*Design Studies and Intelligence Engineering*
*L.C. Jain et al. (Eds.)*

*doi:10.3233/FAIA220007*

# A Multi-Robot Planning Algorithm with Quad Tree Map Division for Obstacles of Irregular Shape

Tongpo ZHANG[a], Chenyuan LI[a], Xu (Judy) ZHU[b], Enggee LIM[a], Fei MA[c], Limin YU[a,1]

[a] *School of Advanced Technology, Xi'an Jiaotong-Liverpool University*
[b] *School of Electronic and Information Engineering, Harbing Institute of Engineering, Shenzhen*
[c] *School of Science, Xi'an Jiaotong-Liverpool University*

**Abstract.** This paper proposes an algorithm to improve the efficiency of the multi-robot system in simulated global information map containing obstacles. The designed multi-AGV scheduling algorithm is based on an optimal shortest path algorithm with the combination of the waiting mode and motion coordination. The proposed shortest path algorithm not only has lower time delay but also decreases the possibility of collision of the multi-robot system. In addition, simulated global information maps are established to test the efficacy of the algorithm.

**Keywords.** Robot planning, shortest path algorithm, collision avoidance, scheduling optimization

## 1. Introduction

A multi-robot system can generally be divided into two types, centralized control architecture and decentralized control architecture. For the former, there is a central control platform to convey useful information to each robot such as moving state of robot, map information and mark information, and it needs to carry out the algorithm for all robots [1]. Hence, the centralized control architecture has the properties of stability, simple realization, low efficiency and high requirement of communication. This architecture can be classified as coupled and decoupled. For the coupled architecture [2] [3], it regards all robots as the same one and carries out the same algorithm with considering the whole state and parameters, which means that its computation volume will be huge if there are multi-robots in the map. In contrast, the decoupled architecture does not consider all information but uses the tricks to deal with the coming conflicts or dead locks. To reduce time complexity, this architecture needs to plan paths and coordinate movements in real time. An apparent disadvantage of it is that it cannot

[1] Corresponding Author.

consider the ideal optimal solution for the system. About movement coordination modes, there are region control, time window and multi-agent system. This paper designs an algorithm to construct the shortest path for each robot and combines the waiting mode and collision avoidance algorithm to ensure there is no conflict in simulated maps.

For a large number of robot systems, the coupled architecture is not feasible of sustainable tasks because of the huge computation content. Thus, our idea emphasizes the decoupled architecture, which means each robot needs to compute the optimal path before moving and then implement a waiting or collision avoidance algorithm. For a path planning algorithm, in the first step, it needs to examine the environment parameters to construct a node-obstacle distribution graph [4]. Then, it uses an efficient shortest path algorithm to avoid obstacles [5]. In this paper, the algorithm adopts the quad tree to construct the lattice for the map. Unlike the general lattice map, using a quad tree can quickly locate a precise position and adjust the resolution more simply. It will also decrease time to compute the path planning and collision avoidance.

To effectively plan the shortest path and coordinate every robot to avoid collision, using decoupled architecture will be more suitable for the combination of waiting mode and avoidance collision algorithm. [6] has proposed a two-layer architecture to control the whole movement and local avoidance. They are macro level and micro level respectively. This architecture strength is that it distributes the calculated amount to different parts of the system. There are several other decoupled architectures with similar technology to deal with different assumed situations appearing in the [7] [8] [9]. In addition, to obtain dynamic information in real time, robot can communicate with neighbors in the specified region [10]. The advantage is its stability, expandability and flexibility. [11] has proposed a strategy that multi-robot system combines the guidance algorithm and disperse policy decision.

For a scheduling algorithm, the most important point is to coordinate robots to avoid collisions on the map. Those solutions can be divided into two types, waiting mode and motion coordination. To combine the stability of waiting mode and efficiency of coordination motion, this paper aims proposing a decentralized algorithm to solve collision avoidance problem of multi-robot system in a simulated map with polygon obstacles. In previous algorithms, they did not consider the obstacles when dealing with collision avoidance. In this paper, the designed algorithm uses quad tree to mark the map and obstacles. The robot only communicates with others in a certain region based on wanted accuracy, which reduces the computation complexity. Furthermore, there is a central control platform to observe global information which has access to collect the data from multi-robot to avoid the dead lock problem, which cannot be solved in [1]

The rest of this paper is described as the structure below. Section 2 introduces a path-planning algorithm containing how to climb obstacles. Section 3 introduces a designed system which combines the waiting mode and motion coordination. Section 4 displays some simulations of the multi-robot system. Section 5 concludes the paper and discusses possible future work.

