The Journal of Supercomputing (2022) 78:12973–13003
https://doi.org/10.1007/s11227-022-04395-6

An improved path planning algorithm based on fuel consumption

Tianbo Liu[1] · **Jindong Zhang**[1,2,3]

Accepted: 23 February 2022 / Published online: 13 March 2022

Abstract
Most path planning algorithms choose path length as the evaluation criterion for algorithm performance. However, considering the driving and environmental costs, the car's fuel consumption is also critical. This paper proposes an improved A* algorithm based on fuel consumption. Because there are driving stages and idling stages in the driving process, the latter mainly occurs when the car meets the red lights. Accordingly, we make the same improvement to the A* algorithm; the fuel consumption of each part corresponds to the composition of the evaluation function of the A* algorithm. Finally, we use the abstract map and set different red light proportions to compare the algorithm's performance. The experimental results show that with the red light proportion increase, the improved A* algorithm can reduce the fuel consumption by up to 16.949% compared with the original A* algorithm.

Keywords A* algorithm · Path length · Idling fuel consumption · Red light proportion

1 Introduction

Path planning is a common problem in any road network [26, 30]. In recent years, the vehicle ownership has increased dramatically, and the proliferation of vehicle numbers has led to the increasing significance of path planning. Choosing a non-optimal path will not only increase travel costs and lead to congestion problems, which significantly impact people's daily life [16, 18]. To this end, how to resolve

✉ Jindong Zhang
zhangjindong_100@163.com

[1] College of Computer Science and Technology, Jilin University, Changchun 130012, China

[2] Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Jilin University, Changchun 130012, China

[3] State Key Laboratory of Automobile Simulation and Control, Jilin University, Changchun 130025, China

the path planning problem, choosing an optimal path has become a top priority now [12].

Path length has been considered a critical factor in most path planning algorithms about vehicles [13, 20]. The most common is the ant colony algorithm [33] and the A* algorithm [29]. Ant colony algorithm is one of the algorithms based on swarm intelligence [32]. The idea of ant colony algorithm is to simulate ant's behavior, which is often used in solving optimization problems [1, 24, 27]. A* algorithm is a typical heuristic search algorithm. The search in the state space evaluates the location of each search, gets the best location, and then searches from this location to the target location [8, 14, 17]. The advantage of the A* algorithm is that by introducing the heuristic function as the auxiliary decision-making information for moving to the target location, it is unnecessary to expand all the expansion locations when looking for the successor location, which reduces a lot of unnecessary searches and saves the time for path planning. In addition, the flexible form of the A* algorithm has good results in various application scenarios, the most common being pathfinding in game development and path planning in navigation. The ant colony algorithm does not possess these advantages.

Compared with how to efficiently complete the path planning, the practical significance of the selected path is equally important. Most path planning algorithms still choose the shortest path or the shortest time-consuming path, but this is often difficult to achieve in natural driving environments. Considering the complex traffic environment and the driving characteristics of the vehicle under different conditions, many factors should be taken into account when path planning, such as the fuel consumption of the vehicle [34], the idling time [31], the congestion level of the path [5].

We propose an improved A* algorithm based on the above reasons. The improved algorithm optimizes the A* algorithm based on retaining the heuristic characteristics of the original A* algorithm. At the same time, we define a new fuel consumption calculation method, which uses the fuel consumption and proportion of different cars. Obtain the mathematical expectation on fuel consumption, and combine it with the driving distance to calculate the driving fuel consumption; use the total idling time to calculate the idling fuel consumption. Furthermore, the path obtained by the improved A* algorithm can effectively reduce fuel consumption. In order to improve the authenticity and convenience of the simulation effect, the red light is introduced into the map to restore the natural traffic environment. The importance of this study is that it explores the relationship between path planning and fuel consumption. This not only gives practical significance to the selected algorithm but also conforms to the city's traffic concept, and the resulting paths are also primarily in line with the actual choice of the driver.

The rest of the paper is structured as follows. The related work is introduced in Sect. 2. The improved ant colony algorithm and A* algorithm are introduced in Sect. 3. The next Sect. 4 is the experimental process, we use the abstract maps and randomly generate red lights to test the performance of the improved A* algorithm. In the last section of the article, we conclude and forward the future research work.

2 Related works

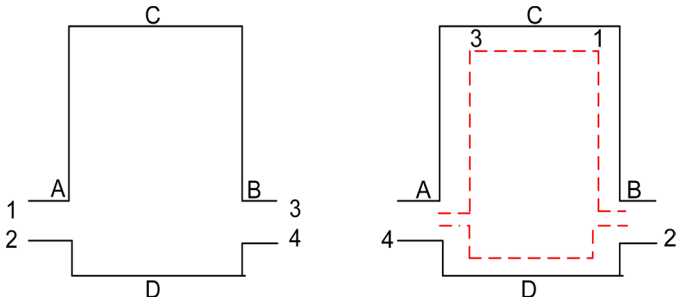
A* algorithm is favored by researchers for its unique search advantages and wide range of application scenarios. However, the disadvantages of the A* algorithm should not be ignored, as the proportion of the heuristic function [6] and the complexity of the environment [3] can affect the algorithm's performance. To overcome the performance limitations of the A* algorithm, many researchers have proposed improvement measures. Alazzam et al. [2] change how the A* algorithm searches, where the algorithm searches parallelized. Zhang and Xu [36] put a new hybrid algorithm, the advantages of each algorithm are fully utilized. Chen et al. [4] combined the A* algorithm with the ant colony algorithm, which greatly improved algorithmic efficiency. Tang and Wu [28] applied the hierarchical search method to the A* algorithm plan paths efficiently while avoiding obstacles. While improving the execution efficiency of the A* algorithm, more researchers are committed to broadening its application fields. Mohammadzadeh et al. [22] uses feature selection for spam detection and a multi-agent system to optimize the search process of the A* algorithm, which eliminates spam more effectively. Min and Xiong [19] present an environment description method combining global navigation based on the improved A* algorithm for autonomous driving vehicles in unstructured environment. Duan and Fan [7] propose a hybrid algorithm of adaptive large-scale neighborhood search to predict the trajectory of marine debris and determine its location, which reduces the cost of positioning and dramatically reduces the search time. Ouyang et al. [21] use the A* algorithm for underwater gravity matching navigation, and formed a grid reference map for adapting the gravity field, effectively overcoming the accumulation of inertial navigation system errors caused by long-term underwater navigation.

3 Proposed algorithms

3.1 Algorithm A-the improved ant colony algorithm

Italian scholars in the 1990s first proposed the ant colony algorithm. The original purpose of this algorithm is to solve the traveling salesman problem [10, 15].

The main idea of the ant colony algorithm is to simulate the communication and cooperation between ants to complete the transportation of food to the nest. In looking for food, ants will leave pheromones on the path, and ants will travel along the path with a high concentration of pheromones. The dynamic changes of the foraging pheromone of the ants who come back early can determine the time and direction for the foraging ants to choose the optimal foraging path. In this way, the ants behind are usually able to automatically determine the next foraging and travel path according to the number and concentration of ant pheromones on each path. Under such a positive feedback mechanism, an optimal path is finally formed, all ants will select this path to find food. The pathfinding principle of the ant colony algorithm is shown in Fig.1.



(a) Initial pathfinding state **(b)** Ant colony algorithm pathfinding ends

Fig. 1 The pathfinding principle of ant colony algorithm

As shown in Fig.1, when ant_2 and ant_4 reach their respective destination, ant_1 and ant_3 have not yet reached the destination. Due to the positive feedback mechanism, all ants will eventually select the path between ant_2 and ant_4. The optimal path is also determined. In using the ant colony algorithm for path selection, under the role of positive feedback mechanism, the search speed is greatly accelerated and has an excellent ability to solve problems. However, the shortcomings of ant colony algorithm cannot be ignored. When faced with a complex environment, such as an urban traffic environment, the operation of the whole algorithm will become particularly slow. There is no way to obtain the optimal solution quickly, which dramatically limits its scope of application [23]. Another disadvantage of the ant colony algorithm is that if there is a non-optimal solution with high pheromone concentration, other ant colonies will not continue to look for other solutions so that they cannot traverse all paths, only obtain the optimal local solution, and thus fall into the optimal local solution.

In order to effectively solve the limitations of the ant colony algorithm, we have made the following improvements.

In the t iteration, the state transition probability of ant_k selecting the next point j by the point i is:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\partial}\left[\eta_{ij}(t)\right]^{\beta}}{\sum_{j\in \mathrm{allow}_k}\left[\tau_{ij}(t)\right]^{\partial}\left[\eta_{ij}(t)\right]^{\beta}}, j \in \mathrm{allow}_k \end{cases} \tag{1}$$

Where allow_k represents the set of all passable path points in the next step, ∂ is the information heuristic factor, β is the expected heuristic factor, τ_{ij} is the pheromone concentration of the path, η_{ij} is the heuristic function, which is usually negatively correlated with the distance between the two points. However, in the iterative process of the algorithm, due to the increasing number of iterations, it will cause uneven distribution of pheromone concentration in the global search, which in turn reflects the tendency of pheromone distribution, so this influence factor with the increasing number of iterations should be taken into account in the heuristic function. That is,

$$\eta_{ij}(t) = \frac{1}{d_{ij}} * \frac{R_{\max} - R_t}{R_{\max}} \tag{2}$$

d_{ij} represents the distance between point i and point j, $R_{\max}$ represents the maximum number of iterations allowed in the search process, R_t represents the current number of iterations. This practice considers the iterative boundary problem of the algorithm to some extent, makes the heuristic function more practical.

