Enhanced Center Constraint Weighted A* Algorithm for Path Planning of Petrochemical Inspection Robot

Xin Lai · JiaHe Li · Jonathon Chambers

Received: 24 November 2020 / Accepted: 11 June 2021 / Published online: 8 July 2021
© The Author(s), under exclusive licence to Springer Nature B.V. 2021

Abstract
In many practical applications of robot path planning, finding the shortest path is critical, while the response time is often overlooked but important. To address the problems of search node divergence and long calculation time in the A* routing algorithm in the large scenario, this paper presents a novel center constraint weighted A* algorithm (CCWA*). The heuristic function is modified to give different dynamic weights to nodes in different positions, and the node weights are changed in the specified direction during the expansion process, thereby reducing the number of search nodes. An adaptive threshold is further added to the heuristic function to enhance the adaptiveness of the algorithm. To verify the effectiveness of the CCWA* algorithm, simulations are performed on 2-dimensional grid maps of different sizes. The results show that the proposed algorithm speeds up the search process and reduces the planning time in the process of path planning in a multi-obstacle environment compared with the conventional A* algorithm and weighted A* algorithm.

Keywords CCWA* algorithm · Dynamic weights · Heuristic function · Path planning

1 Introduction

With the rapid development of intelligence and information technology, it is critical to improve the level of intelligence in monitoring systems in oil fields, petrochemical enterprises and other industries. A high-tech tool is required to build "smart oil fields" and realize the unattended digital management of equipment by introducing petrochemical inspection robots into oil field production inspection work. A rail navigation inspection robot adopts magnetic navigation tracking or radio frequency identification (RFID) and other auxiliary positioning devices. An autonomous navigation inspection robot does not need the support of the track, and relies on the airborne equipment to independently perform the patrol operation with a wide range of activities, maneuverability and all-round monitoring. At present, the application of trackless navigation technology to unmanned autonomous patrol systems has become a research hotspot. The inspection robot uses SINS/GPS for autonomous navigation and positioning, in which the key technology is path planning and trajectory tracking [21, 22]. Under the condition that the map of the actual site is known, an accurate and efficient path planning method is needed in the process of the robot moving from the starting point to the target point.

Path planning for a mobile robot entails finding an optimal or suboptimal safe collision-free path for the robot from the starting point to the target point in an environment with obstacles. The path planning problem is usually constrained by a large amount of environmental information including many obstacles, which has been proven to be an uncertain polynomial difficult problem [7].

The commonly used path planning algorithms are the breadth-first search algorithm [15], the depth-first search algorithm [11], the heuristic search algorithm [25], and Dijkstra’s algorithm [32]. Dijkstra’s algorithm is a typical breadth-first search algorithm, which has the advantage of finding the shortest path between nodes in the map, but the search efficiency is low, and it is difficult to meet the demand...
of fast path planning. The advantage of the depth-first algorithm is that it takes less memory to run, but the search speed is slow, and it cannot guarantee to find the optimal path. The best-first search algorithm [3] is a kind of heuristic search algorithm, and is evaluated based on the distance from the target node, and can quickly direct the search to the target node, thereby greatly improving the search efficiency of the algorithm, but it usually cannot obtain a reasonable shortest path. All of these algorithms eventually find their endpoints while having certain limitations. Therefore, a variety of improved algorithms have been proposed based on these algorithms. A flood filling algorithm proposed by Francy et al. [9] is one of the most commonly used maze solving algorithms, and it is based on an improved depth-first search algorithm. The flood filling algorithm can find the shortest path from the starting point to the target point. Duchon et al. [5] proposed a jump point search strategy to reduce the number of nodes to be visited during the traversal, and the running speed is fast. The quality of routing in the probabilistic roadmap method [8] depends to a large extent on the number of nodes, which can improve the speed of the algorithm, but when the distribution of nodes is unreasonable or the number of nodes is small, the method is not perfect, and the routing is not globally optimal in most cases. An ant colony algorithm [26, 29, 41] has good global optimization ability. The rapidly exploring random tree algorithm [18, 28] can effectively avoid the algorithm falling into the defect of local search by introducing random sampling, and finds a feasible path quickly. The bidirectional rapidly exploring random tree algorithm [13, 35, 36] can effectively solve the problems of high-dimensional space and complex constraint. A particle swarm algorithm [34] has the characteristics of fast convergence speed and few setting parameters. Shyam used probabilistic movement primitives (ProMPs) to learn the trajectory distribution from which any number of trajectories could be sampled [33]. ProMP generated versatile initial solutions for the optimisation, so the chance to find poor local minima is reduced. This method optimized the local trajectory by learning from demonstration results in a much smoother or natural looking trajectory, and it is effective in several complex obstacle avoidance scenarios. Liu [23] proposed a novel geometric analysis-based trajectory planning approach. They imposed a harmonic drive and computed the dynamic projection onto a hyper-manifold to reduce complexity and to characterize coupling, the trajectory planning is converted into geometric analysis and trajectory optimization, such that the optimal locomotion can be achieved.