## 2. Robot planning

First, assuming there is only one robot moving on the map where there are some resource points and polygon obstacles. To avoid the condition that the robot moves from one resource point to another resource point crossing through one unnecessary resource point, the robot is guided to move along with several resource points but not just from the source point to the destination point. To climb a group of obstacles between the resource point and destination point, the positions of the obstacles should be arranged. Then, the designed algorithm will check all obstacles that hinder the robot and find the distance between the source point and the obstacle. The distance value is calculated by the formula of distance between the point and straight line.

A shortest path algorithm to climb the polygon obstacles between the resource points is created. This algorithm is different from the previous shortest path algorithm which used the subdivision of the map and wave front to obtain the absolute shortest path [1]. The realization of the previous algorithms is complex and has higher time complexity of O(n*log(n)) than the shortest path algorithm where n is the number of the vertices of the obstacles. In addition, the absolute shortest path is often one path. If the system has multi-robots, all of them will use this path, which increases the probability of collisions. Hence, the designed shortest path algorithm is more suitable for the multi-robot system.

The algorithm defines Landing point as the first vertex of that obstacle the robot needs to arrive at, and Separation point as the last vertex of that obstacle the robot needs to arrive at. The first step is to find a landing point. The algorithm will compute two angles formed by the source-destination line and source-visible point line where the visible point is defined that the most distant point AGV can see directly. Then, the algorithm chooses the landing point from a smaller angle.

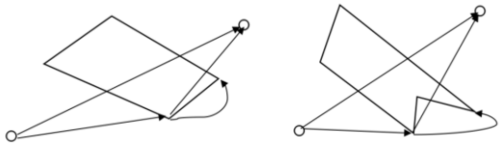

**Figure 1**. Obstacle in square shape and Obstacle in zigzag shape

Because there is a sort of obstacles with different shapes, the algorithm needs to use different solutions to cope with them. If the condition is like Figure 1: square obstacle that the obstacle is just a general square, the algorithm needs to guide the robot to choose a different separation point. If the landing-destination line intersects with the interior of the obstacle, the robot needs to move current point to the next point along with the anticlockwise direction (if the point is located on the right side of the Source-destination line).

If the condition is that the landing-destination line intersects with the obstacle but does not cross the interior of the obstacle in Figure 1: zigzag obstacle, the algorithm needs move robot from landing point to next point in graph along with the anticlockwise direction (if the point is located in the right side of Source-destination line). The algorithm repeats the above steps until the point climbs the obstacle. After dealing with these two conditions, the shortest path algorithm can climb over all types of polygon obstacles.

## 3. The designed system setup

The designed system is the shortest path algorithm with the combination of the waiting mode and motion coordination. Waiting mode is aimed at solving the collision problems in the multi-robot system. Traditional waiting mode sets additional parking points for each robot that wastes space resources. Meanwhile, it needs a large amount of time. Thus, an idea was proposed that the waiting mode will be executed in the eventual condition based on the observation of the center control platform, which means that this algorithm will not be triggered all the time. waiting mode needs to avoid the collision, deadlock and live lock problems in the scheduling field. The algorithm needs to construct a control platform firstly to store all the paths information of the robots so that the control platform can dispatch robots.

Once the path information is undated, the control platform will calculate the shared path of all robots. The shared path was defined as the repetitive part of the path of one robot with other paths of robots. Live lattice was defined as based on the volume of the robot; the algorithm will find the minimum area lattice that can contain one robot. If no three child-nodes are occupied by any object, this lattice is a live lattice. Dead lattice was defined as based on the volume of the robot; the algorithm will find the minimum area lattice that can contain one robot. If anyone lattice of three child-node is occupied, this lattice is dead lattice. If the next hop of the robot is in the shared path but any point in the shared path of this robot is occupied by other robots and any point of lattice of the shared path is dead lock, the robot cannot go to the next point and has to wait for other robots pass. All these behaviors are controlled by the center control platform.