Because the algorithm is only affected by the path length and pheromone, and the positive feedback makes the pheromone accumulate continuously, it is conceivable that the pheromone value on the path will become huge when it reaches a particular time. It will weaken or even eliminate the role of the heuristic function. Therefore, every time the ant completes an iterative process, it must be updated according to the pheromone update formula. The update rules for the original pheromone are

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t, t+1) \tag{3}$$

$$\Delta\tau_{ij}(t, t+1) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t, t+1) \tag{4}$$

Where ρ represents the pheromone volatilization coefficient, m represents the total number of ants, $\Delta\tau_{\mathrm{ij}}(\mathrm{t}, \mathrm{t}+1)$ represents the increment of pheromones in $cycle(i, j)$.

In the original ant colony algorithm, ρ is a constant value. If the value of ρ is significant, although it can accelerate the convergence speed, it is easy to fall into the optimal local solution; if the value of ρ is small, the probability that the already explored points are repeatedly explored becomes larger, which reduces the convergence speed of the algorithm, so we improve the strategy of pheromone volatilization coefficient as follows:

$$\rho(t) = \begin{cases} 0.9\rho(t-1) & 0.9\rho(t-1) > \rho_{\min} \\ \rho_{\min} & \text{otherwise} \end{cases} \tag{5}$$

$$\rho_{\min} = 0.1 \tag{6}$$

We set the minimum threshold $\rho_{\min}$ for the pheromone volatilization coefficient and adjust its volatilization dynamically by piecewise function. Accordingly, we change the pheromone update rules as follows.

$$\tau_{ij}(t+1) = (1-\rho(t)) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t, t+1) + l_{\mathrm{iter}} \tag{7}$$

$$l_{\mathrm{iter}} = \frac{l_{\mathrm{history}} - l_{\mathrm{current}}}{l_{\mathrm{current}}} \tag{8}$$

l_{history} represents the optimal historical path, l_{current} represents the current optimal path length. After ants finish each iteration, if $l_{\mathrm{history}} > l_{\mathrm{current}}$, it means that the path

of this iteration is shorter, and should strengthen the concentration of pheromone of this iteration and preserve the optimal path of this iteration; on the contrary, if $l_{\text{history}} < l_{\text{current}}$, it means that the path found in this iteration is not the shortest, and should reduce the concentration of pheromone of this path. In this way, we introduce the path length factor into the pheromone update rules, making the pheromone update rules more in line with the search process.

According to the different methods of solving $\tau_{ij}(t)$, there are usually three solving models: ant-cycle model, ant-quantity model and ant-density model. Because of the rigor of the ant-cycle model, it is often used to solve the problem in practice.

In the ant-cycle model, the pheromone intensity released by ants through a specific cycle is fixed, that is,

$$\Delta\tau_{ij}^{k}(t,t+1) = \begin{cases} \frac{Q}{L_k}, \text{if ant}_k \text{ pass the path in cycle } (i,j) \\ 0, \text{ otherwise} \end{cases} \tag{9}$$

Where Q represents the pheromone intensity, L_k represents the total length of the path taken by ant_k.

The process of the improved ant colony algorithm can be expressed as follows.

Among them, R_t represents the current number of iterations, $R_{\max}$ represents the maximum number of iterations allowed, m represents the total number of ants, $\text{point}_{\max}$ represents the total number of coordinate points in the map, and L is the final path obtained.

Algorithm 1 The improved ant colony algorithm

```
Parameters initialization
R_t = 1
while R_t < R_max do
    for ant = 1 to m do
        for point = 1 to point_max do
            Calculating the heuristic function according to formula (2)
            Selecting the next point Next_v according to formula (1)
            cur ← Next_v
            L = L + L_{cur←Next_v}
        end for
    end for
    Calculate the path length of each ant
    Update the pheromone according to formula (9)
    R_t = R_t + 1
end while
```

After the algorithm finds the optimal path, we calculate the length of the optimal path, denoted as pre_{dist}, as an essential parameter for the subsequent algorithm.

3.2 Algorithm B-the improved A* algorithm based on fuel comparison

A* algorithm is a heuristic search algorithm. It starts from the starting, searches for the locations adjacent to the starting, and then uses the evaluate function to judge and select the best-extended location as the starting of the following search, and continues to expand until the target is found. Because every step in the search process selects the location with the lowest cost, the resulting path must be the optimal path.

The evaluation function is the core part of the heuristic search. An appropriate evaluation function will significantly speed up the acquisition of the optimal path [25].

The evaluation function of A* algorithm is:

$$F = G(n) + H(n) \tag{10}$$

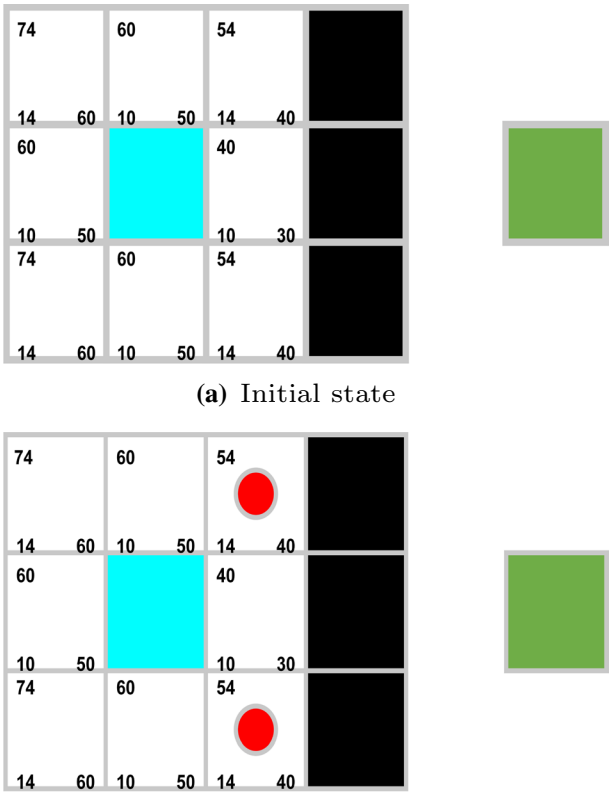
Where n refers to the current location, $G(n)$ refers to the actual cost from the starting to location n, and $H(n)$ is the estimated cost from location n to the target.

In the pathfinding process of the A* algorithm, it is generally necessary to construct two tables: OPEN table and CLOSED table. The OPEN table stores the locations that have been estimated, and the role of the CLOSED table is to store the extended locations that do not need to be concerned.

We use a more specific picture to illustrate the pathfinding principle of the A* algorithm, as shown in Fig.2.

The blue rectangle in Fig.2 represents the starting point, the green rectangle represents the endpoint, the black area represents the impassable area, and the rectangle marked with a red circle inside represents the following moveable location. Each rectangle around the starting point represents a selectable location for the next step. The default horizontal and vertical distance between two adjacent rectangles is 10, and the diagonal distance is 14. The number in the lower-left corner of each rectangle represents the distance between itself and the starting point, which is calculated as Euclidean distance; the number in the lower right corner represents the distance between itself and the endpoint, which is calculated as Manhattan distance; the number in the upper left corner is the sum of the two, which is the total cost of the A* algorithm. The A* algorithm will choose the location with the lowest total cost. After calculation, it is found that the total cost of the rectangle in the upper right corner and the lower right corner of the starting point is equal, so they can be used as the moving location that can be selected for the next step of the starting point.

It is undeniable that the original A* algorithm has some disadvantages. The guidance of the heuristic function removes the locations containing unknown information to some extent, which may cause path deadlock. At the data structure level, the maintenance of the OPEN table is also a critical issue that affects the algorithm's performance. Therefore, we have to improve the algorithm to meet the needs of the problem.



(a) Initial state

(b) The next location to be selected

Fig. 2 The pathfinding principle of the A* algorithm

3.2.1 Optimization of OPEN table storage structure

When expanding the CLOSED table during the searching process, we first need to find a corresponding point with the lowest F value in the OPEN table as we use abstract maps. The storage structure of the OPEN table typically selects a linked list or an array. The time spent during various operations will grow as the storage capacity increases. This paper selects another structure, the minimum heap.

The binary tree implements the minimum heap. The value of each child point in the tree is greater than its parents, so this will make the point with the lowest value always at the top of the tree. The process has a time complexity of $O(\log n)$, compared to the original complexity of $O(n)$, which greatly reduces the time spent of the algorithm. There is another advantage that cannot be neglected. It is more convenient to insert or delete the point. It can be put in a suitable location relatively few times when inserting the point. When deleting the point, the top has the lowest value, and the last point is used as a supplement.

3.2.2 The use of hash table

In the process of A* search, the next step of searching requires the selection and judgment of adjacent points, and it is necessary to determine whether these points exist in the OPEN table, which is usually achieved by traversal of the OPEN table. When the amount of points is too large, this operation can severely limit the algorithm's performance. For this reason, this paper proposes to create a hash table to store the mark value of the point in the OPEN table. It only needs to visit the elements corresponding to the point coordinates to judge the belonging of the members in the OPEN table.

The basic principle of the hash table is key-value mapping. The key value key is converted into a valid number Num through a specially designed calculation [11], called the hash function, and then the value is stored in the storage space marked by the number. The general hash function can be expressed as follows:

$$\mathrm{Num} = f(\mathrm{key}) \tag{11}$$

The principle of the hash function is shown in Fig.3.

In the application of this paper, when the coordinates of a point is $(temp\_x, temp\_y)$, the hash function can be represented as:

$$f(\mathrm{key}) = \left(\mathrm{temp}\_x + \mathrm{temp}\_y\right)\%M \tag{12}$$

Formula (12) is the data retrieval formula to find the planning point from the hash table according to the key value, where M is usually a prime with the same length as the hash table.

