electronics

MDPI

Article

Application of A* Algorithm Based on Extended Neighborhood Priority Search in Multi-Scenario Maps

Zhiyu You *,†, Keyu Shen †, Tao Huang, Yongxin Liu and Xiaofeng Zhang

Key Laboratory of Electronic Information of State Ethnic Affairs Commission, College of Electrical Engineering, Southwest Minzu University, Chengdu 610041, China

* Correspondence: youzhiyu@swun.edu.cn

† These authors contributed equally to this work.

Abstract: The robustness of the traditional A* algorithm of path planning is poor due to its excessive number of traversal nodes, slow search speed, and large turning angle. Aiming to solve the above problems, a multi-scenario adaptive A* algorithm based on extended neighborhood priority search is proposed. Firstly, this algorithm designs the heuristic function that can adapt to various scene changes by quantifying the scene map information, and the search weight is adjusted adaptively to enhance the robustness and adaptability of the algorithm. Secondly, the search strategy based on extended neighborhood priority is adopted to improve the orientation of the algorithm, and the redundant node removal strategy is used to smooth the path to reduce the number of traversed nodes and the turning angle. Finally, simulation tests are conducted in several representative map environments. The test results show that the proposed algorithm is superior to the traditional A* algorithm due to its stronger robustness and significantly improved performance metrics, with an 84.95% reduction in the number of traversal nodes, an 83.84% reduction in the number of path nodes, a 62.28% reduction in turning points on the path, a 77.38% reduction in the total turning angle, and a 58.47% reduction in the search time.

Keywords: A* algorithm; adaptive weights; heuristic function; path planning

Citation: You, Z.; Shen, K.; Huang, T.; Liu, Y.; Zhang, X. Application of A* Algorithm Based on Extended Neighborhood Priority Search in Multi-Scenario Maps. *Electronics* **2023**, *12*, 1004. https://doi.org/10.3390/electronics12041004

Academic Editors: Ionica Oncioiu, Stelian Brad and Fuji Ren

Received: 31 January 2023
Revised: 14 February 2023
Accepted: 15 February 2023
Published: 17 February 2023

1. Introduction

With the rapid development of artificial intelligence and robotics, driverless robots, a hotspot in the academic field around the world, are widely used to replace manual operations in a wide range of areas, such as the service industry, urban safety, space exploration, etc. In unmanned robot technology, path planning is one of the key technologies to achieve driverless robots, which plans a collision-free path from the starting node to the target node for unmanned robots in the working environment with obstacles and ensures the planning path meets certain optimization principles (such as faster search speed, shorter path distance, etc.) as the optimal path [1]. Therefore, quick and efficient planning of a relatively optimal path that is safe and collision-free is of great significance for artificial intelligence and robotics, as it can provide robots with intelligence and movement accuracy. According to the degree of mastery of map scene information, driverless robots' path planning can be divided into local path planning and global path planning [2,3].

Local path planning requires better hardware devices with high computing and information processing capabilities because of its real-time collection of surrounding map scene information to correct the search orientation [4]. Therefore, the robot has good obstacle avoidance ability, and at the same time, has high robustness to environmental errors and noise. Due to the lack of global path information, local path planning cannot guarantee that the planning path will be globally optimal, and it can even finally fail [5]. The accuracy of the planned path given by the global path planning, which is based on prior global map information, depends on the accuracy of obtaining map scene information [6].

If there is an error in the map environment information and too much noise, it will make it less robust [7,8], and the globally optimal path cannot be obtained.

In global path planning algorithms, the A* algorithm [9,10] using the heuristic search method is one of the most commonly used path planning algorithms, with its excellent pathfinding completeness and path optimization. It is widely used for path planning in the field of unmanned driving, for example, urban logistics [11–16], environmental perception [17–20], underwater navigation [21–23], and robotic arm design [23]. Therefore, the traditional A* algorithm has always been the focus of scholars' research. In the literature [24], a new distance calculation method was designed to find the shortest path, but the path-finding time increased. Previous research [25] set the safe distance to make the path smoother and extended the search neighborhood to make the planned path distance shorter, but the search was less efficient. In other research [26], the path-finding time and the number of redundant nodes were reduced by changing the calculation method and function weight, but the weight ratio needed to be redistributed according to the specific map, which was less robust. The Chebyshev distance has been used as a heuristic function weight in the literature [27], and the introduction of parent nodes strengthens its influence, which greatly improves the search speed, but it easily falls into local optimization and the global optimization of the path cannot be guaranteed. For mobile robots with special work needs, for example, in reference [28], the energy loss factor was introduced to ensure that the total energy consumption of the final robot is the lowest, but the algorithm has a large amount of calculation and is not superior in path length. Previous authors [29–32] introduced dynamic windows into traditional A* algorithms to enable better obstacle avoidance, but the search speed was slower and required high device performance.

As can be seen from the above literature, the improved methods of the A* algorithm primarily include the optimization of the distance calculation function, the expansion of the neighborhood, path smoothing, and combination with other intelligent algorithms. However, the performance robustness of these improved algorithms is weak and they cannot guarantee good applications in various scene maps. Furthermore, some of the improved algorithms have high requirements for the hardware and software performance of mobile robots, as shown in Table 1. The improved A* algorithm in the existing literature is optimized at the expense of other performance factors, so it will be superior to the traditional A* algorithm in some performances, but it is not guaranteed to be applicable to any map.

Table 1. Performance comparison of several improved A* algorithms in the literature.

Algorithm	Robustness		Real-Time Adaptability	Search Efficiency	Path Optimization	Path Smoothness
	Stability	Performance Robustness				
Improved A* [24]	moderate	weak	weak	weak	strong	moderate
Improved A* [25]	strong	weak	moderate	weak	strong	strong
Improved A* [26]	strong	weak	weak	moderate	weak	moderate
Improved A* [27]	weak	weak	weak	weak	weak	weak
Improved A* [28]	moderate	moderate	moderate	weak	weak	weak
Improved A* [29–32]	moderate	moderate	strong	weak	moderate	strong

Based on the above literature and aimed at improving the current weaknesses of the traditional A* algorithm, such as poor robustness, too many traversal nodes, and the large path turning angle, a multi-scenario adaptive A* algorithm based on extended neighborhood priority search (ENMSA-A* algorithm) is proposed in this paper. The ENMSA-A* algorithm realizes search weight adaptive adjustment of the heuristic function via the adaptive control strategy and introduces an extended neighborhood priority search and redundant node deletion strategy to guide and smooth the path planning, which causes the ENMSA-A* algorithm to traverse fewer nodes and a smoother path in path planning and has stronger robustness and adaptability to multi-map scenes.

The main contributions and innovations of this paper are summarized as follows:

(1) We designed an ENMSA-A* algorithm that adaptively adjusts the search weight of the heuristic function based on the obstacle distribution of the scene map, which adopts the fitting priority search strategy and can update the movement direction in real time according to the search location, which is suitable for a variety of scene maps, and the comprehensive performance is better than the traditional A* algorithm.
(2) We determined the distribution of obstacles in the scene map by segmentation and then fusion and designed a more robust heuristic function on this basis.
(3) According to the relative position relationship between vectors, we designed a new dynamic programming matrix and assigned priority in the extended 16 neighborhoods, so as to enhance the guidance and search speed of the ENMSA-A* algorithm.
(4) By judging the distance and connectivity between each four turning points, a better route can be selected to obtain a relatively smooth path, which makes the ENMSA-A* algorithm more suitable for the daily work of mobile robots.

The rest of this paper is organized as follows: Section 1 introduces the current research status and related works. Section 2 describes the principle of the traditional A* algorithms. Section 3 describes the specific process of the ENMSA-A* algorithm and designs the algorithm performance robustness evaluation index. Section 4 analyzes the simulation results of various algorithms under various scenario maps. Section 5 focuses on the conclusions of this work.

2. Traditional A* Algorithm

The A* algorithm [7] was proposed by Peter Hart, Nils Nilsson, and Bertram Raphael, and its core idea is to find the minimum distance of the movement path from the starting position to the target position, which draws on the Dijkstra algorithm and the Best-First Search algorithm, and its total movement path distance estimation function $f(n)$ is:

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

where n represents the current position during path planning, $g(n)$ represents the actual distance of movement from the start position to the current location, $h(n)$ represents the estimated distance of movement from the current location to the target location (also known as the heuristic function), and $f(n)$ represents the total movement path estimate from the starting position to the target location. The relationship between the current node n and $g(n)$ in the path planning process is shown in Figure 1.

