. In this paper, the multi-robot path planning in a known map environment was explored. The most commonly used algorithm is grid-based search such as A* [4-5] and Dijkstra.

Grid-based approaches overlay a grid on configuration space. Calculating the current cost and estimating the distance to the goal determines what the next step will be. The computation speed is proportional to the size of the map. The large map needs more computing time. The path is often planned along the edge of the wall. In multi-robot path planning, the grid-based search cannot be used because the map will change when any of the robots moves.

The result of other robots' path will be changed by any move of the robots, called multi-robot coordination. Most multi-robot researches in the past usually used evolutionary algorithm to coordinate the paths among multi-robots. The common evolutionary algorithms are ant colony optimization and genetic algorithm (GA). These algorithms are equipped with bionic concept. Paths will get better with the evolution of generations. Evolutionary computation method gets the optimal path, but the amount of computation is huge.

We referred to Sven Koenig's paper "Fast Re-planning for Navigation in Unknown Terrain" [6] on movement costs experiments. This experiment was applied to the multi-robot path planning. In most cases, the cost of movement did not change significantly. Only when main roads were blocked did the movement cost of grids increase. Based on the results of this experiment, we chose D*Lite Algorithm as the basis of our research, and used previous path information to achieve fast re-planning.

Path planning is divided into two categories, which are global [7] and local [8]. If the knowledge of the environment is known, the global path can be planned offline before the robot starts to move. This global path can facilitate the robot to traverse within real environment because the feasible optimal path has been less constructed within the environment. Another category, local path, is usually assembled online when the robot avoid the obstacles in a real-time environment. Local Path planning only deals with grids around the robot. The purpose is to calculate the robot's next step and ignore obstacles that are too far away in order to achieve fast planning. But it may be caught in the local optimum solution.

We can define multi-robot path as- "is a map of geometric locus of all the points in a given space where the robot has to travel." Robots cannot collision on this path. In our research, we add the idea of a safe distance in the original D * Lite algorithm. The robot can avoid the collision occurred in path planning. Moreover, the concept of the local path was added to enhance the computing speed. Planning a real time and collision-free path for multi-robots can be achieved. The safe distance is proportional to the size of the robot. The larger robot has the larger rotation radius. In practice, it is to increase the movement cost of the grids around the robot.

Section II introduces the background of GA, A*, and D*Lite. Section III presents the proposed Improved D* Lite Algorithm. Section IV describes applications of the Improved D* Lite Algorithm for path planning. The comparison result, from the application of available approaches, is included in section V. Conclusion is given in Section VI.

II. Preliminary

A. A* Algorithm

Peter Hart and others proposed A* Algorithm in 1972. It is one of the greedy algorithms. Every time the decision-making is required to choose the best option. A* algorithm looks for the grid that is the closest to the destination and has the smallest movement cost. Using open list and close list checks every walkable roads and obstacles, and calculates its path function:

$$F(n) = G(n) + H(n) \quad (1)$$

G(n) is the movement cost from the starting point A to a given square on the grid. H(n) is the estimated movement cost from the current square on the grid to the final destination. However, the actual distance between the current point to the goal cannot be predicted, because the quantity and the distribution of obstacles are unknown. Thus, G(n) is just an estimation. Some call this approach "Error trying" or heuristic search. It is interesting to note that when we ignore the function H(n) A * will become Dijkstra Algorithm. If you want A * algorithm to consider the terrain like mountains, lakes, you need to define different movement cost in G(n). Some terrains are hard to pass, but the robot needs to spend higher price to move. Manhattan method is one of the commonly used heuristic searches. The sum of horizontal grids and vertical grids are multiplied by ten. For example, in Fig. 1, the value of Manhattan method between the green starting point and the red goal is H(n)=40.

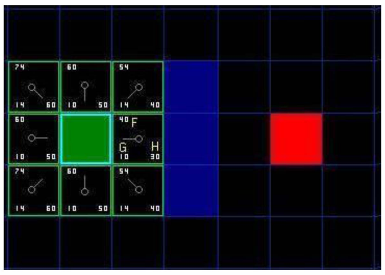

Fig. 1. Manhattan method.