Considering the secondary conflict, the definition of the secondary hash function should also be simple. There is:

$$f(\mathrm{key}) = \left(\mathrm{temp}\_x + \mathrm{temp}\_y + p\right)\%M \tag{13}$$

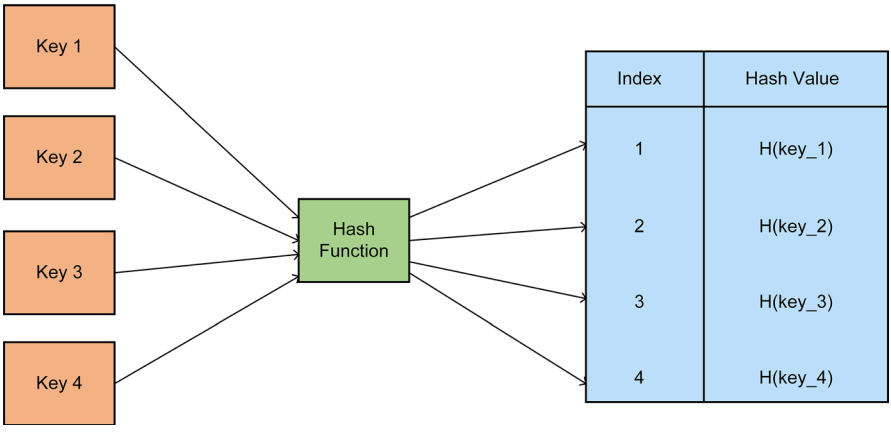


Fig. 3 The principle of hash function

Where p represents the offset, thus, we can resolve the secondary conflict issues. The hash table dramatically reduces the resource waste caused by the algorithm.

3.2.3 The selection of distance calculation method

There are usually three ways to calculate the distance, Manhattan distance, Euclidean distance, and Diagonal distance, shown in Fig.4.

3.2.3.1 Manhattandistance Weassumethatthecurrentpointis$\left(\text{current_x}, \text{current_y}\right)$, and the target point is $\left(\text{goal\_x}, \text{goal\_y}\right)$, then the Manhattan distance $\text{dis}_{\text{Manhattan}}$ is:

$$\text{dis}_{\text{Manhattan}} = \left(\text{current\_}x - \text{goal\_}x\right) + \left(\text{current\_}y - \text{goal\_}y\right) \tag{14}$$

Because Manhattan distance calculates the sum of the absolute difference between the horizontal and vertical coordinates, the algorithm will search in a broader range when performing the search, resulting in the results may be greater than the actual value of the optimal path.

3.2.3.2 Euclidean distance The specific formula of Euclidean distance $dis_{Euclidean}$ is as follows:

$$\text{dis}_{\text{Euclidean}} = \sqrt{\left(\text{current\_}x - \text{goal\_}x\right)^2 + \left(\text{current\_}y - \text{goal\_}y\right)^2} \tag{15}$$

The Euclidean distance is shorter than the Manhattan distance, the probability of obtaining the optimal path is high, but the calculation involves square root operations, making the calculation very cumbersome.

3.2.3.3 Diagonal distance As seen above, we can see that although the finding efficiency of Manhattan distance is high, the path quality is low; the Euclidean distance is just the opposite. So we choose the diagonal distance and achieve a balance of both. The diagonal distance $\text{dis}_{\text{Diagonal}}$ can be expressed as:

$$l(n) = \min\left(\text{current\_}x - \text{goal\_}x, \text{current\_}y - \text{goal\_}y\right) \tag{16}$$

$$\text{dis}_{\text{Diagonal}} = \sqrt{2} \times l(n) + \left(\text{dis}_{\text{manhattan}} - 2 \times l(n)\right) \tag{17}$$

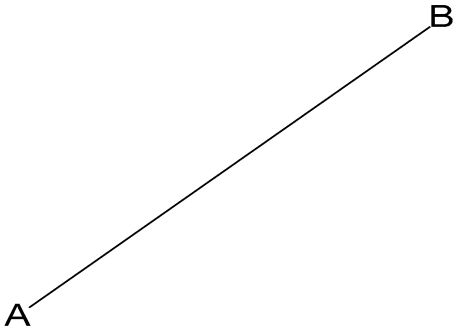
Based on the highlights of diagonal distance, this paper chooses to calculate the distance. To be consistent, the ant colony algorithm in the previous section also chooses the diagonal distance as the distance calculation.

3.2.4 Calculation of driving fuel consumption

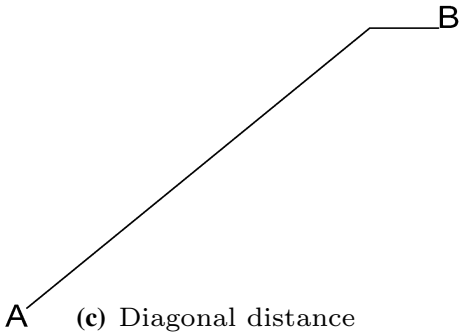
Figure 5 shows the 100 kilometers of driving fuel consumption and its proportion of vehicles with different emissions. The data is collected from China's vehicle market different products focus on proportion displacement in June and July 2012 [35].

Fig. 4 Three common distances

(a) Manhattan distance

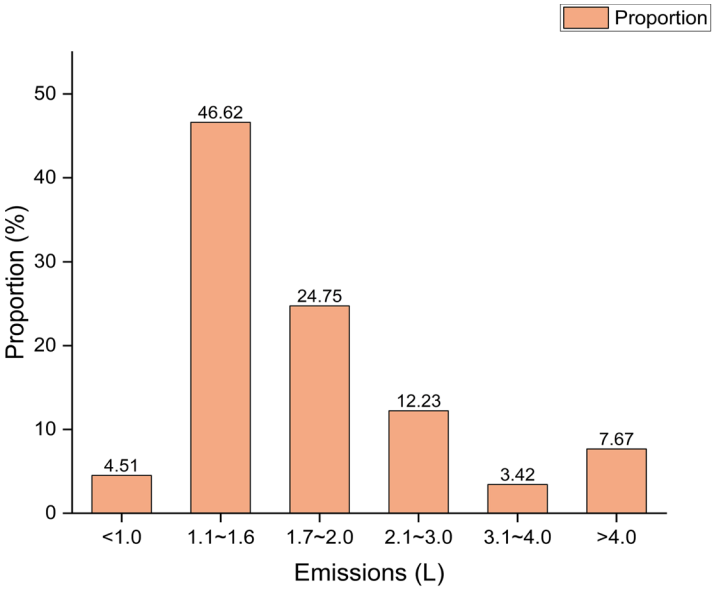


(b) Euclidean distance

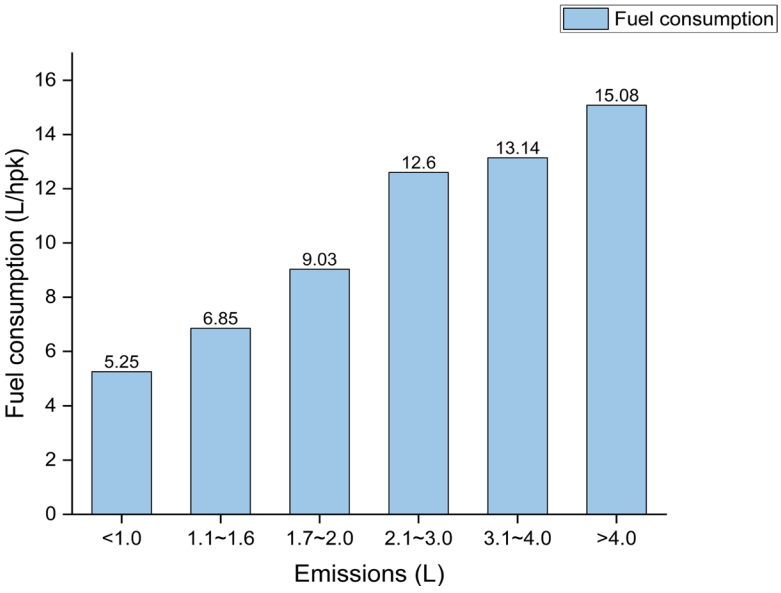


(c) Diagonal distance

As we all know, driving fuel consumption is related to driving distance. Then we can replace the driving fuel consumption according to the overall average level of driving fuel consumption of vehicles with different emissions, that is, mathematical expectations. We use $E_{\text{consumption}}$ to express expectations, $\text{Fuel}_{\text{consumption}}$ for the fuel consumption, and $\text{Pro}_{\text{vehicle}}$ for the proportion of vehicles, then there are:



(a) Different vehicle emissions and corresponding proportion



(b) Different vehicle emissions and corresponding driving fuel consumption

Fig. 5 The 100 kilometers driving fuel consumption and its proportion of vehicles with different emissions

$$E_{\text{consumption}} = \sum \text{Pro}_{\text{vehicle}} * \text{Fuel}_{\text{consumption}} \tag{18}$$

When the driving distance of the vehicle is $dis_{driving}$, then the driving fuel consumption $l_{driving}$ can be expressed as:

$$l_{\text{driving}} = E_{\text{consumption}} * \text{dis}_{\text{driving}} \tag{19}$$

3.2.5 Calculation of idling fuel consumption

The idling fuel consumption is a positive correlation to the idling time [9]. We assume that the total idling time of the vehicle in the driving process is t_{idling}, the idling fuel consumption of the vehicle is $\text{Fuel}_{\text{idling}}$, then the idling fuel consumption l_{idling} during driving can be expressed as:

$$l_{\text{idling}} = \text{Fuel}_{\text{idling}} * t_{\text{idling}} \tag{20}$$

3.2.6 Calculation of the cost of the improved A* algorithm

For the improved A* algorithm based on fuel consumption, assuming that the diagonal distance from the starting to the current location n is S_1, then the actual cost $G(n)$ is:

$$G(n) = E_{\text{consumption}} * S_1 \tag{21}$$

In the same way, when the diagonal distance from the current location n to the endpoint is S_2, the estimated cost $H(n)$ is:

$$H(n) = E_{\text{consumption}} * S_2 * f\left(S_2\right) \tag{22}$$

$$f\left(S_2\right) = \log_2\left(1 + e^{\frac{S_2}{\text{pre}_{\text{dist}}} - 1}\right) \tag{23}$$

Among them, S_2 becomes smaller gradually, and the initial value of S_2 is smaller than pre_{dist}. Because the proportion of $H(n)$ in the initial stage of the algorithm is much more significant than $G(n)$, the algorithm will gradually degenerate to a breadth-first search algorithm. By setting such parameters, we can appropriately reduce the proportion of $H(n)$ in the initial stage of the algorithm so that the proportion of $H(n)$ and $G(n)$ will not differ too much to ensure the characteristics of the algorithm itself.