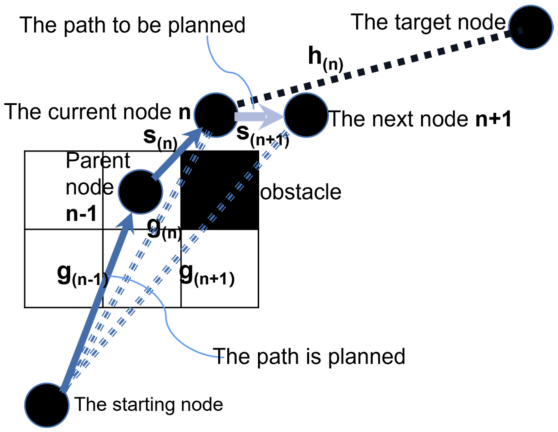


Figure 1. Diagram of the relationship between nodes.

In Figure 1, $S(n)$ represents the step-by-step distance from the parent node (n − 1) to the current node n, $S(n + 1)$ represents the step-by-step distance from the current node n to the next node (n + 1), and $g(n)$ is the actual distance traveled from the starting node to the (n − 1) node, the value of which is the sum of $g(n - 1)$ and $S(n)$. Thus, the actual

distance $g(n)$ can be expressed as the sum of the planned path distances for each segment, from which it can be deduced that the expression of the path planning total movement path estimation function $f(n)$ is:

$$f(n) = \sum_{i=1}^{n} S(i) + h(n) \tag{2}$$

According to the pathfinding principle of the A* algorithm and the expression of $f(n)$, it can be known that the A* algorithm needs to constantly judge and compare the surrounding neighboring nodes in the process of planning the path. When a large number of nodes need to be traversed, the corresponding node movement cost needs to be calculated, so the open list and the closed list need to be established to store the relevant information of each node. Note that it is necessary to establish the open list labeled as OpenList to store the nodes that need to be visited and start up the closed list labeled as ClosedList to store the nodes that have already been visited.

The A* algorithm path planning diagram based on the Manhattan distance is shown in Figure 2, in which the lower-left corner of each square is the true moving cost generation value $g(n)$ from the starting node to the current node, the lower right corner represents the estimated mobile generation value $h(n)$ from the current node to the target node, and the upper left corner represents the overall cost estimate $f(n)$ from the starting node through the current node to the destination node. The green and red grids are the starting and target nodes, and the gray grids are the obstacles. According to the definition of OpenList and ClosedList mentioned above, and in connection with the basic principles of the A* algorithm, it can be seen that the light blue grids are the nodes recorded in ClosedList, and light green grids are the nodes recorded in OpenList but rejected for inclusion by ClosedList.

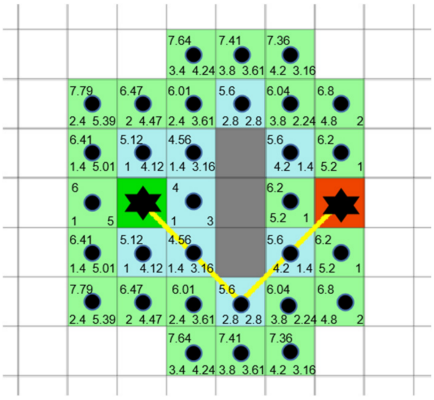


Figure 2. Schematic diagram of A* algorithm path planning based on Euclidean Distance.

First, the A* algorithm path planning needs to iterate all the surrounding neighboring nodes and calculate the corresponding movement distance value, then record the node with the smallest $f(n)$ into the ClosedList, continue to traverse and access other nodes in the OpenList, continuously update the judgment of the ClosedList until the target location is found, and then reverse the ClosedList output as the final path.

3. ENMSA-A* Algorithm

The traditional A* algorithm has poor robustness, and path planning will continue to search back and forth; furthermore, too many unnecessary nodes were traversed. Therefore, this results in a slow search speed and smooth path planning. To solve the above problems of the traditional A* algorithm, this paper optimizes the A* algorithm from three aspects: Adaptability, guiding direction, and path smoothing. It then forms an ENMSA-A* algorithm with fewer nodes, a smaller turning angle, a smoother path, and stronger robustness in the path planning process.

3.1. Adaptive Weight Strategy

In order to obtain the optimal path, the value of $f(n)$ needs to be calculated in real-time to obtain the next planning path. If $h(n)$ is much less than $g(n)$, then $f(n)$ can be approximately equal to $g(n)$; at this time, the A* algorithm is approximated to the Dijkstra algorithm, the traversal nodes will increase, and the search efficiency will be greatly reduced; if $h(n)$ is much larger than $g(n)$, then the A* algorithm gradually evolves into the Best-First-Search algorithm, which will speed up the path planning speed, but it is easy to obtain a locally optimal solution. Therefore, choosing the appropriate heuristic function $h(n)$ will affect the performance of the A* algorithm.

At present, there are three common forms of heuristic function $h(n)$: Euclidean Distance, Manhattan Distance, and Diagonal Distance. We use the starting point coordinate (s_1, s_2) and the end coordinate (g_1, g_2), and the three distances calculation formulas in the cartesian coordinate system of the map are shown in Equation (3), where $Diagonal = \min(|s_1 - g_1| + |s_2 - g_2|)$, $Straight = |s_1 - g_1| + |s_2 - g_2|$.

$$\begin{cases} Euclidean\ Dis\tan ce = \sqrt{(g_1 - s_1)^2 + (g_2 - s_2)^2} \\ Manhat\tan Dis\tan ce = |g_1 - s_1| + |g_2 - s_2| \\ Diagonal \quad Dis\tan ce = 1.4 \times Diagonal + (Straight - 2 \times Diagonal) \end{cases} \tag{3}$$

Among the three distance calculation formulas, the Euclidean distance has the highest calculation accuracy, so this paper chooses to use the Euclidean distance as the heuristic function $h(n)$. On this basis, an adaptive weight factor is added to improve its adaptability in multi-scene maps. Since the weighted heuristic function $h'(n)$ will affect the performance of the A* algorithm, it will even directly lead to an algorithmic imbalance and tilt towards other algorithms. Therefore, in order to solve the problem of an algorithm imbalance and realize the adaptability of $h(n)$ to multi-map scenes, this paper enhances the real-time and robustness of $h'(n)$ by introducing the adaptive weight function $W(Obs\_P)$ of the obstacle distribution rate Obs_P. As the search progresses, the distribution of obstacles in the map scene changes, and $h'(n)$ can adapt to the Obs_P change and adjust its search step size to avoid an algorithm imbalance.

$$h'(n) = W(Obs\_P) \times h(n) \times K \tag{4}$$

where K is the ratio of the actual map to the simulated raster map. The obstacle distribution rate Obs_P will affect the heuristic function $h'(n)$, and the results of the path planning will also change. Therefore, it is necessary to combine the obstacle distribution information in the map and the changing map area to be searched to select the appropriate obstacle distribution rate Obs_P.

This paper will first divide and then fuse the concept, dividing the map into three parts: The Global map, the Real-time map, and the Rear map, as shown in Figure 3. In this figure, the blue square is the current node, the solid yellow circle is the starting node, the red five-pointed star is the target node, the black square is the obstacle, and the black arrow indicates the search direction. The blue area enclosed by the starting node and the target node is the Global map, the red area around the current node during the search process is the Real-time map, and the yellow area surrounded by the current node and the target node is the Rear map. Among them, both the Real-time map and the Rear map will change with the search node. Then, the distribution of the three map obstacles is comprehensively analyzed to obtain the final Obs_P, so as to ensure the real-time adaptability of $W(Obs\_P)$.