B. D*Lite Algorithm

D * Lite can reuse the previous path planning information to quickly update a path. In Fig. 2, we mark each grid point using the Chebyshev distance:

$$\max(|x_{goal} - x|, |y_{goal} - y|) \quad (2)$$

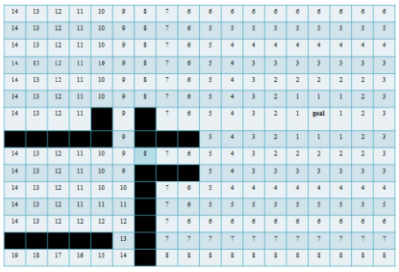

Fig. 2. Chebyshev distance.

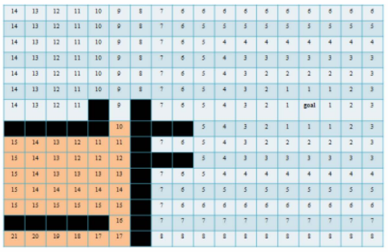

Fig. 3. Add an obstacle.

In Fig. 3, if an obstacle is added, the movement costs of some grids are changed. Orange blocks are where the changes are. Orange blocks accounts for 14.1% of the map. When re-planning the map, only the orange blocks need to be updated, the amount of computation can be reduced and accelerate the processing speed.

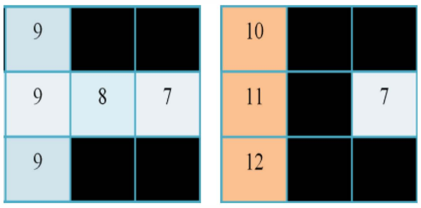

Fig. 4. Local amplification.

How to know which grids' cost change? In Fig. 4, only when the coordinates of added obstacles are known, we view magnified observation. Therefore, we can know the four grids around the obstacle which we added in. The cost of the grids on the left side of the obstacle we added in is changed, and on the right side remains the same. The reason is that the grid on the right side is the predecessor node and the grids on the left sides are the successor nodes. Only when the successor node moves to the movement cost change. Finding all the successor nodes is the only thing required to locate the grids with changes of cost.

S denotes the finite set of vertices of the graph. s denotes any grid point on the map. $Succ(s) \subseteq S$ denotes the set of successors of vertex $s \in S$. $c(s,s')$ denotes movement costs. $0 < c(s,s') < \infty$ denotes the cost of moving from vertex to vertex $s' \in Succ(s)$. The common distance formulas are Chebyshev and Euclidean distance.

1. When *g*=*rhs,* a node is consistent.
2. When *g*>*rhs,* a node is over-consistent.
3. When *g*<*rhs,* a node is under-consistent.

There are four steps in the D* Lite algorithm for planning a path.

Step 1: Initially, *rhs* and g of all the nodes are set to infinity. The open list is set to empty. Then, the goal's *g* value is set to infinity and *rhs* value is set to zero. The goal is put on the open list and the goal is the current node.

Step2: If current node is over-consistent, the current node's *g* value is equal to *rhs* value. Then all of the neighbors are put on the open list. Their *rhs* values are updated. After that, the current node must be removed from the open list. If current node is under-consistent, its *g* value is set to infinity. Then all of the neighbors are put on the open list. And their *rhs* values are updated. After that, the node muse be removed from the open list. Choose the next node to determine the priority of the nodes in the open list.

Step 3: If the robot detects an obstacle, the obstacle's *rhs* value is set to infinity. The costs of neighbors of the obstacles are updated to infinity, so this node cannot be passed through. Hence, the current path is not feasible. Moreover, we re-plan an optimal path from current node to goal node.

Step 4: Repeat the step 2 and 3 until the current states are equal to the start states or no path exists. Thus, we can obtain the shortest path.

III. Improved D* Lite Algorithm

In order to verify whether multi-robots can be applied to the fast re-planning method, an experiment on multi-robot path planning using D*Lite was designed. The process is analyzed as follows.

In Fig .5 (a) the red grid indicates the starting point and the green grid indicates the destination. The yellow, blue, and brown grids represent different robots. During the path planning, other robots were regarded as obstacles. The movement cost is infinite and, in Fig. 5 (b), only the original position cost of the robot was changed from infinite movement cost to a passable point. In Fig. 5 (c), the key passages were blocked so there is 15% of movement cost changed. In Fig. 5 (d), this experiment result shows that unless the key passages are blocked, the movement cost rarely changes.