The total cost F at this time is:

$$F = G(n) + H(n) + l_{\text{cur}-\text{idling}} \tag{24}$$

$l_{\text{cur}-\text{idling}}$ represents the idling fuel consumption so far, calculated in the same way as above.

The process of the improved A* algorithm based on fuel consumption is shown as follows.

Algorithm 2 The improved A* algorithm based on fuel cunsumption

```
Put the starting point into the OPEN table
CLOSED table initialization
while starting!=goal && OPEN table!=NULL do
    Calculate the actual cost G(n) according to formula (21)
    Calculate the estimated cost H(n) according to formula (22)
    Calculate idling fuel consumption l_{cur-idling} according to formula (20)
    Calculate the total cost F according to formula (24)
    Move the location n with the smallest F value into the CLOSED table
    for each location adjacent to n do
        if the adjacent location of n is accessible then
            Put the adjacent location into the OPEN table
            Set the parent of the adjacent location as the current location
            Calculate G(n), H(n) and F
        else
            If the cost of adjacent location is smaller, update F and G(n)
        end if
    end for
end while
```

4 Experimental simulations

In order to confirm the effectiveness of the proposed algorithms, we test the algorithms through simulation experiments. The hardware environment and software environment adopted in the experiment are as follows: (1)Operating System: Ubuntu 18.04; (2)Processor: AMD Ryzen 7 4800H with Radeon Graphics; (3)Memory: 16GB; (4)GPU: NVIDIA 2060M. First of all, we analyze the improved ant colony algorithm; after we apply the abstract map to simulate the traffic environment, we randomly add red lights to the map to test the performance of the improved A* algorithm. Finally, we provide a comprehensive evaluation of the experimental results (Fig. 6)

The specific flow chart for solving the problem is as follows:

4.1 Verify the improved ant colony algorithm

In order to achieve the purpose of comparison, we use different maps to simulate. We set the starting point and endpoint unchanged and changed the distribution of obstacles. We conducted a total of five sets of comparative experiments (Table 1).

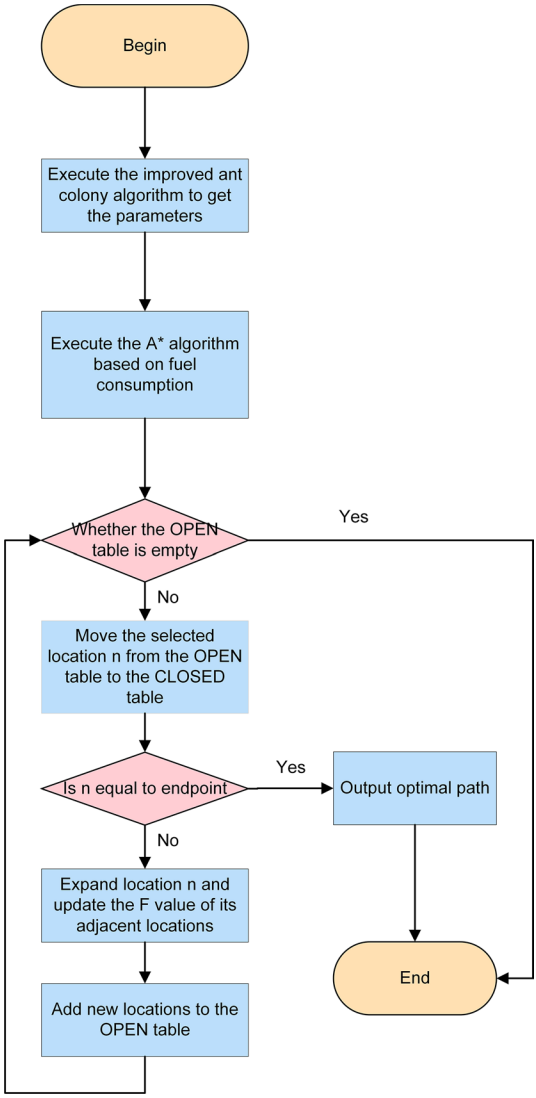


Fig. 6 The flow chart of the path planning problem

Table 1 Parameter settings of the ant colony algorithm

Original expression	Initial value
∂	1
β	5
$R_{\max}$	300
R_t	1
ρ	0.9
$\rho_{\min}$	0.1
Q	1
m	50

Among them, the initialization of each parameter in the ant colony algorithm is as follows:

After the parameter initialization is completed, we will compare the original ant colony algorithm and the improved ant colony algorithm from the following two aspects.

4.1.1 Comparison of path length and smoothness

The gray area in the figure represents the obstacles, and the white area represents the passable area. Among them, the red trajectory represents the path of the original ant colony algorithm, the black trajectory represents the path of the improved ant colony algorithm, and the blue trajectory represents the overlapping part of the two algorithm paths. The pathfinding results of the first group of comparative experiments are shown in Fig.7 (Figs. 8, 9, 10 and 11).

Similarly, the pathfinding results of the other four groups of comparative experiments are:

After five groups of comparative experiments, it can be found that the improved ant colony algorithm has a significant improvement in path length and path smoothness, which not only avoids many unnecessary path deviations but also enhances the overall path quality. From this point of view, the improved ant colony algorithm has a more practical significance for the next step of integration with the A* algorithm.

