2022 19th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE). Mexico City, Mexico. November 9-11, 2022

A Priority-based Modified A* Path Planning Algorithm for Multi-Mobile Robot Navigation

Chadi Riman
College of Engineering and Technology
American University of the Middle East
Egaila, Kuwait
chadi.riman@aum.edu.kw

Pierre E. Abi-Char
College of Engineering and Technology
American University of the Middle East
Egaila, Kuwait
pierre.abichar@aum.edu.kw

***Abstract*—Automated storage and retrieval systems (ASRS) are designed for automated storage and retrieval of parts and items in specific locations within manufacturing, warehouses, institutions, and others. These systems are usually comprised of predefined locations where machines/robots can follow established routes to get items. The Path planning belongs to shortest path problem. For these robots, it is important to find the shortest path, without causing any conflict among them. Several algorithms exist to find shortest path, notably Dijkstra and A*, which are famous to solve this kind of problem. Additionally, a preference is recommended in such environment for the most important robotic task among these robots. The preferred robot will be selected as the one with highest priority to accomplish the task. In this work, we suggested a simple and efficient algorithm based on A* algorithm to find the path planning with collision-free for these robots with priority among these machines, so that to have one machine to be of higher priority than all of the others robots. We proposed two priority algorithms A_1 and A_2. We also did simulation results to show the efficiency of the proposed priority algorithms. The new priority algorithm showed to be very effective with both A_1 and A_2 algorithms.**

Keywords— A* algorithm, Collision-Free, Priority Path Planning, Mobile Robots, Shortest Path, Sensors, Warehouses.

I. INTRODUCTION

An automated storage and retrieval system (ASRS) is a type of warehouse automation technology specifically designed to store and retrieve products on demand. Automated storage and retrieval systems have rapidly advanced in the recent years and a variety of robotic systems have been developed including in smart environments [1], airports [2], malls [3], laboratories [4]. The main challenges to ASRS are: solving the path planning for finding shortest path in a minimum amount of time, avoid collisions, and provide a priority scheme among robots involved in the system. Shortest-path algorithms are used in computer science to find the shortest path between two points on a graph or grid. Shorted-path algorithms are adapted by several applications like network route planning, traffic control, transportation systems, operation research, and others. Dijkstra's algorithm is a famous algorithm that is used for finding the shortest path, from starting node to target node in a weighted graph. This algorithm makes a tree of the shortest path from the starting node to all other nodes in the graph. It makes use of weights of the edges for finding the path that minimizes the total distance from the source node to all other nodes. This algorithm is also known as the single-source shortest path algorithm. The A* algorithm is just like Dijkstra's algorithm. The only difference is that A* algorithm tries to look for a better path by using a heuristic function H which gives priority to nodes that are supposed to be better than others while Dijkstra's algorithm just explores all possible paths.

In an environment with multiple robots going on different paths, the priority issue is often considered. This is important in order to determine which robot will pass first if a conflict occurs. A conflict is when two or more robots pass at an intersection in their different paths at the same time. We need to make sure that other robots will wait for higher priority robots, so that no collision occurs. The highest priority robot is considered first, and then lower priority robots are considered next. Usually each robot searches its own path in advance without considering other robots, using the regular minimal path searching techniques such as Dijkstra, A*, and others. If there exists a potential collision, they can negotiate to solve the incoming issue. The solution is that the higher priority robot will pass first, while the other will wait until its path is clear. The priority can be assigned statically in advance or dynamically during the robots' movement.

In 2021, AbiChar et al. [5] suggested a simple and efficient algorithm to find the path planning with collision-free for non-complex automated storage systems. The algorithm was based on an enhanced version of A* algorithm to find the shortest path from a source node to a destination node, with two heuristic functions H_1 and H_2. They developed a simulation model to evaluate the performance of their proposed scheme using different scenarios, in terms of number of robots, types and numbers of obstacles. Simulation results showed very high efficiency of the proposed collision-free path planning algorithm.

In this paper, we present an improvement to the work proposed in [5] by the addition of priority. Our priority path planning algorithm is using one of two proposed algorithms in order to search for a collision free priority path for the main robot. The environment map for mobile robots is described by using the grid method. Computer experiments for different scenarios are conducted to evaluate the performance of the proposed algorithms. The simulation results demonstrate the effectiveness of the proposed system.

