Research on Mobile Robot Path Planning Based on the Weighted A* Algorithm

Zexiu Teng *

School of Mechanical Engineering, Xi'an Jiaotong University, Xi'an, 710049, China

* Corresponding Author Email: 890001t@stu.xjtu.edu.cn

Abstract. The path planning problem is crucial in different areas like autonomous navigation and so on and has a wide range of applications. The A* algorithm is a classic way to solve the path planning problem, able to give the optimal path within a relatively short period. However, in some situations, the A* algorithm requires a long time to find the solution. In the weighted A* algorithm, a weight factor w is introduced to enlarge the role of heuristic estimation in the algorithm, allowing a faster searching process while giving back a path of acceptable length. In the experiment, the weighted A* algorithm was run multiple times under different values of w in two different maps to evaluate the influence of w on the final result and the algorithm's performance in different conditions. It was found that with a weight factor > 1, generally, the path length became longer, and computational time became quicker. The research illustrates the influence of the value of w in the weighted A* algorithm, allowing people to adjust the value of w that well meets their demands.

Keywords: Path planning; A* algorithm; Heuristic search.

1. Introduction

Nowadays, there are a lot of technologies, like navigation for cars and people, or a sweeping robot cleaning one's house, to solve an important problem, which is the path planning problem. For example, autonomous navigation is vital in mobile robots, helping them be more independent from people's intervention. However, many tasks and problems are required to solve, for example, path planning [1]. Apart from that, it is also crucial for unmanned aerial vehicles to find the optimal path between the starting points to the target [2]. In most cases, path planning can be categorized into global path planning and local path planning, as in global path planning the robot knows the environment in advance and finds a predetermined optimal path, while in local path planning the robot doesn't know the whole environment and have to make real-time decisions based on feedback [3].

In global path planning problems, the Dijkstra algorithm, proposed by E.W. Dijkstra, was one well-known and applicated algorithm used to solve the path planning problem [4]. However, it relies on a greedy strategy, which means it has to expand more nodes and save all potential results, raising a high demand on memory and consuming a longer time to find the optimal path [5].

A* algorithm was developed from the Dijkstra algorithm [4]. It features a heuristic function in calculating the values of each node, thus finding the optimal path [6]. Thanks to the heuristic estimation in the cost function, the A* algorithm can find the shortest path just like the Dijkstra algorithm with fewer nodes expanded, which means a shorter computational time and higher time efficiency. On the other hand, despite the A* algorithm's superior performance compared to the Dijkstra algorithm in memory demand and time efficiency, with the environment getting bigger and more complex, the computational time and memory necessary can still grow beyond acceptance.

The weighted A* algorithm is a variant of the A* algorithm. In the weighted A* algorithm, a weight factor w, generally > 1, is introduced to adjust the role of heuristic estimation in the cost function. With the weight factor *w* in the cost function, the algorithm gives a quicker search process with fewer nodes expanded, saving time and memory, at the cost of sacrificing a bit of optimality, resulting in a slightly longer path length than the Dijkstra algorithm and A* algorithm. Changing the value of weight factor *w* can drastically influence the performance of the algorithm.

In this research, the performance of the weighted A* algorithm in different environments was evaluated under different values of weight factor w. It was found that normally with the inflammation of the weight factor w, the search process accelerates and the path length gets longer, which can be viewed as a trade-off between the optimality of the result and the time-efficiency of the search process because a bigger value of w means less nodes are expanded and the algorithm is more likely to neglect shorter yet less costly paths.

It was also found that in some situations the weighted A* algorithm would not follow the rules mentioned above, which can be a direction for further research.