The motion coordination proposed in this paper is based on the lattices constructed by the quad tree. The quad tree was used to divide the simulated map into many specified areas of lattices. The waiting mode was improved through adding a new limitation condition based on the quad tree lattice. Under this condition, the waiting mode will only be triggered in a few cases. The quad tree is like other trees with parent nodes and child nodes. The difference is that each parent node has four child nodes standing for four regions in the map, left upper, left lower, right upper and right lower. Furthermore, each child region can also be divided into four regions if the system needs a higher accuracy degree. The advantage of this type of data type is that it can be simply realized and have low time complexity. As a result, each child nodes without any child node becomes minimum lattice in the map.

For different robots, they may have different volumes in reality, which means they need different sizes of lattice. Thus, the designed algorithm chooses the minimum size of the lattice that can contain one size of the robot. For example, in the simulation of the multi-robot system, the experiment sets a 100*100 m^2 map and has tried several different numbers of divisions. The floor area of one robot is assumed to be about 1.5 m^2. Eventually, the experiment finds the six divisions are suitable for this size of the map, which means there will be totally $4 + 4^2 + 4^3 + 4^4 + 4^5 + 4^6 = 5460$ lattices in the map. Therein, there are $4^6 = 4096$ minimum size of the lattice and the minimum size is $(100 * 100)/4^6 = 2.44$ that can contain one robot, so it is suitable. In this paper, the minimum precision is 2.44 m^2 and resolution is 1.22 m^2 that can be adjusted by the depth of the quad tree. The algorithm assumes that the map area is S. The AGV floor area is N and the number of the division of the quad tree is n. The function can be concluded as follows:

$$\frac{S}{4^n} \geq N \tag{1}$$

Through this equation, the algorithm can calculate the maximum n value. In the next step, the algorithm needs to assign different states to those lattices and robots, so that the algorithm can be realized quickly through checking these states. Occupied state is defined as if the minimum lattice contains any robot or the part of the obstacle, this lattice will be marked as occupied. Private state is defined as if the parent lattice of the minimum lattice contains any robot, this lattice will be marked as private. Free state is defined as there is nothing in the lattice. Then, the algorithm still needs to define some states for robots. Idle state is defined as if the robot has no transportation task, the robot will be marked as idle. Running state is defined as if the robot has the transportation task, the robot will be marked as running. Removal state is defined as if the robot is avoiding other robots, the robot will be marked as removal. Blocked state is defined as if the robot is in the waiting mode or it meets one robot with state of removal, the robot will be marked as blocked.

The motion coordination is distributed in each robot and it will be executed through checking the state of the lattice in the control platform and state of other lattice by communicating with them. To establish a checking standard, the robot can detect the same size of the region as the size of itself towards the direction of moving, which means each robot can only detect the next minimum lattice in the map towards the direction of moving. During the process of moving, the robot will constantly check the next lattice state and will face two states it needs to solve. If the state is Free, it can freely move to the next lattice. If the state is private, which means collision will take place soon, the robot will implement motion coordination. Firstly, the robot will check the other three child lattices whether they are located in the straight line of the direction of the moving. If any child lattice is so, the robot will ignore it and check the state of the rest lattices. If the state is not occupied, the robot will change its next hop into the center point of this lattice. If there is not any lattice the robot can avoid, it will mark itself as blocked. It should be noticed that this situation would only happen as a result of other robots occupying the lattice because the waiting mode has avoided the occupation of the obstacle.

For the collision avoidance problem, the robot will face three different situations which are head-on collision, rear collision and intersection collision. Head-on collision is described as two robots move at the same time in the opposite direction. Rear collision is described as two robots move in the same direction, but the back robot has higher speed. Intersection collision is described as two robots will pass the same node at the same time. This decision can reduce the time of implementing the motion coordination because only the head-on collision needs to coordinate through analysis and other two types of collision the robot can only wait for others in site.

## 4. Simulation result

The simulation of the multi-robot system is based on the python matplotlib module. This module can create a visual graph to observe the simulation result. For each robot, the designed experiment can adjust their velocity, task and control platform. For the control platform, it contains the lattice information based on the quad tree and all path information of the robots, which will be used to implement the waiting mode and motion coordination. The map is 100*100 size and contains totally 16 resource points and 3 obstacles. The visual graph is in the Figure 3: simulation environment. The blue points represent the resource points which are labeled, the black polygons represent the obstacles, the red point represents the robot, and the yellow points represent the parking points.