The A* algorithm was first proposed in 1968 by Hart et al. [14], it is a heuristic search algorithm proposed by combining the advantages of the best-first search algorithm and Dijkstra’s algorithm [37, 42]. The A* algorithm uses heuristic information to guide the search direction, which has been widely used to find the optimal solution in a short time, and has strong expansibility and adaptability to different scenarios. The A* algorithm usually focuses on the optimal path and ignores the response time of the system. However, the time consumption of the device is also important to evaluate the performance of the algorithm in the actual field environment of a petrochemical inspection robot system. Researchers have made different improvements for different application scenarios. In terms of mobile robot path planning, the heuristic graph search algorithm represented by the sparse A* search algorithm [19, 27] is most widely used in engineering due to its high efficiency and strong robustness. Wang et al. [38] proposed an improved A* path planning algorithm, which solved the problem that the path of the traditional A* algorithm contains too many redundant points, and obtained the minimum turning angle at the inflection point to clarify the robot’s turning. The infinite neighborhood A* algorithm proposed by Xin et al. [43] solves the problem where the direction of motion is limited to a multiple of π/4, but the search speed decreases. Gu [12] proposed an improved algorithm that avoids generating a path through the vertex of the obstacle grid employing the priority sub-node generation strategy. Botea et al. [2] proposed the hierarchical path-finding A* (HPA*) algorithm to reduce the complexity in path-finding on grid-based maps. The HPA* algorithm adapts to dynamically changing environments, and the hierarchy can be extended to several abstraction levels, making it scalable for large space problem. Qin et al. [31] proposed a double-layer grid map method to optimize the A* algorithm. The high-level A* algorithm determines the next large grid local map, and the low-level A* algorithm performs the path planning on the combined local map. This algorithm optimizes the quality of the route and effectively reduces the length, turning times and turning angle of the planned path of the robot. Goldberg et al. [10] developed a new bidirectional variant of A* search, and used A* search in combination with a new graph-theoretic lower-bounding technique based on landmarks and the triangle inequality. Gao [6] proposed a bidirectional time-efficient A* algorithm to find the path, and exploited a multi-nearest neighbor grid distance calculation scheme to improve efficiency and smooth the path, but it is not suitable for large-scale complex maps. Lin et al. [20] proposed an improved real-time A* algorithm based on the introduction of the parent node and the change of the heuristic cost weight to control the path planning of intelligent robots. This method can usually obtain the optimal path, and the speed is greatly improved compared with the A* algorithm.
However, while pursuing a higher speed, there is a higher space complexity, which makes the algorithm take up more storage space. Korf [17] proposed an iterative deepening A* algorithm (IDA*), which optimized the routing strategy of the A* algorithm without the need to make state judgments and evaluate rankings. Wang et al. [39] optimizes the A* algorithm by using straight lines with heuristic factors, thereby attaining high stability and the potential to obtain the shortest path. Wang et al. [40] proposed the smooth A* algorithm. When there are no obstacles on the connection between the front and back nodes of the intermediate node, the intermediate node of the extension line is deleted, and a better path can be obtained at the expense of time efficiency.

To sum up, the A* algorithm is a typical heuristic search algorithm which can significantly reduce the number of extended states while ensuring the optimization of solutions. Although there have been many studies on the improvement of the A* algorithm, which have greatly improved its application in different environments, these advantages will be weakened when the search area is too large or the robot moves faster. These methods do not yet address the inefficiency of algorithms in environments with large search regions and multiple obstacles.

It was found that A* algorithm can search for the optimal solution if the value of the heuristic function from the current state to the target state is less than the cost of the optimal path. Based on this, we can draw a conclusion: the larger value of the heuristic function, the smaller the number of expansion states, thereby improving the search efficiency. When the value of the heuristic function is greater than the cost of the optimal solution, the A* algorithm can quickly find a feasible path, although the optimality of the solution cannot be guaranteed.

Pohl proposed a weighted A* algorithm [30] (WA*) that multiplies the heuristic function $h(n)$ of the A algorithm by an inflation factor $\alpha (\alpha \geq 1)$, which can reduce the search time and search area. Although the weighted A* algorithm sacrifices the optimal path, it can quickly determine a suboptimal path, and its cost does not exceed $\alpha$ times of the optimal path [4]. In this way, the WA* algorithm can solve the problem of planning failure in a limited time. With more time, the WA* algorithm will reduce $\alpha$ and look for a better solution again until the allocation time expires. However, this method does not reuse the previous planning work, and the efficiency is not very high. There is still room for improvement in the WA* algorithm.

A novel improved weighted A* algorithm is proposed in the paper. The new optimization method is based on a raster method graph algorithm [1, 16, 45], and improves the efficiency of the algorithm in a multi-obstacle environment while ensuring that near optimal paths are obtained. Our main contributions are summarized as follows:

1. To improve the efficiency of the A* algorithm, a novel center constraint weighted A* algorithm (CCWA*) is proposed. A new optimization factor for the heuristic function of the A* algorithm is designed - namely a dynamic weight.
2. The CCWA* algorithm assigns the different weights to the heuristic functions of different nodes based on the distance between the node location and the central node. In this way, it can control the nodes to explore the direction of expansion, effectively reducing the search nodes, and ensure that the search direction will not diverge excessively.
3. The expected value $\varepsilon$ is added on the basis of giving the dynamic weights to the heuristic function. The selected expected value is the coefficient of the dynamic weight, it further reduces the node’s exploration area by changing the expected value $\varepsilon$. An adaptive threshold method is proposed to change the weights of the heuristic function adaptively in real time according to the map size, thus making the expansion node toward the direction of the target node, and improving the search efficiency.
4. The experimental results show that, compared with the A* algorithm and the weighted A* algorithm, the CCWA* algorithm can reduce the planning time and improve the path planning efficiency for relatively large and complex environments and ensure that the path is near optimal.

## 2 Environmental Modelling

In the path planning of a petrochemical inspection robot, it is first necessary to fuse the environmental information obtained by various sensors, and establish a map model representing the surrounding environment. A grid unit has been widely used in path planning for its simple, effective and easy-to-implement way to describe environmental information. The abbreviations and symbols used in this paper are shown in Table 1.

In this paper, a map consisting of a grid with side length 1 is used to represent the environmental information around the robot, as shown in Fig. 1. The grid method expresses the working environment of the robot as a series of grids with the same size, and divides the grid into two states according to the environmental information: the occupied state and the
| Notations          | Definitions                                                                 |
|-------------------|-----------------------------------------------------------------------------|
| CCWA*             | A center constraint weighted A* algorithm                                     |
| ATCCWA*           | An adaptive threshold center constraint weighted A* algorithm                |
| WA*               | A weighted A* algorithm                                                      |
| \( \alpha \)      | The inflation factor of the WA* algorithm                                     |
| \( \varepsilon \) | The expected value of the CCWA* algorithm                                     |
| \( \delta \)      | The adaptive parameter of the prediction model of \( \varepsilon \)           |
| \((x_a, y_a)\)    | The node in search of the A* algorithm                                        |
| \((x_t, y_t)\)    | The target node of the A* algorithm                                           |
| \(n_s(x_s, y_s)\) | Starting node of the CCWA* algorithm                                          |
| \(n_d(x_d, y_d)\) | The target node of the CCWA* algorithm                                        |
| \(n_o(x_o, y_o)\) | The center node with the starting point and the target node as the diameter circle |
| \(n_c(x_c, y_c)\) | The node in search of the CCWA* algorithm                                     |
| \(n'(x', y')\)   | The next search node of the CCWA* algorithm                                   |
| \(r_1\)           | Half of the distance difference between the starting node \(n_s\) and the target node \(n_d\) |
| \(r_2\)           | The distance from the node \(n_c\) to the node \(n_o\)                       |
| \(k\)             | The dynamic parameter, which is the absolute value of the difference between \(r_1\) and \(r_2\) |
| \(K(n)\)          | Weighted factor of the CCWA*                                                |
| \(f(n)\)          | The evaluation function                                                      |
| \(g(n)\)          | The actual cost                                                             |
| \(h(n)\)          | The heuristic function                                                       |
| \(H(n)\)          | The new heuristic function                                                   |