Note (s_1, s_2), (g_1, g_2), and (n_1, n_2) are the coordinates of the starting node, the target node, and the current node, respectively. The number of obstacles in the Global map is recorded as x_1 and the obstacle distribution rate is recorded as p_1, the number of obstacles in the Real-time map is recorded as x_2 and the obstacle distribution rate is recorded as p_2, and the number of obstacles in the Rear map is recorded as x_3 and the obstacle distribution rate is recorded as p_3. The formula for calculating the distribution rate of obstacles in

the three maps is shown in Equation (5), where a, b, c, and d are constants, indicating the current node's distance to the surrounding area.

$$\begin{cases} p_1 = x_1/(|s_1 - g_1| \times |s_2 - g_2| - 2) \\ p_2 = x_2/(|(n_1 - a) - (n_1 + b)| \times |(n_2 - c) - (n_2 + d)|) \\ p_3 = x_3/(|(n_1 - g_1)| \times |(n_2 - g_2)| - 2) \end{cases} \quad (5)$$

According to the distribution information of the three obstacles in Figure 3, the *Obs_P* is obtained by Equation (6), where A, B, and C are constant coefficients and $A + B + C = 3$. The proportional relationship of the three constant coefficients indicates the extent to which the distribution of obstacles in the three maps affects pathfinding.

$$Obs\_P = (A \times p_1 + B \times p_2 + C \times p_3)/3 \quad (6)$$

In order to improve the robustness of the A* algorithm, the influence of Real-time map coefficient B and Rear map coefficient C should be appropriately enhanced to determine the final obstacle distribution rate *Obs_P*.

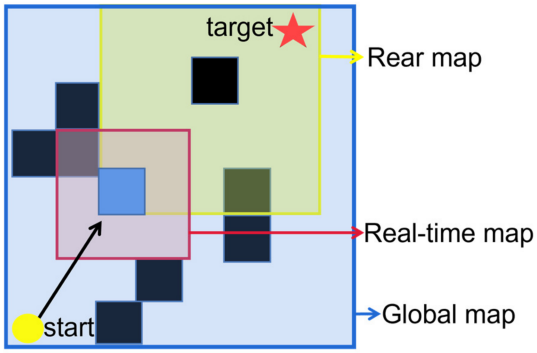


Figure 3. How the map is selected during pathfinding.

From Equation (6), it can be seen that the definition field of *Obs_P* is [1]. When *Obs_P* is large, there will be more obstacles on the map and the scene environment will be more complex. At this time, the weight of $h(n)$ needs to be reduced, so that the A* algorithm is close to the Dijkstra algorithm to ensure that the planned path is relatively better. Similarly, if *Obs_P* is small, there will be few obstacles on the map, the scene environment will be simpler, and the weight of $h(n)$ can be appropriately increased so that the algorithm can approach the Best-First-Search algorithm to improve the search speed.

Through analysis, it can be seen that *Obs_P* and W should be inversely proportional to each other, and the function is designed from this aspect. Four adaptive weight functions $W(Obs\_P)$ based on *Obs_P* in the map scene were designed and are shown in Figure 4.

If the adaptive weight $W(Obs\_P) <= 1$, the value of $h'(n)$ may be less than or equal to the actual distance from the current node to the target node, which can ensure that the optimal path is planned, but the number of nodes to be traversed is large, and the search efficiency will be reduced; if the adaptive weight $W(Obs\_P)$ is too large, the value of $h'(n)$ will be far more than the actual distance from the current node to the target node, which will easily lead the pathfinding to a locally optimal solution. In this case, the pathfinding optimality cannot be guaranteed. To enhance the real-time and search efficiency of the algorithm and ensure the final planning path is relatively short, the curve function *W3* shown in Figure 4 is finally selected as the adaptive weight factor $W(Obs\_P)$ of the heuristic function in this paper, and the total movement path distance estimation function $f(n)$ shown in Formula (7) is obtained.

$$f(n) = \sum_{i=1}^{n} S(i) + h'(n) = \sum_{i=1}^{n} S(i) + (-Obs\_P \times (1 - \ln(Obs\_P)) + 2) \times h(n) \times K \quad (7)$$

To verify the impact of the adaptive weight strategy on path planning, the traditional A* algorithm and the adaptive weight strategy algorithm are simulated in the same map scene, and the test results are shown in Figure 5. Suppose that when the raster map size is 30 × 30, the value of K is 1 in Formula (5) of the test, $a = b = c = d = 3$ in Equation (6), and A, B, and C are calculated in the ratio of 1:3:2 for the *Obs_P* of the obstacle distribution rate.

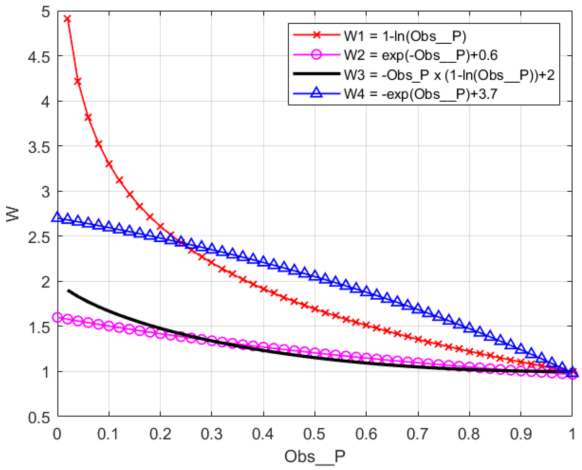
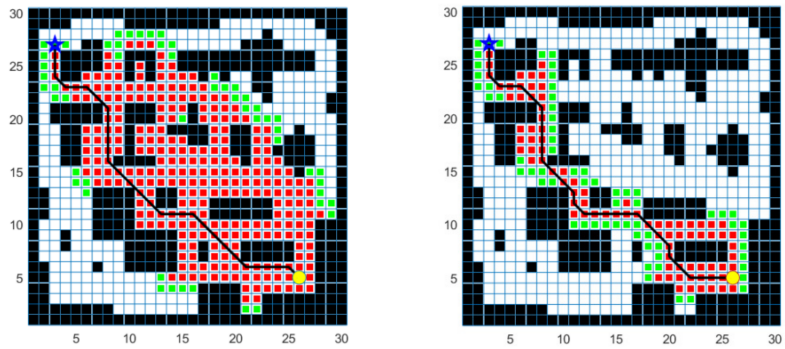


Figure 4. Four adaptive weight function curves based on barrier distribution rate.



(**a**) (**b**)

Figure 5. Two algorithmic tests: (**a**) Traditional A* algorithm; (**b**) adaptive weight strategy algorithm.

In Figure 5, the solid yellow circle is the starting node, the blue solid five-pointed star is the target node, the black node is the obstacle, the red node is the traversed and recorded node, and the green node is the discarded node. From Figure 5, it can be seen that the number of nodes traversed by the traditional A* algorithm is much greater than the number of nodes traversed by the adaptive weight strategy algorithm, and its performance indicators are shown in Table 2.

Table 2. Comparison between the traditional A* algorithm and the adaptive weight strategy algorithm.

Algorithm	Number of Nodes	Path Nodes	Number of Turns	Turn Angles (°)	Path Time (s)	Path Length (m)
Traditional A*	280	32	9	405	0.702	36.7990
Adaptive weight strategy	87	34	12	540	0.241	37.9706

The data in Table 2 show that the adaptive weight strategy causes the number of nodes traversed and the pathfinding time to be less than the traditional A* algorithm. The efficiency of the algorithm traversing nodes after taking the adaptive scene weight is higher

than the traditional A* algorithm. The variation curve of *Obs_P* and *W*(*Obs_P*) for some obstacles implemented by the adaptive weight strategy in Figure 5b is shown in Figure 6.

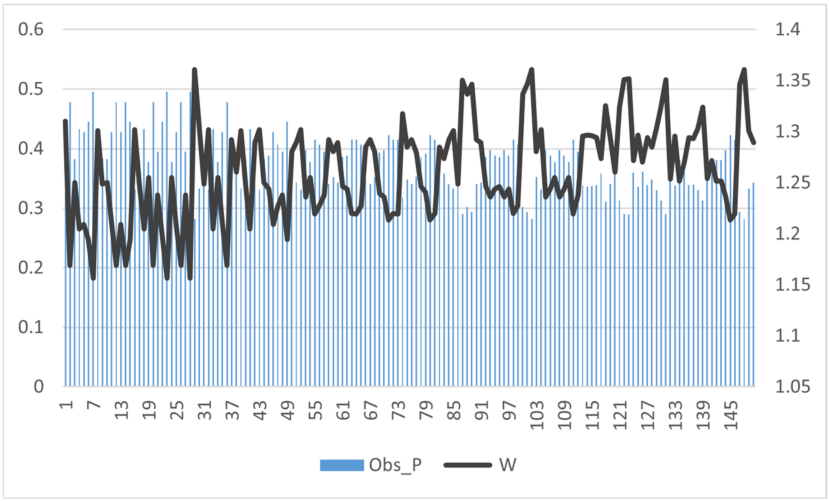


Figure 6. Function variation plot.

