Path Planning and Task Allocation of Many Robots with Motion Uncertainty in a Warehouse Environment

Yinbin Shi and Biao Hu and Ran Huang

Abstract— **Currently, robots have been widely used in the warehouse environment to improve the logistics operation efficiency. In general, the more robots are deployed, the higher efficiency logistics operation will be. However, more robots also bring more challenges to their task allocation and path planning. Towards the two problems, this paper presents an efficient task allocation and path planning solution that can scale up the robots greatly in a warehouse environment. In particular, the decentralized auction-bid scheme is used to allocate tasks, i.e., each robot provides its estimated task completion time as a bid, and the task is allocated to the robot with the lowest bid. In order to estimate the bid accurately, we take the robots motion uncertainty into account and predict the robot density in the map at runtime, and then use the Floyd algorithm to the plan the robot path. We also design an effective scheme to sufficiently avoid the robot collision. The experimental results of a hundred robots demonstrated that our solution can quickly allocate the task and plan the path. The average task completion time is also minimized compared to some other state-of-the-art approaches. The simulation code has been open sourced[1].**

I. INTRODUCTION

Now it has become a common sense that robots can improve the logistics operation efficiency. Compared to humans, warehouse robots require smaller working space, and have better mobility and flexibility, and can be scheduled to achieve the optimal performance. The success of Kiva robot system in Amazon warehousing and logistics has also motivated more and more manufacturer to develop intelligent warehousing system using mobile robots instead of manual handling in today's logistics industry [1]. By now Amazon has 200,000 robots working in its warehouses, and robots help Amazon warehouses improve around 50% efficiency in sorting packages [2].

Along with the increase of robots deployed in the warehouse system, how to allocate tasks and plan the robot path becomes very crucial for the system performance. On the one hand, the growth of robots increases the decision dimension that makes the task allocation and path planning more complicated. On the other hand, due to the influence of noise, friction and inaccurate control input, the robot's behavior on the fly is not deterministic, which makes the collision hard to be avoided. In addition, tasks are often released frequently and dynamically, especially in the busy period such as double 11 or 618 shopping festival [3]. This means that tasks need to be handled quickly, because otherwise more and more tasks will be pended and finally lead to the system collapse.

The task allocation problem is how to allocate multiple tasks to each robot reasonably, so as to maximize the efficiency of task execution. There are centralized and decentralized ways to allocate tasks [4]. Centralized allocation rely on the central controller to make the allocation decisions based on the latest system information. In order to make optimal decisions, some classic intelligent algorithms such as genetic algorithm, ant colony algorithm, particle swarm optimization can be used [5]. These algorithms are effective for small-size problems, but need a long time to find good solutions when the problem size becomes large. The decentralized allocation rely on individual robot to make its own decision. Because the decision is made without considering the other robot information, it is often fast but not optimal.

The essence of path planning problem is how to control the robot move from its start point to the target point without collision. Depending on the map, there can be a variety types of path planning to be used, such as probabilistic roadmap and rapidly-exploring random tree in a continuous space [6], and A* algorithm in a grid map [7]. The warehouse map is often a grid map that divides the environment into a series of grid and each grid is associated with a cost that a robot needs to pay for its pass. The optimal path is the one that needs the least cost to complete the whole route. It is often easy to find an optimal solution for a single robot, but not easy for multiple robots because multiple paths may be deeply interweaved together and are hard to detached, especially when robot size is large.

Being aware of the aforementioned difficulties, this paper proposes an efficient task allocation and path planning solution that can scale the robot size to a hundred in a warehouse environment. We adopt the auction-bid scheme to allocate tasks. The task is defined as a pick-up mission that needs a robot to reach $1 \sim 3$ shelves. For each task, every robot searches a path to complete it and uses the path cost as the bid. The task is allocated to the one robot with the least bid. In order to reduce the computation time of path planning on the fly, we use the Floyd algorithm with offline computation and online match, where the optimal path between every pair of nodes is computed offline and the path is searched out by the online match. In addition, by predicting the robot's future position, we take the influence of robot's density on the path cost into account. To avoid

*This work has been supported in part by National Key R&D Program of China 2019YFB1312102 and in part by the National Natural Science Foundation of China under Grant 91848103.