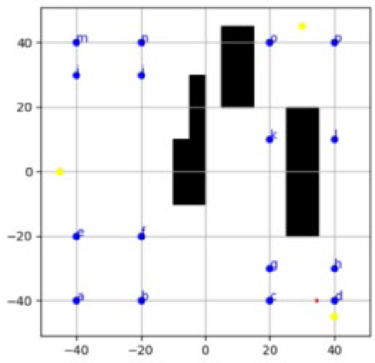

**Figure 2**. Simulation environment

The designed simulation can increase the number of robots in the system to observe the efficiency of the system. Main indexes are completed task number, waiting time and ratio which is the waiting time over the running time. To observe them constantly, each robot will get a random task after they complete a task. The completed task numbers can represent the efficiency of the whole system. The waiting time and ratio of the waiting time over the running time can represent the efficiency of the algorithm where waiting time represents the total time of waiting mode and motion coordination when there will be a collision happening. Running time represents the time of executing the tasks. The unit of waiting time and running time is seconds. Ratio meansWaitingTime divided by RunningTime.

The experiment has done the 2-robot system, 3-robot system 4-robot system simulation tables during the 300 seconds of the experiment. For each experiment, we choose the time when each robot completes 2, 10, 20 and 30 task numbers and analyze their waiting time and running time.

**Table 1.** 2-robot system

| Robot number | Task number | Waiting time (s) | Running time (s) | Ratio (%) |
|---|---|---|---|---|
| 1 | 2 | 0 | 3.4 | 0 |
| 2 | 2 | 0 | 8.5 | 0 |
| 1 | 10 | 0 | 66 | 0 |
| 2 | 10 | 0.8 | 101 | 0.8 |
| 1 | 20 | 0.5 | 158 | 0.3 |
| 2 | 20 | 1.3 | 195 | 0.6 |
| 1 | 30 | 0.8 | 243 | 0.3 |
| 2 | 30 | 1.3 | 294 | 0.4 |

**Table 2.** 3-robot system

| Robot number | Task number | Waiting time (s) | Running time (s) | Ratio (%) |
|---|---|---|---|---|
| 1 | 2 | 0 | 4.5 | 0 |
| 2 | 2 | 0 | 11 | 0 |
| 3 | 2 | 0 | 15 | 0 |
| 1 | 10 | 0.6 | 62 | 0.9 |
| 2 | 10 | 0.8 | 88 | 1.0 |
| 3 | 10 | 0.8 | 83 | 1.0 |
| 1 | 20 | 2.9 | 182 | 1.5 |
| 2 | 20 | 1.5 | 172 | 0.9 |
| 3 | 20 | 1.7 | 145 | 1.1 |
| 1 | 30 | 4.3 | 277 | 1.5 |
| 2 | 30 | 3.9 | 270 | 1.4 |
| 3 | 30 | 3.3 | 245 | 1.3 |

**Table 3.** 4-robot system

| Robot number | Task number | Waiting time (s) | Running time (s) | Ratio (%) |
|---|---|---|---|---|
| 1 | 2 | 0 | 5.7 | 0 |
| 2 | 2 | 0 | 5.7 | 0 |
| 3 | 2 | 0 | 14 | 0 |
| 4 | 2 | 1.1 | 15 | 7.3 |
| 1 | 10 | 1.9 | 100 | 1.9 |
| 2 | 10 | 1.9 | 122 | 1.6 |
| 3 | 10 | 0.9 | 112 | 0.8 |
| 4 | 10 | 3.0 | 148 | 2.0 |
| 1 | 20 | 6.3 | 233 | 2.7 |
| 2 | 20 | 6.2 | 246 | 2.5 |
| 3 | 20 | 1.8 | 244 | 0.7 |
| 4 | 20 | 4.3 | 296 | 1.4 |

We also plot the line chart to compare the different systems in the Fig 3. The blue line is the 2-robot system, the orange line is the 3-robot system and the green line is the 4-robot system. From this chart, we can know that the total waiting time of the system within the 300 second time frame increases about 8 seconds per one additional robot which is acceptable, and if the map is larger, this parameter will be lower.

