




Received August 14, 2019, accepted September 1, 2019, date of publication September 4, 2019,  
date of current version September 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2939423

# A New Algorithm of the Best Path Selection Based on Machine Learning

XIAO-HUAN LIU<sup>1</sup> , (Member, IEEE), DE-GAN ZHANG<sup>2</sup> , (Member, IEEE),  
HAO-RAN YAN, (Member, IEEE), YU-YA CUI<sup>1</sup> , (Member, IEEE),  
AND LU CHEN, (Member, IEEE)

<sup>1</sup>Tianjin Key Lab of Intelligent Computing and Novel software Technology, Tianjin University of Technology, Tianjin 300384, China

<sup>2</sup>Key Laboratory of Computer Vision and System, Ministry of Education, Tianjin University of Technology, Tianjin 300384, China

Corresponding author: De-Gan Zhang (2310674826@qq.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61571328, in part by the Tianjin Key Natural Science Foundation under Grant 18JCZDJC96800, in part by the Major Projects of Science and Technology in Tianjin under Grant 15ZXDZSGX00050, in part by the Training Plan of Tianjin University Innovation Team under Grant TD12-5016 and Grant TD13-5025, in part by the Major Projects of science and Technology for their Services in Tianjin under Grant 16 ZXFZWX00010 and Grant 17YFZC GX00360, and in part by the Training Plan of Tianjin 131 Innovation Talent Team under Grant TD2015-23.

**ABSTRACT** This paper proposes and designs a best path selection algorithm, which can solve the problem of path planning for intelligent driving vehicles in the case of restricted driving, traffic congestions and accidents. We tried to solve the problem under these emergency situations in path planning process for there's no driver in intelligent driving vehicle. We designed a new method of the best path selection with length priority based on the prior knowledge applied reinforcement learning strategy, and improved the search direction setting of A\* shortest path algorithm in the program. This best path planning algorithm can effectively help different types of intelligent driving vehicles to select the best path in the traffic network with limited height, width and weight, accidents and traffic jams. Through simulation experiments and practical test, it is proved that the proposed algorithm has good stability, high efficiency and practicability.

**INDEX TERMS** Reinforcement learning, intelligent driving, path planning, shortest path algorithm.

## I. INTRODUCTION

With the development of artificial intelligence technology and the concept of intelligent transportation system, the technology of intelligent driving vehicles has become a hot spot of research [1]. The huge and complex transportation network environment poses even greater challenges for smart driving technology [2], [3]. In order to ensure that obstacles are avoided in the path selection of intelligently driven vehicles, current optimal path algorithms may miss the best choice due to overcorrection [4]. With the rapid development of artificial intelligence technology and automobile industry, frequent traffic congestion and accidents, residents' travel efficiency and safety issues have received more and more attention. In this context, the idea of intelligent driving traffic system came into being [5].

As an important part of intelligent transportation system, intelligent driving vehicles have outstanding research value [6], [7]. Vehicle drivers take action in the event of an

emergency, but intelligently driven vehicles can only learn and try to avoid the danger of situation like traffic accident, jam and temporary limits, etc. in the process of continuous optimization of the path plan, therefore we propose this method to solve best path planning of intelligent driving vehicle. The path planning of intelligent vehicles is based on the driving tasks and real-time changing environment given by intelligent decision-making to provide the driving area and driving guidance process for intelligent vehicles, and this process is based on the controller interacting with the environment to get feedback and use intelligent determination technology to plan the path. It is divided into global path planning and local path planning [8]. Global path planning is a combination of optimization and feedback mechanisms using local information in the case of known map databases to determine feasible regions and optimal paths [9]. Since the path generated by global path planning can only be a rough path, it does not consider the direction, width, curvature, road intersection and roadblock details of the path, and the uncertainty of the local environment and its state during the driving process of the intelligent vehicle, the vehicle may

The associate editor coordinating the review of this manuscript and approving it for publication was Soon Xin Ng.

encounter various unpredictable situations during driving, so local path planning must be based on local environmental information [10]. The local path planning is based on the route of the travelable area generated by the global path plan. According to the requirements of each sub-goal of the intelligent vehicle and the local environmental information, the road conditions and accidents sensed by the sensors are quickly and accurately judged, and the optimal controllable driving path of the intelligent vehicle is developed [11].

The A\* algorithm is the fastest algorithm for solving the shortest path in state space search in static road network. However, the road network in practical applications is not static, so we need to use other technologies. Reinforcement Learning (RL) is an important branch of Machine Learning (ML). Its goal is to give the machine the wisdom to think and react like a human being [12]. The greatest feature of reinforcement learning is that it can obtain the optimal strategy by maximizing long-term compensation by giving the current state reward. It has gained more applications in the field of robotics, and it can be applied to unknown environmental information path planning through trial and learn from errors with the environment [13]. The best path for intelligent driving vehicle to travel is to optimize the path according to certain performance indicators, including the shortest path, the minimum total cost, and the shortest travel time. In this paper, the combination of priori reinforcement learning technology and searching-optimal shortest path algorithm to find best path for intelligent driving vehicle. This path optimization method can effectively help different types of intelligently driven vehicles to plan the best path in the traffic network under the conditions of limits including height, width and weight, accidents and congestions.

## II. RELATED WORKS

The most commonly used Q-Learning algorithm in reinforcement learning evaluates the quality of an action by establishing an evaluation function to learn an overall optimal strategy [14]. The Markov decision process provides a theoretical framework for reinforcement learning. The process can be described by a quaternion array  $\langle S, A, P, R \rangle$ , where  $S$  stands for state set;  $A$  stands for action set;  $P$  stands for state transition probability matrix, and  $P_{ss'}^a = P[S_{t+1} = s' | S_t = s, A_t = a]$ , that is,  $P$  is the probability of the situation when the agent in the current state  $s$  at the current time transferred to the state  $s'$  after the action  $a$ ;  $R$  represents the reward function, and  $R_s^a = E[R_{t+1} | S_t = s, A_t = a]$ , that is,  $R$  is the reward obtained when the agent is in the state  $s$  and take the action of  $a$  [15]. The idea of Q-Learning to solve such problems is to learn an action value function  $Q(s, a)$  first, that is the Q value obtained after taking action  $a$  in state  $s$ , and then select action according to certain strategies. The strategy is a rule in which an agent selects an action in a given state. For example, the random greedy method strategy is to randomly select actions below a certain probability value, and select actions with a maximum Q value above the certain probability

value. At the same time, such action-value function, also called reward function or evaluation function. Action selection under certain strategy is based on the maximum of the target value function instead of the current instantaneous reward function [16].

Reinforced learning, also known as reward-learning, learns the optimal behavioral strategies of dynamic systems by perceiving changes in the dynamic environment and obtaining uncertain rewards and punishments from the resulting actions, and evaluating the pros and cons of the movements [17]. An important aspect of the application of this method in the practical field is the path planning of intelligently driven vehicles. Through the continuous perception of the environment by the vehicle, the acquired information is continuously learned and fed back through the intensive learning strategy, and finally the optimal path is obtained. Many excellent researchers and scholars have conducted a lot of researches and explorations, and made a lot of progress in the studies of reinforcement learning applied in path planning.

By introducing a frequency-maximum Q-value heuristic learning algorithm, the researchers improved the hierarchical reinforcement learning method to solve the problem of optimal behavior strategy learning for agents in a large state space and dynamic change environment, they introduced attribute maintenance operators and commitments and the planning awareness attribute enables the agent to have the ability to conduct online learning in a dynamic environment. Through the configuration of the driving environment and the continuous learning of the driving state, the optimal path is finally obtained [18]. Aiming at the slow convergence and low efficiency of path planning algorithm, a multi-agent path planning algorithm based on hierarchical reinforcement learning and artificial potential field is proposed. The environment-free model learning and local update ability using the hierarchical reinforcement learning method limit the strategy update process to the smaller local space or lower-dimensional high-level space, improve the performance of the learning algorithm, and the advantages of efficiency and convergence speed of algorithm is also proved through 3D simulation [19].

Scholars Misel Brezak and Ivan Petrović use professional mathematical methods to smooth the planning path of mobile robots to solve the problem of turns in the initial planning [20]. Some scholars in China have used the time difference method in reinforcement learning to solve the problem that traditional algorithms have too many inflection points, which are not suitable for practical application. For excessive exploration increases unnecessary training time and excessive utilization can cause the agent hardly converge to correct solution, these scholars solved the balance of exploring and utilization of the environment by dynamically adjusting the exploration factor.