The rest of this paper is organized as follows. In Section II, we briefly survey the relevant literature review. Section III

978-1-6654-5508-4/22/$31.00 ©2022 IEEE

presents system model requirements and preliminaries. Section IV describes the proposed path planning with collision-free algorithm. In sections V, applications to AS systems withsimulation discussions are presented. Finally, we conclude the paper in Section VI and present possible future work.

II. Review of Previous Research

In the recent past years, several approaches have been proposed to handle path planning for robots in general, and for automated storage and retrieval systems in particular. A few of them handled the collision avoidance, while others also added a priority scheme.

The standard shortest path algorithms have been developed by Bellman [6], Dijkstra [7], and Dreyfus [8]. In 2019, Zheng et al. [9] optimized the node search mode and search speed, and add the angle evaluation cost function to the cost function of A^* algorithm to find the path with the least inflection point. In 2021, Li et al. [10] suggested an improved greedy algorithm to evaluate the shortest path planning of *ASRS* systems. The algorithm proved to be faster than the conventional greedy algorithm. It included path-priority concept based on Dijkstra, and obstacle avoidance strategies. In 2004, authors [11] suggested a priority based algorithm to solve dynamic multiple robot path planning. The algorithm relied of having coordination among different mobile robots. The method can guarantee collision-free paths for each robot but needed further investigation and implementation. However, it was not applied to *ASRS* systems. In 2020, authors [12] suggested a priority-based speed control strategy in conjunction with the Dijkstra depth-first search algorithm to achieve conflict-free path planning. This model was applied on automatic guided vehicles(*AGV*). The path was planned based on *AGV s* having high or low priority. In 2015, Novak et al. [13] proposed two algorithms: a revised prioritized planning, and another asynchronous implementation of this revised prioritized planning with a faster conversion. The experiments showed that the first method solves priority cases with failure, and that the second method provides a faster solution.

Authors in [14] proposed a Priority-based Optimized-Hybrid Asynchronous Centralized and Decentralized algorithm for multi-robot path planning issue where multiple robots communicate with each other. The algorithm prevented unnecessary communication by transmitting global information and limited specific information that is only in the interest of certain robots. In 2021, Huang et al. [15] suggested a navigation strategy with path priority for multiple robots moving in a large flat space. The robots can be both dynamic and static. The highest priority robot does not need to change its path, while lower priority robots might change their path to avoid collisions with higher priority robots. In 2016, authors [16] suggested a modified A^* algorithm to determine the heuristic function's value just before the collision phase rather than initially. It showed a much smaller processing time with higher speed than regular A* algorithm, but with a little bit longer path.

Authors in [17] proposed a priority-based coordination of robots in order to pass through an intersection. Priorities are assigned as robots enter the circuit; first one entering is given the highest priority, and so on, until the last one entering getting the least priority. This system ensures that all collisions are avoided in order to pass through the intersections. In 2016, Zeng et al. [18] applied Hierarchical Task Network (HTN) method to multi-robot path planning on a grid. The method proved to be more effective than A∗ method and the testing showed smaller total path length and completion time than A*. Authors in [19] introduced a prioritized method to address the problem of motion planning for multiple robots. The method proved to be fast where even problems more than ten robots in confined environments were solved in just seconds of computation time. However, it was assumed that the robots were collision-free, so that the work done did not deal with collision failures in the experiments.

III. Model Requirements and Preliminaries

In this section, we describe our requirements for designing a path planning collision-free scheme for automated storage systems. Next, We briefly present the A∗ algorithm which our proposed algorithm is based on.

A. Model Requirements and Assumptions

The model environment for solving a path planning can be reduced to an undirected connected graph G = V, E, L , where V is the set of nodes, E is set of edges, and L represents the effective length between two nodes. The requirements and assumptions are as follows:

- The path width between two nodes can only accommodate one mobile robot at a time.
- The running speed is constant and same for all mobile robots under normal operation condition.
- Mobile robots direction is considered at initial starting point.
- The path taken into consideration can be single way from start to target.
- Priority for a mobile robot in terms of reaching a goal before another one is considered and will be defined in the next section

B. Our Previous Modified A* Algorithm

In our previous work [5], we proposed a modified Dijkstra A* algorithm to solve the path planning problem for ASR systems by taking into consideration the dynamic obstacles avoidance. The Dijkstra A* algorithm has a static table that contains approximate distances to the goal.