In Figure 6, the left vertical axis is the obstacle distribution rate and the right vertical axis is the corresponding adaptive weight size. From Figures 5b and 6 and Table 2, it can be seen that the adaptive weight strategy can effectively judge the surrounding obstacles in real-time when planning the road and adjust the adaptive weights according to the distribution rate of obstacles, thereby reducing the number of traversal nodes and the pathfinding time and improving the efficiency of traversal nodes.

In order to verify the robustness of the algorithm and ensure that the algorithm can adapt to the scene map change and regenerate a new route, the traditional A* algorithm and adaptive weight algorithm are tested by randomly generating obstacles in the map of Figure 5, and the planning results are shown in Figure 7, and the specific values of the corresponding performance parameters are shown in Table 3. In Figure 7, the solid blue line represents the adaptive weighting strategy algorithm, and the solid black line represents the traditional A* algorithm.

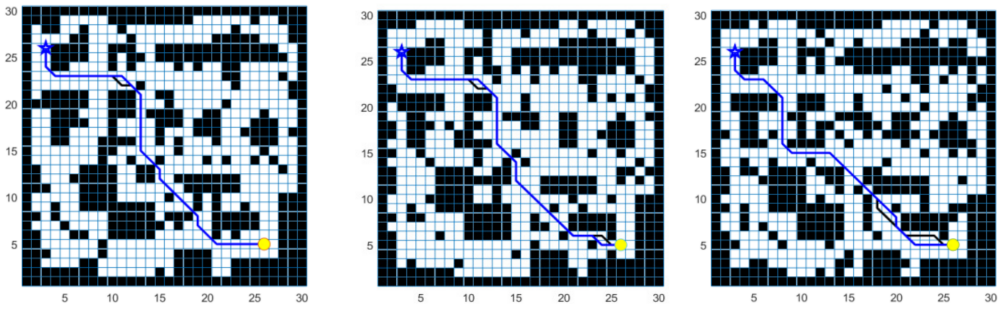


Figure 7. Algorithm robustness verification: (**a**) Randomly generate 120 obstacles; (**b**) randomly generate 150 obstacles; (**c**) randomly generate 180 obstacles.

In Figure 7, although the traditional A* algorithm and the adaptive weight strategy algorithm differ in the selection of some nodes, they can both find a safe and collision-free path. Therefore, both algorithms are acceptable in terms of stability, and the adaptive weight strategy algorithm can adapt itself according to the map obstacle distribution. The search step size can be adjusted in real time, and the pathfinding completeness of the algorithm can also be ensured. In addition, according to the data in Table 3, the adaptive weight

strategy algorithm is excellent in traversing node efficiency and search speed compared to the traditional A* algorithm. In order to ensure that the final comprehensive performance robustness of the algorithm proposed in this paper is better, the adaptive weight strategy algorithm will continue to be designed. In this way, the algorithm can be tilted towards a better search direction, so as to ensure its robustness and real-time adaptability.

Table 3. Comparison of the planning results of the two algorithms on a map with randomly generated obstacles.

Maps	Algorithm	Number of Nodes	Path Nodes	Number of Turns	Turn Angles (°)	Path Time (s)	Path Length (m)
(a)	Traditional A*	242	32	12	540	0.673	37.5563
	Adaptive weight strategy	93	34	10	450	0.255	37.5563
(b)	Traditional A*	230	33	12	540	0.614	36.9706
	Adaptive weight strategy	78	33	10	450	0.231	36.9706
(c)	Traditional A*	163	32	12	540	0.412	36.3848
	Adaptive weight strategy	60	32	10	450	0.189	36.3848

3.2. Extend the Neighborhood Priority Search Strategy

The traditional A* algorithm searches for four neighborhoods or eight neighborhoods, resulting in a limited search space and an optimal search path. So, it is necessary to design and select the scope of the search neighborhoods for the specific situation. If the neighborhood increases, the probability of finding the optimal path will increase, while the search speed will become slower; if the neighborhood decreases, the search speed will increase, but the optimal path will be not guaranteed, and pathfinding may even fail.

Considering the advantages and disadvantages of extending and reducing neighborhoods, in this paper, an extended neighborhood strategy based on the vector location relationship is designed to assign the search priority to neighbors and then search according to priority. Given that the search speed is reduced by too many neighborhoods, the 24 neighborhood directions that coincide with the traditional 8 neighborhoods are removed to form 16 neighborhood expansion directions, as shown in Figure 8.

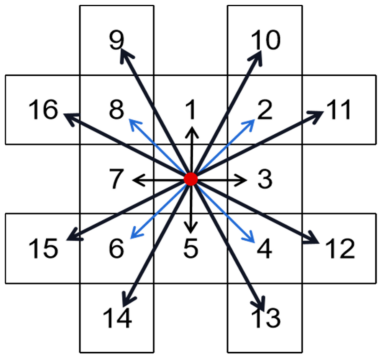


Figure 8. Extended 16 neighborhoods.

Note that (dx, dy) is the change in the horizontal coordinate between the current node and the adjacent node, and $S(i)$ is the step-by-step distance from the current node to the adjacent node. The dynamic programming matrix when searching using the traditional A* algorithm is:

$$\text{Motion} = [dx, dy, S(i)] \tag{8}$$

Based on the extended neighborhood, this paper assigns priority to the search neighborhood to enhance the heuristics of $g(n)$ so that it can improve the search speed while ensuring the optimal fit of the pathfinding direction. The black dotted line in Figure 9 represents the search vector from the starting node (parent node) to the target node, the

solid black line is the movement vector from the starting node to the child node, and the angle between the vectors is denoted as θ.

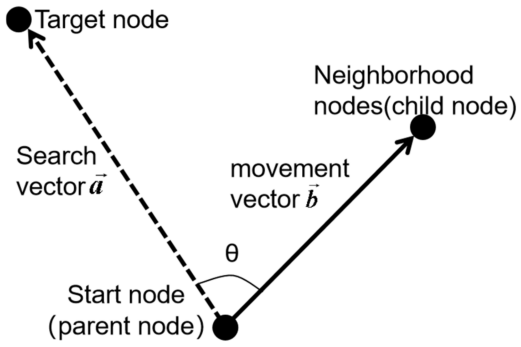


Figure 9. Vector angle representation.

Let $\vec{a} = (ax, ay)$, $\vec{b} = (bx, by)$, the cosine of the angle θ between the vectors is:

$$cos\theta = \frac{\vec{a} \cdot \vec{b}}{\left|\vec{a}\right| \times \left|\vec{b}\right|} = \frac{ax \times bx + ay \times by}{\sqrt{(ax)^2 + (ay)^2} \times \sqrt{(bx)^2 + (by)^2}} \tag{9}$$

According to Equation (9), the smaller the angle between vectors θ, the more its direction of movement fits the optimal search direction. Therefore, the function of $cos\theta$ is designed as the weight of the step-by-step movement distance $S(i)$ when searching for neighborhoods, ensuring that the neighborhood suitable for the search vector is preferentially selected during the pathfinding process, thereby enhancing the heuristic and guiding of $g(n)$ and improving the search speed. However, the value of this function should be moderate, because if it is too large, it cannot be applied to complex map environments, and if it is too small, the path optimization is not obvious enough. The improved $S(i)'$ is:

$$S(i)' = [(1 - cos\theta)/N + 1] \times S(i) = [(1 - cos\theta)/N + 1] \times \sqrt{(dx)^2 + (dy)^2} \tag{10}$$

In order to enhance the heuristics of $g'(n)$ and ensure the relative equilibrium between $g'(n)$ and $h'(n)$, the value of function $S(i)'$ should be moderate, if it is too large, it cannot be applied to complex map environments, and if it is too small, the path optimization is not obvious enough. From the above analysis, it can be seen that the value of N will directly affect the performance of the algorithm. In this paper, through the simulation results, the variables are compared after control, and the following table is finally obtained.