In addition, some scholars have combined fuzzy neural networks with reinforcement learning to study path planning. They study the path planning success rate and shortest path

problem of mobile objects in complex network environments through pre-processing and post-processing strategies to optimize paths [21]–[23]. In addition, the research fields based on ant colony algorithm, genetic algorithm and particle swarm optimization algorithm have attracted the attention of many researchers. These studies can solve some outstanding problems in the intelligent mobile path planning, however, there are certain defects or shortcomings that need to be resolved and balanced in future research.

### III. THE PRINCIPLE OF OPABRL ALGORITHM

Path selection is an important aspect for intelligent driving vehicle. This part will design a length-first best path algorithm based on reinforcement learning strategy, referred to as OPABRL (Optimized Path Algorithm Based on Reinforcement Learning).

#### A. REINFORCEMENT LEARNING STRATEGY BASED ON PRIOR KNOWLEDGE

While the intelligent vehicle driving, there are two problems we have to solve: path planning and path selection. In order to simplify the system, we first describe the reinforcement learning strategy based on prior knowledge and the update rules:

Assuming that the intelligent vehicle is always driving on a road with a certain width, the problem of path planning can be understood as solving the problem of planning one or more paths that can reach the end point from the starting point and successfully avoid the obstacle in the road environment. The shortest path algorithm can solve the first part of the problem. For the second part, we consider rasterizing the road network, and determine whether it is an obstacle based on whether the grid is safe or not. Here, the obstacle mainly contains three cases, including traffic limits, accidents and congestions. The limits mainly refer to three conditions: height limit, width limit and weight limit; traffic accidents are divided into three levels: mild, moderate and heavy congestion; traffic congestion is divided into slight congestion, slow and heavy congestion. Where the grid is included in any of the above, it is considered an obstacle. When the system detects that the safe grid is non-obstacle, it is divided into two cases: correct judgment and wrong judgment. The correct judgment is that the grid meets the conditions as the next step of the path, and can be used as an option by the optimal path selection step; the wrong judgment means that the grid is unsafe while the obstacle is not detected, and the vehicle passes through such a grid. The total cost of the grid will increase due to misidentification.

*Definition 1:* We define the cost equivalent of misidentification as virtual time  $t_{ee}$ . The error equivalent time here includes the cost equivalent time  $t_{de}$  due to the wrong judgment, and the cost equivalent time  $t_{ce}$  of turnings and crossings. Then

$$t_{ee} = t_{de} + t_{ce} \quad (1)$$

When the system detects an unsafe grid with obstacle, the probability of wrong judgment for a non-obstacle grid is

almost zero due to the prior knowledge, here we ignore this situation.

The problem of path selection can be understood as that we select the best solution according to the judging criteria set by the user based on successful path planning, i.e. filter out the relatively optimal path, and this path satisfies the parameters required by the user. Taking the length-first best path as an example, in equation (1), we combine the calculation of the virtual equivalent time  $t_{ee}$  with the shortest length  $l_i$  between the initial node and the destination node and the minimum number of turns  $n_i$  required to avoid obstacle. In practical applications, we idealize the traffic conditions of intersection traffic lights as 1/3 for each color of light, and the probability of vehicles passing directly at intersections is 1/3, that is, it can pass directly only when the light is green. Note that the length of the actual path and the number of turns are  $l$  and  $n$ , respectively.

*Definition 2:* Define virtual equivalence ratio  $\varphi$ :

$$\varphi = \frac{l - l_i}{l_i} + \frac{n - n_i}{n_i} * 1/3 \quad (2)$$

*Theorem 1:* The smaller the virtual equivalence ratio, the smaller the cost of misidentification, and the better the performance of the algorithm.

*Proof:* According to the actual situation, we know:

$$t_r = (1 + \varphi)t_{ee} \quad (3)$$

where  $t_r$  is the actual travel time of the vehicle. Assume that the driving speed of an intelligent vehicle is  $v$ , then:

$$t_r = \frac{l}{v} \quad (4)$$

That is, there is the following relationship between the virtual equivalent time and the actual driving length:

$$\frac{l}{v} = (1 + \varphi)t_{ee} \quad (5)$$

Transform the form into:

$$l = (1 + \varphi)t_{ee}v \quad (6)$$

That is, when the speed is constant, the smaller the virtual equivalence ratio, the smaller the length of the actual planning path, the smaller the cost of the misidentification in the path planning and the better the performance of the algorithm.

*Theorem 2:* When cost is used as the reference standard parameter, the problem is simplified to the solution of the minimum  $t_{ee}$ . At this time, the greater the total cost time  $t_{ee}$ , the smaller the reward, the slower the update of the Q value, and the slower convergence of the reinforcement learning algorithm.

*Proof:* In the Markov process, the quaternion arrays  $(S, A, P, R)$  represent the finite state set, the finite action set, the transition probability and the reward function [24], respectively, and the update rules are as shown in equation (7) and (8):

$$Q(s, a) \leftarrow r + \gamma \max_a Q(s', a') \quad (7)$$

$$Q(s, a) = Q(s, a) + \alpha \left[ r + \gamma \max_a Q(s', a') - Q(s, a) \right] \quad (8)$$

where  $Q(s, a)$  is the initialization state. When the system take the action of  $a$  in the state  $s$ , the probability of transferring to the state  $s'$  is recorded as  $p(s, a, s')$ , and the learning factor  $\alpha$  determines the proportion of the new data covering the original value. For example, if the learning factor is 0, it means that no new date is learned, only the original values are stored, and the learning factor is 1 means that all of the data are replaced with newly learned values. When an environmental problem needs to be characterized by supposing, a fixed constant can be set to the learning factor, and its range is  $\alpha \in [0, 1]$ . The rewards obtained at this time can be expressed as  $R(s, a) = \sum_{s'} p(s, a, s') R(s, a, s')$  first, our goal is to

find the biggest reward which is denoted as  $\sum_{t=0}^{\infty} pr_t$ , where  $0 \leq p < 1$ ,  $r_t$  is the instantaneous reward at time  $t$ .

*Definition 3:* The Q-value function is used to obtain the optimal strategy under unknown conditions, and the Q-value function under the optimal strategy is called the best Q-value function, which can be understood as the fixed point of a certain operator  $G_M$ , and it meets E.q.(9) and E.q.(10):

$$(G_M Q)(s, a) = R(s, a) + \alpha \sum_{s'} p(s, a, s') Q(s', a) \quad (9)$$

$$R(s') = \max_a Q(s', a) \quad (10)$$

In a time-continuous complex system, the convergence of Q-learning requires that each state- action pair be accessed innumerable times. Therefore, we try to make the above discrete process continuous, i.e. to obtain a reward function expression in a continuous system.

*Definition 4:* In continuous systems, the reward function for reinforcement learning is:

$$R(s, a) = \sum_{s'} p(s, a, s') \int_0^{\infty} \int_0^t e^{-\theta t} dt dr \quad (11)$$

where:  $\theta$  is the continuous discount factor of the system used to represent the discount offset in a continuous system. The discount factor determines the importance of future reward values. If the discount factor is 0, it means that only the current return value is considered, regardless the impact of future actions; if the discount factor is 1, it is considered that all subsequent actions have the highest impact on the current action reward value. When the discount factor is between 0 and 1, the larger the value is and the greater the influence of the previous action is.

According to the update rule defined by Q-learning, the discretization reward  $R(s, a)$  in continuous time system after rasterization process can be expressed as:

$$R(s, a) = \sum_{s'} p(s, a, s') R(s, a, s') \quad (12)$$

According to the definition 4,  $R(s, a)$  and  $t_{ee}$  meets:

$$R(s, a) = \theta t_{ee}^{-1} \quad (13)$$

$$Q(s, a) = Q(s, a) + \alpha \left[ \theta t_{ee}^{-1} + \theta \max_{a'} Q(s', a') - Q(s, a) \right] \quad (14)$$

Assume that the vehicle can find a path from the starting point to the goal point in all cases. The best path selection problem is how to reasonably select the path from starting point to the goal point under the premise of avoiding the obstacle. The best path mentioned in this paper includes the following metrics: time  $T$  taken to complete the best path, including the algorithm execution time  $t_a$  and the vehicle travel time  $t_r$ , they meet:

$$T = t_a + t_r \quad (15)$$

Therefore, when the time is taken as the main consideration parameter, the path with the smallest  $T$  is selected. At this time, the larger total time  $T$  is, the smaller reward is, the slower update of Q value is, and the slower Q-learning convergence is.