IV. Proposed Priority-Based Path Planning With Collision-Free Algorithm

The proposed scheme consists of three main parts, namely the modified Dijkstra A* with priority algorithm, the two detailed priority methods and the return path calculation. These three parts are described as follow:

A. Modified A* with Priority Algorithm

We propose a modified Dijkstra A* algorithm that takes into consideration the priority in addition to the dynamic obstacles avoidance. The Dijkstra A* algorithm has a static table that contains approximate distances to the goal. This

table is usually given or calculated once before finding the shortest path. By changing the algorithm so that this table H is calculated once and then locking the path, the lock will make sure that no other moving robots will jump into the locked path. This gives us the ability to make the priority robot move as quickly as possible without having to avoid any other robot since they will avoid stepping into its path. Furthermore, the modified Dijkstra A* algorithm will take into consideration the steps spent to rotate the robot into the right direction. We assume that the time needed to rotate the robot in +/- 90 degrees is equal to one move. So to rotate the robot in 180 degrees, it will take two moves. The general priority algorithm is shown next.

Algorithm 1 Modified A* Algorithm With Priority

1. Select the Start node V, and the Goal node G.
2. Create a set S with only the starting node V, Make current node i=V, Dist(i)=0, Dist is the traveled distance so far from V. Direction=initial direction of robot located in V (right/left/up/down).
3. Calculate approximate distance from G to all nodes removing obstacle nodes. Put result in table H. H(i) designates approximate distance from G to node i, without the needed rotations, if any. There are two functions: H1 and H2. H2 adds the approximate minimum needed angle rotations for the travel.
4. Check if (H(j) = ∞), wait one cycle and go back to step 3. Repeat this for 10 cycles. After that, announce failure to find a path. For practicality, infinity is 99.
5. Loop from step 6 to 11 to build a priority path.
6. If (H(i)<= ∞), select all neighbor nodes to i as the candidate intermediate nodes. Keep record of the direction of the neighbor node with respect to the current node (right/left/up/down).
7. Select the node j with the smallest (2*Number + Needed Rotation) among the candidate intermediate nodes and add it into set S. Each movement step is considered twice as important as the rotation, and thus the multiplication by 2 in the above expression. Also update new Direction=current direction after needed rotation.
8. Update Dist(j) = Dist(i) + C(i,j), where C(i,j) is the direct link distance between i and j including any needed rotation.
9. Add node i to the visited path.
10. Regard j as the new intermediate node. Mark: i = j
11. Check if Goal G is reached (i=G)? If so, save the priority path and the needed distance. If not, go back to step 6.
12. Using the saved priority path, get the next node.
13. If next node is occupied, and it will not be released in one step (Algorithm 2), release the priority path and go back to step 3.
14. Add node i to the visited path.
15. Regard j as the new intermediate node. Mark: i = j
16. Check if Goal G is reached (i=G)? If so, announce success, the needed distance, and the traveled path. If not, go back to step 12.
17. End of Algorithm

B. *Priority Algorithms A_1 and A_2*

We propose two ways for the non-priority robots to check their paths: A_1 and A_2. A_1 will check the next node for each non-priority robot and compare it with the next two nodes of the priority robot's path. If there is an intersection, then the non-priority robot will not move. A_2 will check the next two nodes for each non-priority robot and compare them with the next node of the priority robot's path. If there is an intersection, then the non-priority robot will not move. The specific priority algorithms A_1 and A_2 are shown next.

Algorithm 2 Priority Algorithm 1 (A_1): applied to non-priority robots:

1. For each non-priority robot, check the next node N .
2. Compare the next node N with next two nodes of the priority robot's path: PriorityPath[nextStep] and Priority-Path[nextStep+1].
3. If there is intersection N=PriorityPath[nextStep] or N=PriorityPath[nextStep+1], wait for one step, them go back to step 2.
4. Move the robot to the next node.

Algorithm 3 Priority Algorithm 2 (A_2): applied to non-priority robots:

1. For each non-priority robot, check the next two nodes N and N2.
2. Compare the next two nodes N and N2 with next node of the priority robot's path PriorityPath[nextStep].
3. If there is intersection N=PriorityPath[nextStep] or N2=PriorityPath[nextStep], wait for one step, then go back to step 2.
4. Move the robot to the next node

V. Application to Automated Storage Systems

To verify the effectiveness of the proposed algorithm, extensive simulations for a range of scenarios for automated storage, AS, are carried out. The developed program was written in C# language and run on Window 10 Operating System. Updates were included into the original program to have the priority path.