As can be seen from Table 4, the smaller the N value, the larger the $S(i)'$, and the algorithm performance is weaker than that of the traditional A* algorithm. The larger the N, the smaller the $S(i)'$, and the algorithm traverses the number of nodes, and the search speed is improved. Among them, N = 8 is the inflection point of the state transition, and the algorithm gradually tends to be stable. However, in different application scenarios, the value of N needs to be redetermined. In order for the algorithm proposed in this paper to be able to adapt to various occasions, the state transition inflection point value of 8 in the table is used as N.

At this point, the $g'(n)$ dynamic programming matrix for extending neighborhoods and assigning priorities is:

$$Motion = [dx, dy, S(i)'] \tag{11}$$

The pathfinding principle of algorithms (1), (7), and (10) and A* algorithms can be fitted to the priority search $f(n)$ as shown in equation (12).

$$f(n) = \sum_{i=1}^{n} S(i)' + h'(n) \tag{12}$$

Table 4. The influence of the selection of N value on the performance of the path planning algorithm.

		Number of Nodes	Path Nodes	Number of Turns	Turn Angles (°)	Path Time (s)	Path Length (m)
Traditional A* algorithm	8 search neighborhood	280	32	9	405	0.709	36.7990
	16 search neighborhood	35	17	15	321.0588	0.394	41.0197
Extend Neighborhood Priority Search Strategy	$N = 1$	35	21	13	386.3406	0.355	41.4274
	$N = 4$	35	22	15	480.3623	0.331	41.6055
	$N = 8$	35	21	15	321.0588	0.302	41.0197
	$N = 12$	35	21	15	321.0588	0.302	41.0197
	$N = 16$	35	21	15	321.0588	0.302	41.0197
	$N = 20$	35	21	15	321.0588	0.302	41.0197

3.3. Redundant Node Deletion Strategy

Aimed at the problem of too many summary storage traversal nodes and turning angles in the process of mobile robot path planning, this paper designs a redundant node deletion strategy based on the relationship between the location of the turning point and the connection to smooth the initial path. The specific steps are as follows:

Step 1: Calculate the position relationship between the three nearby nodes in the initial path, and judge and store the turning nodes. When the total number of turning nodes is less than 4, the loop ends.

Step 2: Determine if the four adjacent turning nodes (Nodes 1, 2, 3, and 4) can be connected directly without passing through obstacles.

Yes: Record the location of the endpoint in the shortest path and delete the other nodes.

No: Take node 4 as the starting point. Use Step 2 again to judge the three turning nodes after that. When the number of nodes is judged to be less than 3, go to Step 4.

Step 3: Take the record node as the starting node. Go to Step 2.

Step 4: Connect the recorded turning nodes and update the path. Go to Step 1

After several judgments, the redundant nodes are removed, resulting in a smoother path and fewer nodes to pass. The schematic diagram of the redundant node deletion strategy is shown in Figure 9.

In Figure 10a, A->B->C->D is the planning path, the yellow squares A, B, C, and D are the turning points, the red dotted lines indicate that the barriers between two points can be directly connected, and the blue dotted lines indicate that there are obstacles between two points that cannot be directly connected. Since both A->D and B->D in Figure 10a pass through obstacles, the redundant node deletion strategy can be used to conclude that A->C-> D is the optimal path. Assuming that the paths A->D, A->C, and B->D in Figure 10b do not pass through obstacles, the redundant node deletion strategy directly determines that A->D is the optimal path based on the principle of the shortest line segment between two points. If B->D in Figure 10c also does not pass through obstacles, the redundant node deletion strategy will compare the distance between A->C->D and A->B->D and then select the path with the shortest distance as the optimal search path.

To test the effectiveness of the redundant node deletion strategy, the map in Figure 10d was used for the path planning test, and the search path shown in Figure 10d was obtained. The solid black line is the planned path for the undeleted redundant nodes, and the solid

red line is the planning path for the deletion of the redundant node. The results in this figure show that after deleting the redundant nodes, the number of traversal nodes that need to be stored in the planned path is reduced and the path is smoother, which alleviates the storage pressure of the mobile robot, reduces the probability of collision between the mobile robot and obstacles, and also solves the problem of additional energy loss and waiting time generated by the robot when turning.

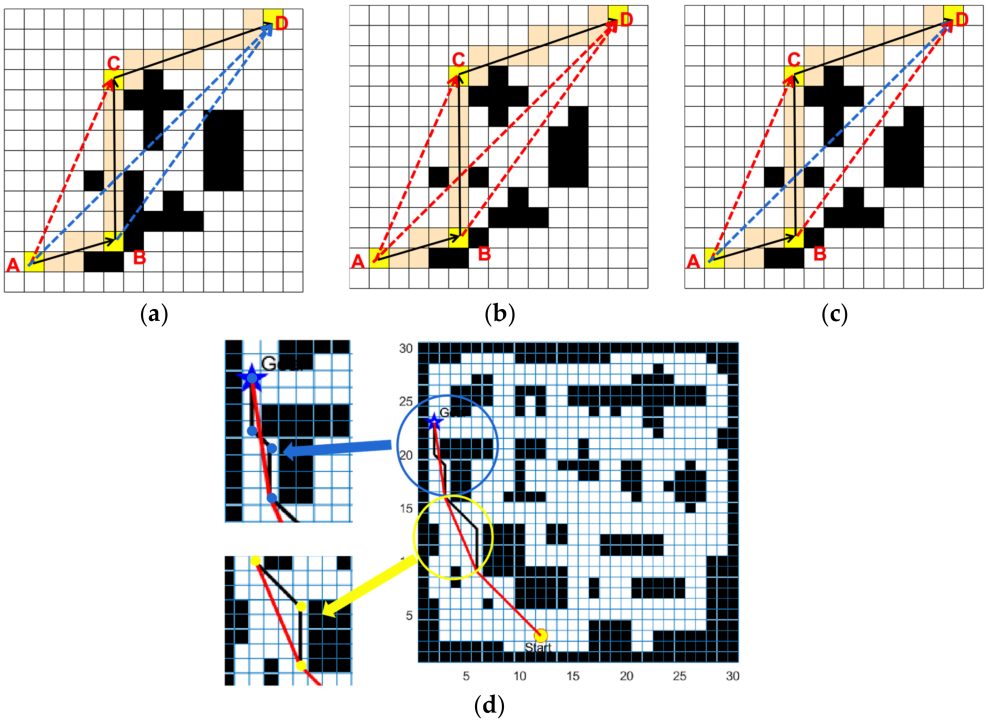


Figure 10. Redundant node removal principle and test: (**a**) No direct path; (**b**) three direct paths; (**c**) no direct path; (**d**) redundant node deletion test.

3.4. ENMSA-A Algorithmic Process*

The adaptive weight strategy grants the algorithm adaptability to arbitrary maps and improves the pathfinding speed, but the path is not smooth enough and the path length cannot be guaranteed to be shorter. The extended neighborhood priority search strategy uses 16 extended neighborhoods and assigns the priority search strategy to further reduce the total number of traversal nodes, the number of path nodes, and the total turning angle of the path, which can ensure that the path planned by the algorithm is relatively better and can appropriately reduce the pathfinding time. The redundant node deletion strategy can reduce the redundant nodes on the path, smooth the path processing, and reduce the node storage pressure of the mobile robot. In order to ensure the A* algorithm can effectively reduce the number of nodes traversed and the total turning angle, and improve the search speed in any map path planning, in this paper, the adaptive weight adjustment strategy was integrated with the extended neighborhood priority search strategy and the redundant node deletion strategy, which combines the advantages of the three aspects to make up for the shortcomings of a single aspect, finally forming the ENMSA-A* algorithm proposed in this paper whose algorithm flow is shown in Figure 11.

In Figure 11, the blue box is an adjustment of the adaptive weight of the traditional A* algorithm, the red box is the extended neighborhood priority search strategy introduced, and the green box part is the redundancy node deletion strategy introduced. By introducing a three-sided adjustment strategy, the robustness of ENMSA-A* proposed in this paper can be improved, and it can be optimized in terms of the number of nodes traversed and the smoothness of the path.

