Multi-Robot Task Allocation in E-commerce Warehouses: A Comparative Analysis of Distance Minimization and Priority-based Approaches

1st Student 1
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Student 2
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Rahul Singhal
Dept. of Mechanical-Mechatronics Engg.
LNMIIT Jaipur, India
rahul.singhal@lnmiit.ac.in

4th Mohit Makkar
Dept. of Mechanical-Mechatronics Engg.
LNMIIT Jaipur, India
mohit.makkar@lnmiit.ac.in

***Abstract*—This paper presents a methodology for warehouse management with multiple robots for the efficient picking and dropping of consignments. Task allocation is a critical challenge in this process, as it requires determining a collision-free path for all robots based on warehouse management preferences. Two task allocation preferences are considered: (1) minimizing the total distance travelled by all robots to reduce power consumption and improve overall efficiency, and (2) prioritizing task allocation by assigning tasks to the nearest available robot for faster delivery. The proposed approach uses the A* algorithm to determine the shortest path for each robot and evaluates the overall lower cost with the Kuhn-Munkar algorithm. Simulation results show the effectiveness of the methodology in achieving efficient task allocation and reducing the overall distance travelled.**

***Index Terms*—Multi-robot, warehouse management, task allocation, path planning, task-priority collision avoidance**

I. Introduction

E-commerce logistics involves the complex process of receiving, sorting, storing, and distributing goods to multiple customers. One of the crucial tasks in this process is order fulfilment, where employees inside the warehouse physically initiate item search operations and pick-up tasks. However, picking operations can be repetitive and routine for humans, and they may cost up to 55% of the entire warehouse's operating expense. To overcome these challenges, robots have been proposed to sort, store, and distribute goods inside the warehouse. The high cost of picking operations and the need for faster order fulfilment have been the driving force behind the adoption of robots in the logistics industry [1]. Moreover, mobile robots have been found to improve warehouse efficiency and reduce labour costs [2]. The warehouse facilities will have 30% mobile robots by the end of 2023, requiring automation for some of their processes. Mobile robots have shown promise in sending and receiving items in multiple orders. However, determining the best path for these robots can be increasingly complicated. Therefore, an efficient strategy for warehouse management's preference in choosing a robot's path is of utmost importance.

Recent research has suggested several techniques for implementing multi-robot systems in e-commerce logistics. [3] proposed a multi-robot scheduling strategy for e-commerce order fulfilment to optimize the processing time of orders. [4] developed a multi-robot coordination approach to enhance order fulfilment efficiency by allocating tasks to the robots based on their respective capabilities. [5] proposed a deep reinforcement learning approach to improve multi-robot cooperation for warehouse order fulfilment. The study used a centralized training method to teach the robots how to cooperate, communicate, and allocate tasks for optimal performance. [6] proposed a dynamic routing strategy for multi-robot order picking in e-commerce warehouses. The approach used a genetic algorithm to determine the optimal route for each robot based on task allocation and inventory information. [7] conducted a case study of collaborative order picking by multiple robots in e-commerce warehousing. The study compared the performance of different collaboration modes and identified the benefits of collaborative order picking in reducing processing time and increasing efficiency. These studies demonstrate the potential of multi-robot systems in improving order fulfilment in e-commerce logistics. However, further research is needed to address challenges such as collision avoidance, path planning, and task allocation to fully exploit the benefits of using robots in warehouse operations.

In this paper, a methodology for task allocation using multiple robots in the picking operation of items is presented. The methodology considers two preferences for task allocation: (1) to obtain an optimized path such that the total distance representing the cost of all the robots is minimum, and (2) to assign tasks to the robots based on their given priority. For task allocation, a cost matrix is obtained using the A* algorithm, which represents the shortest distance of each robot