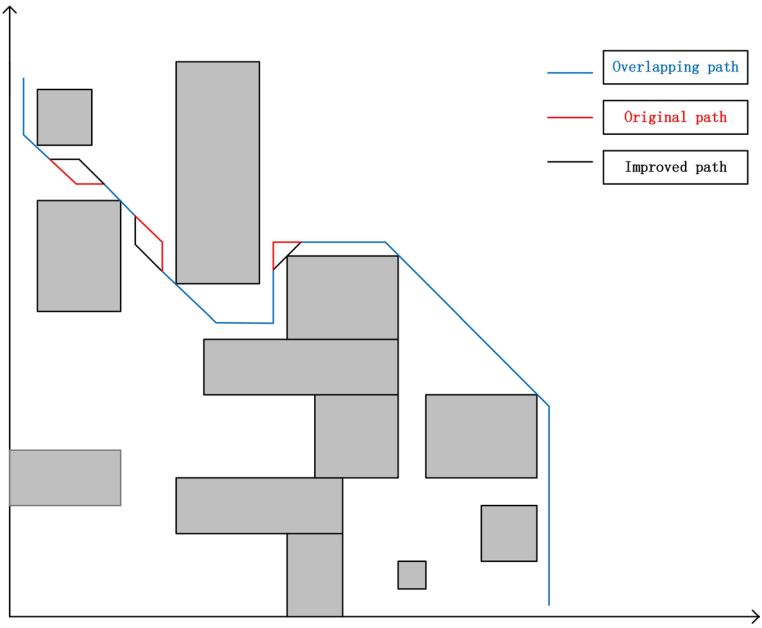


Fig. 7 The pathfinding results of the first map distribution

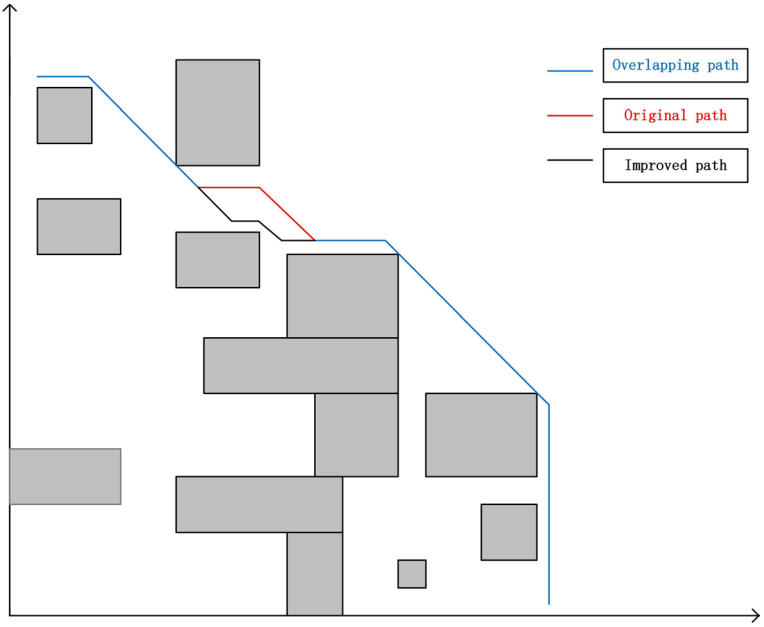


Fig. 8 The pathfinding results of the second map distribution

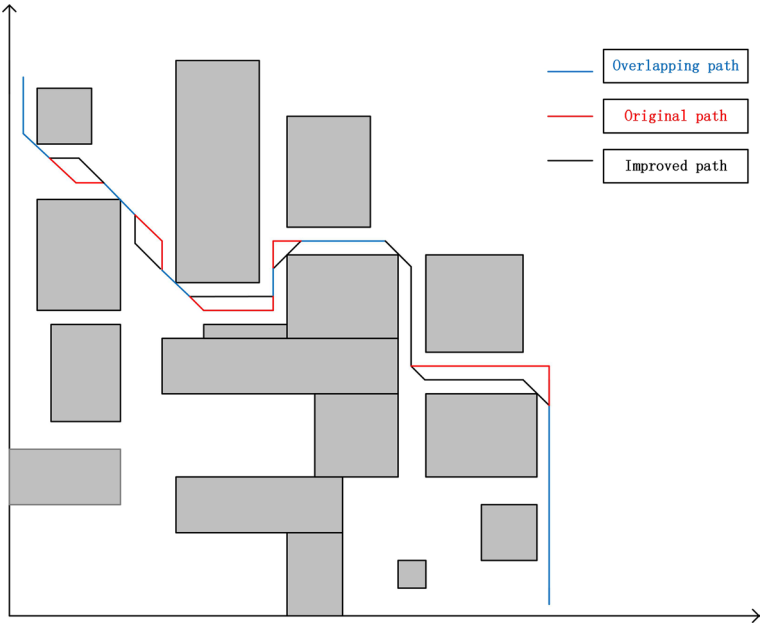


Fig. 9 The pathfinding results of the third map distribution

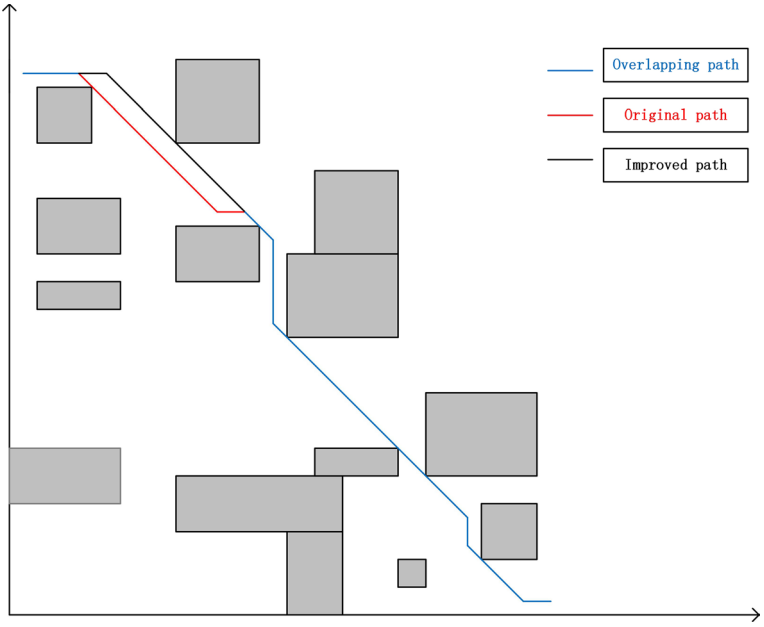


Fig. 10 The pathfinding results of the fourth map distribution

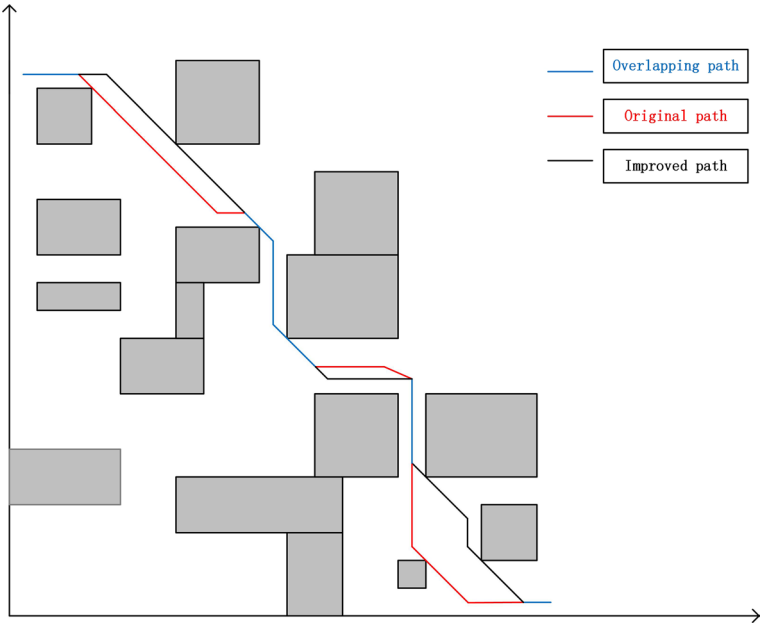


Fig. 11 The pathfinding results of the fifth map distribution

Table 2 Time performance comparison of the ant colony algorithm

	Original algorithm time(s)[a]	Improved algorithm time(s)[b]	Improvement[c] (%)
Experiment 1	12.019	11.564	3.786
Experiment 2	12.475	11.642	6.677
Experiment 3	11.634	10.651	8.449
Experiment 4	11.479	10.259	10.628
Experiment 5	11.281	9.892	12.313

[a]The total time of the original ant colony algorithm.

[b]The total time of the improved ant colony algorithm.

[c]The improvement in total time.

4.1.2 Comparison of pathfinding time

We compare the algorithm time of the above five groups of experiments (Table 2).

From the above results, we can see that the improved ant colony algorithm has an apparent improvement in total time, which proves that the improved ant colony algorithm has a better performance.

4.2 Verify the improved A* algorithm

In the experiments, we randomly generate red lights to test the performance of the improved A* algorithm.

For calculation purposes, the horizontal and vertical moving cost for each step in the map is equivalent to 100m in practice. To simplify the problem, we assume that the idling time at each red light is the same, equating the idling fuel consumption at each red light as the mathematical expectation of driving fuel consumption multiplied by the moving cost for each step. Namely, the idling fuel consumption at each red light is equivalent to the driving fuel consumption at each step. This assumption minimizes the path deviation caused by excessive idling fuel consumption setting at each red light. If the setting is too big, the red lights will often be avoided when selecting the following location, thus making the introduction of red lights meaningless. If the setting is too small, it does not play a reference role for path selection at all, which will cause more severe path deviation, resulting in the path is not the optimal path. In addition, to ensure the quality of pathfinding and avoid many complex calculations, the distance measurement in the A* algorithm also adopts the diagonal distance.

4.2.1 Comparison of time performance

In order to verify the necessity of each improvement measure in the A* algorithm, we conduct the following comparative experiments. First, for the following

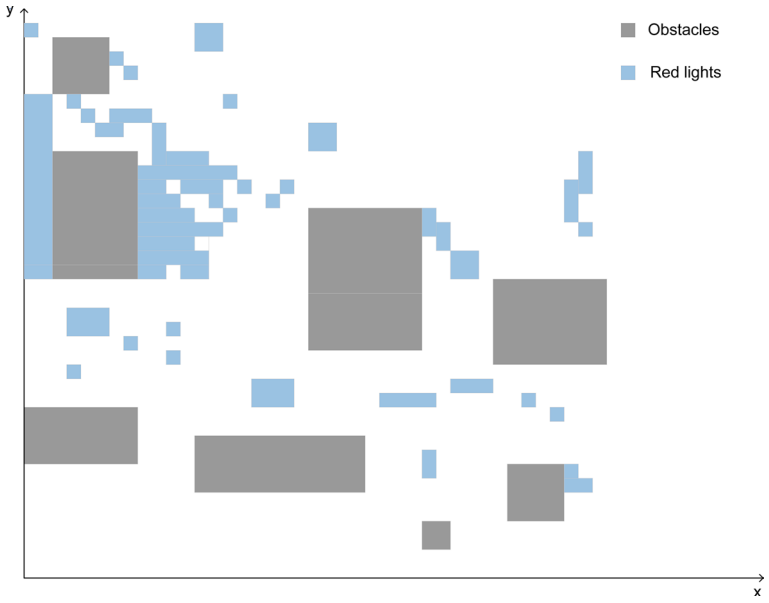


Fig. 12 The first map distribution

Table 3 Time performance comparison of the first map distribution

	Minimum heap time(s)[d]	Hash function time(s)[e]	Improved time(s)[f]
(1,1)->(37,39)	0.37	0.41	0.28
(1,1)->(24,39)	0.32	0.38	0.24
(1,3)->(26,39)	0.34	0.30	0.25

[d]Time for A* algorithm without minimum heap.

[e]Time for A* algorithm without hash function.

[f]Time for A* algorithm with minimum heap and hash function.

maps, we set three sets of points to explore the time performance of the algorithm without minimum heap, without hash function, and improved algorithm (Table 3).

When the map distribution is as shown in Fig.12, the corresponding pathfinding time in different situations is:

We change the map distribution as shown in Figs.13 and 14, and continue to explore the time performance of the improved algorithm.

The results corresponding to Figs.13 and 14 are as follows (Tables 4 and 5).

It can be seen from the above comparison that the time performance of the improved algorithm is better than that without minimum heap or hash function. The above improvement measures are indispensable.