2. Methodology

2.1. Basic A* Algorithm

First presented by E.W. Dijkstra, the Dijkstra algorithm is a classic algorithm for finding the shortest path, featuring searching from the starting point, and then expanding to the target [4]. By two vertices, each edge from the map is shaped into an ordered element pair [4]. The weight function is used to represent the value of the edges [4]. There are two vertex sets in the algorithm, which can be called set A and set B. Set A, at the start, is empty. Each time only one vertex from B can make sure the overall sum of the edge weight from the start to the current node is chosen and transported into a [4]. The traditional Dijkstra algorithm is heavily dependent on a greedy strategy in solving path-planning problems, and to find the shortest path, the Dijkstra algorithm needs to calculate all potential results, making it reliable yet memory-heavy [5]. Having to go through more nodes, the efficiency of the Dijkstra algorithm is lower compared to other algorithms [4].

Therefore, based on the Dijkstra algorithm, the A* algorithm was put forward [4]. In the A* algorithm, an evaluation function (1) shown below plays an important role in selecting the next node.

$$f(n) = g(n) + h(n) \tag{1}$$

Where f (n) is the cost of the path from the starting node, going through the current node, to the target node, which consists of two parts, g (n) that refers to the actual cost from the starting node to the current node, and h (n) that evaluates the cost from the current node to the target node through an optimal path [7]. Fig. 1 is a flowchart of how the A* algorithm works [8].

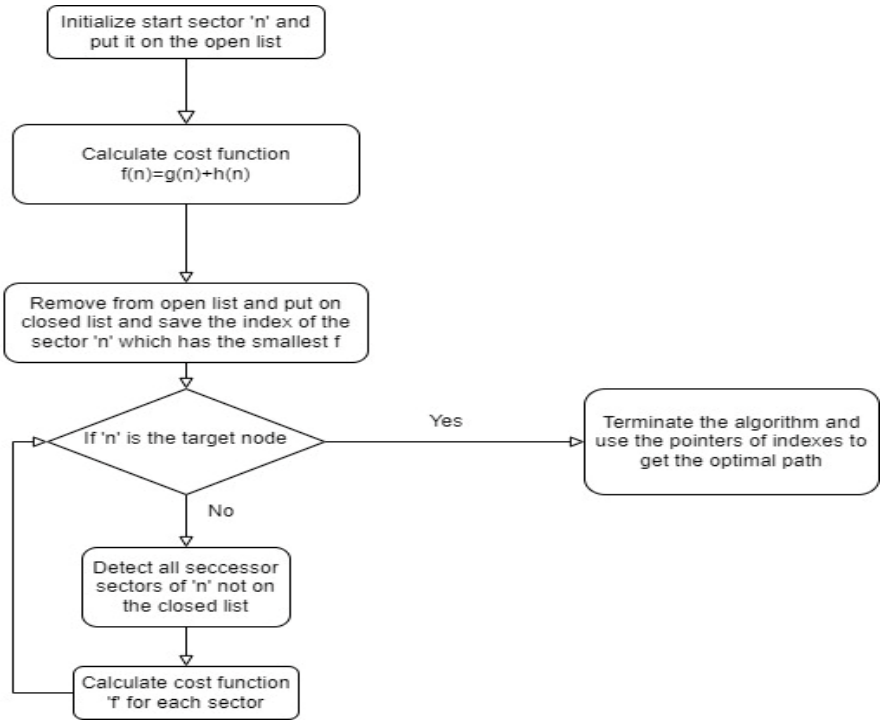


Fig 1. A flowchart on how the A* algorithm works [8].

In the evaluation function f (n), the evaluation cost h (n) is crucial given the fact that the actual cost g (n) is a constant [7]. Often the Euclidean distance between the current node and the target node is applied as the value of the evaluation cost h (n) [4].

Despite that the A* algorithm has an excellent performance in finding the shortest path, a long computational time is necessary to achieve the shortest distance [9].

2.2. Weighted A* Algorithm

One of the A* algorithm's big disadvantages are that in some extreme situations, the computational time and the memory it requires are huge, resulting in different kinds of variants of the A* algorithm [10]. It was first suggested by Pohl to multiply a factor by the heuristic function, which is the idea of the first weighted A* algorithm, also called by many researchers the WA* algorithm.