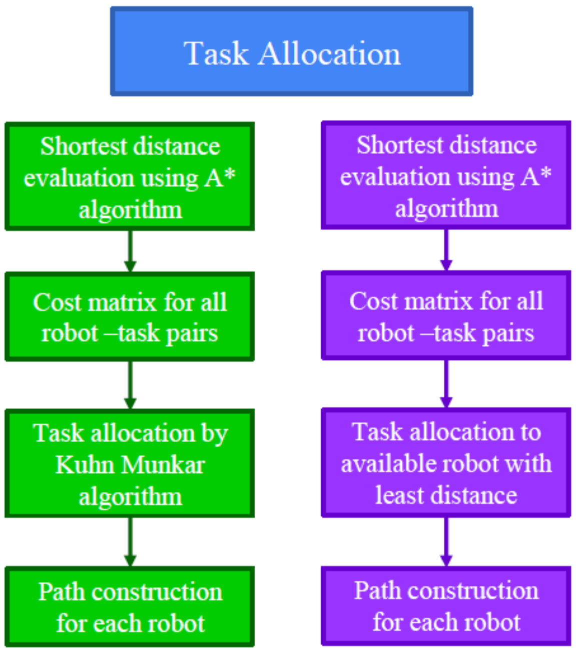

Task Allocation
Shortest distance evaluation using A* algorithm
Cost matrix for all robot –task pairs
Task allocation by Kuhn Munkar algorithm
Path construction for each robot
Shortest distance evaluation using A* algorithm
Cost matrix for all robot –task pairs
Task allocation to available robot with least distance
Path construction for each robot

Fig. 1: Task allocation methodologies considered for warehouse management. For minimum total distance, the methodology is presented in green. The prioritised task allocation methodology is shown in purple.

to all tasks.

In section 2, the methodology for task allocation is presented. The approach involves using the A* algorithm to determine the shortest robot path for the picking operation. The overall lower cost is evaluated with the Kuhn-Munkar algorithm. For the prioritized task allocation, tasks are allocated based on their priority to available robots with the least distance. In section 3, the results obtained from the simulation scenario are presented, followed by a discussion of the findings. Finally, section 4 presents the conclusions of the overall minimum distance and priority task allocation for the multi-robots.

II. Methodology

Warehouse management involves the task of allocating orders to robots in order to fulfil the needs of multiple customers. This task allocation process can be repetitive and easily automated by using mobile robots for tasks such as storing, distributing, and delivering items in the warehouse. In order to ensure efficient and effective movement of the robots, it is essential to plan their paths in advance, thereby minimizing the time and distance required to complete the tasks.

This paper proposes two task allocation methodologies for warehouse management, as shown in Figure 1. The cost matrix is used to determine the shortest path between the robots and the individual tasks, which is calculated using the A* algorithm. The first methodology uses the Kuhn-Munkres algorithm to generate optimal paths by assigning tasks to robots with the least total distance for all robots.

The second methodology involves priority task allocation, where the highest priority task is assigned to the nearest available robot from the cost matrix. In this methodology, only the task priorities are considered in sequence from highest to lowest, without any concern for the overall total cost. These two methodologies are implemented separately to analyze their respective efficiencies and trade-offs.

A. *A* Algorithm*

The A* algorithm [8] is a widely used heuristic search algorithm that can be used to find the shortest path between two points in a graph or grid. The algorithm works by expanding nodes in the graph or grid based on the estimated cost of reaching the goal. The cost of each node is calculated as the sum of the cost of reaching that node from the starting point and the estimated cost of reaching the goal from that node. The estimated cost is usually calculated using a heuristic function that provides an estimate of the remaining distance to the goal. By considering both the cost and the heuristic, A* can find the optimal path quickly and efficiently, even in large graphs.

Algorithm 1: A* algorithm

```
Input: start node, goal node, cost function, heuristic
       function
Output: shortest path from start to goal
Initialize an empty set called "open", and add the start
  node to it; Initialize an empty set called "closed";
while open is not empty do
    Find the node in the open set with the lowest
      f-score and call it "current"; Remove "current"
      from the open set and add it to the closed set; if
      "current" is the goal node then
        Return the path;
    end
    for each neighbor of "current" do
        if neighbor is in the closed set then
            Skip to the next neighbor;
        end
        Calculate the tentative g-score for the neighbor;
          if neighbor is not in the open set then
            Add neighbor to the open set;
        end
        else if the tentative g-score is greater than the
          neighbor's current g-score then
            Skip to the next neighbor;
        end
        Set the parent of neighbor to "current"; Update
          the g-score, h-score, and f-score for neighbor;
    end
end
Return failure;
```

The pseudocode of the A* algorithm is given in Algorithm [1] to find the shortest path between two nodes in a graph, and it includes the heuristic function to guide the search towards the goal node. The algorithm uses sets to keep track of the nodes that have been visited (closed set) and the nodes that are yet to be visited (open set). It also calculates the g-score, h-score, and f-score for each node, where g-score is the cost of the path from the start node to the current node, h-score is the estimated cost of the path from the current node to the goal node, and f-score is the sum of g-score and h-score. The algorithm iteratively expands the node with the lowest f-score until it reaches the goal node or fails to find a path.