A. *Application to AS.: Priority Scenarios*

a) 1-Moving Obstacle-A_1/A_2: : scenario with one main robot and one dynamic obstacle robot: The first experiment was conducted by having one mobile obstacle. One mobile obstacle robot was moving from location 4, going to 12, then 20, turning then moving to location 21. Three simulations were conducted. In all simulations, the starting point is node 17 and the goal is node 36. The simulation results are shown in figure 1, where circles indicate source and goal, squares indicate mobile obstacle, and the arrows indicate the selected shortest path. For the simulation, non-priority and priority (algorithms A_1 and A_2) were implemented and the initial robot's direction was right. For the non-priority, the main robot changed direction at position 19 to avoid obstacle robot (figure 1). In priority algorithm A_1, the mobile obstacle robot waited 3 steps at position 12 to let main robot advance. In priority algorithm A_2, the main robot advanced normally and the other robot did not have to wait for it. Both A_1 and A_2 showed the same efficiency to find the shortest path which is 17-18-19-20- 20-28-36 and had a total cost of 12, including one turning step at nodes 20. Algorithm A_2, however, showed better efficiency for the other robots. Moreover, tables I and

II, summarize the position with respect to time for the robot and the other obstacle.

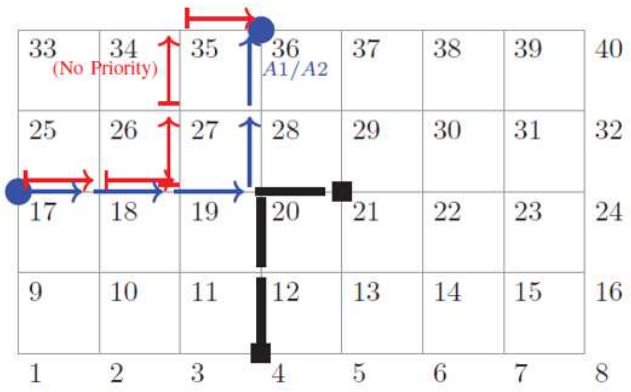

Fig. 1. Exp1-Path planning from Source to Goal

TABLE I. SCENARIO EXP1: 1 MOVING OBSTACLE-NO PRIORITY

Time	*Robot's Location*	*Robot's Direction*	*Obstacle's Location*
0	17	Right	4
1	18	Right	12
2	19	Right	20
3	19	Turn Up	21
4	27	Up	21
5	35	Up	21
6	35	Turn Right	21
7	36	Right	21

TABLE II. SCENARIO EXP1: 1 MOVING OBSTACLE-USING A_1/A_2

Time	*Robot's Location*	*Robot's Direction*	*Obstacle's Location– Using A_1*	*Obstacle's Location– Using A_2*
0	17	Right	4	4
1	18	Right	12	12
2	19	Right	12	20
3	20	Right	12	21
4	20	Turn Up	12	21
5	28	Up	20	21
6	36	Up	21	21

b) Application to AS: Priority Scenario-2- Multi Moving Obstacle-A_1/A_2: scenario with one main robot and two dynamic obstacle robots: The second experiment was conducted by having two mobile obstacles. The first mobile obstacle robot was moving from location 6, going to 5, then 4. The second mobile obstacle robot was moving from location 17, going to 9, turn to 10 (without turning delay), then 11, then 12, and then stopping at 13. Three simulations were conducted. In all simulations, the starting point is node 1 and the goal is node 36. The simulation results are shown in figures 2 and 3. For the simulation, non-priority and priority (algorithms A_1 and A_2) were implemented and the initial robot's direction was right. For the non-priority, the main robot changed direction at position 3 to avoid obstacle robot (figure 2). In priority algorithm A_1, the first mobile obstacle robot waited 3 steps at position 5 to let main robot advance. The second obstacle robot waited 2 steps at position 11 to let main robot advance. In priority algorithm A_2, similar to the non-priority algorithm, the main robot changed direction at position 3 to avoid first obstacle robot. A_1 showed the best efficiency to find the shortest path which is 1-2-3-4-4-12-20-28-36 and had a total cost of 8, including one turning step at nodes 20. Figure 3 shows the output path for both A_1 and A_2 algorithms. Moreover, tables III, IV, V, summarize the position with respect to time for the main robot and the other obstacles.