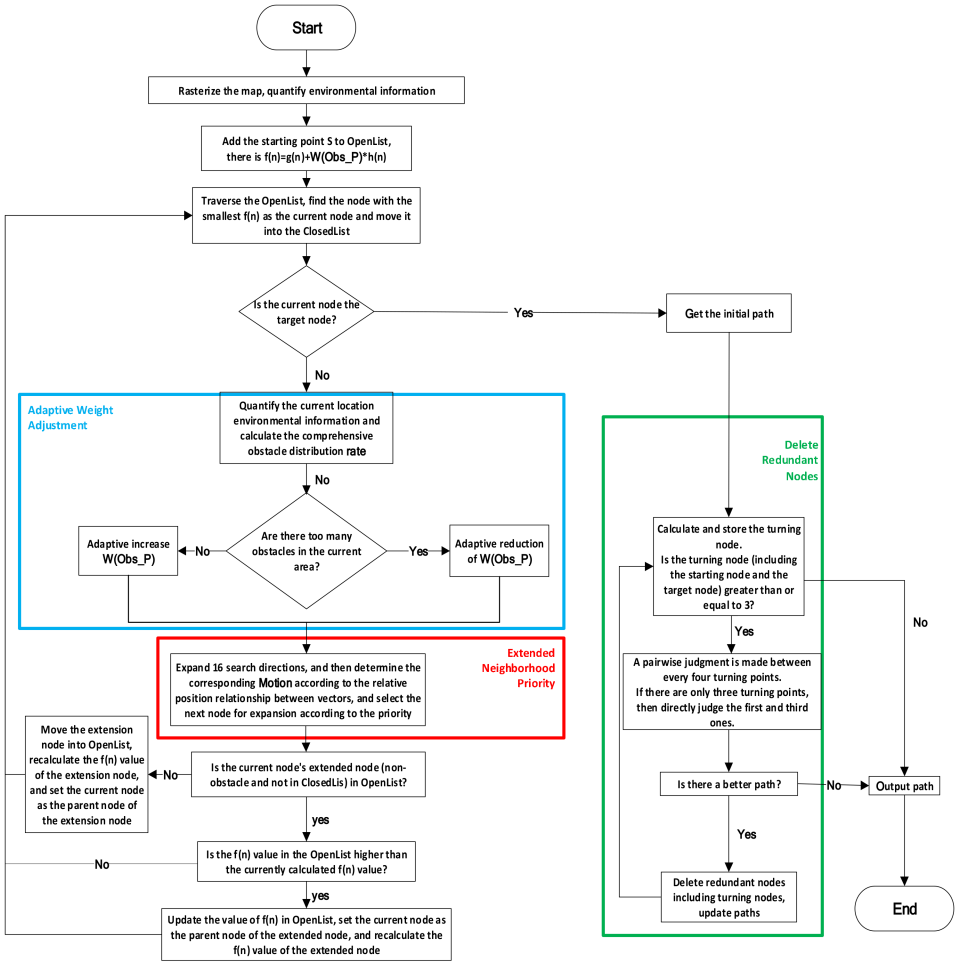


Figure 11. ENMSA-A* algorithm flowchart.

3.5. Performance Robustness Evaluation Index Design of Algorithms

Because of the different emphases of the algorithm researchers on the improvement of the traditional A* algorithm, it is difficult to compare the advantages and disadvantages of various improved A* algorithms. In order to facilitate the comparison of the performance robustness of different improved algorithms, this paper proposes the algorithm validity V as the criterion for judgment. The number of nodes traversed during the pathfinding process is recorded as S, the number of turning nodes is recorded as Wp, the final planned path length is recorded as L, and the pathfinding time is recorded as T. By synthesizing the five performance indicators, namely, the number of nodes, the turning point, the total turning angle, the pathfinding time, and the path length, the algorithm validity V evaluation index shown in Equation (13) is designed.

$$V = w \times (1 - \frac{S_n}{S_0}) + x \times (1 - \frac{Wp_n}{Wp_0}) + y \times (1 - \frac{L_n}{L_0}) + z \times (1 - \frac{T_n}{T_0}) \quad (13)$$

Among them, w, x, y, and z are custom constant coefficients, subscript 0 represents the traditional A* algorithm indicator value, and subscript n represents an improved A* algorithm indicator value. This paper is an improvement of the problem in which the number of nodes and the total turning angle of the traditional A* algorithm are too large when planning the path. To ensure that the final planned path can be relatively optimal, $w = x = y = 3$ and $z = 1$. From Equation (13), it can be seen that the algorithm validity V of the traditional A* algorithm is 0, and if the algorithm validity V of the improved algorithm is greater, the comprehensive performance of the improved algorithm is better. Since the algorithm validity V value depends on multiple performance indicators, even in the same

map scenario, the location of the starting node and the target node will lead to changes in the relevant performance indicators. Therefore, it is necessary to ensure that the algorithm validity V calculation of various algorithms is based on the same scene map, the same starting node and target node, and the same size raster map. The resulting V can be used as a performance robustness evaluation index to evaluate the advantages and disadvantages of each algorithm.

4. Comparative Analysis of ENMSA-A* Algorithm

In order to test the performance of the ENMSA-A* algorithm proposed in this paper, MATLAB 2020b was used on the Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz computer to simulate the traditional A* algorithm, the improved A* algorithm [25], the improved A*algorithm [26], the improved A*algorithm [27], and the ENMSA-A* algorithm proposed in this paper in six scenario maps.

The path planning simulation results are shown in Figure 12, and the related data are plotted as a line chart in Figure 13.

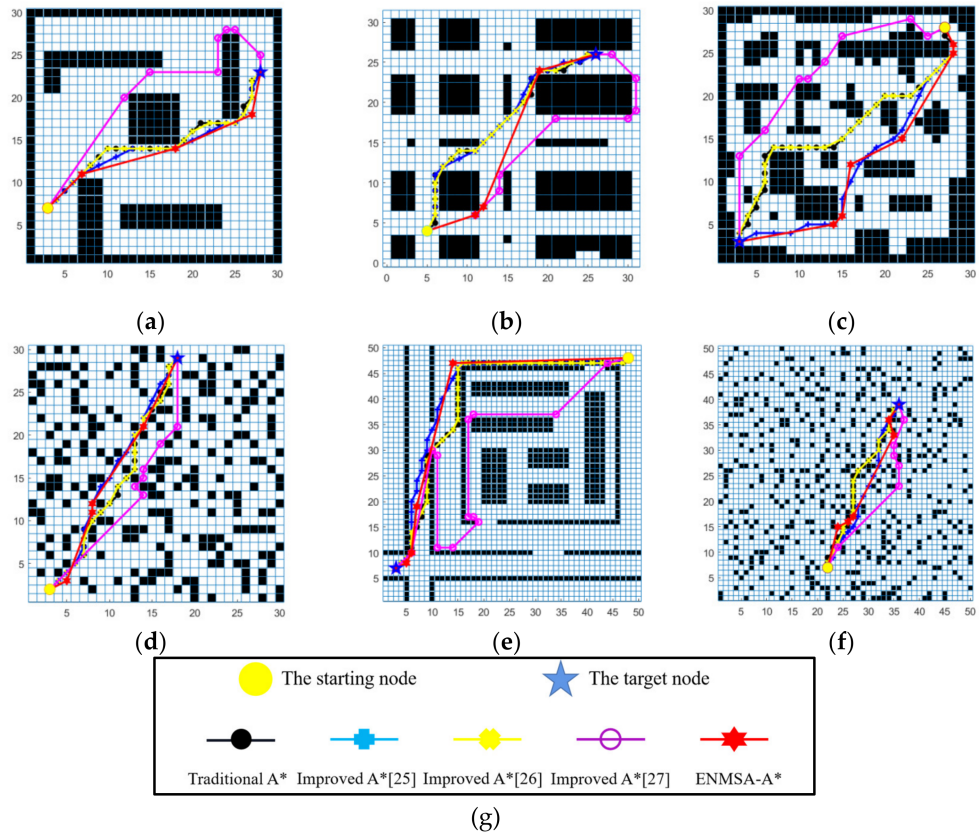


Figure 12. Demonstration of improved algorithm path planning for different map environments: (**a**) Simple Map 1; (**b**) Simple Map 2; (**c**) Complex map; (**d**) 30 × 30 random map; (**e**) 50 × 50 extended map; (**f**) 50 × 50 random map; (**g**) the shapes and various color curves in this figure refer to the content.