When the shortest travel length path is taken as the best path, we assume each grid is one step, the length problem is converted into a step problem. At this point, the problem is converted to find the least number of grid steps from the starting point to the goal point. That is, the best path problem can be convert to the minimum number of steps  $N$ , which can be realized by a counter in the algorithm program. In the problem studied in this paper, the part of the path selection is carried out according to the user-set parameters and as mentioned above, it can be the shortest path length, or the minimum traveling time, or the minimum comprehensive cost. As a standard, these parameters can be converted according to different weights, and the weights need to fixed by multiple reinforcement learning and finally reach a reasonable range.

## B. ESTABLISHMENT OF NETWORK TOPOLOGY

In order to realize the auto-positioning of intelligent vehicles and the path planning in the whole region, it is necessary to use the sensors carried by them to detect the environment data of the outside world and use the data to model the external environment. The basic idea of grid map is to divide the working environment of intelligent vehicles into identical grids and each grid corresponds to a specific small area in the environment. When the sensor detects that the environment changes, the grid map will maintain timely. We use the grid method to construct the environment map, when using this method to model the environment, the more grids are divided, the smaller the map and the higher the accuracy of the map for the same working environment [25]. This process can be divided into two steps:

### 1) BOUNDARY LEARNING PROCESS

The process of boundary learning means that the intelligent driving vehicle starts searching driving area in a specific direction along the boundary of the area or the boundary of the obstacle next to the boundary, and learns the contour of the entire driving environment and distribution of obstacles by the boundary in the process [26]. The black grids represent obstacles and white part represents the safe area.

## 2) ENVIRONMENT MAP GENERATION

The environment map is in the form of rectangular grid. By dividing the entire environment area into grids, the actual environment is mapped to grids to realize discretization of the driving environment. If the size of the vehicle is  $d$  and the grid size is the same as the size of the vehicle, then the number of grids in this map is:  $XY/d^2$ . The grid is represented by  $G$ .  $G(x, y)$  is used to describe the data of the area represented by the grid, and  $x, y$  represent the horizontal and vertical coordinates of the grid in the entire area;  $G(x, y)$  indicates whether the grid contains an obstacle. When  $G(x, y)$  is 1, it means that the grid is an unsafe grid (that is, the grid contains obstacles) and the grid is output black. When  $G(x, y)$  is 0, it means the grid is a safe grid (that is, this grid does not contain obstacles) and the grid is output white. The entire working area is then represented by a binary matrix with values of 0 and 1 and output as shown in figure 1:

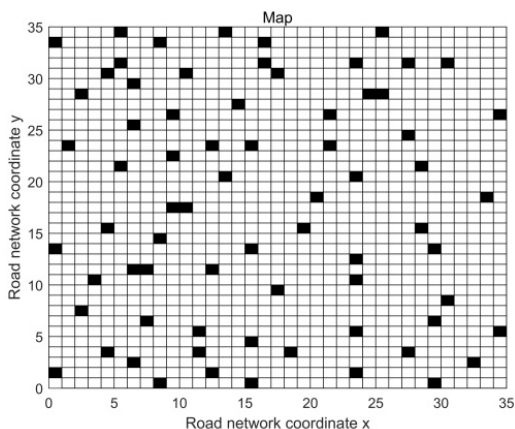


FIGURE 1. Map for test.

As can be seen from the figure, the grid map can clearly identify unsafe area and safe area, on which we can plan the path of the intelligent vehicle.

## C. PATH PREPROCESSING AND PLANNING

The traffic network without negative margins is abstractly represented as a topological map of  $G(V, E, W)$ , where  $V, E, W$  represent the set of nodes, the set of edges and the set of weights of the edges, respectively.

*Definition 5:* The connected edge  $e = (i, j) \in E$  represents the edge from node  $i$  to node  $j$ ,  $w(e)$  represents the weight of edge  $e$ , and  $w'(e)$  represents the weight after the  $e$  changes.

*Definition 6:*  $Dist(i, j)$  represents the shortest path from node  $i$  to node  $j$ , and the corresponding shortest path value is  $d(i, j)$ . Here  $Dist(i, j)$  is not equivalent to  $Dist(j, i)$ ,  $d(i, j)$  and  $d(j, i)$  may not equal either.

The mathematical model to solve the shortest path problem of point pairs  $(i, j)$  is expressed as:

$$\min d(i, j) = \sum_{e \in Dist(i, j)} w(e) \quad (16)$$

where  $\min d(i, j)$  is the conventional shortest path. In practical applications, for example, when path length is the best

path evaluation criterion, we will calculate the budget cost  $h(n)$  in the evaluation function of the A\* algorithm. The virtual time  $t_{ee}$  of the intelligent driving vehicle avoiding obstacles is linked, that is, when we run the shortest path algorithm, the evaluation is based on the equivalent virtual time in each iteration of the algorithm each time the estimated cost of the subsequent points is calculated. The problem can be simplified as:

$$f(n) = g(n) + h(n) \quad (17)$$

$$g(n) = t'_{ee} \quad (18)$$

$$h(n) = d(i, j) \quad (19)$$

According to equation (6):

$$d(i, j) = (1 + \varphi)t_{ee}v \quad (20)$$

where the velocity  $v$  is a constant, which is a function that converts the heuristic evaluation cost into a function of the path length.

According to the pre-processed road network topology, a priori reinforcement learning training is performed. After obtaining the experience, the reference path calculated according to the optimized shortest path algorithm determines the next grid position of the starting point, and at the same time, by continuously interacting with the environment to obtain information, whether the reference path is safe is judged, this operation is repeat until we get the path to the end point.

## D. BEST PATH SELECTION AND DETERMINATION

*Theorem 3:* When the best path is selected, the number of grids in the planned path is used as the criterion for determination.

*Proof:* The vehicle did not actually travel during the path planning process, the actual travel distance could not be obtained, but in the road grid rasterization part, the size of the grid is fixed, so we can actually plan the path. The length of the route is equivalent to the number of planned route paths. Although the specific value of the route cannot be directly indicated, it can be accurately determined that the shortest path under this condition is the path with the smallest number of grids.

*Theorem 4:* In the case of unforeseen parameters related situation (including congestion, limit line, fuel consumption in case of accident, time, etc.), the virtual equivalent ratio is calculated according to the estimated value in the parameter setting, and it is taken as the criteria for the determination of the path.

*Proof:* The intelligent driving vehicle obtains the environmental information through the sensor in the unknown environment in the local optimization method designed in this paper, adjusts and selects best path without collision in real time. In practical applications, in order to reduce the workload, we consider to take the initial path calculated by the shortest path algorithm. When the obstacle is found and identified, the path is re-planned. In fact, although the

traditional A\* algorithm can solve the shortest path problem well, in the unknown environment, the path planning and optimization of intelligent vehicles cannot be realized by the shortest path algorithm only. Therefore, we design best path selection strategy based on reinforcement learning and shortest path algorithm. Firstly, the reinforcement learning training based on prior knowledge is carried out through the intelligent driving vehicle controller. By reducing the state set calculation time, the convergence speed of the reinforcement learning algorithm is accelerated, and the time consumption caused by the large sample learning and training is reduced. Then, the feedback from the unknown environment is continuously obtained through the vehicle's sensor system, and the corresponding action instruction is obtained according to the reinforcement learning algorithm. Intelligent driving vehicle go along the path calculated by shortest path algorithm according to the selected reference standard while avoiding the obstacle. And this reference standard can be converted to the actual value by the virtual equivalent increase ratio  $\varphi$  to achieve the optimal path selection from the starting point to the end point.

It should be noted that the estimated value here is initial. As the algorithm implements, the reinforcement learning strategy will continuously update the reference estimate value to adjust it to a reasonable range. Therefore, we use the initial estimated value to judge when implementing the algorithm, while when the value is updated, it is more accurate to calculate with the updated value.

#### IV. THE DESIGN OF OPABRL ALGORITHM

In practical applications, intelligent driving vehicles work in environment with multiple obstacles, including moving obstacles, that is, the environment is dynamically changed, so the path selection problem becomes more complicated. Therefore, the OPABRL algorithm designed in this paper proposes a more reasonable method and the process is shown in figure 2:

##### A. OPABRL ALGORITHM

###### 1) PARAMETER SETTING

The intelligent driving vehicle best path algorithm needs certain evaluation criteria to evaluate the function and performance of the algorithm. Therefore, we set the following evaluation parameters to evaluate the algorithm:

Firstly, during the running of the vehicle, the length of the path needs to be calculated in advance, and the best path that meets the condition can be selected according to the calculation result.

*Definition 7:* In the vehicle path planning process, the number of the network grid covered by the planned path is used as the basis for calculating the path length.

In addition, the main purpose of path planning is to be able to reach the goal point from the start point and to successfully plan the path, i.e. the probability of path planning algorithm can obtain the best solution.