Yinbin Shi and Biao Hu and Ran Huang are with the College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China. hubiao@mail.buct.edu.cn

[1]https://github.com/nucleusbiao/Robot-simulation-in-a-warehouse-environment

the robot collision, we use some traffic rules to restrict the robot's motion, and alert robots to avoid possible collision by the online obstacle detection. In the end, experimental results show that our solution can greatly improve the system performance in terms of robot size, average task completion time, and computation overhead on the task allocation and path planning.

This paper is organized as follows: the related work is reviewed in Section II. The problem is presented in Section **??**. The details of our proposed solution is presented in Section **??**. Experimental results are provided in Section **??**. Conclusions and future work are provided in Section **??**.

II. Related Work

As two crucial techniques of warehousing robots, task allocation and path planning issues have been studied actively for decades of years, and there are a plethora of approaches towards them.

Task allocation among multiple robots is a NP-hard problem that is not easy to find an optimal solution [8]. Auction-based approaches have recently become popular because of their advantages such as flexibility, decentralized nature, and robustness to failure [9], [10], [11]. By auctioning a task for all robots, the computation burden is moved onto individual robots. Auction is also robust because it can continue with the remaining robots even after some robots malfunction. For tasks that have temporal and precedence constraints, their allocation can be formulated as a distributed constraint optimization problem and some approximate methods such as Max-Sum can be used to find a nearly optimal result [12]. A unified auction-based algorithm called TePSSI (Temporal- and Precedence-constrained Sequential Single-Item aution) is proposed in [13] that also provides a solution for such a problem. These solutions are effective when the robot number is small, but cannot be scaled to a large number of robots well.

There exists some optimal solvers for the multi-robot path planning problem, such as conflict-based search [14], combinatorial auctions [15] and ILP-solver [16]. These approaches often find out the best path by exhaustively comparing every potential path. They are not suitable for searching the path of many robots, because the computation complexity increases exponentially with robot number. Some heuristic path planning approaches are thus used for many robots. In [17], a holistic approach with two layers is proposed to coordinate a fleet of automated guided vehicles (AGVs) in an industrial environment, and in [18], a hierarchical framework is developed to resolve the traffic flow prediction, robot path planning, and motion coordination problem. The two approaches only focus on the robot movement and ignores the fact that task allocation has a big influence on robot path. Unlike them, this paper considers the problem of task allocation and robot path planning together.

III. Problem Formulation

As shown in Figure 1, we conduct research on large-scale warehouse environments, including shelves (robot working

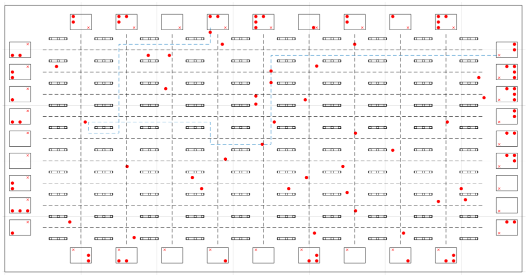

Fig. 1. Example of robot warehouse environment,the red circle represents the robot

points), robot docking stations (such as robot charging area or maintenance area, and cargo delivery area). Consider a large group of mobile robots $R_n, n = 1, 2, ..., N_r$, all free robots are parked at the robot docking station, waiting for the assignment of order tasks.