idle state. The blank grid in Fig. 1 represents the area that the inspection robot can pass through, which reflects being in the idle state; the black grid represents the obstacles in the environment, which the robot cannot pass at this time. To facilitate the study and experiments, two assumptions are used in the paper:

1. It is assumed that both map boundaries and obstacle boundaries are established, taking into account the safe distance of the detected robot, and the robot is considered as a particle that can only move within the grid. The black areas are obstacles and the white areas are reachable roads.

2. In the absence of a boundary and obstacles, the robot can move to the grid in the four directions. Based on the raster map environment, four motions are defined without considering the effect of the robot’s own height, as shown in Fig. 2: up, down, left and right.

3 A* Algorithm

3.1 Basic Principles

The basic principle of the A* algorithm is a heuristic path search method. The heuristic information contained in the evaluation function quickly lock the target direction. The general form of the evaluation function of the A* algorithm

![Fig. 1 Grid map](image1)

![Fig. 2 The direction of robot motion](image2)
is shown in (1):

\[ f(n) = g(n) + h(n) \]  

where \( f(n) \) is the evaluation function from the starting point through the node to the end point, which represents the total cost value of node \( n \) to the end point. \( g(n) \) is the actual cost from the starting point to the node \( n \), which represents the real cost value of the path from the starting point to node \( n \). \( h(n) \) is the heuristic function, which represents the estimated cost from node \( n \) to the end point. \( h(n) \) makes the algorithm have the potential of heuristic search, so the selection of the heuristic function \( h(n) \) is crucial to the entire algorithm. It is generally assumed that the optimal cost value from node \( n \) to the end point is \( h^*(n) \), and \( h(n) \) and \( h^*(n) \) have the following three relationships:

(1) If \( h^*(n) = h(n) \), each expansion point belongs to a point on the optimal path, and the expansion will definitely have an optimal solution.
(2) If \( h^*(n) > h(n) \), that is an optimal path can be found, but it may pass through many invalid points. In the extreme cases, when \( h(n) = 0, f(n) = g(n) \), it becomes the breadth-first search algorithm, thus making the algorithm lose its heuristic.
(3) If \( h^*(n) < h(n) \), it can find a path to the destination in a relatively short time, but may not be the optimal solution.

Only when \( h^*(n) > h(n) \) holds, the algorithm has the ability of completeness and optimal solution. The key of the A* algorithm to find the optimal solution (the shortest path) lies in the selection of the evaluation function \( f(n) \): the value of \( h(n) \) is less than the actual value of the distance from \( n \) to the target node. In practice, if you want to accurately obtain the optimal path, you need to construct the function closer to the actual shortest path. This needs to refer to more heuristic information, but the more constraints, the more \( h(n) \) needs to be calculated. This means that the A* algorithm can theoretically obtain the optimal solution, but its biggest disadvantage is that it requires too much space. We can use some time-for-space methods to improve performance, for example, by choosing an evaluation function to reduce the solution space.

### 3.2 A* Algorithm Heuristic Function

The design of the heuristic function for the A* algorithm is very important, which determines the search efficiency. The construction of the heuristic function has no rules to follow, so long as it is conducive to the solution of the problem. In practice, a real-valued function is usually used to optimize through the experimental process. We suppose that the coordinate \((x_a, y_a)\) is any node in the search, and the coordinate \((x_t, y_t)\) is the target node. The most commonly used methods for the A* algorithm are:

(1) Manhattan distance \([46]\), the sum of the projection distances of line segments on the two coordinate axes in the rectangular coordinate system of Euclidean space:

\[ h(n) = |x_a - x_t| + |y_a - y_t| \]  

(2) Euclidean distance \([44]\), the linear distance between two points in Euclidean space:

\[ h(n) = \sqrt{(x_a - x_t)^2 + (y_a - y_t)^2} \]  

(3) Chebyshev distance \([24]\), the maximum value of each component difference between two vectors:

\[ h(n) = \max \{|x_a - x_t|, |y_a - y_t|\} \]

The calculation of Euclidean distance is simple. It only needs to calculate the straight-line distance between two points, but the error is large when encountering a complicated map; Manhattan distance and Chebyshev distance are easy to realize, and they are widely used in search and have high efficiency.

We can observe a petrochemical site from Fig. 3. The topographic structure of petrochemical plants is more complex, the number of nodes in the network is large, and the search width is deeper. An inspection robot working in this kind of field finds a near optimal path in a limited time, and its heuristic function must be efficient and accurate. Considering the practical application scenarios, Manhattan distance is chosen as the heuristic function to be more accurate in the multi-obstacle environment.

**Fig. 3** Petrochemical plants with complex terrains in real life
On the basis of (1), an improved heuristic function method in the WA* algorithm takes the form [30]:

\[
f(n) = g(n) + \alpha \times h(n)
\] (5)

where \( \alpha \) is the inflation factor of the WA* algorithm. The heuristic function is endowed with dynamic weights. In the early stages, speed is given priority to speed up the search, while in the later stages, accuracy is given priority to expand the search. The heuristic function including distance and direction can ensure that the search direction will not deviate excessively, thereby greatly increasing the amount of heuristic information, which can speed up the search process and accuracy. For the weight that changes dynamically during the search process, the value of \( \alpha \) keeps decreasing, and repeated attempts to find a relatively better path are made until the path converges or \( \alpha = 1 \). The disadvantage of this algorithm is that the prediction process of \( \alpha \) is difficult to control, so the algorithm is complicated to calculate and difficult to implement.