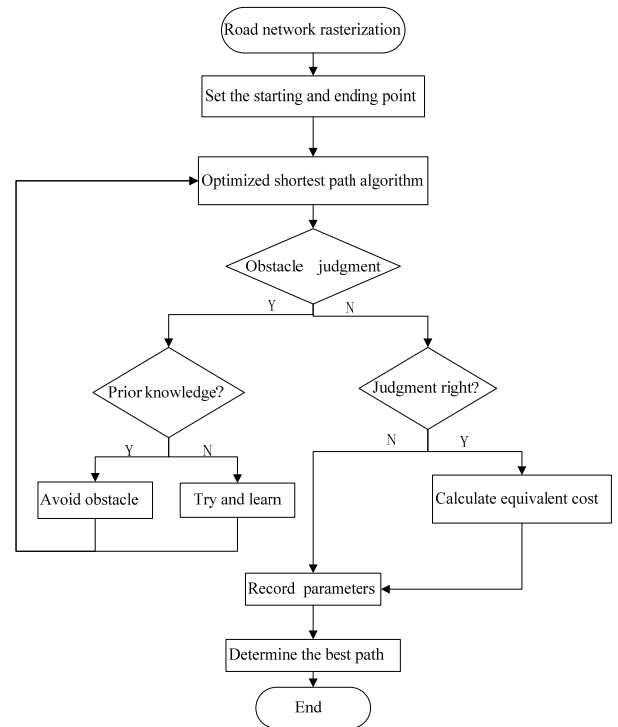


FIGURE 2. Flow of the algorithm.

*Definition 8:* We define the ratio of successful best path planning to the total plans in the process of calculating the shortest path, and it is called the best solution probability  $p$ .

In practical time-varying traffic networks, the best solution may not be obtained every time. Therefore, the probability of finding the best solution is an important indicator for evaluating a path planning algorithm. Even if the path from the starting point to the end point is obtained, the cost of this path is also a very important parameter. In different application scenarios, the meaning of the cost is not exactly the same.

*Definition 9:* Among the successfully planned paths, according to the actual needs, the cost parameters such as the shortest travel distance, the shortest travel time, the minimum number of turns and the lowest actual fuel consumption can be used as the basis for determining the best path. It can be converted to a virtual equivalent ratio calculation by conversion.

In the actual road network, due to urban construction, bridges and underground passages, etc., different kinds of limit are set. For obstacles of different heights limit, in the path planning, according to the height of the intelligent vehicle, we convert the time of the vehicle pass these limits into different virtual equivalence ratios to select the best path. Since the intelligent driving vehicle is assumed to travel at a constant speed, it can be directly converted into the driving distance. Similarly, for different levels of traffic accidents and traffic jams, different estimated bypass distances  $l_{est}$  are set, the estimated bypass distance and no-obstacle distance  $l$  also

**TABLE 1. Virtual equivalent increase ratio setting.**

	Level	Estimated extra length	$\varphi$
Height limit	<3.5m	1.1 <i>l</i>	0.1
	<4.5m	1.3 <i>l</i>	0.3
	<5m	1.5 <i>l</i>	0.5
Width limit	<3m	1.1 <i>l</i>	0.1
	<4m	1.3 <i>l</i>	0.3
	<5m	1.5 <i>l</i>	0.5
Weight limit	<5t	1.2 <i>l</i>	0.2
	<10t	1.5 <i>l</i>	0.5
	<20t	1.8 <i>l</i>	0.8
Traffic accident	Mild	1.4 <i>l</i>	0.4
	Moderate	1.6 <i>l</i>	0.6
	Heavy	1.8 <i>l</i>	0.8
Traffic congestion	Slight	1.1 <i>l</i>	0.1
	Slow	1.2 <i>l</i>	0.2
	Heavy	1.5 <i>l</i>	0.5

meets:

$$\varphi = \frac{l_{est} - l}{l} \quad (21)$$

As shown in Table 1, it should be pointed out that this is only a reference value within a reasonable range set by experience for research and the shortest path without obstacle is  $l$ . In practical applications, continuous learning and correction of reinforcement learning strategies can fix these reference values in more reasonable ranges.

In this paper, we convert the determination of the best path into the virtual equivalent ratio and the number of turns required to avoid the obstacle.

Finally, the performance of the algorithm itself, including algorithm running time and convergence speed, are also important indicators to evaluate the algorithm.

## 2) SHORTEST PATH ALGORITHM OPTIMIZATION

There are many algorithms based on grid for path planning, including A\*, D\*, D\* Lite, etc., where A\* can calculate the shortest path between two known points quickly. However, there is a problem with the grid-based A\* algorithm: it can be searched as a node in the state space based on the center point of the grid [27], while the path searching directions will be limited, so that the final path may not be the best one for there are a large number of turns and vertices [28], and it may not be the shortest in terms of path length. The advantage of the A\* algorithm is that it combines the Dijkstra algorithm with the information block of the BFS algorithm, and it use the cost  $g(n)$  from the initial node to arbitrary node  $n$  and the heuristic evaluation cost  $h(n)$  from node  $n$  to the target point to evaluate the current node, its evaluation function is:

$$f(n) = g(n) + h(n) \quad (22)$$

where:  $f(n)$  is the evaluation function from the initial point to the target point via node  $n$ ;  $g(n)$  represents the actual

cost from the initial node to node  $n$ ;  $h(n)$  represents the heuristic evaluation cost from current node  $n$  to the target node, which plays a key role in the evaluation function. It determines the efficiency of the A\* algorithm [29]. A\* algorithm can design heuristic function according to the specific path planning require, and make the search direction closer to the target state. The process of A\* algorithm as follows:

(1) The starting point is stored in the open table, and obstacle points are stored in the close table.

(2) Select node  $n$  with the smallest  $f$  value in the open table and put it into the close table.

(3) Judge whether  $n$  is the target point. If it is the target point, generate a path according to its forward pointer; otherwise expand node  $n$  to generate a sub-node  $m$ .

(4) Establish a pointer in the open table from the sub-node  $m$  back to  $n$ , calculate  $f(m) = g(m) + h(m)$ .

(5) Add a judgment statement to determine whether there is a node  $m$  in the open table. If the judgment fails,  $m$  should be into the open table; if the judgment is successful, compare the  $f(m)$  values of different forward pointers, and select smaller  $f(m)$  value.

(6) Update  $g(m)$ ,  $f(m)$  and the forward pointer of the sub-node  $m$ .

(7) According to the positive sequence of the numerical values, the  $f$  values are reordered in the open table, and the process returns to step (2).

When implementing the A\* algorithm in this paper, we optimized the step (2). When there is more than one node with the smallest  $f$  value in the table, we calculate the value between the candidate node and the connection of initial point and the destination node, and select the node with the smallest value. Before step (2), this connection will also limit the eight consecutive search directions set by the A\* algorithm itself, and try to ignore the four directions with larger values to the connection. We search the four directions close to the connection between the start point and the destination, although the calculation amount is increased when selecting the next node candidate for each node, experiments have shown that the calculation is still simplified on the whole and the computational efficiency of the algorithm is improved. The pseudo code of its searching optimization is as follows:

```

while ~strcmp(fieldpointers{posind}, 'S')
%if current node is not the starting point
setpointers(setOpen(I)) = movementdirections(jj);
%update move direction
if(jj-ii>0)
%if parent node of current node is from positive direction
case 'R'
px = px + 1; %searching direction move right
case 'U'
py = py - 1; % searching direction move up
elseif(jj-ii<0)
% if parent node of current node is from negative direction
case 'L'

```

```

px = px - 1; % searching direction move left
case 'D'
py = py + 1; % searching direction move down
end
end

```

## B. THE FLOW OF OPABRL ALGORITHM

### 1) THE PROCEDURES OF OPABRL

In the OPABRL algorithm for intelligent driving vehicle designed in this paper, the reinforcement learning strategy based on prior knowledge provide strategy for vehicles to plan path in unknown environment. The principle of this algorithm to solve this problem is to transform from one state to another after the agent performs an action in an action set and provides an immediate reward value. The goal of the agent is to maximize its total reward value, and learn to find the best selection of corresponding action of each state. The best action means that after been executed, the maximum reward value can be obtained from the final result. The calculation of this reward is to add the results of multiple predicted reward values of all subsequent actions in the current state by its weight. One of the advantages of the reinforcement learning algorithm is that it can compare the expected reward value of the optional actions without learning the environment model. In addition, another advantage is that it can handle random conversion and return value problems without any modification.