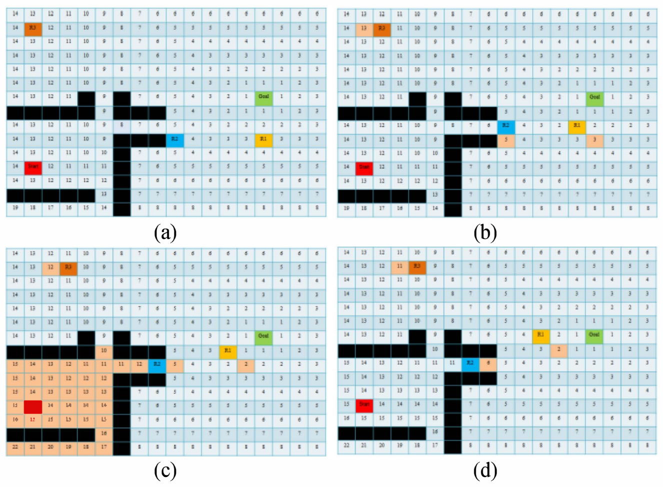

Fig. 5. (a) Initial state, (b) First step, (c) Second state, (d) Third step.

In this paper, different sizes of the robots are considered. From smaller robots like NXT Lego to bigger tracked stair-climbing robot, they have to be adapted to this algorithm. So, the following experiment is for the large size robots.

In the experiment, with bigger size of the robots In Fig. 6 (a) 、(b) 、(c) 、(d) and more grids have costs changes, most of the movement costs of grids remain the same. Through experimental prediction D* Lite algorithm can be applied to the multi-robot path planning. Each robot can posit itself autonomously, and the map is known. The mobile robot uploads self-positioned information to the MySQL server and provides updated movement costs to other robots in the system. The way to update movement costs is the same as the D * Lite. As long as any robot moves on the map, the movement costs will be updated and the paths will be re-planned. So the paths will be constantly updated.

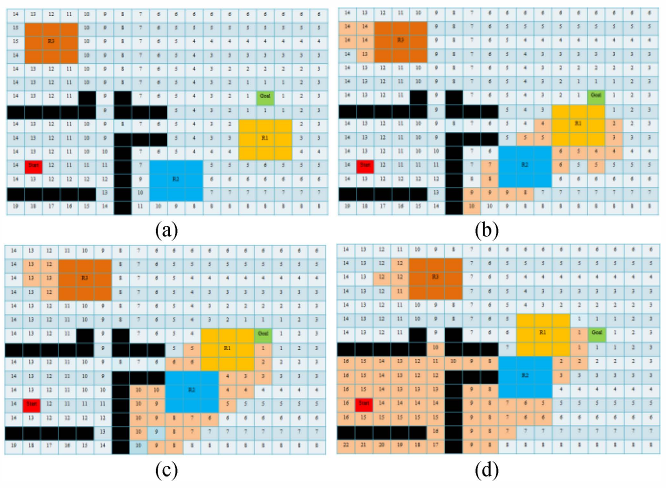

Fig. 6. bigger size of the robot (a) Initial state, (b) First step, (c) Second state, (d) Third step.

The planned path is shown in Fig. 7. The purple, red and yellow dots are the position of the robots. The green dots are the destinations. Blue and orange lines are the paths. It is found that the paths planned keep a safe distance from the walls and avoid other robots.

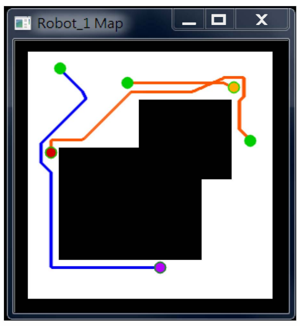

Fig. 7. Multi-robot path.

The method for avoidance:

$$rhs(s)=\begin{cases}0, \text{if } s=S_{start}\\ \min_{s'\in Pred(s)}(g(s')+T(s)*c(s,s')), \text{otherwise}\end{cases}$$

T(s) is the movement costs through *s* grid point. *T(s)* is customizable. In this paper, the costs within safe distance are multiplied 1.1 proportionally. Robots in different sizes have different path planning. In Fig. 8, Fig. 9, the red robot is smaller than the chair. So the red robot chose to go through the chair. The blue robot is bigger than the chair, so it chose to go around the chair to avoid collusion.