Based on the above experimental and simulation data analysis, combined with the algorithm validity value V, the traditional A* algorithm, other improved algorithms in the literature, and the ENMSA-A* algorithm mentioned in this paper, the data comparison is shown in Table 5.

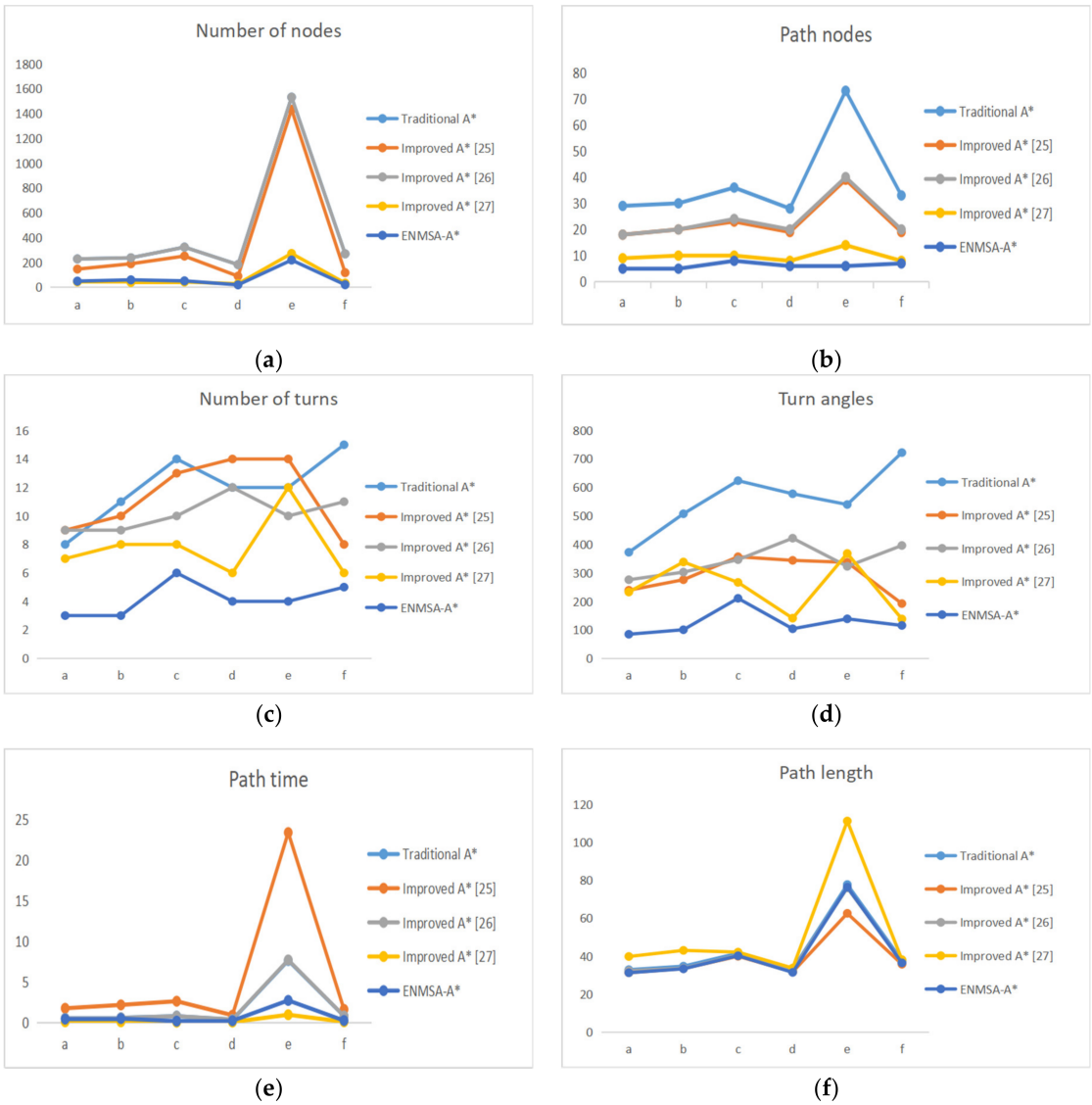


Figure 13. Comparison data of algorithm path planning simulation: (**a**) Number of nodes; (**b**) path nodes; (**c**) number of turns; (**d**) turn angles (°); (**e**) path time (s); (**f**) path length (m).

Table 5. Percentage of performance reduction for each improved A* algorithm in the literature compared to the traditional A* algorithm.

	Traditional A*	Improved A* [25]	Improved A* [26]	Improved A* [27]	ENMSA-A*
Number of nodes	1	19.67	0	83.54	84.95
Path nodes	1	39.74	38	74.24	83.84
Number of turns	1	5.56	15.28	34.72	65.28
Turns angles	1	47.78	38.16	55.54	77.38
Path time	1	−200.79	−1.1	85.85	58.47
Path length	1	8.92	2.28	−19.47	3.36
V	0	0.2827	1.2026	4.4438	5.5555

The data in Table 5 shows that the improved A* algorithm in the literature [25] introduces a security factor and an extended 20-neighborhood search, so it is the shortest in terms of the length of the planning path, but its pathfinding time increased by 200.79% compared with the traditional algorithm, and the number of traversal nodes and the total inflection angle were also weaker than the ENMSA-A* algorithm proposed in this paper. The improved A* algorithm in the literature [26] is not robust due to the improved

weighting ratio needing to be averaged by multiple artificial weighting adjustments based on different scene maps, which are pre-planned and rely on scene environment information, and the pathfinding time is 1.1% higher than the traditional A* algorithm. Although the evaluation index of algorithm performance robustness is higher than the traditional algorithm by 1.2026, the improvement in other aspects is not optimal. The improved A* algorithm in the literature [27] introduces the parent node and enhances the Chebyshev distance as the heuristic function weight, which reduces the total number of traversing nodes by 83.54% compared with the traditional algorithm, and the pathfinding time is also reduced by 85.85%, but at the cost of increasing the path length by 19.47%. There are multiple locally optimal solutions in the path, which cannot guarantee global path optimization, so the evaluation index of algorithm performance robustness V is 4.4438.

The ENMSA-A* algorithm proposed in this paper is optimal in four aspects: Traversal nodes, path nodes, the path turning angle, and the total turning angle. Compared with the traditional A* algorithm, the number of traversal nodes is reduced by 84.95%, the path nodes are reduced by 83.84%, the turning angle on the path is reduced by 62.28%, the total turning angle is reduced by 77.38%, the pathfinding time is reduced by 58.47%, and the total distance of the planned path is reduced by 3.36%. In summary, the ENMSA-A* algorithm proposed in this paper has good performance, and its performance robustness evaluation index V is 5.555, which is the highest among the four improved algorithms, indicating that the ENMSA-A* algorithm has excellent performance robustness.

Through the simulation test of Figure 12 and the comparative analysis of the data in Figure 13 and Table 5, the robustness of the ENMSA-A* algorithm proposed in this paper is further verified. Among them, stability is reflected in the ENMSA-A* algorithm, which can be applied to a variety of map scenarios and can adjust the search weight adaptively. Because the ENMSA-A* algorithm traverses fewer nodes and has faster speed, smoother path planning, and optimal or relatively optimal total path length, the performance robustness of the ENMSA-A* algorithm is superior and more suitable for the daily work of mobile robots.

5. Conclusions

Aimed at the shortcomings of the traditional A* algorithm, such as poor robustness, the large number of nodes traversed, and the excessive total turning angle, this paper proposes an ENMSA-A* algorithm based on the combination of the adaptive weight adjustment heuristic function strategy, the extended domain priority search strategy, and the redundant node deletion strategy.

Firstly, the heuristic function of the traditional A* algorithm is introduced to the adaptive weight adjustment factor that can judge the distribution of obstacles in real time, which enhances the robustness of the algorithm, reduces the number of traversal nodes, and improves the search speed. Secondly, by extending the 16 neighborhoods, the algorithm can find a better path and reduces the number of traversal nodes and the pathfinding time by assigning a priority search strategy; then, by removing the redundant nodes between turning angle, the planned path is further smoothed so that the total turning angle is smaller and the number of traversal nodes is less.