It can be seen that in the Markov process of reinforcement learning, the Q-valued function under the optimal strategy is actually the sum of the expected returns  $R(s, a)$  in the whole learning process, which explains the convergence in reinforcement learning. In the process, if the action set is given, the size of the state set is the main factor affecting the convergence speed of the Q-learning process. Therefore, we will guide the action state determined by the prior knowledge to guide marking in the learning process, which can greatly improve the learning efficiency.

The main steps of the best path algorithm based on reinforcement learning are as follows:

(1) The vehicle controller interacts with the known environment to obtain prior knowledge at first. We record this process as a learning process. The controller records the state action pairs in the learning process and learns the trend of such benign feedback. At the same time, through continuous learning, the referenced estimate values in the parameter setting are periodically updated until the values are kept within a reasonable range;

(2) Implement optimal A\* algorithm to obtain a reference best path;

If the vehicle controller encounters the obstacle grid during the travel, it first detect whether it belongs to the prior knowledge storage, and compares the obstacle parameters encountered with the storage obstacle parameters:

If the two sets of parameters meet the belonging relationship, the measures are taken according to the trend that has been learned and recorded, and the reward of the operation

is weighted, that is, the reward weight is increased for the benign operation;

If the two sets of parameters do not meet belonging relationship and are recorded as new obstacles, the corresponding obstacle sub-algorithms are called to perform the trial and error process. These sub-algorithms are correspond to traffic congestion, accidents and limited driving, respectively.

Each trial and error process is firstly executed according to the Q-learning algorithm according to the unknown environment, that is, when the vehicle controller detects the grid obstacle in the current state, according to the greedy strategy it select an action and learn experience knowledge, the next state and instant reward are obtained, then the value of the state action pair is updated according to the iterative rule until the target state is reached, and the trial and error process is completed.

The pseudo code of judging process is as follows:

```

[ temp, ii ] = min( setOpenCosts + setOpenHeuristics );
% select the smallest cost point from the OPEN table
if ( ii >= 1 && ii <= length( setOpen ) )
% temp has been trained in prior knowledge
setOpen = [ setOpen( 1:ii-1 ); setOpen( ii+1:end ) ];
setOpenCosts = setOpenCosts( ii:end ) + 1;
% reward weight increase
ri = Virtualincrease( ii, goali, n, N );
% call sub-algorithm to calculate virtual increase
else % temp hasn't been trained
setOpen = setOpen( 1:end-1 );
setOpenCosts = setOpenCosts( ii:end ) - 1;
% reward weight decrease
end

```

The sub-algorithm to calculate the virtual equivalence ratio  $\varphi$  is as follows:

```

function r = Virtualincrease( ii, goali, ni, N )
r = ( goali - ii ) / ii + [ ( N - ni ) / ni ] * 1/3;
end

```

(3) If the path is obtained, the parameters of the best path are calculated and recorded, otherwise, step (2) is looped from the step before the obstacle is encountered until the path is obtained.

(4) According to the recorded and calculated parameters, the best path is selected according to the value of the reference parameter.

The implementation of the prior knowledge learning and the initial road network rasterization part can be logically arranged side by side, while because the whole learning process is not involved in the running of the algorithm, only the result is applied and continuously updated and improved, this process can be performed separately, and the data is periodically updated when the operation is implemented later.

## 2) IMPLEMENTATION PROCESS

In the following part we will introduce the process of intelligent driving vehicle avoid obstacles of height limit by the proposed algorithm.



(1) First, rasterize the planned range area to establish a network topology;

(2) The user inputs the starting point and the ending position;

(3) Obtaining a reference path by optimizing the A\* algorithm;

(4) Reinforcement learning strategy to optimize the path process:

Initialize state set  $S$ ; action set  $A$ ;

Define learning factor  $\alpha = 0.7$ , discount factor  $\theta = 1$ ;

Initializing the state of the Markov process, starting at time  $t = 0$ ;

Learn  $Q(s, a)$  when encountering obstacle: take action  $a_t$  according to different limit conditions and current state  $s_t$ , update state to  $s_{t+1}$ , then update  $Q(s, a)$ ;

Enter the following loop until the stop condition is met:

a) Determine whether the obstacle is a height limit obstacle, if it is, implement state-action pair in prior learning and update  $Q(s, a)$ , jump to step c);

b) Otherwise, according to the current state  $s_t$  and greedy strategy, calculate  $a_t$ , then get the state  $s_{t+1}$  and update  $Q(s, a)$  and the initial estimated value according to the iterative formula;

c) Let  $t = t + 1$ , return to step (1);

d) End.

(5) Select the best path that can bypass the obstacle according to the determination condition.

### C. THEORETICAL ANALYSIS OF ALGORITHM COMPLEXITY

Space complexity and time complexity are two important factors in measuring whether an algorithm is suitable for actual execution. Time complexity refers to the amount of computation required to run an algorithm; space complexity refers to the amount of memory required to collectively run an algorithm. The time and space complexity of the algorithm embody the total amount of resources required for the algorithm to run on a computer, and is also the two main factors that make up the complexity of the algorithm. The basic principle is to designate the key operations to solve the problem as basic operations, and the number of these operations in the algorithm is called the time complexity of the algorithm. The total storage space occupied by all operations during this period is called the space complexity of the algorithm, which is a function of the model of the problem [30]–[32].

*Theorem 5:* The worst time complexity of the OPABRL algorithm is  $O(n^2)$ .

*Proof:* The algorithm designed in this paper can be used to calculate the optimal path of different starting point and ending position every time. In the initial search operation, it involves the operation of strengthening the learning prior knowledge, but because this operation is not executed every time. Algorithms must be executed from the beginning, but are constantly updated during the operation. This update can be considered as the basic operation of the total number  $n$  of nodes, so when considering the complexity of the algorithm

time, the complexity  $O(n^2)$  of the shortest path algorithm itself can be ignored. The worst case of the shortest path algorithm is that all the nodes in the network are connected in pairs. Since the scanning is performed on all nodes, the worst time complexity of the algorithm is  $O(n^2)$ . However, in an actual traffic network, the number of associated nodes of a node generally does not reach  $n-1$ , and neither node can be connected to each other, and the calculation amount is greatly reduced in the algorithm due to the setting of the search mode. The number of associated points in complexity can be considered as a constant, that is, in some cases the time complexity of the algorithm can be reduced to  $O(Cn)$  (where  $C$  is a constant).

*Theorem 6:* The space complexity of the OPABRL algorithm is  $\Omega(n^2)$ .

*Proof:* For large sparse data such as road networks, the time required to construct and cancel the graph is considerable. Therefore, adjacency matrix is a better choice for large sparse graphs such as road networks to store data. In terms of space complexity, the storage method used by the road network involved in this paper is the adjacency matrix, that is, the data is stored in a space of up to  $n \times n$ , so the space complexity is  $\Omega(n^2)$ .

## V. CONCLUSIONS AND FUTURE WORKS

### A. SIMULATION TEST AND ANALYSIS

#### 1) TEST ENVIRONMENT AND PARAMETER SETTINGS

We will target on algorithms that are commonly used in intelligent robot path planning: GA(Genetic Algorithm) and ACO(Ant Colony Optimization), ANNs (Artificial Neural Networks), and PSO (Particle Swarm Optimization). The four most widely used and most practical algorithms [33] are compared with the designed algorithm OPABRL in terms of algorithm function and performance. The advantages of the designed algorithm are analyzed by the simulation results and the parameter settings are shown in table 2 :

The simulation process will be carried out in two parts on the MATLAB platform. The first part will first prove the path planning and obstacle avoidance function of the designed algorithm. The second part will firstly be implemented in the same environment where the obstacle is set, between the same start and goal point, compare the differences among the planned paths of different algorithms in the same environment. The main measurement parameters of the simulation test include:

- (1) Algorithm convergence speed;
- (2) The length of the planned path of different algorithms and the number of turns under the same conditions;
- (3) The best solution probability  $P$ ;
- (4) Virtual equivalence ratio  $\varphi$ ;
- (5) Algorithm running time.

#### 2) SIMULATION EXPERIMENT RESULTS AND ANALYSIS

Experiment 1: After several simulation experiments, in order to better display and illustrate the effect, this paper will first

TABLE 2. Parameters setting.