### 4 CCWA* Algorithm Principle and Implementation

#### 4.1 CCWA* Algorithm Principle

The A* algorithm takes more time and sacrifices efficiency in finding the shortest paths, especially for large and complex maps. The actual working environment of petrochemical inspection robots is complex, and the requirements for optimal path during work are reduced, but the requirements for a real-time response are higher to deal with different work tasks.

In general, the heuristic function \( h(n) \) is set to the Euclidean distance between two nodes, which is often smaller than the actual shortest distance. Under the premise of ensuring the acceptability of the algorithm, the proposed algorithm is shown in Fig. 4, the heuristic function \( h(n) \) in the A* algorithm is improved. The modification of \( h(n) \) is used to control the process of selecting subsequent nodes, which enhances the node expansion direction and depth of the A* algorithm, and ensures the direction of the next search node \( n' (x', y') \) will not be excessively divergent. The amount of heuristic information is greatly increased, it can speed up the search process and improve its accuracy and efficiency. If the search node is beyond the circular region with a diameter of the distance from the starting node to the target node, the farther away the node will be from this circular region, and the greater the value of the heuristic function \( h(n) \).

![Fig. 4 Geometric representation of the concept underlying the CCWA* algorithm](image)

**Lemma 1** Each step of the A* algorithm always has a node \( n^* \) before termination, which has the following characteristics on the OPEN table:

1. \( n^* \) is the best way to achieve the goal.
2. A* algorithm has found the best path to \( n^* \).
3. \( f(n^*) \leq f(n_0) \).

Then this algorithm is acceptable. If further extended, any function whose distance is not greater than \( h(n_0) \) is suitable, so these improvements are still feasible.

In this paper, the proposed algorithm is improved based on the above lemma. We set two parameters: the distance threshold \( r_1 \) is half of the distance difference between the starting node \( n_s \) and the target node \( n_d \); \( r_2 \) is the distance from the node \( n_e \) to the node \( n_o \).

\[
r_1 = \frac{|x_s - x_d| + |y_s - y_d|}{2}
\] (6)

\[
r_2 = |n_e - n_o|
\] (7)

where \( (x_s, y_s) \) denotes the X and Y coordinates of the starting node \( n_s \), \( (x_d, y_d) \) denotes the X and Y coordinates of the target node \( n_d \), \( n_e \) denotes the exploration node, \( n_o \) denotes the center node with the starting point and the end point as the diameter circle. We compare \( r_1 \) with \( r_2 \), and define \( k \) as the dynamic parameter

\[
k = |r_1 - r_2|
\] (8)

The heuristic function \( h(n) \) is given a weighted factor \( K(n) \),

\[
K(n) = (1 + k \times \varepsilon)
\] (9)

where \( \varepsilon \) is the expected value, and the specific parameters of its range are the desired paths. We give constraints to
the heuristic function, so as to add directional guidance to the selection of the exploration nodes during the operation of the OPEN table. Employing (6), (7), (8) and (9), a new heuristic function is obtained,

\[ f(n) = g(n) + K(n) \ast h(n) \quad (10) \]

Defining the new heuristic function \( H(n) \), (10) becomes the following

\[ f(n) = g(n) + H(n) \quad (11) \]

where \( H(n) = K(n) \ast h(n) \). Assuming that you do not want your path to exceed 1000 steps, we can make \( \varepsilon = 0.001 \), which can reduce the search area of the A* algorithm and limit the value of \( K(n) \). If the value of \( K(n) \) is too large, it will cause \( H(n) \gg g(n) \), which will result in the g(n) function failing and the A* algorithm will become a breadth-first search algorithm reducing the optimality of the path. The threshold is set in the algorithm, when the node is in the region of a circle with the starting point and the end point as the diameter (when \( r_2 \leq r_1 \)), \( k = 0 \), the CCWA* algorithm performs an optimal path search in this region. When the nodes are far from the target area, the \( H(n) \) value will be larger to ensure that the search will not deviate excessively; when the search node is close to the target area, the \( H(n) \) value will be smaller, which reduces the importance of the heuristic function.

### 4.2 CCWA* Algorithm Implementation

The steps of the proposed algorithm are shown in Algorithm 1. The OPEN table stores nodes that have not yet been searched, the CLOSED table stores the nodes that are fetched from the OPEN table to fit the algorithm.

### 5 Experiment and Simulation of the Improved A* Algorithm

The proposed algorithm and existing algorithms are coded in C++ and run on a computer with a 2.50 GHz Intel Core i7-6500U CPU. The simulation experiments are conducted in the Windows environment and Linux environment respectively, and the map in which the length of each grid is 1 and the cost of moving each grid is 1.

#### 5.1 CCWA* Algorithm Simulation Experiment

**Scenario 1:** In the Windows environment, a simulation experiment is conducted on a map with a map size of 2,251 nodes. The simulation results are shown in Fig. 5. The black areas indicate the obstacle area, the white areas indicate the access area, and the red line indicates the planned route. The upper opening is the starting point and the lower opening is the end.

Comparing the red paths in Figs. 6a and b, it is intuitive to see that the path of the proposed algorithm is suboptimal. It can be seen from Table 2 that the proposed algorithm...
Fig. 5 Simulation diagram of A* algorithm and CCWA* algorithm in the Scene 1

(a) A* algorithm simulation results. (b) The proposed algorithm with $\varepsilon = 0.01$. 

Fig. 6 Simulation diagram of A* algorithm and CCWA* algorithm in Scene 2

(a) A* algorithm simulation results. (b) The proposed algorithm with $\varepsilon = 0.001$. (c) The proposed algorithm with $\varepsilon = 0.01$. (d) The proposed algorithm with $\varepsilon = 0.1$.