The function (2) below is the evaluation function of the weighted A* algorithm.

$$f(n) = g(n) + w * h(n) \quad (2)$$

Similar to the standard A* algorithm, g(n) here stands for the actual cost between the starting node to the current node, and h(n) here is a heuristic estimation of the cost from the current node to the target node, often the Euclidean distance. Compared to the standard A* algorithm, the weighted A* algorithm introduces a weight factor w to adjust, in most cases inflate the influence of the heuristic function, resulting in the heuristic estimation playing a bigger role in the path planning process. Generally, the value of the weight factor w here is bigger than 1. Adjusting the weight factor w influences the performance of the algorithm, for example, the computational time and the path length.

2.3. Experimental Condition and Research Tools

The experiment was performed virtually using Python. In the experiment, the standard A* algorithm and the weighted A* algorithm was executed in two different maps. Map1 is a standard maze automatically generated by a computer. Map1 is also a simply connected maze that can be solved using the Hand on Wall Rule. Map2 features point-shaped obstacles that are irregularly distributed in the whole map. The weighted A* algorithm was run multiple times under different values of weight factor w to explore the effect of changing the parameter w's value on the algorithm. To evaluate the performance of the algorithm, three indicators, respectively path length, average computational time, and the number of nodes expanded were recorded and compared under different values of w in both maps.

3. Results and Discussions

3.1. Weighted A* Path Planning Results

3.1.1. Path planning results in map 1

Fig. 2 shows the result of the standard A* algorithm path planning with its performance matrix in map 1.