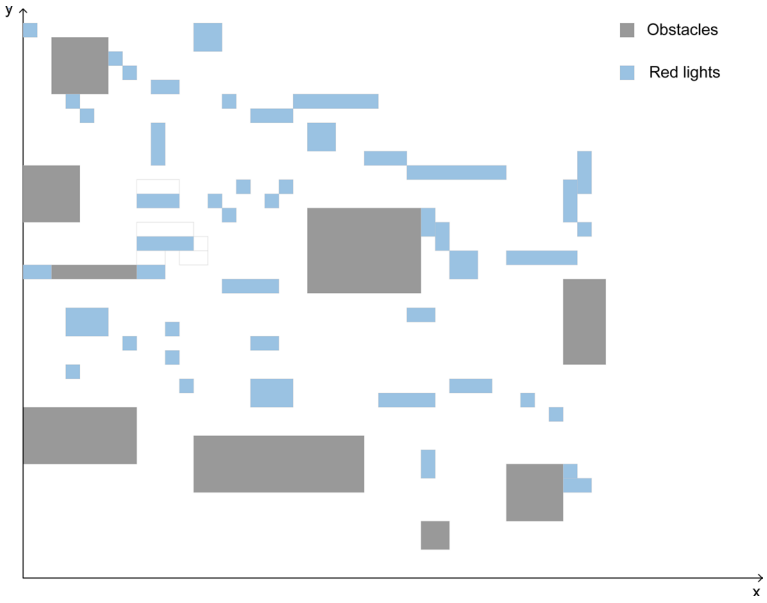


Fig. 13 The second map distribution

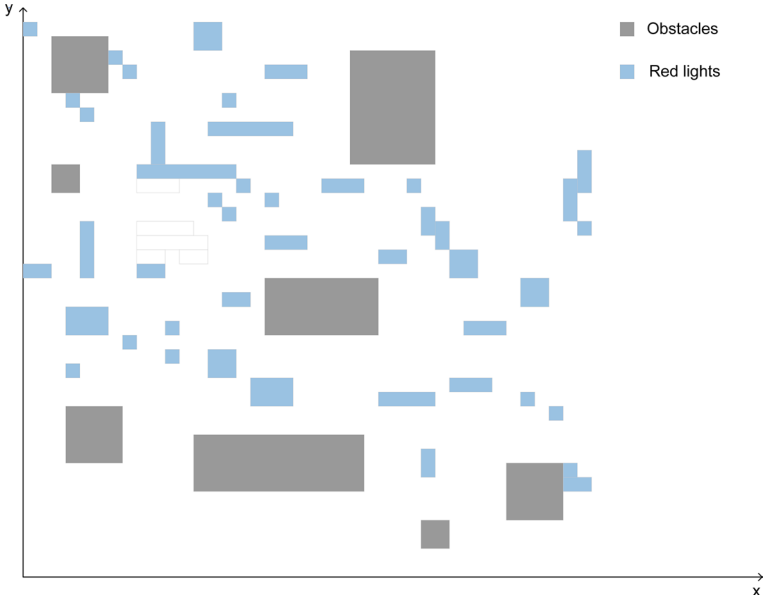


Fig. 14 The third map distribution

Table 4 Time performance comparison of the second map distribution

	Minimum heap time(s)[d]	Hash function time(s)[e]	Improved time(s)[f]
(1,1)->(37,39)	0.22	0.26	0.14
(1,1)->(24,39)	0.29	0.25	0.18
(1,3)->(26,39)	0.24	0.21	0.17

[d]Time for A* algorithm without minimum heap.

[e]Time for A* algorithm without hash function.

[f]Time for A* algorithm with minimum heap and hash function.

Table 5 Time performance comparison of the third map distribution

	Minimum heap time(s)[d]	Hash function time(s)[e]	Improved time(s)[f]
(1,1)->(37,39)	0.29	0.31	0.21
(1,1)->(24,39)	0.37	0.43	0.26
(1,3)->(26,39)	0.32	0.35	0.23

[d]Time for A* algorithm without minimum heap.

[e]Time for A* algorithm without hash function.

[f]Time for A* algorithm with minimum heap and hash function.

4.2.2 Comparison of fuel consumption

After verifying the necessity of each improvement measure, we compare the improved A* algorithm with the original A* algorithm and the Dijkstra algorithm to explore the difference in fuel consumption.

To this end, we set three different red light proportions and points. In order to visually compare the differences in the paths, we put the paths of the improved A* algorithm and the paths of the other two algorithms on a figure and marked them with different colors. Among them, the green trajectory represents the path of the original A* algorithm, the orange trajectory represents the path of the improved A* algorithm, the purple trajectory represents the path of the Dijkstra algorithm, and the red trajectory represents the overlapping part of the paths of different algorithms.

When the red light proportion is ρ_1, the pathfinding results of the first set of points are shown in Fig.15.

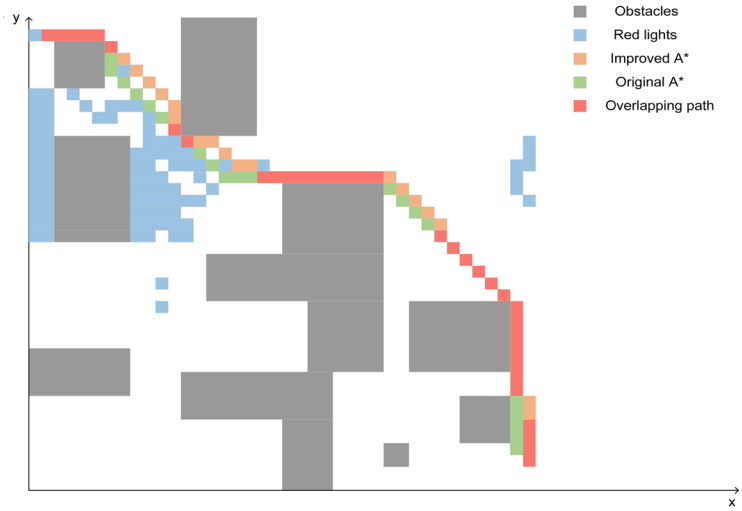
When the red light proportion is ρ_1, the pathfinding results of the second set of points are shown in Fig.16.

When the red light proportion is ρ_1, the pathfinding results of the third set of points are shown in Fig.17.

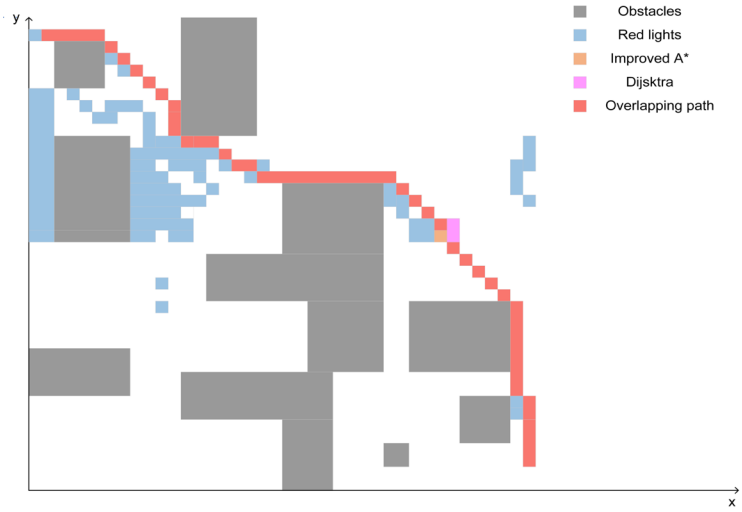
After getting the path, we count the number of red lights in the different paths above (Tables 6, 7, 8 and 9). There are:

The total fuel consumption can be calculated from formula (24), that is:

We continue to increase the proportion of red lights, recorded as ρ_2 and ρ_3, and the corresponding fuel consumption is:



(a) The pathfinding results of the improved A* algorithm and original A* algorithm



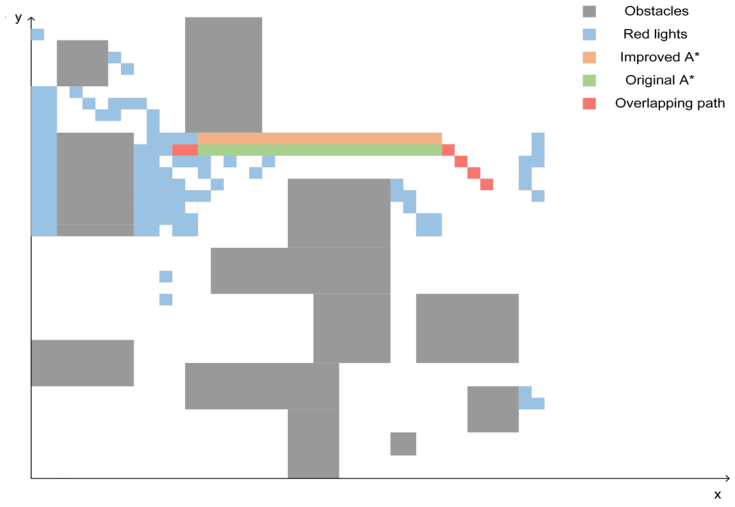
(b) The pathfinding results of the improved A* algorithm and Dijkstra algorithm

Fig. 15 The pathfinding results of the first set of points

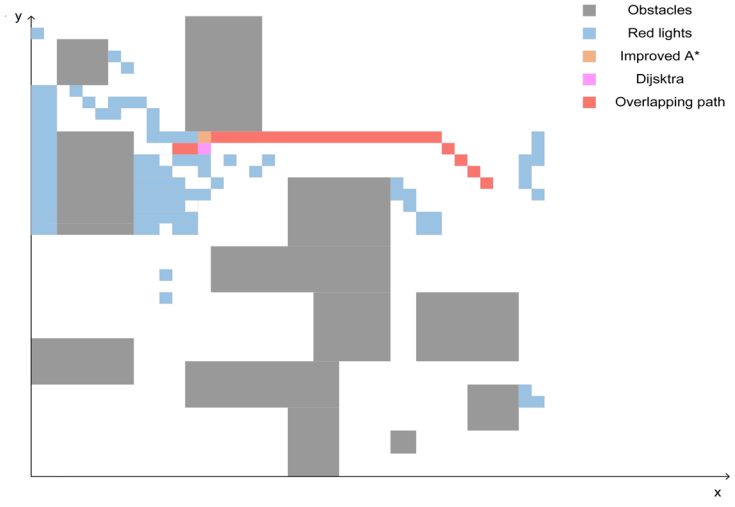
4.3 Overall evaluation

We comprehensively evaluate the experimental results, as shown in Figs.18 and 19.

Based on the above comparison, it can be seen that compared with the original A* algorithm and Dijkstra algorithm, the improved A* algorithm based on fuel consumption proposed by us significantly reduces the total fuel consumption. At the same time, the complexity of the abstract map restores the actual traffic environment to a certain extent, so the selected path is convenient to be applied to the actual environment.