Table 2 Performance comparison of the proposed algorithms and A* algorithm in Scenario 1

| Algorithm                        | Time (s) | Path length |
|----------------------------------|----------|-------------|
| A* algorithm                     | 0.0195   | 203         |
| The proposed algorithm $\varepsilon = 0.01$ | 0.0153   | 227         |

Table 3 Performance comparison of the proposed algorithms and A* algorithm in Scenario 2

| Algorithm                        | Time (s) | Path length | Exploration nodes |
|----------------------------------|----------|-------------|-------------------|
| A* algorithm                     | 0.0657   | 137         | 4850              |
| The proposed algorithm $\varepsilon = 0.001$ | 0.0652   | 137         | 4816              |
| The proposed algorithm $\varepsilon = 0.01$ | 0.0589   | 139         | 4200              |
| The proposed algorithm $\varepsilon = 0.1$ | 0.0438   | 141         | 3158              |
is compared with the A* algorithm. 1) The search time is reduced from 19.5 ms to 15.3 ms, and the time is optimized by 21.6%. 2) The number of path steps increases from 203 to 227, and the path increases by 11.8%. The simulation results show that in the map scene with many obstacles, the proposed algorithm sacrifices a smaller path, in exchange for a greater improvement in algorithm operation time.

Scenario 2: In the Ubuntu environment, a simulation experiment is conducted on a map with a map size of 14,539...
Table 4 Performance comparison of the proposed algorithms and A* algorithm in Scenario 3

| Algorithm                | Time (s) | Path length | Exploration nodes |
|--------------------------|----------|-------------|-------------------|
| A* algorithm             | 0.1101   | 296         | 7948              |
| The proposed algorithm $\varepsilon = 0.001$ | 0.0933 | 298         | 7036              |
| The proposed algorithm $\varepsilon = 0.01$  | 0.0579  | 304         | 4062              |
| The proposed algorithm $\varepsilon = 0.1$   | 0.0563  | 340         | 3911              |

We employ the time, path length and exploration nodes as performance indexes. Table 3 is comparison of the proposed algorithms and A* algorithm in Scenario 2. Compare with the A* algorithm, 1) when $\varepsilon = 0.001$, the time is optimized by 0.2%, the path length is unchanged, and the exploration node is optimized by 0.7%. 2) when $\varepsilon = 0.01$, the time is optimized by 10.3%, the path length is increased by 1.4%, and the exploration node is optimized by 13.4%. 3) when $\varepsilon = 0.1$, the time is optimized by 33.2%, the path length is increased by 2.9%, and the exploration node is optimized by 34.8%.

It can be seen from Table 3 above that the expected value $\varepsilon$ has a greater influence on the search time, path and exploration nodes. When $\varepsilon = 0.1$, the time and exploration nodes are greatly optimized. Although the path has a certain increase, it is still within the acceptable range. When $\varepsilon = 0.001$, the path is optimal, but the time and exploration nodes are optimized less than that of the other $\varepsilon$ cases.

Figure 7 demonstrates that the expected value $\varepsilon$ has a great influence on the search path of the algorithm, and selecting the appropriate expected value will yield more desirable results. When the value $\varepsilon$ is large, the value $H(n)$ is obviously large, which leads to the increase of the path and the contraction of the exploration node towards the target area; when the value $\varepsilon$ is small, the value $k^*\varepsilon$ in (9) is too small, and the value of $K(n)$ approaches 1, and the effect of the proposed algorithm is not obvious. Because Scenario 2 is more complex than scenario 1, search nodes need to go through more and further areas. The farther the weighting factor $K(n)$ is from the central node, the greater the effect of the value of $\varepsilon$. It can be seen that when $\varepsilon = 0.1$, a significant deviation occurs in the value of the heuristic function $H(n)$ for nodes far from the central point, leading to the path obviously being offset from the center region. It is this deviation from the central area that causes the path to increase.

Scenario 3: In the Ubuntu environment, a simulation experiment is performed on the complex scene map with a map size of 54,979 nodes. The simulation results are shown in Fig. 8, the definition of each color area and the starting point and the end point are the same as in Scenario 2.
Table 4 lists the search time, path and exploration nodes under different expected value $\epsilon$. As shown in Fig. 8, compared to the A* algorithm, 1) when $\epsilon = 0.001$, the time is optimized by 15.3%, the path length is increased by 0.6%, and the exploration node is optimized by 11.5%. 2) when $\epsilon = 0.01$, the time is optimized by 47.3%, the path increases by 2.7%, and the exploration node is optimized by 48.9%. 3) when $\epsilon = 0.1$, the time is optimized by 48.8%, the path length is increased by 14.8%, and the exploration node is optimized by 50.8%. In Scenario 3, the simulation results again demonstrate that the expectation value $\epsilon$ has a significant impact on the search path.

As can be seen from Figs. 7 and 9, compared to the traditional A* algorithm, we can see that the proposed algorithm is optimized by 0.8%−48.8% in path node time, and the space utilization is increased by 0.7%−50.3%, but the proposed algorithm has a path length increased by 0%−14.8%. Therefore, the A* algorithm only uses the shortest path length as the criterion to generate the optimal path, in contrast, the proposed algorithm also considers the path length, the algorithm response time, and the space utilization.

The proposed algorithm can greatly improve the efficiency and the space utilization of the algorithm whilst slightly increasing the path length, and has certain advantages. In this case, we further compare the proposed algorithm with the WA* algorithm [30] under the inflation factor $\alpha$ in two cases. The simulation results of scenario 3 using the WA* algorithm are shown in Figs. 10 and 5.

Table 5 Performance comparison of the WA* algorithms in Scenario 3

| Algorithm           | Time (s) | Path length | Exploration nodes |
|---------------------|----------|-------------|-------------------|
| WA* algorithm $\alpha = 1.80$ | 0.0659   | 304         | 4691              |
| WA* algorithm $\alpha = 2.35$ | 0.0633   | 310         | 4020              |

It can be seen from Tables 4 and 5: 1) when $\alpha = 1.80$ in the WA* algorithm and $\epsilon = 0.01$ in the proposed algorithm, the path length is the same for both algorithms, the search time increases by 13.8%, and the search nodes increase by 15.5%. 2) when $\alpha = 2.35$ in the WA* algorithm and $\epsilon = 0.01$ in the proposed algorithm, the search node is reduced by 1%, the search time is increased by 9.3%, and the path length is increased by 1.9%. The simulation results show that the proposed algorithm optimizes the search range, reduces search time, and improves search efficiency compared to the WA* algorithm.