The A* algorithm is known for its efficiency in finding optimal paths in large search spaces. It guarantees to find the shortest path if the heuristic function used is admissible, meaning it never overestimates the true cost of reaching the goal. Additionally, the algorithm can be modified to include additional constraints or preferences, such as avoiding certain areas or favouring specific paths. Overall, the A* algorithm is a powerful tool for solving pathfinding problems in various applications, including robotics, video games, and navigation systems.

B. *Kuhn-Munkres Algorithm*

The Kuhn-Munkres Algorithm [9], also known as the Hungarian Algorithm, is a well-known algorithm for solving the Assignment Problem, a combinatorial optimization problem. The Assignment Problem involves assigning n workers to n jobs with different costs. The goal is to find an assignment that minimizes the total cost or maximizes the total profit. The Kuhn-Munkres algorithm solves this problem in polynomial time $O(n^3)$, where n is the size of the problem.

The algorithm works by creating a matrix of costs, where the rows represent the workers and the columns represent the jobs. Each cell (i,j) of the matrix contains the cost of assigning worker i to job j. The algorithm then proceeds in two phases.

In the first phase, the algorithm transforms the cost matrix into a matrix of weights, where each weight is obtained by subtracting the minimum cost in the row and column of the corresponding cell. The goal of this phase is to find a set of independent zeros in the resulting matrix. If there are n independent zeros, the problem is solved, and the zeros represent the optimal assignment.

In the second phase, the algorithm searches for augmenting paths in the weight matrix. An augmenting path is a path of alternating zeros and non-zeros that starts and ends with zeros. The algorithm uses a depth-first search to find an augmenting path and updates the assignment matrix accordingly. This phase continues until no more augmenting paths can be found, at which point the algorithm terminates and returns the optimal assignment.

The Kuhn-Munkres Algorithm is a powerful tool for solving the Assignment Problem, and it has been applied in a wide range of fields, including computer vision, logistics, and economics.

III. Results and Discussion

The multi-robot simulation consists of six robots. The six-picking item operation is considered for task allocation. The arena considered with robots, tasks, and racks is shown in Fig. 2. The racks shown in the black box represent the storage space in the warehouse for the storage of the items. The several pickup operations of an item as tasks for robots are shown in a green box. The robot is allowed to move up, down, left, right, and diagonally across the free space represented with white blocks. The robot movement consisting of moving in either up, down, left, or right has been considered as one unit, and for diagonal motion 1.4 units.

TABLE I: Simulation preference considered for picking task allocation

Case	**Description**
Case A	Overall lowest total distance for all robots
Case B	Prioritised task allocation

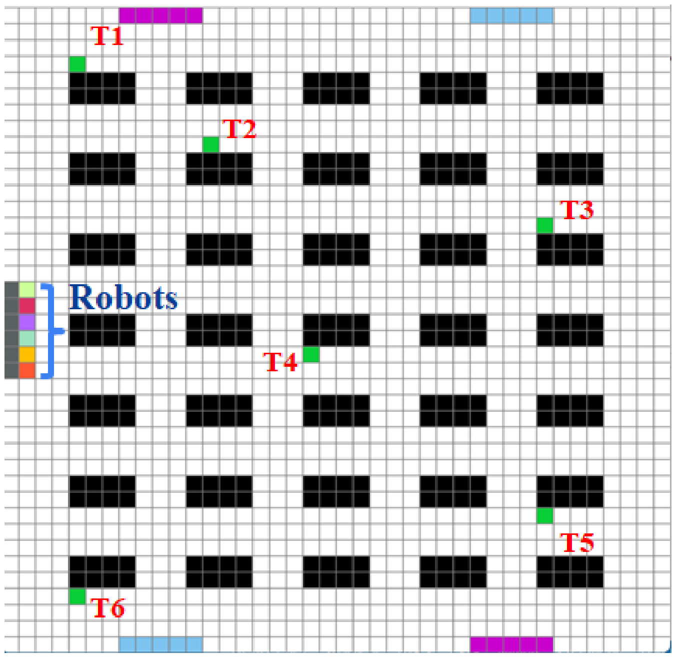


Fig. 2: Warehouse considered for task allocation. The racks for storing items are shown as ■. The picking task sites for robot are shown in 🟩.