The specific warehouse environment is described as follows.There are n rows and m rows of shelves distributed inside the warehouse, and each shelf has several pick-up locations. After receiving the task, the storage robot will work at the corresponding pick-up location. At the periphery of the warehouse, there are several robot docking areas where the robots are maintained, charged, docked, etc. Each robot docking station has only one entrance and exit and can park up to 9 robots. The robots are searched in order from the inside out. To park in an empty position, in order to ensure that all robots will eventually have parking spaces, the robot's stopping point must be greater than the total number of robots, as shown in Figure 2.For the roads in the warehouse, the trajectory is given in advance. There are nodes $Q = [q_1, q_2, ..., q_N]$ distributed in the warehouse. There are roads between adjacent nodes, and all road sections constitute the road network of the entire warehouse. In order to reduce robot congestion and improve operating efficiency, each section of the road is a one-way road and follows the principle of driving on the right.Then the adjacency matrix $G_(NN)$ of the entire road can be obtained, $G(i, j)$ represents the weight of node i to node j, if node i and node j are not adjacent, then $G(i, j) =$, If $i = j$, then $G(i, j) = 0$. Figure 2 shows a partial road map.

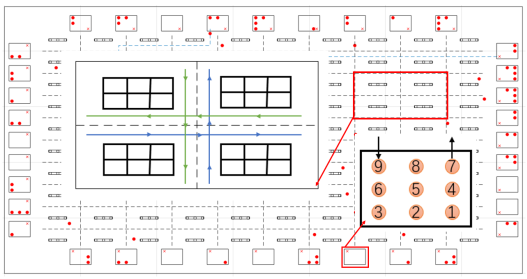

Fig. 2. Examples of robot stops and local roads, the arrow indicates the passable direction

When the warehousing system starts to run, one order task after another will be generated online. The generation of order tasks $W_n, n = 1, 2, ...$ follows the following rules: each time an order task is generated is random;Each order task will contain 1 to 3 different working points (pickup locations) that need to be reached, and the work points will be randomly generated from all the pickup locations; the final destination of each order task is the robot docking station. After the order is generated, it needs to be assigned to the appropriate mobile robot, and a reasonable execution path is generated for the robot. Considering the mobile robot model, the robot R_n runs at a constant speed of v when performing the order task, but there will always be a certain deviation in the speed in the actual process, so the traveling speed of the robot R_n is $v_n = v+e$, where e is a normal distribution $N(0, \sigma^2)$, the mean is 0, and σ is the standard deviation.In order to solve the problem of task allocation and path planning in this kind of storage environment, we have adopted some strategies.

Algorithm 1 Floyd algorithm

```
Input: adjacency matrix $G$
Output: shortest distance matrix $A$ and shortest path matrix $path$
 1: set $A = G$;
 2: $\forall\ i, j\ path(i, j) = -1$;
 3: for $k \in [1, N]$ do
 4:     for $i \in [1, N]$ do
 5:         for $j \in [1, N]$ do
 6:             if $A(i, k) + A(k, j) < A(i, j)$ then
 7:                 set A(i,j)=A(i,k)+A(k,j);
 8:             end if
 9:         end for
10:     end for
11: end for
12: return $A$ and $path$;
```

IV. Solutions to path planning and task scheduling problems

We consider the auction-bid method for task allocation, using the total time cost of the mobile robot to complete the order task as its bid for the order task. In order to calculate the cost of each mobile robot for each order task, it is necessary to know the travel path of the robot to execute the order. We start with the path planning of a single robot, and use the Floyd algorithm to generate the shortest path for a single robot to execute the current bidding order.