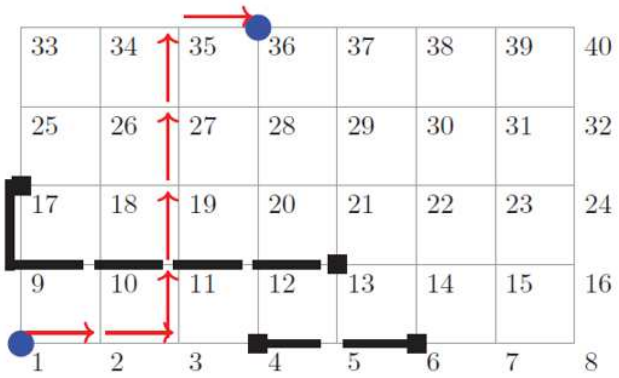

Fig. 2. Exp2-Path planning from Source to Goal - No Priority

TABLE III. EXP2: 2 MOVING OBSTACLES - NO PRIORITY

Time	*Robot's Location*	*Robot's Direction*	*1st Mobile Obstacle's Location*	*2nd Mobile Obstacle's Location*
0	1	Right	6	17
1	2	Right	5	9
2	3	Right	4	10
3	3	Turn Up	4	11
4	11	Up	4	12
5	19	Up	4	13
6	27	Up	4	13
7	35	Up	4	13
8	35	Turn Right	4	13
9	36	Right	4	13

TABLE IV. EXP2: 2 MOVING OBSTACLES-USING A_1

Time	*Robot's Location*	*Robot's Direction*	*1st Mobile Obstacle's Location*	*2nd Mobile Obstacle's Location*
0	1	Right	6	17
1	2	Right	5	9
2	3	Right	5	10
3	4	Right	5	11
4	4	Turn Up	5	11
5	12	Up	4	11
6	20	Up	4	12
7	28	Up	4	13
8	36	Up	4	13

TABLE V. EXP2: 2 MOVING OBSTACLES-USING A_2

Time	*Robot's Location*	*Robot's Direction*	*1st Mobile Obstacle's Location*	*2nd Mobile Obstacle's Location*
0	1	Right	6	17
1	2	Right	5	9
2	3	Right	4	10
3	3	Turn Up	4	11
4	11	Up	4	12
5	19	Up	4	13
6	27	Up	4	13
7	35	Up	4	13
8	35	Turn Right	4	13
9	36	Right	4	13

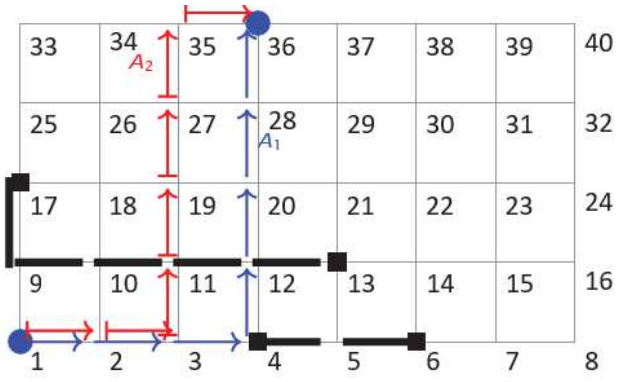

Fig. 3. Exp2-Path planning from Source to Goal-A_1/A_2

c) Application to AS: Priority Scenario-3- Multi-Moving Obstacle-A_1/A_2: The third experiment is the same as the second experiment, but with one extra step for the first mobile robot: from location 7, then 6, going to 5, then 4. Three simulations were conducted. In all simulations, the starting point is node 1 and the goal is node 36. The simulation results are shown in figure 4. For the simulation, non-priority and priority (algorithms A_1 and A_2) were implemented and the initial robot's direction was right. For the non-priority, the main robot crashed into the first obstacle robot at position 4. In priority algorithm A_1, the first obstacle robot waited 2 steps at position 5 to let main robot advance. The second obstacle robot waited 2 steps at position 11 to let main robot advance. In priority algorithm A_2, similar to algorithm A_1, the first obstacle robot waited 2 steps at position 5 to let main robot advance. However, the second obstacle did not need to wait. Both A_1 and A_2 showed the same efficiency to find the shortest path which is 1-2-3-4-4-12-20-28-36 and had a total cost of 8, including one turning step at nodes 20. Algorithm A_2, however, showed better efficiency for the second obstacle robot. Moreover, tables VI and VII, summarize the position with respect to time for the main robot and the other obstacles.