Environment parameters setting	Grid number	35*35
	Unsafe grid ratio	0.5
OPABRL parameters setting	Learning factor $\alpha$	0.7
	Discount factor $\theta$	1
	Greedy strategy $\epsilon$	0.5
ACO parameters setting	Ant number m	80
	Heuristic factor a	1
	Pheromone volatility $\rho$	0.3
GA parameters setting	Population size M	100
	Cross probability $P_1$	0.6
ANNs parameters setting	Mutation probability $p_2$	0.01
	Network layers	3
	Number of input nodes	6
	Number of output nodes	1
PSO parameters setting	Number of hidden layer nodes	12
	Particle swarm size	80
	Learning factor $c_1$	1.5
	Learning factor $c_2$	2.0
	Inertia weight parameter $\omega$	0.9

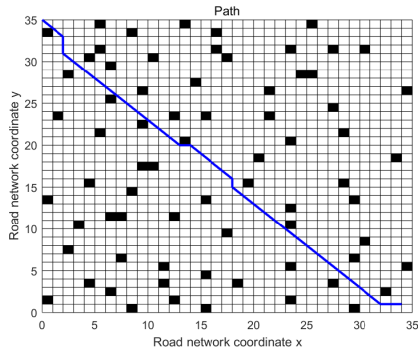


FIGURE 3. Path calculated by OPABRL.

simulate the path planning function of the algorithm from the starting point to the end point in the 35\*35 grid to prove the feasibility of the algorithm, then set random obstacles output as black grids, and the black grid percentage is first set to 5%. The simulation result is shown in figure 3.

As can be seen from the above figure, when no obstacle is set, the algorithm can successfully plan the path from the start point to the end point, and the path is very close to the straight line of the start point and the end point with slightly deviation. It is mainly because of the actual limits of the vehicle. It is almost impossible to drive directly along the straight line of the starting point and the end point. Therefore, in order to shorten the distance as much as possible, the path planned by the algorithm in this design is connected with the straight line of the starting point and the ending point in the trend. The path can avoid driving in areas that cannot be driven directly. It can be seen from the figure that although many obstacles are randomly generated and occupy part of the grid, the algorithm can successfully avoid these areas. Although the number of turns is significantly big, we will verify the advantages of this algorithm compared with other algorithms in the next simulation experiments.

Experiment 2: This part of the simulation experiment will compare OPABRL algorithm with the traditional path planning algorithms including ACO algorithm, GA algorithm, ANNs algorithm and PSO algorithm in the same area between same start point and end point. The convergences of these algorithms are shown in figure 4- figure 8.

In the above experimental results, y-axis is the shortest path length change in the algorithm process, and the x-axis is the number of iterations. It can be seen from the above experimental results that all these algorithms show a convergence trend as the same, although there are fluctuations at the beginning of the calculation, but as the calculation proceeds, the number of random searches becomes less and less, making the shortest path length tend stable, and finally they all achieve convergence. All the algorithms can achieve

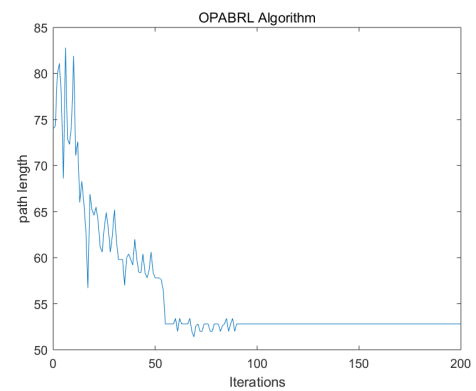


FIGURE 4. Convergence rate of OPABRL.

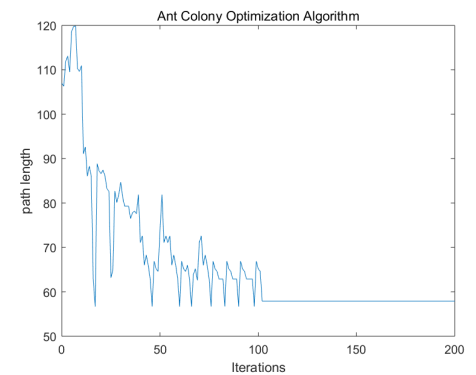


FIGURE 5. Convergence rate of ACO.

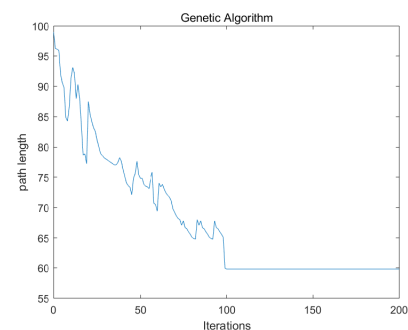


FIGURE 6. Convergence rate of GA.

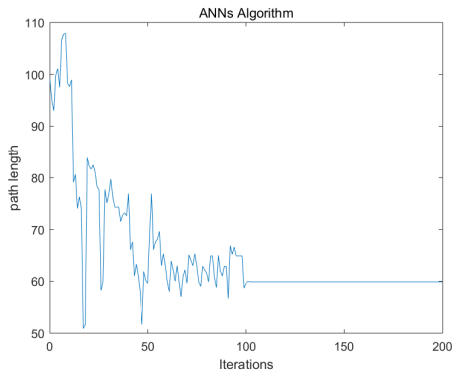


FIGURE 7. Convergence rate of ANNs.

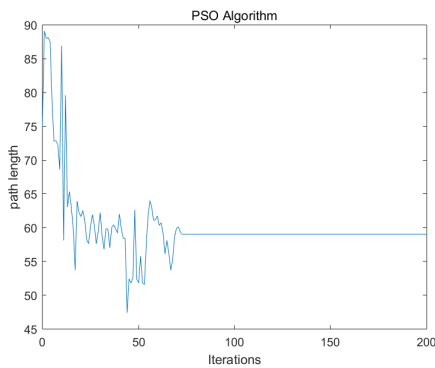


FIGURE 8. Convergence rate of PSO.

convergence after multiple iterations, while the number of iterations to reach convergence is obviously bigger than that of the proposed algorithm except PSO, and the experimental results show that the ACO algorithm has some instability after convergence. Although the OPABRL algorithm proposed in this paper fluctuate obviously in the early stage, it can converge on the shortest path result earlier in the same experimental environment and keep stability well.

In order to better display the experimental results, the results of path planning by different algorithms in the same environment are shown in figure 9:

It can be seen from the above experimental results that compared with traditional algorithms, the OPABRL

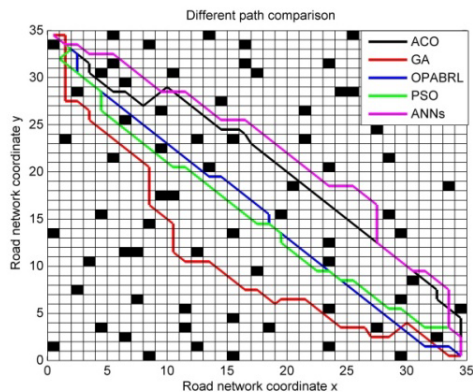


FIGURE 9. Comparison of different paths.

algorithm proposed in this paper can plan a more reasonable path with relatively shorter length and less turns. In order to better show the results of the simulation experiment, we compare the results of the path planning simulation experiments of different algorithms under different scale networks (only the number of nodes and the ratio of obstacles are unchanged). The specific situation is shown in figure 10 and 11.

Figure 10 and 11 show the comparison results of the planned paths of different algorithms under different network scales. Figure 10 shows the average number of turns of planning paths by different algorithms under different network sizes. The x-axis is the number of nodes. At this time, the size of the network is  $N \times N$ , and the proportion of random obstacles in the network is still 0.5, and the displayed data is the average of 20 repeated experiments. It can be clearly seen from the figure that in the multiple simulations of the proposed algorithm, the planned path can significantly reduce the number of driving turns and optimize the path cost.

Figure 11 shows the comparison of the lengths of different algorithm planning path lengths at different network sizes. Similarly, the data displayed is still the average of 20 repeated experiments when different parameters are set. It can be seen from the figure that when the network is small, the difference of the five algorithms is not obvious, while with

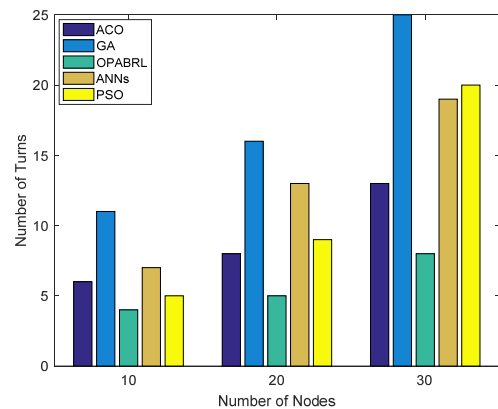


FIGURE 10. Comparison of turns.