In order to verify the EMSA-A* algorithm proposed in this paper, simulation tests were performed in MATLAB. By controlling variables, several algorithms were tested on path planning in the same scene map, and relevant data were recorded. We chose a different scene map later to avoid accidents. Simulations and comparison experimental tests showed that the robustness, real-time adaptability, search efficiency, path optimization, and path smoothness of the proposed ENMSA-A* algorithm were significantly improved.

However, the ENMSA-A* algorithm proposed in this paper relies more on map information, and if the map accuracy error is large, it will directly affect the final path planning result. In future work, the ENMSA-A* algorithm will be integrated with the local path planning algorithm and the trajectory tracking algorithm to reduce its dependence on

the global map information, realize dynamic obstacle avoidance, and become more suitable for the daily work of mobile robots.

Author Contributions: Conceptualization: Z.Y. and K.S.; resources: Z.Y. and K.S.; validation: K.S., Y.L., and X.Z.; writing—original draft: Z.Y., K.S., and T.H.; writing—review and editing: Z.Y. and K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Fundamental Research Funds for the Central Universities of Southwest Minzu University (No. 2021101) and the Science and Technology Project of Chengdu Science and Technology Bureau (No. 2021-RK00-00079-ZF).

Acknowledgments: The authors would like to thank anonymous referees for their kind suggestions and corrections that helped improve the original manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhu, D.Q.; Yan, M.Z. Survey on technology of mobile robot path planning. *Control Decis.* **2010**, *25*, 961–967.
2. Bao, Q.Y.; Li, S.M.; Shen, H.; Men, X.H. Survey of local path planning of autonomous mobile robot. *Transducer Microsyst. Technol.* **2009**, *28*, 1–4.
3. Lu, X.H.; Zhang, G.L. Summarization on indoor service robot navigation. *Robot* **2003**, *25*, 80–87.
4. Liang, J.; Han, D.; Pan, Z.; Chen, L.; Chen, F.; Du, W. Review on Critical Technologies of Robot-based Intelligent Garages. *J. Mech. Eng.* **2022**, *58*, 1–20.
5. Hernández, B.; Giraldo, E. A Review of Path Planning and Control for Autonomous Robots. In Proceedings of the 2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA), Barranquilla, Colombia, 1–3 November 2018; pp. 1–6.
6. DíazBayona, J.L.; GonzálezPilonieta, J.D.; MolinaLache, R.M.; Prada, S.R. Development of a control system for path following applications in an AGV using computer vision. In Proceedings of the 2020 IX International Congress of Mechatronics Engineering and Automation (CIIMA), Cartagena, Colombia, 4–6 November 2020; pp. 1–6.
7. Hart, P.E.; Nilsson, N.J.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
8. Wang, Z.; Hu, X.; Li, X.X.; Du, Z.Q. Overview of Global Path Planning Algorithms for Mobile Robots. *Comput. Sci.* **2021**, *48*, 19–29.
9. Mac, T.T.; Copot, C.; Tran, D.T.; De Keyser, R. Heuristic approaches in robot path planning: A survey. *Robot. Auton. Syst.* **2016**, *86*, 13–28. [CrossRef]
10. Shen, K.; You, Z.; Liu, Y.; Huang, T. Mobile robot planning based on improved A* algorithm. *Appl. Res. Comput.* **2023**, *40*, 75–79.
11. Zhang, H.; Li, H.; Liu, H.; Xu, W.; Zou, Y. Path Planning for Logistics Unmanned Aerial Vehicle in Urban Area. *J. Transp. Syst. Eng. Inf. Technol.* **2020**, *20*, 22–29.
12. Zhang, X.Y.; Zou, Y.S. Collision-free path planning for automated guided vehicles based on improved A* algorithm. *Syst. Eng.-Theory Pract.* **2021**, *41*, 240–246.
13. Pang, L.; Cao, Z.; Yu, J. A pedestrian-aware collision-free following approach for mobile robots based on A* and TEB. *Acta Aeronaut. Astronaut. Sin.* **2021**, *42*, 524909.
14. Qi, Z.G.; Huang, P.F.; Liu, Z.X.; Han, D. Research on Path Planning Method of Spatial Redundant Manipulator. *Acta Autom. Sin.* **2019**, *45*, 1103–1110.
15. Zhang, W.; Zhang, Y.; Zhang, H. Path planning of coal mine rescue robot based on inproved A* algorithm. *Coal Geol. Explor.* **2022**, *50*, 185–193.
16. Liu, G.; Ma, Y.; Qi, F.; Xu, Y. Flight path planning for urban logistics UAV based on improved A*-APF algorithm. *Flight Dyn.* **2022**, *40*, 16–23.
17. Lian, Y.; Xie, W. Improved A* path planning algorithm for vision-guided multi-AGV system. *Control Decis.* **2021**, *3*, 1881–1890.
18. Jiao, Q.; Chen, X.; Zheng, Z.; Bai, Y.; Liu, Y.; Zhang, Z.; Sun, L. Dynamic path planning of unmanned aerial vehicle based on crowd density prediction. *Prog. Geogr.* **2021**, *40*, 1516–1527. [CrossRef]
19. Wang, H.; Yin, P.; Zheng, W.; Wang, H.; Zuo, J. Mobile Robot Path Planning Based on Improved A* Algorithm and Dynamic Window Method. *Robot* **2020**, *42*, 346–353.
20. Yan, X.; Chang, T.; Guo, L. Research on path planning of unmanned vehicle in off-road battlefield environment. *J. Ordnance Equip. Eng.* **2022**, *43*, 288–293.
21. Ouyang, M.; Ma, Y. Path planning for gravity aided navigation based on improved A* algorithm. *Chin. J. Geophys.* **2020**, *63*, 4361–4368.
22. Zhang, D.; Sun, X.; Fu, S.; Zheng, B. Cooperative path planning in multi robots for intelligent warehouse. *Comput. Integr. Manuf. Syst.* **2018**, *24*, 410–418.
23. Zhao, C.; Jiang, H.; Xu, M.; Man, W.; Yang, W.; Chen, F. Application of improved A algorithm in unmanned ship path planning. *J. Zhejiang Univ. Technol.* **2022**, *50*, 615–620.

24. Ju, C.; Luo, Q.; Yan, X. Path Planning Using an Improved A-star Algorithm. In Proceedings of the 2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan), Jinan, China, 23–25 October 2020; pp. 23–26.
25. Junwei, Y.; Jing, H.; Guang, C. Improved Safety-First A-Star Algorithm for Autonomous Vehicles. In Proceedings of the 2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM), Shenzhen, China, 18–21 December 2020; pp. 706–710.
26. Wang, Z.; Zeng, G.; Huang, B.; Fang, Z. Global optimal path planning for robots with improved A* algorithm. *J. Comput. Appl.* **2019**, *9*, 2517–2522.
27. Liu, S.; Ma, Y.; Meng, S.; Sun, S. Improved A* algorithm for path planning of AGV. *J. Comput. Appl.* **2019**, *39*, 41–44.
28. Zhang, H.; Zhang, Y.; Liang, R.; Yang, T. Energy-efficient pathplanning method for robots based on improved A* algorithm. *Syst. Eng. Electron.* **2023**, *45*, 513–520.
29. Li, X.; Hu, X.; Wang, Z.; Du, Z. Path Planning Based on Combinaion of Improved A-STAR Algorithm and DWA Algorithm. In Proceedings of the 2020 2nd International Conference on Artificial Intelligence and Advanced Manufacture (AIAM), Manchester, UK, 15–17 October 2020; pp. 99–103.
30. Gao, H.; Ma, Z.; Zhao, Y. A Fusion Approach for Mobile Robot Path Planning Based on Improved A* Algorithm and Adaptive Dynamic Window Approach. In Proceedings of the 2021 IEEE 4th International Conference on Electronics Technology (ICET), Chengdu, China, 7–10 May 2021; pp. 882–886.
31. Yang, H.; Li, Y.; Sun, H.; Li, Z. Research on path planning of mobile robots with optimal path. *J. Mach. Des.* **2022**, *39*, 58–67.
32. Zhong, X.; Tian, J.; Hu, H.; Peng, X. Hybrid Path Planning Based on Safe A* Algorithm and Adaptive Window Approach for Mobile Robot in Large-Scale Dynamic Environment. *J. Intell. Robot. Syst.* **2020**, *99*, 65–77. [CrossRef]