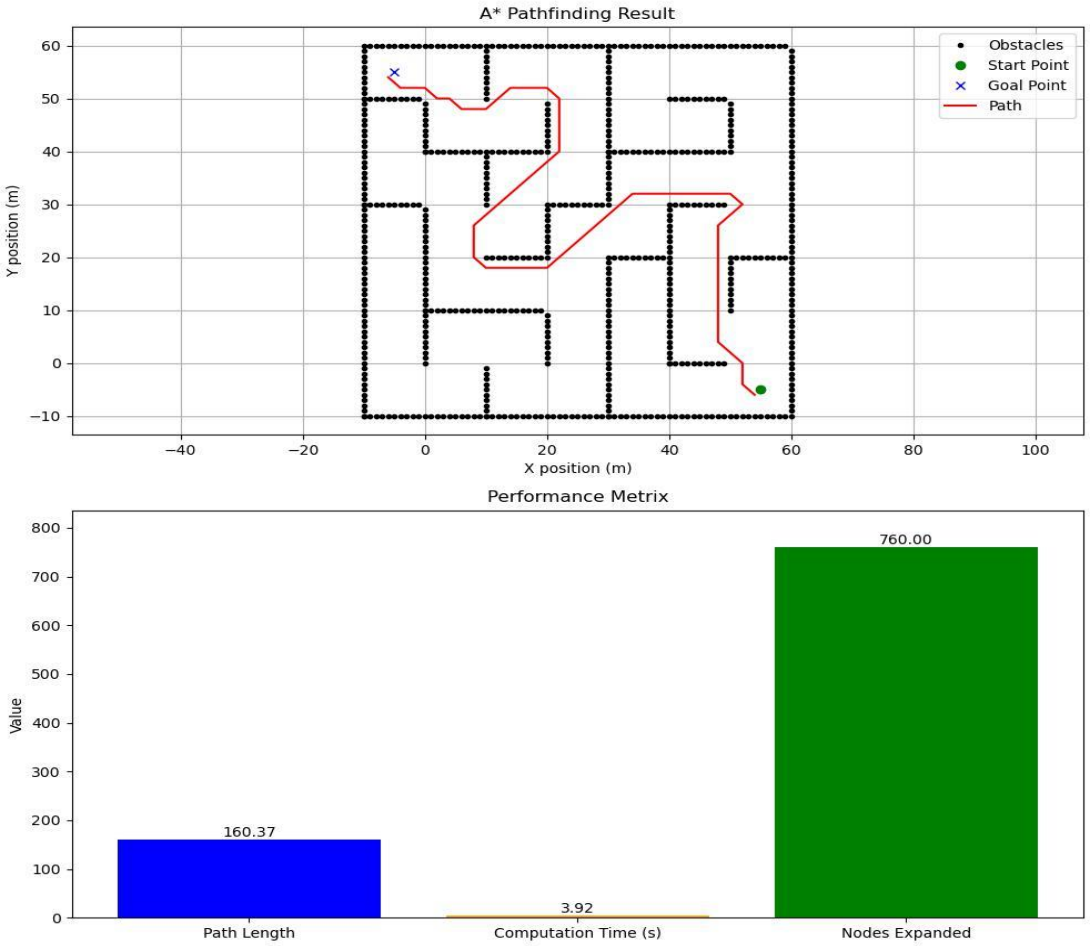


Fig 2. Path planning result and performance matrix of standard A* in map 1. (Picture credit: Original).

Table 1 illustrates the performance of the weighted A* algorithm in map1 under different values of weight factor w.

Table 1. Performance of weighted A* algorithm in map 1

Value of w	Path length	Average computational time(s)	Nodes expanded
0	160.37	5.678	969
1	160.37	4.014	760
1.5	162.02	3.291	662
2	165.34	3.076	621
3	171.97	1.404	349
5	175.97	1.201	303
10	186.79	1.074	275
100	187.97	0.986	258
10000	187.97	0.996	257

Each time the weighted A* algorithm was executed, the computational time differed. Therefore, under each value of w, the experiment was repeated several times, and the average computational time was introduced to evaluate the performance of the algorithm.

When the value of the weight factor w = 0, the weighted A* algorithm turns into Dijkstra's algorithm. Similarly, when the weight factor w = 1, the weighted A* algorithm becomes the standard A* algorithm.

From Table 1 above it can be concluded that with the weight factor w increasing, the average computational time was significantly quicker and the number of nodes expanded became smaller. On the other hand, the path length got longer with the inflation of the value of w.

Apart from that it is also found that when the value of w got big enough, the path length topped out at a certain value, as the number of nodes expanded and the average computational time stopped shrinking.

3.1.2. Path planning results in map 2

Fig. 3 shows the result of the standard A* algorithm path planning with its performance matrix in map 2.