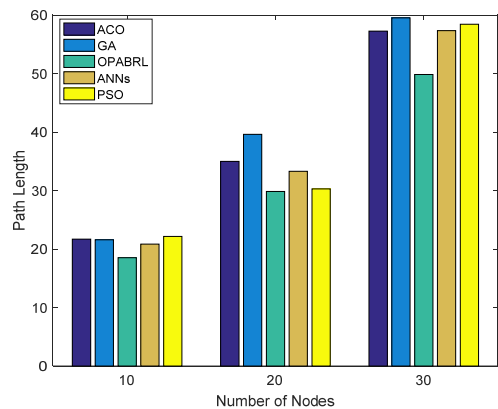


FIGURE 11. Comparison of path length.

TABLE 3. Comparison of different algorithm.

	Best solution probability $P$	Virtual equivalent increase ratio $\varphi$
GA	0.85	0.54
ACO	0.91	0.55
OPABRL	0.98	0.16
ANNs	0.95	0.40
PSO	0.93	0.49

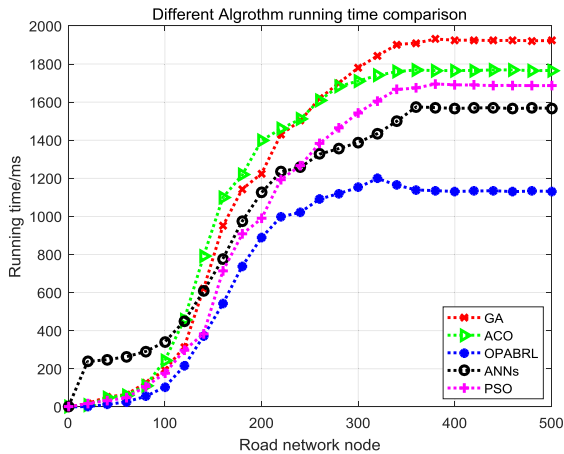


FIGURE 12. Running time of different algorithm.

the increase of network nodes, the advantages of the algorithm are more and more obvious. Obviously, this trend fully demonstrates that the algorithm will show greater advantages in the larger network.

Table 3 shows the statistics of the probability that the average solution is averaged for different algorithms in multiple simulation experiments with a network size of  $35 \times 35$ , and virtual waiting for different algorithms according to equation (7) in Chapter 3. The time increase ratio is compared. It can be seen from the data in the table that the proposed algorithm has a good advantage in finding the optimal solution probability and virtual time increase ratio. Compared with other algorithms, OPABRL algorithm can achieve a smaller virtual time increase ratio and find the optimal solution in most cases.

The experimental results of the running time of different algorithms under different network sizes are shown in figure 12. From the experimental results shown in the above figure, it can be seen that as the number of network nodes increases, the running time of different algorithms show a corresponding growth trend, which is due to the inherent time consumption brought by the topology. Because the number of nodes is small, the network structure is simple initial, and the algorithms show better performance, and the results can be calculated in a shorter time. As the network nodes increase, the calculation time of the algorithms starts to increase, and the increase faster and faster. This is because the increase of network nodes at this time will not only increase the scope of the whole algorithm, but also increase the number of optional next nodes for each node.

Another major reason is that in complex networks, when considering the obstacles in the road network, the algorithm needs to judge whether the added network nodes grid safe or unsafe, which will greatly increase the running time of the algorithm. When the number of nodes increases to a certain number, the algorithm time continues to increase, but the increase speed is obviously slowed down. This is because the algorithm can automatically move the searching direction close to the direction of the line from the start point to the end point in the current network scale, so the search time does not increase as quickly as before. When the network reaches a certain scale, the calculation time of the algorithms will gradually become stable and will not increase with the increase of the number of nodes. This is determined by the characteristics of genetic algorithm and ant colony algorithm. These two algorithms have biological gene characteristics, when the range of search is too large, the influence of each parameter on the calculation result is weakened, and the calculation result will tend to a stable range [32]–[34]. However, when searching area of OPABRL algorithm is increased to a certain extent, since the reinforcement learning algorithm based on prior knowledge is going on, the random operation in the following searching is reduced, and the node is directly judged and planned to be a feasible node. Its calculation is significantly less than the other algorithms. From the experimental results, the running time performance of the algorithm in this paper is obviously better than other algorithms. As the searching progresses, until most of the possible obstacles are classified as prior knowledge by the reinforcement learning algorithm, the OPABRL algorithm also shows good practicality and high efficiency and the calculation time tends to be stable.

### B. TEST OF PRACTICAL SCENES

In general, the solution to the path planning problem can be divided into two types: one is to initialize the map by abstracting the environment into a graph or a grid, and then use the existing search method combined with the abstract map for path planning. The preliminary work of this method is large for it is very difficult to describe and model complex environment. Its advantage is that once the model is established, following path planning will be relatively easy, but the time complexity is high. Moreover, in complex and ever-changing environments, it is no longer applicable, and the significance of the previous modeling work is not worth mentioning compared with the cost paid, so this method is gradually replaced in the current work [35]–[37]. Another type is to use the random sampling method for path planning. It is not required to model the complex environment network map first, but the randomness makes the path generated by it generally very tortuous, and the path smoothing is processed later. It has also been confirmed that it may not be able to find a path in some cases [38], [39].

In addition to the above two methods, with the development of information communication technology and the popularization of the application of reinforcement learning,

the intelligent path planning method based on the continuous interaction between intelligent driving vehicles and the environment has been favored by people [40]. This method can obtain environment information based on the interaction between the intelligent driving vehicle and the environment, and implement obstacle avoidance according to the reinforcement learning algorithm, and then combine the shortest path algorithm to seek the best path.

The urban driving network is complex and variable. The principle of the optimal path planning for simplifying the complex driving route network can be illustrated by the following example. Firstly, preliminary path planning is carried out, and several shortest paths are calculated by the optimized A\* algorithm. By reinforcement learning strategy based on prior knowledge, the vehicle can continuously learn, judge and avoid limit paths, accidents and congestion, and find optional paths from the start point to the end point; Then, according to the requirements, a path with parameters matching the conditions is selected as the final optimal path. At this time, according to the demand, for example, the user selects the path length priority as best path means to select the path with the shortest travel distance among the multiple paths between the set starting point and the destination point.

The following picture shows the interception map of a certain area in Tianjin. We mark the nodes that may be used in the path planning from the Quanyechang street (point A) to Tianjin Modern City (point K), there is a variety of optional paths can be found as follows:

- (1)A-B-E-F-H-I-K;(2)A-C-G-H-I-K;(3)A-C-D-J-I-K;(4)A-B-D-J-I-K;(5)A-B-E-G-J-I-K;(6)A-C-G-J-I-K

According to experience, the Jinzhou section (including E-G-J) is usually a congested section, so paths cover this section will be set as the unsafe area according to the degree of congestion, and will be bypassed in the path planning; Path (6) is not preferred for there are too many turns, and such optimization can be directly achieved through a reinforcement learning strategy based on prior knowledge. The first three paths will be selected as the optimal path according to the reinforcement learning strategy. Then the one with shorter length or less cost path would be selected as the best path according to the user's requirements. The cost can be converted into comprehensive cost according to different requirements. In this paper, the corresponding parameter settings will be given in the algorithm design part. When using this algorithm for path planning, the set parameters can be directly used as the reference parameters for different best path selection.

The application of intelligent driving vehicles is very broad, it is not limited to urban intelligent transportation or transportation and exploration in bad environment and the path planning of unknown environment is the key issue of intelligent driving vehicle research, so we design path planning algorithm to make smart driving vehicles more suitable for actual scenarios. Therefore, after several simulation experiments and analysis of the results, we design practical scene tests on the OPABRL algorithm in the setup

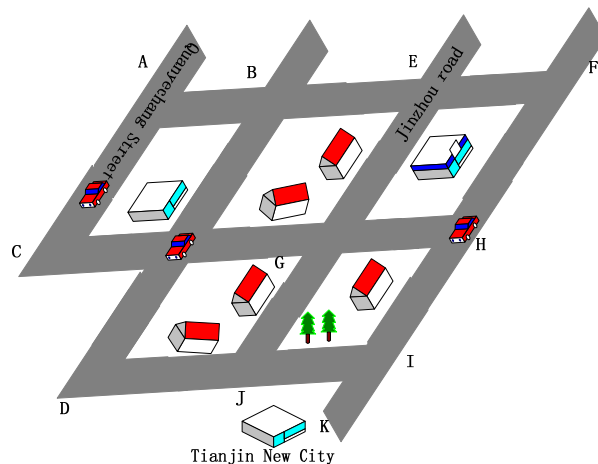


FIGURE 13. Road network example.