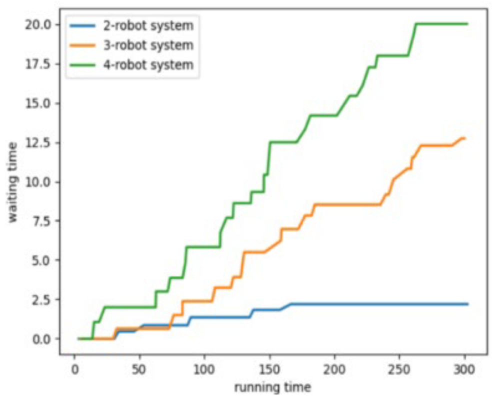

**Figure 3.** Performance Comparison

When the robot number increases, the total completed task numbers are 69, 99 and 93 respectively. It can be shown that the total system efficiency becomes saturated and waiting time does not increase more than 10 seconds with the robot number is increasing. Regarding the ratio of waiting time and running time, we want the ratio value to be as low as possible, which means the majority of the time is used for the task. After analysis of the data, we find that it has a similar trend to the last index. The saturated ratio value is about 2% in the designed system. It demonstrates that each robot is executing its tasks continuously with efficiency.

## 5. Conclusion

This paper proposed a new scheduling algorithm for the multi-robot system. It contains the functions of path-planning, obstacle climbing, waiting mode and motion coordination. These functions can help multi-robot to find path, avoid obstacles, and adjust their paths. Furthermore, with the application of the quad tree, it is convenient for the robot to mark the map using the adjustable lattice structure. More importantly, this strategy can not only be applied to a multi-robot system but also be used to assist other agent devices like artificial intelligence robot to achieve information marking and acquisition.

The innovation of the system design lies in the shortest path algorithm with combination of the waiting mode and motion coordination, as well as the new motion coordination method based on quad tree division. For the shortest path algorithm, it not only has a lower time delay but also decreases the possibility of collision in the multi-robot system. The quad tree division of the map plays an important role. It has reduced computation complexity than the general graph like Voronoi graph, and is flexible for the multi-robot system.

For the future work, more attention should be paid to the study of dynamic nature of the robot colony. More limitations and different costs imposed by the environment and hardware will be considered.

## Acknowledgment:

This research was funded by Research Enhancement Fund of XJTLU (REF-19-01-04), National Natural Science Foundation of China (NSFC) (Grant No. 61501380), and by AI University Research Center (AI-URC) and XJTLU Laboratory for Intelligent Computation and Financial Technology through XJTLU Key Programme Special Fund (KSF-P-02).

## References

[1] I. Draganjac, D. Miklić, Z. Kovačić, G. Vasiljević and S. Bogdan, "Decentralized Control of Multi-robot Systems in Autonomous Warehousing Applications," *IEEE Transactions on Automation Science and Engineering*, **13** (2016), 4: 1433-1447.

[2] M. Cirillo, F. Pecora, H. Andreasson, T. Uras and S. Koenig, "Integrated motion planning and coordination for industrial vehicles", *Proc. 24th Int. Conf. Autom. Planning Scheduling*, 2014.

[3] H. Andreasson et al., "Autonomous transport vehicles: Where we are and what is missing", *IEEE Robot. Autom. Mag.*, **22** (2015), 1:64-75

[4] C. D. B. Borges, A. M. A. Almeida, I. C. P. Júnior and J. J. D. M. Sá Junior, "A strategy and evaluation method for ground global path planning based on aerial images", *Expert Syst. Appl.*, **137** (2019), 232-252

[5] T. T. Mac, C. Copot, D. T. Tran and R. De Keyser, "Heuristic approaches in robot path planning: A survey", *Robot. Auto. Syst.*, **86** (2016), 13-28

[6] S. Manca, A. Fagiolini and L. Pallottino, "Decentralized coordination system for multiple robots in a structured environment", *IFAC Proc.*, **44** (2011), 1: 6005-6010.

[7] K. Zheng, D. Tang, W. Gu and M. Dai, "Distributed control of multi-robot system based on regional control model", *Prod. Eng.*, **7** (2013), 4: 433-441.

[8] V. Digani, L. Sabattini, C. Secchi and C. Fantuzzi, "Hierarchical traffic control for partially decentralized coordination of multi robot systems in industrial environments", *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 6144-6149, May/Jun. 2014.

[9] L. Cancemi, A. Fagiolini and L. Pallottino, "Distributed multi-level motion planning for autonomous vehicles in large scale industrial environments", *Proc. IEEE 18th Conf. Emerg. Technol. Factory Autom. (ETFA)*, pp. 1-8, Sep. 2013.

[10] W. Malopolski, "A sustainable and conflict-free operation of robots in a square topology", *Comput. Ind. Eng.*, **126** (2018), 472-481

[11] J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, **28** (1999), 6:2215-2256