To sum up, the proposed algorithm has advantages over the A* algorithm and WA* algorithm. Although the proposed algorithm can ensure that the path is near optimal and improves the efficiency of the path planning, it is greatly affected by $\epsilon$, which makes it have certain limitations. Therefore, the proposed algorithm still has room for improvement.

Fig. 11 Simulation diagram of ATCCWA* in Scene 3
### Table 6 Performance of ATCCWA* in Scenario 3

| Algorithm            | Time (s) | Path length | Exploration nodes |
|----------------------|----------|-------------|-------------------|
| The ATCCWA* algorithm| 0.0618   | 298         | 4549              |

#### 5.2 An Adaptive Threshold CCWA* Algorithm Experiment

It has been found experimentally that when the node searches outside the target region ($r_1 < r_2$), as the search node is farther away from the center, the proposed algorithm will make the value of $K(n)$ larger as the $k$ becomes larger. The algorithm will become a breadth-first search algorithm when $K(n)$ is too large, causing the $H(n)$ function to be too large and the $g(n)$ function to fail. Therefore, to enhance the search accuracy based on improving search efficiency, we set an threshold for the CCWA* algorithm. Set the upper and lower limits of $k^* \varepsilon$ to be:

$$k^* \varepsilon = \begin{cases} H, & \omega < H \\ \omega, & L \leq \omega \leq H \\ L, & \omega \geq L \end{cases}$$

(12)

$H$ and $L$ are respectively the upper and the lower limit of $k^* \varepsilon$, and $\varepsilon$ is dynamically adjusted during the search process. When the search node is closer to the target area, the smaller the $\omega$ is, the higher the search accuracy is, which ensures the authenticity of the path. When the search node is far away from the target area, the greater the weight of $\omega$, the search speed will increase, and the accuracy of the search range will also decrease. If $\omega$ is too large, the algorithm will fail, so set the upper limit $H$ to ensure that the function does not diverge excessively.

An adaptive threshold CCWA* algorithm (ATCCWA*) is proposed in this paper to further improve the proposed algorithm. It limits the search range of nodes and changes the weight of the heuristic function in real time. On the basis of ensuring the search accuracy, the search speed is accelerated. The parameter settings in the simulation are: the upper limit $H = 1$, the lower limit $L = 0$, and $\varepsilon$ is

$$\varepsilon = \begin{cases} \delta, & k < 10 \\ \delta/10, & 10 < k < 100 \end{cases}$$

(13)

We build a prediction model of $\varepsilon$, and it is determined by

$$\delta = \frac{|x_s - x_d|}{x_{max}} + \frac{|y_s - y_d|}{y_{max}}$$

(14)

where $\delta$ denotes the adaptive parameter of the prediction model of $\varepsilon$, which predicts the expected value of based on the ratio of the distance from the start point to the end point and the map size. When the search node range exceeds the distance set by the observation model, the value of $\varepsilon$ will be reduced. $x_{max}$ and $y_{max}$ denote the maximum value in the $x$ direction and $y$ direction on the map, respectively.

We use the ATCCWA* in scenario 3. The experimental results are shown in Fig. 11 and Table 6. Compared with the A* algorithm in Table 4, the ATCCWA* algorithm is optimized by 43.9% in time, and the exploration node is optimized by 42.8%, while the path length only grows 0.5%. ATCCWA* algorithm is superior to the A* algorithm in terms of search time and search nodes, thus improving the overall performance. The performance of the proposed algorithm is similar to that of the CCWA* algorithm. The proposed algorithm can adjust the $\varepsilon$ adaptively according to

### Table 7 Performance of the ATCCWA* algorithms in the actual map

| Algorithm            | Time (s) | Path length | Exploration nodes |
|----------------------|----------|-------------|-------------------|
| The A* Algorithm     | 0.2879   | 593         | 19824             |
| The ATCCWA* Algorithm| 0.2231   | 593         | 18700             |
the map size in real time, which has better adaptability and optimality of the path.

We conducted experiments on the actual petrochemical map. The simulation results are shown in Fig. 12 and Table 7. The simulation shows that the ATCCWA* algorithm still performs well in the actual environment, the path is optimal, the exploration node and exploration time are reduced.

6 Conclusion

A novel CCWA* algorithm and ATCCWA* algorithm are proposed in the paper. Dynamic weights are assigned to the heuristic function in these algorithms. The adaptive threshold of the ATCCWA* algorithm further enhances the performance and adaptability of the algorithm. The simulation results demonstrated that the proposed algorithm has higher search speed and accuracy than the A* algorithm and WA* algorithm. The proposed algorithm can improve the efficiency of path planning in a multi-obstacle large map environment, make sure that the search direction does not deviate excessively, and achieve a path which is near optimal. It can be used for inspection mobile robots in petrochemical plants with complex terrain structures. For future work, we plan to further explore and optimize the weight functions to improve algorithm performance.

Author Contributions Xin Lai is responsible for overall design, programming and writing, Jiahe Li is responsible for programming and writing, and Jonathon Chambers is responsible for overall guidance and proofreading.

Funding This work was supported in part by the Open Fund of the State Key Laboratory of Oil and Gas Reservoir Geology and Exploitation under Grant PLN2020-10, in part by the Sichuan Science and Technology Department Application Foundation Project under Grant 2019YJ0311, in part by the State Administration of National Security Project under Grant Sichuan-0006-2018AQ, in part by the Open Fund of the Key Laboratory of Oil and Gas Equipment, Ministry of Education under Grant OGE201702-06.

Declarations

Ethical Approval Conflict of interests The authors declare that they have no conflict of interest.

Consent to Participate The authors consent to participate.

Consent to Publish The authors consent to Publish.

Competing Interests The authors declare no competing interest.