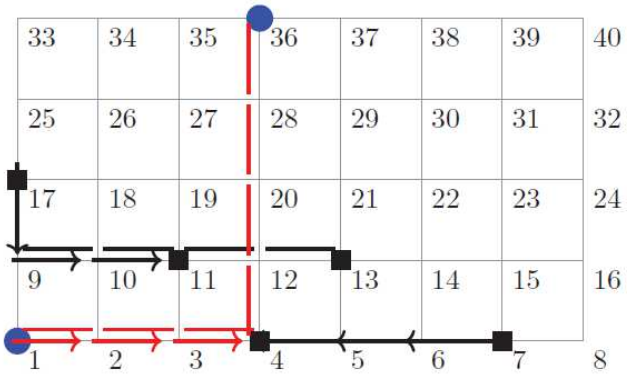

Fig. 4. Exp3-Path planning from Source to Goal Using A_1/A_2

TABLE VI. SCENARIO EXP3: 2 MOVING OBSTACLES-NO PRIORITY

Time	*Robot's Location*	*Robot's Direction*	*1st Mobile Obstacle's Location*	*2nd Mobile Obstacle's Location*
0	1	Right	7	17
1	2	Right	6	9
2	3	Right	5	10
3	4	Right	4	11

TABLE VII. SCENARIO EXP3: 2 MOVING OBSTACLES-USING A_1/A_2

Time	*Rob.'s Locat.*	*Rob.'s Direct.*	*1st Mobile Obstacle's Location Using A_1/A_2*	*2nd Mobile Obstacle's Location Using A_1*	*2nd Mobile Obstacle's Location Using A_2*
0	1	Right	7	17	17
1	2	Right	6	9	9
2	3	Right	5	10	10
3	4	Right	5	11	11
4	4	Turn Up	5	11	12
5	12	Up	4	11	13
6	20	Up	4	12	13
7	28	Up	4	13	13
8	36	Up	4	13	13

B. *Experimental Results Comparison and General Discussion*

Overall, the scenarios for multiple robots with priority were considered. Priority was statically determined in advance for a single or master robot. Conflict detection and solution was determined using one of two suggested priority algorithms: A_1 and A_2. Both A_1 and A_2 priority algorithms were tested after conducting several experiments. These results were shown above in the previous sections. The A_1 algorithm, by checking only the next node for the non-priority robots, works more towards the advantage of the priority robot, mostly ignoring the other robots. On the other hand, the A_2 algorithm checked two steps ahead for the non-priority robots, thus gave them a way to express their needed path, by allowing them to pass if the second step does not conflict with the priority robot. Usually A_1 and A_2 showed similar results, with A_2 being less constraining for other robots, and thus making them faster.

C. Complexity Analysis

The complexity analysis depends on the heuristic functions (namely H1 and H2), and on the success of the priority path selection. Overall, if the priority path is successful, O(n) is achieved where n is the number of nodes. However, if the priority path keeps failing, the penalty will be an extra O(n) loop inserted into the algorithm, making the overall complexity become $O(n^2)$. On the other hand, Dijkstra algorithm's complexity is $O(n^2)$, and A* algorithm's complexity is $O(b^d)$, where b is the branching factor and d is the depth of the shortest path.

VI. Conclusion and Future Work

In this paper, a collision free path planning algorithm to find effectively the shortest-time path for robots under a given logistics warehouse environment has been presented. Our path planning algorithm is an improvement over the original A* algorithm. In addition, we added two priority path calculation and handling methods, namely A_1 and A_2. The developed algorithm was simulated using C Sharp. In order to demonstrate and evaluate the performance of the proposed model, several experiments are carried out and a comparison between different scenarios is discussed. The simulated results show that the presented algorithm can make the system more robust and improves the performance of the navigation system and prioritize the main robot. The A_1 and A_2 algorithms performed similarly for the priority robot. However, the A_2 algorithm gave the same results while at the same time enhanced the path performance time for the other robots. In the future, we plan to implement the proposed algorithm in a real environment. Furthermore, we plan to add the return path planning, so that the robot can go back after taking/putting the load.

References

[1] Mustafa Al-Khawaldeh, Ibrahim Al-Naimi, Xi Chen, and Philip Moore. 2016. Ubiquitous robotics for knowledge-based auto-configuration sys- tem within smart home environment. In 2016 7th international confer- ence on information and communication systems (ICICS), pages 139- 144, 2016.