A. Generate path by The Floyd algorithm

Like Dijkstra algorithm, Floyd algorithm is also an algorithm for finding the shortest path between vertices in a given weighted graph.Unlike Dijkstra algorithm, Floyd algorithm is used to calculate the shortest distance between any two points in the graph. The Floyd algorithm needs to know the adjacency matrix of the graph. the storage environment determines that the adjacency matrix of the entire road network we know is G.In addition, two NN two-dimensional arrays A and Path need to be introduced, where N is the total number of nodes.The array A is used to store the shortest path weight between any two nodes, and the array $Path$ is used to store the shortest path between any pair of nodes. The content of each cell represents the node passing by from node i to node j.Initially, the adjacency matrix is copied to array A, and all elements of $Path$ are -1. Next, for each node k, check whether the distance from node i to node j is greater than the distance from node i to node k and then to node j, that is, judge $A(i, k) + A(k, j) < A(i, j)$ is established, if it is established, update the array A, another $A(i, j) = A(i, k) + A(k, j)$, and another $Path(i, j) = k$, when the traversal is complete All nodes k, $A(i, j)$ record the shortest distance from node i to node j, and the path from node i to node j can be recursively printed according to the array Path.

The time complexity of the Floyd algorithm is $O(N^3)$, and its time complexity is relatively large, but after executing Floyd once, we can get the array A and $Path$. Considering that the path of the storage environment is fixed, the array A And the array $Path$ does not need to be continuously updated, so it only needs to execute the Floyd algorithm once to store the array A and $Path$, and the time complexity does not affect the auction process.However, the Floyd algorithm only gives the shortest distance between any two points, and each order task will generate up to 3 working points. The order of arrival at the working point will affect the total distance. In order to obtain the total shortest path distance, at the same time Considering that the number of working points is at most 3, we permutate and combine the working points, which will produce at most 6 cases and at least 1 case, traverse each case to obtain a sort of the shortest path distance, the path is The shortest path for the robot to execute the order task.

B. Task allocation based on Auction-bid method and traffic flow prediction

Through the shortest distance matrix A generated by Floyd, for the free robot R_n, we can know its shortest path distance d_n to the newly generated order task. The average driving speed of the robot R_n is $\bar{v} = v$, then its travel time is $t_n = d_n/barv$, in addition, the working time of the robot at the working point needs to be considered. If the working time of the robot at each working point is t_w, then the total working time spent by the robot to complete the order is $tt_n = N_w t_w$,N_w Is the number of work points in the order task, so the total time it takes for the robot R_n to complete the order is $m_n = t_n + tt_n$.If the task list of the robot R_n already has tasks, we follow this principle, regardless of the priority of the order, the robot will complete its orders in turn, when calculating the cost of the new order, it will consider all outstanding tasks in the R_n task list T_n is the total time taken by the robot R_n to complete all orders in the task list (including the order tasks currently under bidding), and W_n is the time it takes for the robot R_n to complete all orders at the work point, then the robot R_ns bid for the latest order It can be expressed as:

$$Bid_n = \alpha \times T_n + (1 - \alpha) \times W_n \tag{1}$$

Where α is a parameter used to specify the importance of travel time and working time. When R_n is an idle robot, $T_n = m_n$, $W_n = tt_n$.

In a large-scale warehousing environment, there are a large number of robots. When the local robot flow or density in the warehousing environment is too large, it is easy to cause robot congestion, which makes the collision between robots more likely and the execution efficiency will be reduce. We consider predicting the flow of regional robots, and use the flow of robots passing through the area as a factor that affects their bids.Consider dividing the cargo hold into m areas, as shown in Figure 3.

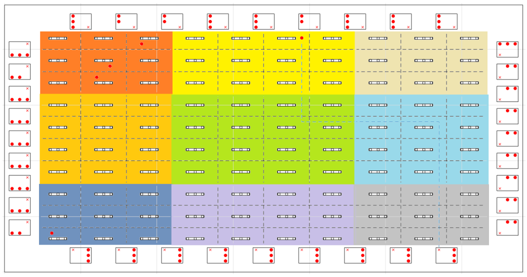
Fig. 3. Example of compartment area division, different colors represent different areas