References

1. Antikainen, H.: Using the hierarchical pathfinding A* algorithm in GIS to find paths through rasters with nonuniform traversal cost. ISPRS Int. J. Geo-Inf. 2(4), 996–1014 (2013). https://doi.org/10.3390/ijgi2040096
2. Botea, A., Müller, M., Schaeffer, J.: Near optimal hierarchical path-finding (HPA*). J. Game Dev. 1, 7–28. (2004)
3. Cui, Z.X., Gu, Z.H.: Shortest path search of game map based on A* algorithm. Softw. Guide 17, 145–147. https://doi.org/10.19734/j.issn.1001-3695.2017.010982 (2019)
4. Dean, T., Boddy, M.: An analysis of time-dependent planning. In: Seventh National Conference on Artificial Intelligence (AAAI 88), pp. 49–54. Springer, Paul (1988)
5. Duchoň, F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T., Jurišica, L.: Path planning with modified a star algorithm for a mobile robot. Procedia Eng. 96, 59–69 (2014). https://doi.org/10.1016/j.proeng.2014.12.098
6. Gao, M.D., Zhang, Y.N., Zhu, L.Y.: Bidirectional time-efficient A* algorithm for robot path planning. Appl. Res. Comput. 36(3), 792–795+800. https://doi.org/10.19734/j.issn.1001-3695.2017.010982 (2019)
7. Ge, K.W., Wang, B.P.: Global path planning method for mobile logistics robots based on raster graph method. Bullet. Sci. Technol. 35(11), 72–75+80. https://doi.org/10.13774/j.cnki.kjtb.2019.11.013 (2019)
8. Geraerts, R., Overmars, M.H.: A Comparative Study of Probabilistic Roadmap Planners. Springer, Berlin. https://doi.org/10.1007/978-3-540-45058-0_4 (2002)
9. Golda, A.F., Aridha, S., Elakkiya, D.: Algorithmic agent for effective mobile robot navigation in an unknown environment. In: 2009 International Conference on Intelligent Agent and Multi-Agent Systems, pp. 1–4. Chennai. https://doi.org/10.1109/IAMAI.2009.5228050 (2009)
10. Goldberg, A.V., Harrelson, C.: Computing the shortest path: A* search meets graph theory. In: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, pp. 156–165. Vancouver, British (2004). https://doi.org/10.1137/1070432.1070455
11. Gong, J.h.: Depth first search algorithm and its improvement. Modern Electron. Technol. (22):90–92. https://doi.org/10.3909/j.issn.1004-371X.2007.22.032 (2007)
12. Gu, C.: Application of improved A* algorithm in robot path planning. Electron. Des. Eng. 22(19), 96–98+102. https://doi.org/10.3969/j.issn.1674-6236.2014.19.031
13. Han, J., Seo, Y.: Mobile robot path planning with surrounding point set and path improvement. Appl. Soft. Comput. 57, 35–47 (2017). https://doi.org/10.1016/j.asoc.2017.03.035
14. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4(2), 100–107 (1968). https://doi.org/10.1145/1056777.1056779
15. Jetto, L., Longhi, S., Venturini, G.: Development and experimental validation of an adaptive extended Kalman filter for the localization of mobile robots. IEEE Trans. Robot. Autom. 15(2), 219–229 (1999). https://doi.org/10.1109/70.760343
16. Jiang, L., Li, J., Ma, X.X., Nie, W.K., Zhu, J.Y., Lei, B.: Voronoi path planning based on improved skeleton extraction. J. Mech. Eng. 56(13), 138–148 (2020). https://doi.org/10.3901/JME.2020.13.138
17. Korf, R.E.: Depth-first iterative-deepening: an optimal admissible tree search. Artif. Intell. 27(1), 97–109 (1985). https://doi.org/10.1016/0004-3702(86)90035-4
18. LaValle, S.: Planning Algorithms. https://doi.org/10.1017/CBO9780511546877 (2006)
19. Li, M., Zhang, Y., Li, S.: The Gradational route planning for aircraft stealth penetration based on genetic algorithm and sparse a-star algorithm. MATEC Web Conf. 151, 04001. https://doi.org/10.1051/matecconf/201815104001 (2018)
20. Lin, M., Yuan, K., Shi, C., Wang, Y.: Path planning of mobile robot based on improved A* algorithm. In: 2017 29th Chinese Control and Decision Conference (CCDC), pp. 3570–3576. Chongqing (2017). https://doi.org/10.1109/CCDC.2017.7979125
21. Liu, P., Yu, H., Cang, S.: Optimized adaptive tracking control for an underactuated vibro-driven capsule system. Nonlinear Dyn. 94(3), 1803–1817 (2018). https://doi.org/10.1007/s11071-018-4458-9
22. Liu, P., Yu, H., Cang, S.: Adaptive neural network tracking control for underactuated systems with matched and mismatched disturbances. Nonlinear Dyn. 98(2), 1447–1464 (2019). https://doi.org/10.1007/s11071-019-05170-8
23. Liu, P., Yu, H., Cang, S.: Geometric analysis-based trajectory planning and control for underactuated capsule systems with viscoelastic property. Trans. Inst. Meas. Control. 40(7), 2416–2427 (2018). https://doi.org/10.1177/0142331217708833
24. Liu, S.W., Ma, Y., Meng, S.F., Sun, S.Q.: Improved A* algorithm for path planning of AGV. J. Comput. Appl. 39(S2), 41–44. https://doi.org/10.1016/S1877-0509(19)30148-X
25. Lu, Y.L., Fan, X.P., Zhang, H.: Heuristic search assisted active localization for mobile robot. Robot. 34(5), 590–595 (2012). https://doi.org/10.3724/SP.J.1218.2012.00590
26. Mac, T., Copot, C., Tran, D., Keyser, R.: Heuristic approaches in robot path planning: A survey. Robot. Auton. Syst. 86, 13–28 (2016). https://doi.org/10.1016/j.robot.2016.08.001
27. Meng, B., Gao, X.: UAV path planning based on bidirectional sparse A* research algorithm. In: 2010 International Conference on Intelligent Computation Technology and Automation, vol. 3, pp. 1106–1109. Changsha. https://doi.org/10.1109/ICICTA.2010.235 (2010)
28. Neto, A., Macharet, D., Campos, M.: On the Generation of Trajectories for Multiple UAVs in Environments with Obstacles. J. Intell. Robot. Syst. 57, 123–141 (2011). https://doi.org/10.1007/s10866-009-9365-3