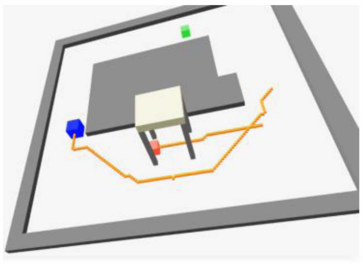
Fig. 8. Different size robot.

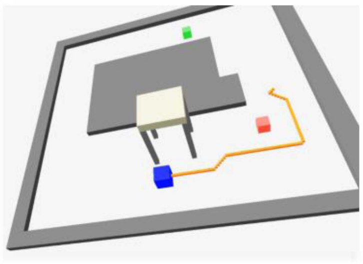
Fig. 9. Safe distance.

In order to avoid collisions among robots, the planned paths also avoid each robot.

IV. Experimental Results

In Fig.10, these experiments discuss path re-planning time under three different algorithms and quantity of robots in the same map. This represents the environment will be changed when multi-robots move. The results are shown in Table I, Table II.

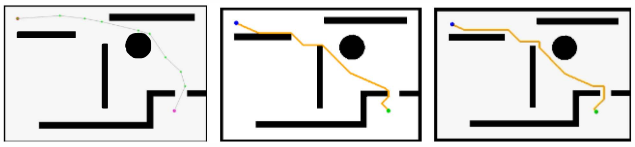
Fig. 10. The planning path of (a) GA (50 generation) ,(b) D* Lite ,(c) proposed Improved D* Lite method

TABLE I. Algorithm comparison(One robot case)

Algorithms	Average re-planning time (400*284)		
	GA(50 generation)	D*Lite	Proposed method
One robot case	5.76 sec	0.46 sec	0.42 sec

TABLE II. Algorithm comparison(Three robot case)

Algorithms	Average re-planning time (400*284)	
	D*Lite	Proposed method
Three robot case	1.32 sec	0.44 sec

V. Conclusion

From the experiment, GA needs to require more computing time. But the computation speed by using the proposed method, Improved D* Lite Algorithm, is far better than GA algorithm in One robot case. We propose the method Improved D* Lite Algorithm using the concept of parallel processing. It is better than D*Lite. This paper proves that through parallel processing and re-planning maps, multi-robots can plan faster and more flexible paths.

Acknowledgment

This work was supported by the Ministry of Science and Technology, Taiwan, under Grants MOST 103-2221-E-003-026-, MOST 103-2221-E-003-027- and MOST 103-2221-E-234-004-.

References

[1] R. Siegwart and I. Nourbakhsh, Introduction to Autonomous Mobile Robots, The MIT Press, 2004.

[2] A. Willms and S. Yang, "An efficient dynamic system for real-timerobot-path planning," *IEEE Trans. on Systems, Man, and Cybernetics—Part B*, vol. 36, no.4, pp.755-766, 2006.

[3] G. Jan, K. Chang, and I. Parberry, "Optimal path planning for mobile robot navigation," *IEEE Trans. on Mechatronics*, vol. 13, no.4, pp. 451-460, 2008.

[4] M. Nakamiya, T. Terada, and S. Nishio, "A route planning method for multiple mobile sensor nodes," in *Proc. of the 5th International Networked Sensing Systems, Kanazawa*, 2008, pp.103-106.

[5] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100-107, 1968.

[6] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Trans. on Robotics and Automation*, vol. 21, no. 3, pp. 354-363, June 2005.

[7] O. Khatib, "Real time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 10, pp. 90-98, 1985.

[8] G. Nagib and W. Gharieb, "Path planning for a mobile robot using genetic Algorithm," in *Proc. of the International Conference on Electrical, Electronic and Computer Engineering,* 2004, pp185-189.

[9] J. Guo, L. Liu, Q. Liu, and Y. Qu, "An improvement of D* algorithm for mobile robot path planning in partial unknown environment," *Intelligent Computation Technology and Automation*, Changsha, Hunan, vol. 3, pp. 394-397, 2009.

[10] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *Proc. of the International Conference on Robotics and Automation*, 2002, pp. 968-975.

[11] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," Coll. Comput., Georgia Inst. Technology, Atlanta, GA, Tech. Rep. GIT-COGSCI-2002/3, 2001.

[12] T. Ersson and X. Hu, "Path planning and navigation of mobile robots in unknown environments," in *Proc. of the International Conference on Intelligent Robots and Systems*, 2001, pp. 858-864.