Let $N(k) = [N_1(k), N_2(k), , N_m(k)]$ denote the number of robots in each area at time k, then the density of robots in area P_i at time k is:

$$\rho_i(k) = \frac{N_i(k)}{S_i}$$

, where S_i is the area of the region P_i. The paths of all robots are known, let $Q^n = [q_1^n, q_2^n, ..., q_i^n, ...]$ be the path of robot R_n, so the positions of all robots at a certain time can be obtained.The appropriate predicted step length m can be determined by considering the path length and the area size, starting from the q_1^n position, calculate the number of robots in the area P_i where the robot R_n is currently located every h path points.For position q_i^n, if

$$q_{i+Kh}^n \in P_j, K = 0 \pm 1 \pm 2, ...$$

When the robot is located in q_{i+Kh}^n, the number of robots in the area P_j is counted as $M_j^n(i + Kh)$.Then let M_j^n is the peak value of robot flow when robot R_n passes through area P_i.

$$M_j^n = max(M_j^n(i + Kh), K = 0, \pm 1, \pm 2, ... \qquad (2)$$

We take the peak traffic of each area of the robot R_n path as a factor that affects its robot's bidding.Use $M^n = [M_1^n, M_2^n, M_m^n]$ to represent the peak traffic in each area passed by the robot R_n on the path generated by the latest order. If the robot R_n does not pass through the area P_j, then $M_j^n = 0$.Then the robot density when the robot R_n passes through each area is $^n = [^n_1, ^n_2, ..., ^n_9]$, so the amount of influence that the flow of the robot path area has on the robot bid is:

$$F = \sum_{i=1}^{m} \rho_i^n \qquad (3)$$

Then, after adding the consideration of regional traffic factors, when the traffic of the robot passing through the area is larger, it will increase its cost of the order, and the robot's bid becomes:

$$Bid_n = \alpha \times T_n + (1 - \alpha) \times W_n + \beta \times F \qquad (4)$$

β is a parameter indicating the importance of the area traffic of the robot pathway to the total bid.

Add the flow of the robot's route area to the auction cost. When the robot needs to pass through the area with high traffic, its bid will be higher, then the possibility of order allocation to it will be reduced, so the aggregation of robots can be reduced. To a certain extent, reduce the possibility of robots encountering each other, and also reduce the possibility of blockage.

C. Robot conflict problem

When we used Auction-bid for task allocation, we used the Floyd algorithm in advance to generate the path for each robot to complete the order task in order to calculate the robot's bid. But in this way, the paths generated by each mobile robot are independent of each other, and the mutual influence between the robots in the entire storage system is not considered. In the actual driving process, robots may influence each other and cause conflicts. For this reason, we adopt a simple method of concession to resolve conflicts and avoid collisions.In fact, since the road network of the warehouse has been planned and the principle of one-way driving is followed, the conflicts between robots are greatly reduced. There are mainly two possible collisions: cross-collision and rear-end collision, as shown in Figure 4. In the discrete environment model, for the robot R_n, its position at time k is $q^n(k)$, then the following situations need to consider the conflict problem:

1) $\forall R_n, R_m, n \neq m, \|q^n(k+1) - q^m(k+1)\| < \epsilon$
2) $\forall R_n, R_m, n \neq m, \|q^n(k+1) - q^m(k)\| < \epsilon$

ϵ is the safety distance between robots.

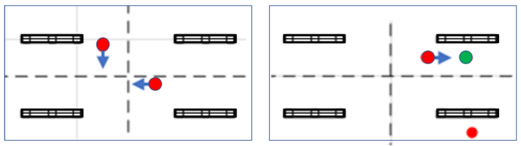
Fig. 4. The left side is a cross collision, the right side is a rear-end collision, the green robot indicates the working state, and the red robot indicates the driving state