29. Wang, P., Lin, H.T., Wang, T.S.: An improved ant colony system algorithm for solving the IP traceback problem. Inform. Sci. 326(1), 172–187 (2016). https://doi.org/10.1016/j.ins.2015.07.006
30. Pohl, I.: Heuristic Search Viewed as Path Finding in a Graph 1(3), 193–204. https://doi.org/10.1006/ijins.1999.0123 (1997)
31. Qin, Y.X., Wang, H.Q., Du, C.J.: Path Planning of Mobile Robot Based on Double-layer A* Algorithm. Manuf. Autom. 36(24), 21–25+40. https://doi.org/10.3969/j.issn.1009-0134.2014.24.006 (2014)
32. Shim, H.S., Kim, H.S., Jung, M.J., Choi, I.H., Kim, J.H., Kim, J.O.: Designing distributed control architecture for cooperative multi-agent system and its real-time application to soccer robot. Robot. Auton. Syst. 21, 149–165 (1997). https://doi.org/10.1016/0921-8890(97)00023-7
33. Shyam, R.A., Lightbody, P., Das, G., Liu, P., Gomez-Gonzalez, S., Neumann, G.: Improving local trajectory optimisation using probabilistic movement primitives. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2666–2671. https://doi.org/10.1109/IROS40897.2019.8967980 (2019)
34. Song, B., Wang, Z., Zou, L.: On global smooth path planning for mobile robots using a novel multimodal delayed PSO algorithm. Cogn. Comput. 9, 5–17. https://doi.org/10.1007/s12559-016-9442-4 (2017)
35. Sudhakara, P., Ganapathy, V., Balasubramanian, P., Sundaran, K.: Obstacle avoidance and navigation planning of a wheeled mobile robot using amended artificial potential field method. Procedia Comput. Sci. 133, 998–1004. https://doi.org/10.1016/j.procs.2018.07.076 (2018)
36. Tahir, Z., Qureshi, A., Ayaz, Y., Nawaz, R.: Potentially guided bidirectionalized RRT* for fast optimal path planning in cluttered environments. Robot. Auton. Syst. 108, 13–27. https://doi.org/10.1016/j.robot.2018.06.013 (2018)
37. Uttendorf, S., Eilert, B., Overmeyer, L.: Combining a fuzzy inference system with an A* algorithm for the automated generation of roadmaps for Automated Guided Vehicles at - Automatisierungstechnik 65(3). https://doi.org/10.1515/auto-2016-0081 (2017)
38. Wang, D.J.: Indoor mobile-robot path planning based on an improved A*algorith. J. Tsinghua Univ. (Sci. Technol.) 52(8), 1085–1089. https://doi.org/10.1016/j.issn.1000-9787.2012.08-013 (2012)
39. Wang, H.M., Zhou, X.Z.: Improvement and realization of Bee-linear optimizing A* algorithm in shortest path problem. J. Eng. Graph. 30(6), 121–126 (2009). https://doi.org/10.1360/72009-1549
40. Wang, H.W., Ma, Y., Xie, Y., Guo, M.: Mobile robot optimal path planning based on smooth A* algorithm. J. Tongji Univ. (Natural Sci. Edn.) 38(11), 1647–1650+1655. https://doi.org/10.3969/j.issn.0253-374X.2010.11.016 (2010)
41. Wang, Z., Feng, X., Qin, H.D., Guo, H.M., Han, G.J.: An AUV-aided routing protocol based on dynamic gateway nodes for underwater wireless sensor networks. J. Internet Technol. 18(2), 333–343 (2017). https://doi.org/10.6138/JIT.2017.18.2.2016122
42. Wodzinski, M., Krzymienowska, A.: Sequential classification of palm gestures based on A* algorithm and MLP neural network for quadrocopter control. Metrol. Measur. Syst. 24(2), 265–276 (2017). https://doi.org/10.1515/mms-2017-0021
43. Xin, Y., Liang, H.W., Du, M.B., Mei, T., Wang, L.Z., Jiang, R.H.: An improved A* algorithm for searching infinite neighborhoods. Robot 36(5), 627–633 (2014). https://doi.org/10.13973/j.cnki.robot.2014.0627
44. Xu, S.Y.: On two distances sets in Euclidean space. J. China Inst. Metrol. 13(1), 16–18 (2002). https://doi.org/10.3969/j.issn.1004-1540.2002.01.002
45. Zhao, X.C., Luo, Q.S., Han, B.L.: Review of robot multi-sensor information fusion. Transd. Microsyst. Technol. 27(8), 1–4+11. https://doi.org/10.3969/j.issn.1000-9787.2008.08.001 (2008)
46. Zhu, H.Y., Yuan, Y.: Optimal path search based on improved A* algorithm. Comput. Technol. Dev. 28(4), 55–59 (2018). https://doi.org/10.3969/j.issn.1673-629X.2018.04.012

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.
JiaHe Li received the B.S. degrees in Measurement and control technology and instrument, Southwest Petroleum University, Chengdu, China, in 2018. He is currently pursuing the M.E. in measuring, and testing technology and instrument. His main research interests include path planning and multi-sensor data fusion.

Jonathon Chambers (S’83–M’90–SM’98–F’11) received the Ph.D. and D.Sc. degrees in signal processing from the Imperial College of Science, Technology and Medicine (Imperial College London), London, U.K., in 1990 and 2014, respectively. He is Emeritus Professor of Signal and Information Processing and Former Head of the School of Engineering at the University of Leicester. He is also an International Honorary Dean and Guest Professor at Harbin Engineering University, China. His research interests include adaptive signal processing and machine learning and their applications, and he has advised approaching 100 researchers through to Ph.D. graduation.

Dr. Chambers is a Fellow of the Royal Academy of Engineering, U.K., and the Institution of Electrical Engineers. He is serving on the organizing committees of ICASSP 2019 Brighton, U.K., and ICASSP 2022. He has served on the IEEE Signal Processing Theory and Methods Technical Committee for six years, the IEEE Signal Processing Society Awards Board for three years, and the Jack Kilby Medal Committee for three years.