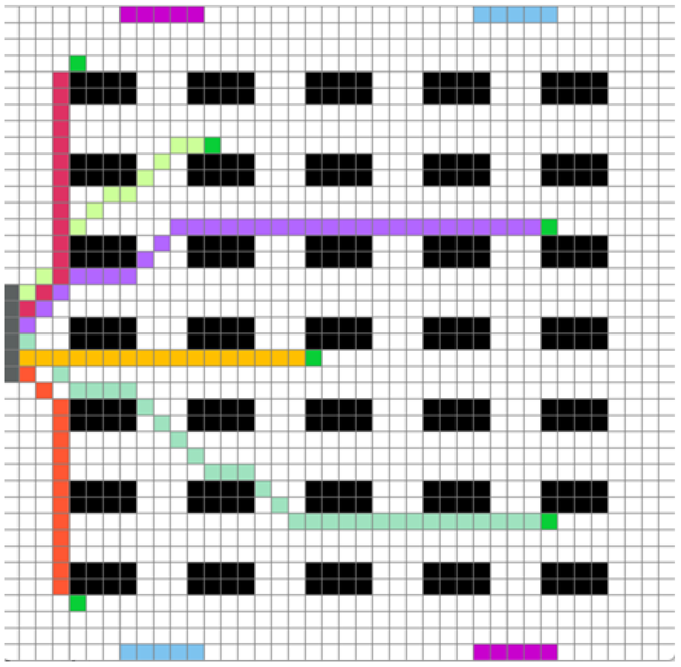
The task allocation preferences for warehouse management are presented in Table I. Case A represents the scenario requiring minimum total distance for all robots, signifying the lowest power consumption for all robots combined. Case B represents the situation in which each task of the picking operation has different priorities. This can be due to the user membership type, delay in expected delivery time, battery SOC, etc. For both cases, the cost map is obtained using A* algorithm. For Case A, the least total distance for all robots is obtained using the Kuhn-Munkres algorithm. For Case B, the tasks are ranked and allocated based on the given priority list, and the task is allocated to the available robot having the least distance in the cost matrix.

TABLE II: Case A : Tasks allocation with least total distance for all robots by Kuhn-Munkres algorithm.

	Task 1	**Task 2**	**Task 3**	**Task 4**	**Task 5**	**Task 6**
Robot 1	15.2	15.2	33.2	19.2	37.2	20.2
Robot 2	16.2	16.2	34.2	17.8	35.8	18.2
Robot 3	17.2	17.8	33.4	17.8	35.8	18.2
Robot 4	18.2	18.2	33.8	17.4	35.4	17.2
Robot 5	19.2	18.6	34.2	17	36.2	16.2
Robot 6	20.2	19.6	35.2	17.4	35.2	15.2
Total Distance	16.2 + 15.2 + 33.4 + 17 + 35.4 + 15.2 = **132.4**					

TABLE III: Case B : Prioritised task allocation of robots.

	Task 1 (Priority 1)	**Task 2 (Priority 2)**	**Task 3 (Priority 3)**	**Task 4 (Priority 4)**	**Task 5 (Priority 5)**	**Task 6 (Priority 6)**
Robot 1	15.2	15.2	33.2	19.2	37.2	20.2
Robot 2	16.2	16.2	34.2	17.8	35.8	18.2
Robot 3	17.2	17.8	33.4	17.8	35.8	18.2
Robot 4	18.2	18.2	33.8	17.4	35.4	17.2
Robot 5	19.2	18.6	34.2	17	36.2	16.2
Robot 6	20.2	19.6	35.2	17.4	35.2	15.2
Total Distance	15.2 + 16.2 + 33.4 + 17 + 35.2 + 17.2 = **134.2**					


(a) Case A task allocation with minimum total cost

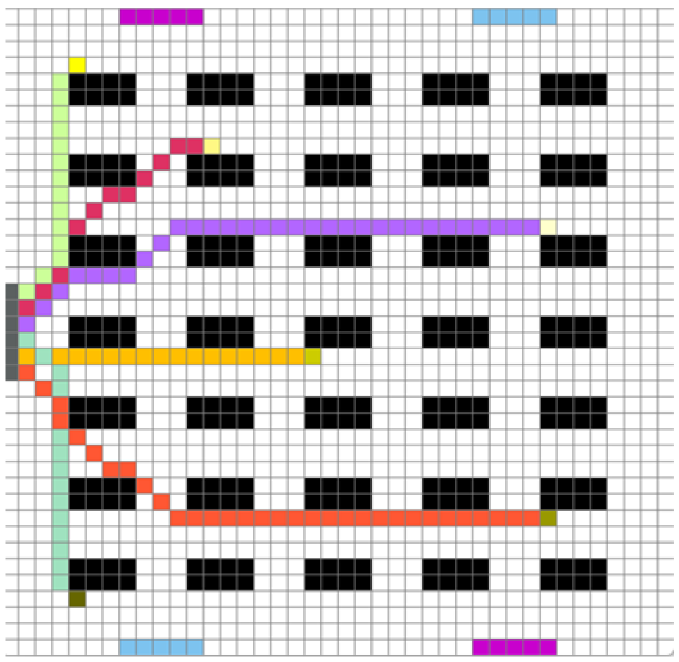

(b) Case B Prioritised task allocation to robots

Fig. 3: Path obtained for the considered simulation scenarios.