For the state s^n of the robot R_n, there are three situations: stop state(s_1^n), driving state(s_2^n), and working state(s_3^n).When the robots R_n and R_m are both in the driving state, a cross-collision may occur at the intersection; while the robot R_n is in the working state, the R_m in the driving state behind may have a rear-end collision with it. Let the two-digit array *crash* indicate whether there will be conflicts between the robots, that is, the distance between the robots is less than the safety distance ϵ.$crash(i, j) = 1$ means that the robot R_i conflicts with the robot R_j. Since the paths of all robots are known, we predict the positions of all robots after time T in order to react in advance to avoid conflicts.At time k, if $s^i = s_2^n$, $s^j = s_2^n$, and $crash(i,j) = 1$ at $k+T$ time, then we consider letting the robots R_i and R_j have the smaller number Give in, that is, stop and wait until $crash(i,j) = 0$.If $s^i = s_2^n$ and $s^j = s_3^n$, let the robot R_i stop and wait until the robot R_j finishes its work.Since we have forecasted regional traffic, we have reduced the possibility of a large number of robots gathering in the same area, making it acceptable for time delays due to occasional conflicts.

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

References

[1] C. Liang, K. Chee, Y. Zou, H. Zhu, A. Causo, S. Vidas, T. Teng, I. Chen, K. Low, and C. Cheah, "Automated robot picking system for e-commerce fulfillment warehouse application," in *The 14th IFToMM World Congress*, 2015.

[2] A. Gharehgozli and N. Zaerpour, "Robot scheduling for pod retrieval in a robotic mobile fulfillment system," *Transportation Research Part E: Logistics and Transportation Review*, vol. 142, p. 102087, 2020.

[3] X. Zhao, X. Chen, X. Song, W. Zhang, L. Gao, and R. De Ciel, "A look at alibaba double 11 shopping festival," *Journal of Student Research*, vol. 8, no. 1, 2019.

[4] I. Jang, H.-S. Shin, and A. Tsourdos, "Anonymous hedonic game for task allocation in a large-scale multiple agent system," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1534–1548, 2018.

[5] P. K. Muhuri and A. Rauniyar, "Immigrants based adaptive genetic algorithms for task allocation in multi-robot systems," *International Journal of Computational Intelligence and Applications*, vol. 16, no. 04, p. 1750025, 2017.

[6] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[7] J. Yu and S. M. LaValle, "Optimal multi-robot path planning on graphs: Structure and computational complexity," *arXiv preprint arXiv:1507.03289*, 2015.

[8] C. Sarkar, H. S. Paul, and A. Pal, "A scalable multi-robot task allocation algorithm," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5022–5027, IEEE, 2018.

[9] D. Sarne and S. Kraus, "Solving the auction-based task allocation problem in an open environment," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 20, p. 164, 2005.

[10] E. Schneider, E. I. Sklar, S. Parsons, and A. T. Özgelen, "Auction-based task allocation for multi-robot teams in dynamic environments," in *Conference Towards Autonomous Robotic Systems*, pp. 246–257, Springer, 2015.

[11] J. Shi, Z. Yang, and J. Zhu, "An auction-based rescue task allocation approach for heterogeneous multi-robot system," *Multimedia Tools and Applications*, vol. 79, no. 21, pp. 14529–14538, 2020.

[12] S. Ramchurn, A. Farinelli, K. Macarthur, M. Polukarov, and N. Jennings, "Decentralised coordination in robocup rescue," *The Computer Journal*, 2009.

[13] E. Nunes, M. McIntire, and M. Gini, "Decentralized multi-robot allocation of tasks with temporal and precedence constraints," *Advanced Robotics*, vol. 31, no. 22, pp. 1193–1207, 2017.

[14] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[15] O. Amir, G. Sharon, and R. Stern, "Multi-agent pathfinding as a combinatorial auction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, 2015.

[16] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016.

[17] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Ensemble coordination approach in multi-agv systems applied to industrial warehouses," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 922–934, 2015.

[18] Z. Liu, H. Wang, H. Wei, M. Liu, and Y.-H. Liu, "Prediction, planning, and coordination of thousand-warehousing-robot networks with motion and communication uncertainties," *IEEE Transactions on Automation Science and Engineering*, 2020.