[2] Jie-Hua Zhou, Ji-Qiang Zhou, Yong-Sheng Zheng, and Bin Kong. 2016.Research on path planning algorithm of intelligent mowing robot usedin large airport lawn. In 2016 international conference on information system and artificial intelligence (ISAI), pages 375-379, 2016.

[3] Takayuki Kanda, Masahiro Shiomi, Zenta Miyashita, Hiroshi Ishiguro, and Norihiro Hagita. 2009. "An affective guide robot in a shopping mall". In 2009 4th ACM/IEEE international conference on human-robotinteraction (HRI), pages 173-180, 2009.

[4] Chen, Chiu-Hung, Tung-Kuan Liu, and Jyh-Horng Chou. 2014. A novelcrowding genetic algorithm and its applications to manufacturing robots.IEEE Transactions on Industrial Informatics 10 (3): 1705-1716.

[5] P. ABI-CHAR, C. Riman "A Collision-Free Path Planning Algorithm for Non-Complex ASRS Using Heuristic Functions" in Proceeding of the 44nd IEEE 2021 International Conference on Telecommunications and Signal Processing (TSP), 26-28 July, 2021, pp. 52-57, 2021, doi: 10.1109/TSP52935.2021.9522682

[6] Bellman E., "On a routing problem," Quarterly of Applied Mathematics 16, pp. 87-90, 1958.

[7] Dijkstra E. W., "A note on two problems in connection with graphs," Numerical Mathematics 1, pp. 269-271, 1959.

[8] Dreyfus S., "An appraisal of some shortest path algorithms," OperationsResearch 17, pp. 395-412, 1969.

[9] T. Zheng, Y. Xu, and D. Zheng, "AGV path planning based on improvedA-star algorithm," in Proceedign of the IEEE 3rd Adv. Inf. Manage., Communicates, Electron. Autom. Control Conf. (IMCEC), Oct., pp.1534-1538, 2019.

[10] Li, D.; Wang, L.; Geng, S.; Jiang, B. "Path Planning of AS/RS Based on Cost Matrix and Improved Greedy Algorithm". Symmetry 2021, 13, 1483. https://doi.org/10.3390/sym13081483.

[11] Taixiong Zheng, D. K. Liu, and Ping Wang; Priority based Dynamic Multiple Robot Path Planning. 2nd International Conference on Autonomous Robots and Agents. December 13-15, 2004 Palmerston North,New Zealand. Pages 373-378.

[12] Meisu Zhong et al. Priority-based speed control strategy for automated guided vehicle path planning in automated container terminals. Transactions of the Institute of Measurement and Control. 2020. Volume: 42 issue: 16, page(s): 3079-3090. https://doi.org/10.1177/0142331220940110.

[13] M. Cap, P. Novak, A. Kleiner and M. Selecky, "Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots," in IEEE Transactions on Automation Science and Engineering, vol. 12, no.3, pp. 835-849, July 2015, doi: 10.1109/TASE.2015.2445780.

[14] Dewangan, R.K.; Shukla, A.; Godfrey, W.W. A solution for priority-based multi-robot path planning problem with obstacles using ant lion optimization. Mod. Phys. Lett. B 2020, 34, 2050137.

[15] Huang, Sheng-Kai, Wang, Wen-June, Sun, Chung-Hsun. (2021). A Path Planning Strategy for Multi-Robot Moving with Path-Priority OrderBased on a Generalized Voronoi Diagram. Applied Sciences. 11. 9650. 10.3390/app11209650.

[16] Akshay Kumar Guruji, Himansh Agarwal, D.K. Parsediya, Time-efficient A* Algorithm for Robot Path Planning, Procedia Technology, Volume 23, 2016, Pages 144-149, ISSN 2212-0173, https://doi.org/10.1016/j.protcy.2016.03.010.

[17] Jean Gregoire, Silvre Bonnabel, Arnaud de la Fortelle. Priority-based coordination of robots. 2014. ffhal-00828976v3

[18] S. Zeng, Y. Zhu and C. Qi, "HTN-based multi-robot path planning," 2016 Chinese Control and Decision Conference (CCDC), 2016, pp. 4719-4723, doi: 10.1109/CCDC.2016.7531837.

[19] J. P. van den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005, pp. 430-435, doi: 10.1109/IROS.2005.1545306.