(a) The pathfinding results of the improved A* algorithm and original A* algorithm

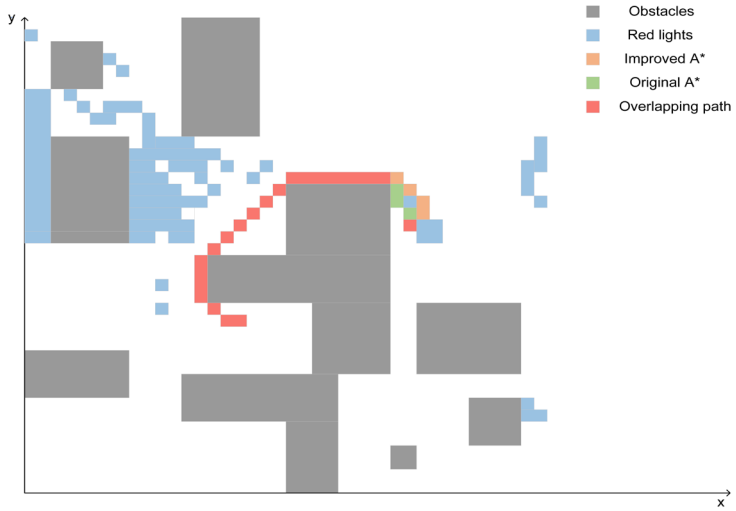


(b) The pathfinding results of the improved A* algorithm and Dijkstra algorithm

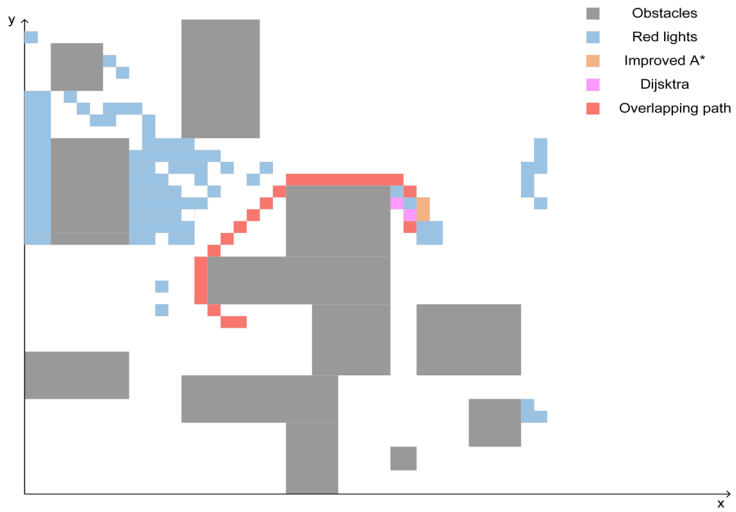
Fig. 16 The pathfinding results of the second set of points

5 Conclusion and future works

In this paper, an improved A* algorithm based on fuel consumption is proposed, combining the characteristics of the A* algorithm with the calculation method of fuel consumption to find the corresponding optimal path. In order to facilitate the simulation, we simplify the abstract map and introduce the red light to restore the natural traffic environment. Statistics and analysis show that it has high feasibility and good performance. As the proportion of red lights



(a) The pathfinding results of the improved A* algorithm and original A* algorithm



(b) The pathfinding results of the improved A* algorithm and Dijkstra algorithm

Fig. 17 The pathfinding results of the third set of points

increases, when the abstract map is getting closer and closer to the actual traffic environment, the improved A* algorithm can reduce fuel consumption by up to 16.949% compared with the original A* algorithm. However, several limitations of our study need to be addressed in future investigations. The restoration of the actual traffic environment is not accurate enough, and factors such as traffic congestion need to be considered. The improved A* algorithm also has a certain complexity and needs to be further optimized. Future research should consider

Table 6 Red lights comparison of the first red light proportion

	Original red lights[g]	Dijkstra red lights[h]	Improved red lights[i]
The first set of points	9	4	3
The second set of points	4	3	2
The third set of points	2	1	0

[g]The total number of red lights passed by the original A* algorithm.

[h]The total number of red lights passed by the Dijkstra algorithm.

[i]The total number of red lights passed by the improved A* algorithm.

Table 7 Total fuel consumption comparison of the first red light proportion

Original consumption(ml)[j]	Dijkstra consumption(ml)[k]	Improved consumption(ml)[l]
617.669	597.936	569.994
257.628	264.926	247.306
269.178	281.661	264.041

[j]The total fuel consumption of the original A* algorithm.

[k]The total fuel consumption of the Dijkstra algorithm.

[l]The total fuel consumption of the improved A* algorithm.

Table 8 Total fuel consumption comparison of the second red light proportion

Original consumption(ml)[j]	Dijkstra consumption(ml)[k]	Improved consumption(ml)[l]
652.933	592.928	578.804
240.008	238.672	229.686
278.011	287.391	266.541

[j]The total fuel consumption of the original A* algorithm.

[k]The total fuel consumption of the Dijkstra algorithm.

[l]The total fuel consumption of the improved A* algorithm.

Table 9 Total fuel consumption comparison of the third red light proportion

Original consumption(ml)[j]	Dijkstra consumption(ml)[k]	Improved consumption(ml)[l]
617.534	536.316	512.980
248.818	239.150	232.958
304.418	293.664	276.500

[j]The total fuel consumption of the original A* algorithm.

[k]The total fuel consumption of the Dijkstra algorithm.

[l]The total fuel consumption of the improved A* algorithm.

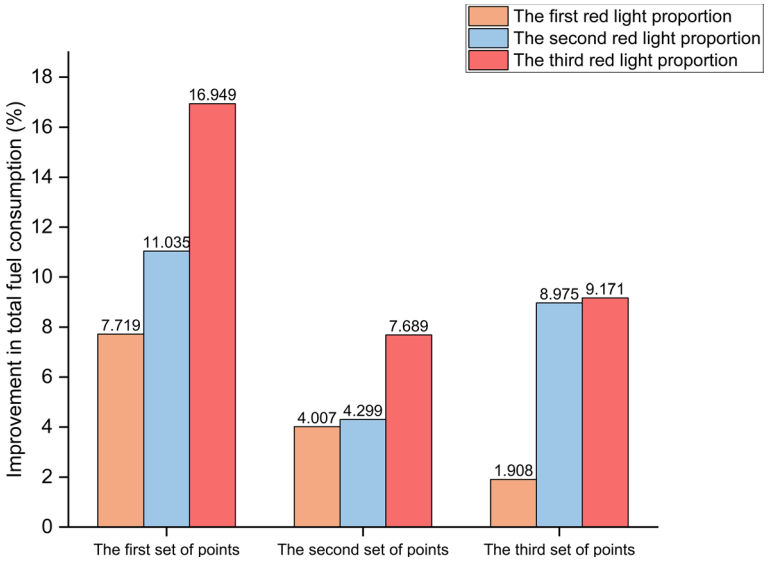


Fig. 18 Overall evaluation of the improved A* algorithm and original A* algorithm

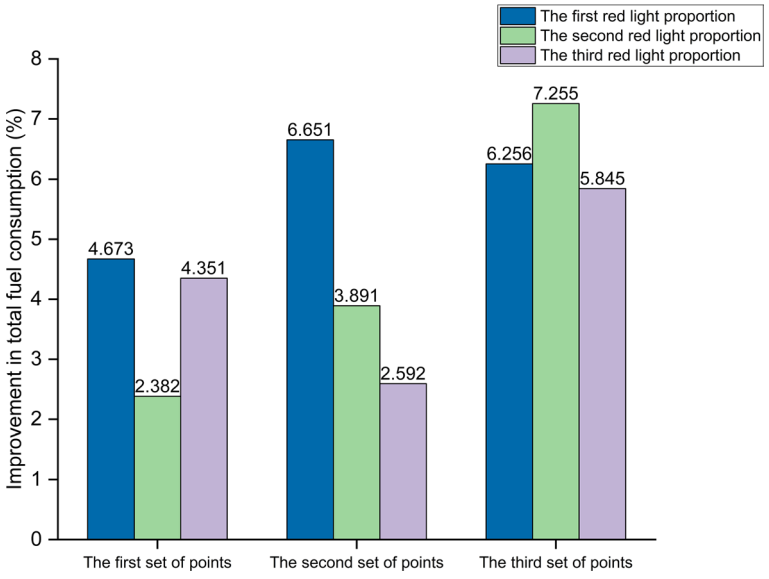


Fig. 19 Overall evaluation of the improved A* algorithm and Dijkstra algorithm

the potential effects of traffic conditions more carefully, thereby more accurately calculating idling time and idling fuel consumption. In the following work, we will focus on reducing the algorithm's complexity so that the conclusion has general applicability.