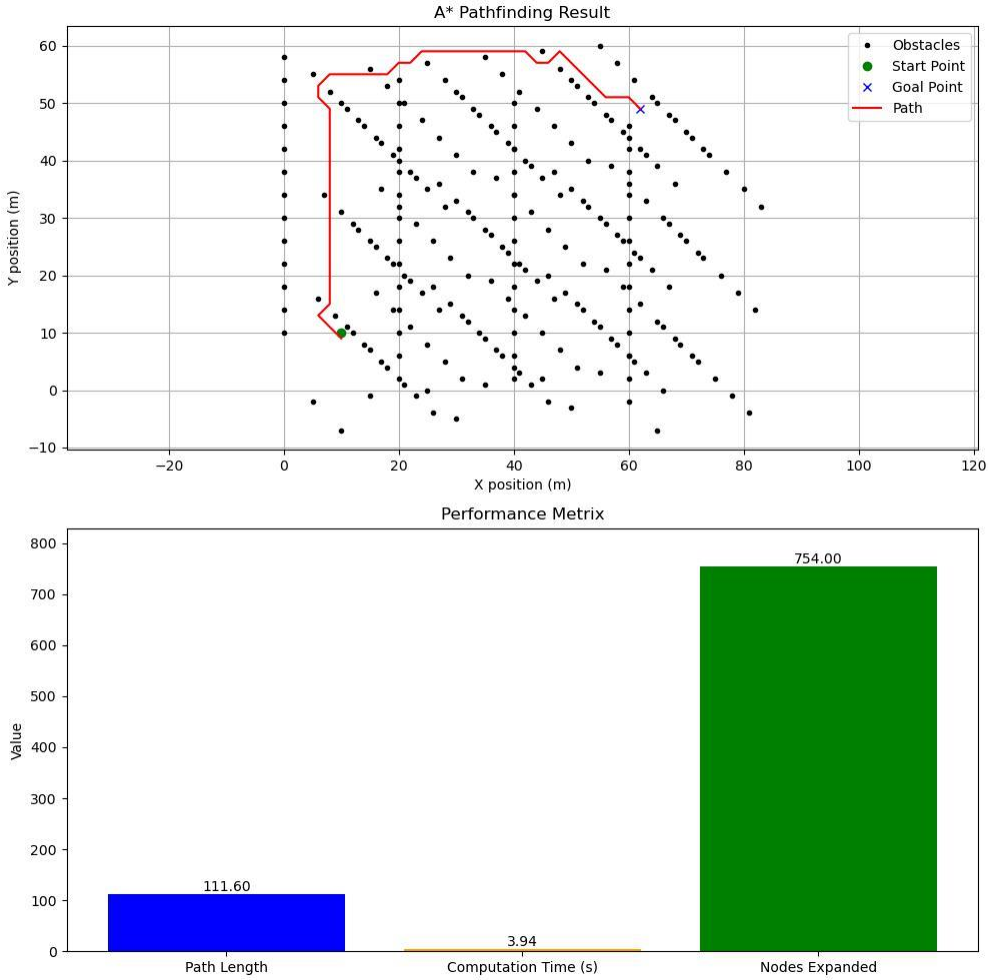


Fig 3. Path planning result and performance matrix of standard A* in map 2. (Picture credit: Original).

Table 2 shows how the weighted A* algorithm performed in path planning in map2 with the value of weight factor w constantly changing. The experiment is also repeated multiple times trying to find representative values of the average computational time.

Table 2. Performance of weighted A* algorithm in map 2.

Value of w	Path length	Average computational time(s)	Nodes expanded
0	111.60	6.276	1024
1	111.60	3.909	754
1.5	113.25	3.329	661
2	131.20	2.249	493
3	136.17	1.470	363
5	128.85	0.825	228
10	122.82	0.813	229
20	147.97	0.556	156
100	149.62	0.541	148
10000	149.62	0.529	146

Similar to the result in map 1, with the value of weight factor w inflating, the average computational time became quicker, and the number of nodes expanded became smaller. Also, with a weight factor w whose value was bigger than 1, the path length got longer along with the increase in the value of the weight factor w.

The path length also reached a maximum value with a value of weight factor w big enough, while the number of nodes expanded and the average computational time was approaching their minimum value.

The results in map 2 were not completely in line with that in map 1 though. From Table 2 above it can be learned that when the value of the weight factor w was 2 and 3, there was a sudden jump in the path length from just over 110 to over 135, then dropping back to over 120 when the value of the weight factor w was 10.

When the weight factor w here equals 0, the weighted A* algorithm becomes the same as Dijkstra's algorithm. Similarly, when the value of the weight factor w is 1, the weighted A* algorithm turns into the standard A* algorithm. Both the Dijkstra's algorithm and the standard A* algorithm are able to find the shortest path in the path planning problem, despite different efficiency. However, the Dijkstra's algorithm and the standard A* algorithm here gave back two slightly different paths of the same length from the starting point to the target. The result of the standard A* algorithm in map 2 is shown in Fig. 3, and Fig. 4 is the result and performance matrix of the Dijkstra's algorithm in map 2.