The result obtained for Case A representing the cost matrix and task allocation for the respective robot is shown in Table II. The least total cost obtained from the Kuhn-Munkres algorithm was 132.4 units. The selection of the respective robot for the task is highlighted in the table with a blue background. The obtained path of each robot for the arena is shown in Fig. 3a.

The results obtained for Case B are shown in III. The tasks are allocated from highest to lowest priority based on the given priority list. For this scenario, Task 1 has the highest priority, and Task 6 has the least priority. Task 1 owing the highest priority is considered first for robot allocation for the fastest task completion. Robot 1 had been allocated to this task because it had the least distance in the cost matrix for that task. Similarly, other tasks were assigned based on the priority order to the available pool of robots depending on the least distance they had for the particular task. The total cost obtained for this scenario was 134.2 units. The path obtained for all robots is shown in Fig. 3b.

The different total distances for task allocation scenarios in warehouse management can be observed for both cases. The partial dissimilarity in robots' paths can be observed in Fig. 3a and Fig. 3b. The diverse task allocation can be observed primarily for Task 5 and Task 6. In Case A, if the Task 1 and Task 2 allocation were interchanged would yield the same total distance. The changes in task allocation are because of the mismatched preferences considered. The total distance obtained from Case A should always be greater than Case B, as Case A objective is to bring the optimal solution having the least total distance for all robots, whereas Case B has no preference for the least total distance.

Overall, the results show that the proposed methodology can effectively allocate tasks in warehouse management scenarios with different preferences. The Kuhn-Munkres algorithm can be used to achieve optimal task allocation when the objective is to minimize the total distance for all robots, while prioritized task allocation can be used to allocate tasks based on their priority when the objective is to complete tasks in a specific order.

IV. Conclusions

This paper represents the task allocation methodologies for warehouse management based on two preferences: minimum total distance for all robots and prioritized task allocation. The A* algorithm was used to determine the shortest distance for each robot to all the tasks and generate a cost matrix. The Kuhn-Munkars algorithm was then used to obtain the minimum total distance from the cost matrix. In prioritized task allocation, the tasks were assigned based on the priority list and assigned to the nearest available robot from the cost matrix. The chosen task allocation preference can impact the overall strategy for warehouse management, with prioritized task allocation providing more flexibility but potentially resulting in the same or higher total distance for all robots. Overall, this study provides insights into effective task allocation methodologies for warehouse management using mobile robots.

References

[1] B. Melián-Batista, P. Mendoza-Moreno, and J. L. Torres-Moreno, "An overview of mobile robots in warehouse management," in *Mobile Robotics*. Springer, 2021, pp. 129–145.

[2] E. Lopez-Rojas, A. Aguilar-Meléndez, and J. Rodriguez-Molina, "Mobile robots for warehouse logistics: A review," *Robotics*, vol. 10, no. 1, p. 34, 2021.

[3] S. Wang, W. Liu, K. Jiang, and H. Cai, "A multi-robot scheduling strategy for e-commerce order fulfillment," *Robotics and Computer-Integrated Manufacturing*, vol. 48, pp. 81–88, 2017.

[4] Y. Zhang, Y. Song, and J. Yan, "Multi-robot coordination for order fulfillment in e-commerce warehouse," *Journal of Intelligent & Robotic Systems*, vol. 94, pp. 137–153, 2019.

[5] D. Tan and X. Xu, "Multi-robot cooperation for warehouse order fulfillment: A deep reinforcement learning approach," *IEEE Access*, vol. 9, pp. 10 321–10 331, 2021.

[6] X. Chen, W. Zhang, and J. Gu, "A dynamic routing strategy for multi-robot order picking in e-commerce warehouses," *Computers & Industrial Engineering*, vol. 142, p. 106305, 2020.

[7] C. Wang, Z. Wang, and H. Zhang, "Collaborative order picking by multiple robots in e-commerce warehousing: A case study," *International Journal of Production Research*, vol. 57, no. 11, pp. 3652–3671, 2019.

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

[9] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proceedings of the second Berkeley symposium on mathematical statistics and probability*, vol. 1. University of California Press, 1955, pp. 481–492.