Appendix A List of variables and abbreviations

See Tables 10 and 11

Table 10 The meaning of variables and abbreviations in the ant colony algorithm

Original expression	Meaning
ant_k	The kth ant
allow_k	The set of all passable points
∂	The information heuristic factor
β	The expected heuristic factor
τ_{ij}	The pheromone concentration of the path
η_{ij}	The heuristic function
d_{ij}	The distance between i and j
$R_{\max}$	The maximum number of iterations
R_t	The current number of iterations
ρ	The pheromone volatilization coefficient
l_{history}	The historical optimal path
l_{current}	The current optimal path length
Q	The pheromone intensity
L_k	The total length of the path taken by ant_k
m	The total number of ants
$\text{point}_{\max}$	The total number of points in the map
pre_{dist}	The length of the optimal path

Table 11 The meaning of variables and abbreviations in the A* algorithm

Original expression	Meaning
$G(n)$	The actual cost from the starting to location n
$H(n)$	The estimated cost from location n to the endpoint
M	The prime with the same length as the hash table
$\text{dis}_{\text{Manhattan}}$	The Manhattan distance
$\text{dis}_{\text{Euclidean}}$	The Euclidean distance
$\text{dis}_{\text{Diagonal}}$	The diagonal distance
$E_{\text{consumption}}$	The expectations about fuel consumption
$\text{Pro}_{\text{vehicle}}$	The proportion of vehicles
$Fuel_{\text{consumption}}$	The fuel consumption
$\text{dis}_{\text{driving}}$	The driving distance
l_{driving}	The driving fuel consumption
l_{idling}	The idling fuel consumption
t_{idling}	The idling time in the driving process
S_1	The distance from the starting to the current location n
S_2	The distance from the current location n to the endpoint
$l_{\text{cur-idling}}$	The idling fuel consumption from the starting to location n
F	The total cost

Author Contributions All authors contributed to the study conception and design. The first draft of the manuscript was written by [Tianbo Liu] and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding This study was supported by the National Key Research and Development Program of China (2017YFB0102500), the National Natural Science Foundation of China (61872158,62172186), the Science and Technology Development Plan Project of Jilin Province (20190701019GH), the Korea Foundation for Advanced Studies' International Scholar Exchange Fellowship for the academic year of 2017-2018, the Fundamental Research Funds for the Chongqing Research Institute, and Jilin University (2021DQ0009).

Availability of data and materials All data, materials generated or used during the study appear in the submitted article.

Code availability The codes used or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Consent to participate Informed consent was obtained from all individual participants included in the study.

Consent for publication The participant has consented to the submission of the case report to the journal.

References

1. Abdollahzadeh B (2021) A multi-objective optimization algorithm for feature selection problems. Eng Comput. https://doi.org/10.1007/s00366-021-01369-9
2. Alazzam H, AbuAlghanam O, Sharieh A (2021) Best path in mountain environment based on parallel A* algorithm and apache spark. J Supercomput. https://doi.org/10.1007/s11227-021-04072-0
3. Bagheri SM, Taghaddos H, Mousaei A et al (2021) An A-star algorithm for semi-optimization of crane location and configuration in modular construction. J Intell Robot Syst 121(3):58. https://doi.org/10.1016/j.autcon.2020.103447
4. Chen Y, Guo J, Yang H et al (2021) Research on navigation of bidirectional A* algorithm based on ant colony algorithm. J Supercomput 77(2):1958–1975. https://doi.org/10.1007/s11227-020-03303-0
5. Chiabaut N, Faitout R (2021) Traffic congestion and travel time prediction based on historical congestion maps and identification of consensual days. Transp Res Part C Emerg Technol 124(102):920. https://doi.org/10.1016/j.trc.2020.102920
6. Costa LdS, Tonidandel F (2021) DVG plus A* and RRT path-planners: a comparison in a highly dynamic environment. J Intell Robot Syst 101(3):58. https://doi.org/10.1007/s10846-021-01326-0
7. Duan G, Fan T, Chen X et al (2021) A hybrid algorithm on the vessel routing optimization for marine debris collection. Expert Syst With Appl. https://doi.org/10.1016/j.eswa.2021.115198
8. Foead D, Ghifari A, Kusuma MB, et al (2021) A systematic literature review of A* pathfinding. In: 5th International Conference on Computer Science and Computational Intelligence (ICCSCI). https://doi.org/10.1016/j.procs.2021.01.034
9. Ghaffarpasand O, Talaie MR, Ahmadikia H et al (2021) Real-world assessment of urban bus transport in a medium-sized city of the middle east: driving behavior, emission performance, and fuel consumption. Atmos Pollut Res 12(3):113–124. https://doi.org/10.1016/j.apr.2021.02.004

10. Gharehchopogh FS, Abdollahzadeh B (2021) An efficient harris hawk optimization algorithm for solving the travelling salesman problem. Clust Comput J Netw Softw Tools Appl. https://doi.org/10.1007/s10586-021-03304-5
11. Gharehchopogh FS, Maleki I, Dizaji ZA (2021) Chaotic vortex search algorithm: metaheuristic algorithm for feature selection. Evol Intell. https://doi.org/10.1007/s12065-021-00590-1
12. Huang Y, Ding H, Zhang Y et al (2020) A motion planning and tracking framework for autonomous vehicles based on artificial potential field elaborated resistance network approach. IEEE Trans Ind Electron 67(2):1376–1386. https://doi.org/10.1109/TIE.2019.2898599
13. Li M, Zhou P, He X et al (2021) Parallel parking path planning and tracking control based on adaptive algorithms. Int J Automot Technol 22(4):949–965. https://doi.org/10.1007/s12239-021-0086-3
14. Liang C, Zhang X, Watanabe Y et al (2021) Autonomous collision avoidance of unmanned surface vehicles based on improved A-star and minimum course alteration algorithms. Appl Ocean Res 113(2):102,755. https://doi.org/10.1016/j.apor.2021.102755
15. Liu Z, Han Z (2020) Commodity search algorithm based on ant colony algorithm. In: 2020 IEEE Intl Conf on Parallel Distributed Processing with Applications. Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom) https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00161
16. Lu H, Zhang Y, Li Y et al (2021) User-oriented virtual mobile network resource management for vehicle communications. IEEE Trans Intell Transp Syst 22(6):3521–3532. https://doi.org/10.1109/TITS.2020.2991766
17. Mandloi D, Arya R, Verma AK (2021) Unmanned aerial vehicle path planning based on A* algorithm and its variants in 3d environment. Int J Syst Assur Eng Manag 12(5):990–1000. https://doi.org/10.1007/s13198-021-01186-9
18. Mattioli G, Roberts C, Steinberger JK et al (2020) The political economy of car dependence: a systems of provision approach. Energy Res Soc Sci 66(101):486. https://doi.org/10.1016/j.erss.2020.101486
19. Min H, Xiong X, Wang P et al (2021) Autonomous driving path planning algorithm based on improved a* algorithm in unstructured environment. Proc Inst Mech Eng Part D J Automob Eng 235(2–3):513–526. https://doi.org/10.1177/0954407020959741
20. Ming Y, Li Y, Zhang Z et al (2021) A survey of path planning algorithms for autonomous vehicles. SAE Int J Commer Veh 14(1):97–109. https://doi.org/10.4271/02-14-01-0007
21. MingDa O, YueYuan M (2021) Path planning for gravity aided navigation based on improved A* algorithm. Chin J Geophys-Chin Ed 63(12):4361–4368. https://doi.org/10.6038/cjg2020N0391
22. Mohammadzadeh H, Gharehchopogh FS (2021) A multi-agent system based for solving high-dimensional optimization problems: a case study on email spam detection. Int J Commun Syst. https://doi.org/10.1002/dac.4670
23. Mohammadzadeh H, Gharehchopogh FS (2021) Feature selection with binary symbiotic organisms search algorithm for email spam detection. Int J Inf Technol Decis Mak 20(01):469–515. https://doi.org/10.1142/S0219622020500546
24. Mohammadzadeh H, Gharehchopogh FS (2021) A novel hybrid whale optimization algorithm with flower pollination algorithm for feature selection: case study email spam detection. Comput Intell 37(1):176–209. https://doi.org/10.1111/coin.12397
25. Mohmmadzadeh H, Gharehchopogh FS (2021) An efficient binary chaotic symbiotic organisms search algorithm approaches for feature selection problems. J Supercomput 77(8):9102–9144. https://doi.org/10.1007/s11227-021-03626-6
26. Pae DS, Kim GH, Kang TK et al (2021) Path planning based on obstacle-dependent gaussian model predictive control for autonomous driving. Appl Sci. https://doi.org/10.3390/app11083703
27. Shayanfar H (2018) Farmland fertility: a new metaheuristic algorithm for solving continuous optimization problems. Appl Soft Comput 71(2):728–746. https://doi.org/10.1016/j.asoc.2018.07.033
28. Tang B, Hirota K, Wu X et al (2021) Path planning based on improved hybrid A* algorithm. J Adv Comput Intell Intell Inform 25(1):64–72
29. Tang G, Tang C, Claramunt C et al (2021) Geometric A-star algorithm: an improved A-star algorithm for agv path planning in a port environment. IEEE Access 9:59196–59210. https://doi.org/10.1109/ACCESS.2021.3070054
30. Thoresen M, Nielsen NH, Mathiassen K et al (2021) Path planning for UGVs based on traversability hybrid A*. IEEE Robot Autom Lett 6(2):1216–1223. https://doi.org/10.1109/LRA.2021.3056028

31. Vignesh R, Ashok B, Jeevanantham AK et al (2021) Enhancement of idling characteristics using multi-objective approach in light-duty diesel vehicle fuelled with orange peel biofuel. Fuel 291(120):222. https://doi.org/10.1016/j.fuel.2021.120222
32. Yu J, You X, Liu S (2021) Ant colony algorithm based on magnetic neighborhood and filteringrecommendation. Soft Comput 25(13):8035–8050. https://doi.org/10.1007/s00500-021-05851-w
33. Zhang G, Wang H, Zhao W et al (2021) Application of improved multi-objective ant colony optimization algorithm in ship weather routing. J Ocean Univ China 20(1):45–55. https://doi.org/10.1007/s11802-021-4436-6
34. Zhang H, Zhao F, Hao H et al (2021) Effect of chinese corporate average fuel consumption and new energy vehicle dual-credit regulation on passenger cars average fuel consumption analysis. Int J Environ Res Public Health. https://doi.org/10.3390/ijerph18147218
35. Zhang J, Feng Y, Shi F et al (2016) Vehicle routing in urban areas based on the oil consumption weight -dijkstra algorithm. IET Intell Transp Syst 10(7):495–502. https://doi.org/10.1049/iet-its.2015.0168
36. Zhang T, Xu G, Zhan X et al (2021) A new hybrid algorithm for path planning of mobile robot. J Supercomput. https://doi.org/10.1007/s11227-021-04031-9