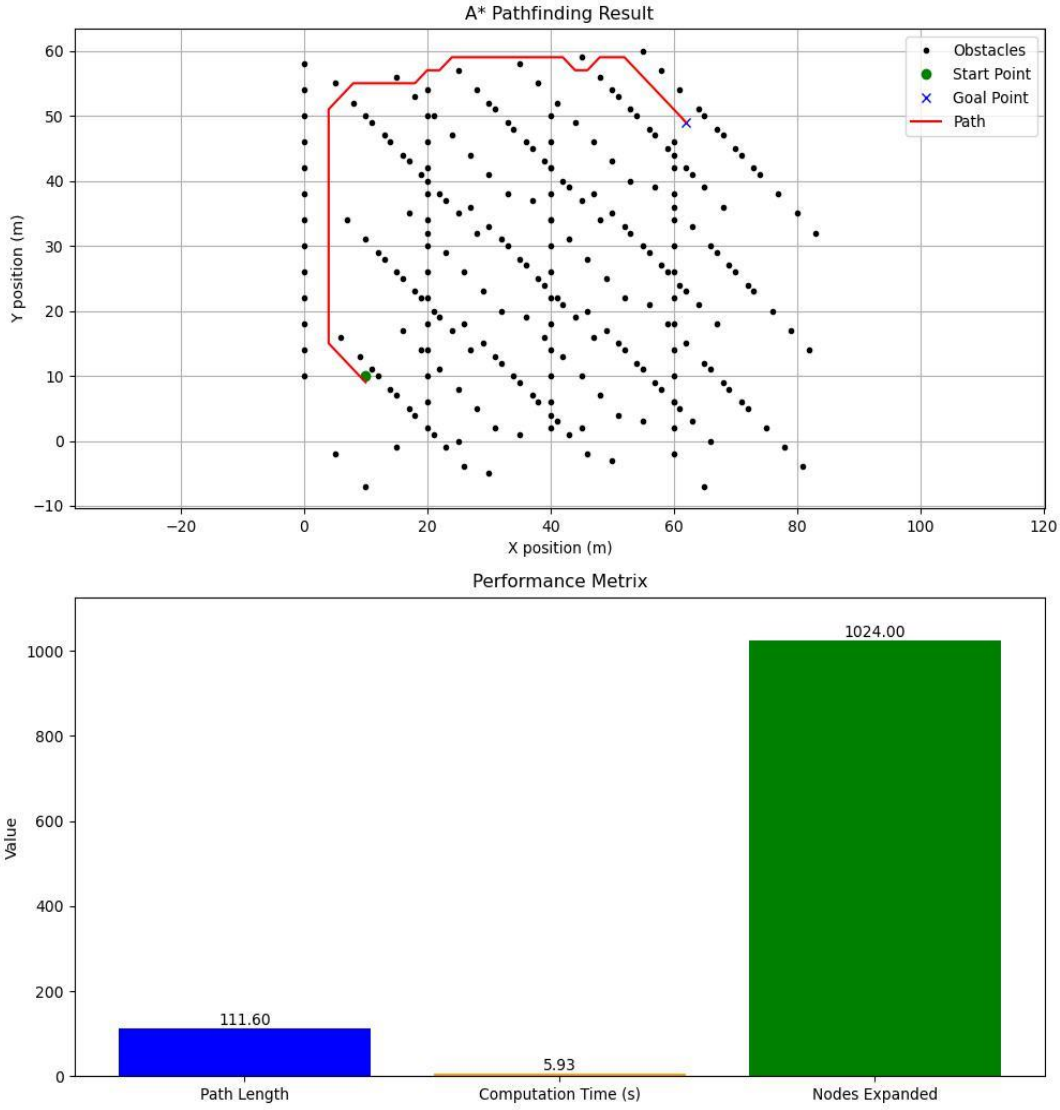


Fig 4. Path planning result and performance matrix of Dijkstra's algorithm in map 2. (Photo/Picture credit: Original).

3.2. Discussion and Evaluation of Results

With the results in map 1 and map 2 above it is clear that the weighted A* algorithm has some certain properties.

When the weight factor w = 0, the weighted A* algorithm degenerates to Dijkstra's algorithm. When the weight factor w = 1, the weight A* algorithm degenerates to the standard A* algorithm. Therefore, in most cases, the value of w is bigger than 1.

Adjusting the weight factor w changes the role of heuristic estimation in the path planning process. A bigger value of w means more influence on the heuristic estimation in the cost function, hence accelerating the path planning process, yet reducing the optimality of the path planning results. Actually, when w is bigger than 1, the adjustment of the value of the weight factor w here is a trade-off between the optimality of the result and the efficiency of the solution.

Generally increasing the value of the weight factor w leads to a more efficient yet less optimal path planning process and result, reducing the computational time while lengthening the path result. Still, as the weight factor w keeps inflating, it is highly likely that the path length would encounter a maximum value, while the computation time also stops decreasing. That is possibly because when the weight factor is big enough, the actual cost g(x) plays a very small role in the cost function, therefore increasing the value of w can barely increase the influence of the heuristic estimation in the path planning process.

There are, however, some exceptions to the patterns put forward above. In some situations, there would be a sudden surge in the path length which does not come in line with the value of the weight factor w. In such conditions, the algorithm loses both optimality and efficiency. This is likely because under certain conditions, the heuristic estimation and with specific values of weight factor w, the cost does not always come alone with the actual path length. The increase of weight factor w means the algorithm may overlook some shorter yet more costly paths, and under some specific conditions with certain values of w, the algorithm may thus find a longer path in a certain phase resulting in a surge in the overall path length.

4. Conclusion

In this research, the weighted A* algorithm was used to solve the path planning problem in two different maps. The same process was repeated many times to make sure the result was accurate and representative and to visualize how changing the value of the weight factor w influenced the path length of the path planning result, which stood for the optimality of the solution, and the computational time in the path planning process, which indicated the time efficiency of the algorithm.

Through the result, it was found that generally, when w > 1, if the value of the weight factor w was increased, the path length would inflate, and the computational time would be accelerated. This means that the inflammation of weight factor w is a trade-off between the optimality of the result and the efficiency of the solution. Increasing the weight factor w would cause the algorithm to overlook shorter yet more costly paths, resulting in a longer path length. The computational time, however, shortens because fewer nodes are expanded.

Such a rule does not apply to all situations. In some conditions, there would be a sudden surge in the path length with the value of the weight factor w in a certain range. This may be due to the distance from the start node to the target node through the selected node does not always come in line with the cost function in some specific environment.

This research shows how different values of weight factor w influence the performance of the weighted A* algorithm, which enables researchers and developers to change the value of w as they want so that it satisfies their requirements in both optimality and efficiency.

For further research, two directions can be looked into deeper. One is if there is a way to find a specific value of w that can perfectly balance between optimality and efficiency given different conditions and requirements, and if there is, how to achieve that. Another one is in what situation and

under what condition the result of the weighted A* algorithm would go against the rules found above, and the reason why the algorithm behaves abnormally.

References

[1] Sanchez-Ibanez J R, Pérez-del-Pulgar C J, García-Cerezo A. Path planning for autonomous mobile robots: A review. Sensors, 2021, 21(23): 7898.

[2] Aggarwal S, Kumar N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. Computer communications, 2020, 149: 270-299.

[3] Liu L, Wang X, Yang X, et al. Path planning techniques for mobile robots: Review and prospect. Expert Systems with Applications, 2023, 227: 120254.

[4] Zhang H, Lin W, Chen A. Path planning for the mobile robot: A review. Symmetry, 2018, 10(10): 450.

[5] Karur K, Sharma N, Dharmatti C, et al. A survey of path planning algorithms for mobile robots. Vehicles, 2021, 3(3): 448-468.

[6] Guruji A K, Agarwal H, Parsediya D K. Time-efficient A* algorithm for robot path planning. Procedia Technology, 2016, 23: 144-149.

[7] Fu B, Chen L, Zhou Y, et al. An improved A* algorithm for industrial robot path planning with a high success rate and short length. Robotics and Autonomous Systems, 2018, 106: 26-37.

[8] Zidane I M, Ibrahim K. Wavefront and a-star algorithms for mobile robot path planning. In Proceedings Of The International Conference On Advanced Intelligent Systems And Informatics 2017. Springer International Publishing, 2018, pp. 69-80.

[9] Korkmaz M, Durdu A. Comparison of optimal path planning algorithms. In 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET). IEEE, 2018, pp. 255-258.

[10] Ebendt R, Drechsler R. Weighted * A search–unifying view and application. Artificial Intelligence, 2009, 173(14): 1310-1342.