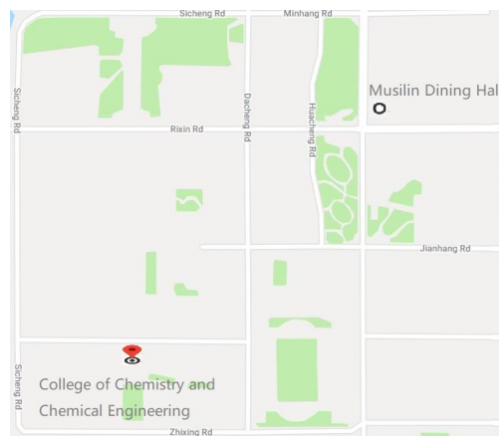


FIGURE 14. Map of testing area.

scenario to verify the practicability and feasibility of the algorithm. Through the tests of path planning, we also prove the advantages of planned path by the proposed algorithm in terms of length, number of turns, running time and time delay.

A random area about 500m × 500m in the campus of Tianjin University of Technology is chosen for our test, all crosses can be turned and two-way driving, all buildings, vehicles and other traffic participants including people walking on the road are obstacles that need to avoid. We tested different algorithms to plan the path from the starting point of the Musilin dining hall to the end point of the Institute of Chemistry and Chemical Engineering in the actual map shown in figure 14.

The path planning test results are shown in figure 15. All the algorithms can reasonably plan the path from the starting point to the end point as marked. The rationality and practicability of the path planned by the proposed algorithm is obviously seen from the results. Although other algorithms can also plan the path from the starting point to the end point, there is a disadvantage in the bypass of the obstacle area (marked black ×). In the test, due to the actual situation, the road structure is not complicated, so there will be overlapping

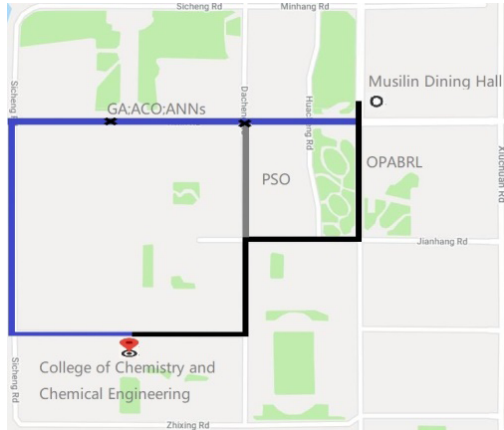


FIGURE 15. Path planning results.

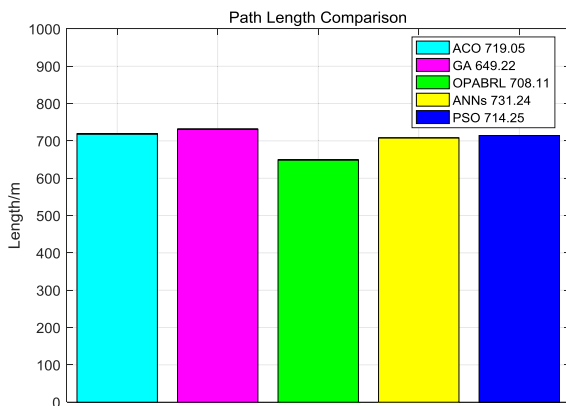


FIGURE 16. Comparison of path length.

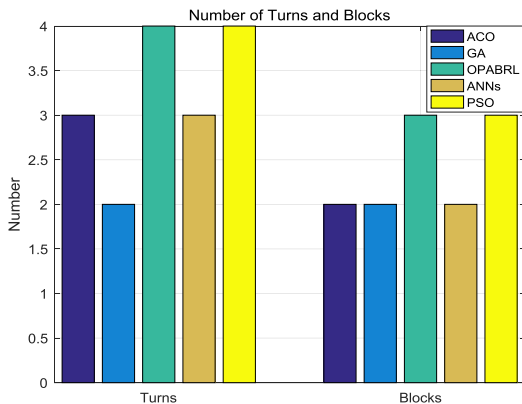


FIGURE 17. Comparison of turns and blocks.

paths. Therefore, we will explain and analyze the situation of paths planned by different algorithms through experimental data.

Figure 16 and 17 show the results of different algorithms in the experiment. Figure 16 shows the length comparison of the planned paths of different algorithms. Figure 17 shows the results of turns and bypass obstacles on the paths by different algorithms.

It can be clearly seen from the results shown in these two figures that the algorithm proposed in this paper has obviously advantages. This algorithm can minimize the

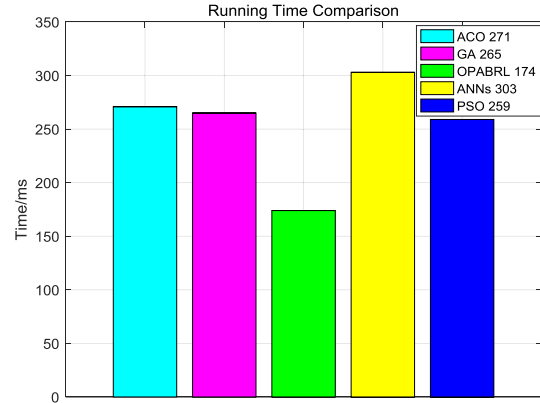


FIGURE 18. Comparison of running time.

TABLE 4. Comparison of test result.

	Planned path length $l/m$	Shortest length $l_i/m$	Planned turns $n$	Virtual equivalent increase ratio $\phi$
GA	731.24	431.13	3	0.86
ACO	719.05	431.13	3	0.83
OPABRL	649.22	431.13	2	0.51
ANNs	708.11	431.13	4	0.98
PSO	714.25	431.13	4	0.99

number of turns even in a simple network, and can plan the best path after avoiding obstacles.

Figure 18 shows the comparison of the running time of different algorithms. Due to the limitation of the environment, these algorithms do not show obvious large gaps in running time, we can still see that the proposed algorithm calculates the best path in a shorter time than other ones from the results.

Table 4 records different algorithms tested in the experiment to calculate the virtual equivalence ratio. The shortest length  $l_i$  is 431.13 meters and minimum turns  $n_i$  is 2. From the table, it can be seen that the proposed algorithm shows good characteristics in the actual test. According to Theorem 1, the proposed algorithm can calculate the best driving path with a small virtual equivalence ratio, which provides a good basis for the practical application of the algorithm. Specifically as shown in Table 4:

## VI. CONCLUSIONS AND FUTURE WORKS

Based on the reinforcement learning algorithm of machine learning, the judgment of prior knowledge is added, combined with the searching-optimized A\* algorithm, we propose a best path planning algorithm that is suitable for traffic jams, accidents and temporary limits for intelligent driving vehicle. The best path determination and virtual equivalence ratio are proposed to convert the cost. Simulations and actual test show that the proposed algorithm has obvious advantages in terms of path length and virtual equivalence ratio. And the algorithm itself has an advantage in terms of running time and search efficiency. In view of the rapid development of intel-

ligent driving vehicle technology, it is of great significance to find a comprehensive cost path planning method based on intelligent decision making.

## REFERENCES

- [1] J.-C. Yang and S.-X. Li, "Research and development of path planning algorithm," *Control Eng.*, vol. 24, no. 7, pp. 1473–1480, 2017.
- [2] D.-G. Zhang, X.-H. Liu, Y.-Y. Cui, and L. C. Zhang, "A kind of novel RSAR protocol for mobile vehicular Ad hoc network," *CCF Trans. Netw.*, vol. 3, no. 2, pp. 111–125, Aug. 2019. doi: [10.1007/s42045-019-00019-5](https://doi.org/10.1007/s42045-019-00019-5).
- [3] T. Zhang, D. Zhang, X. Zhang, J. Qiu, P. Zhao, and C. Gong, "A kind of novel method of power allocation with limited cross-tier interference for CRN," *IEEE Access*, vol. 7, pp. 82571–82583, 2019. doi: [10.1109/ACCESS.2019.2921310](https://doi.org/10.1109/ACCESS.2019.2921310).
- [4] F. Zong, M. Zeng, W. Zhong, and F. Lu, "Hybrid path selection modeling by considering habits and traffic conditions," *IEEE Access*, vol. 7, pp. 43781–43794, 2019. doi: [10.1109/ACCESS.2019.2907725](https://doi.org/10.1109/ACCESS.2019.2907725).
- [5] H. Kim, S.-H. Kim, J. Kim, S. Song, K.-J. Paik, and M. Jeon, "A study on path optimization method of an unmanned surface vehicle under environmental loads using genetic algorithm," *Ocean Eng.*, vol. 142, pp. 616–624, Sep. 2017.
- [6] D.-G. Zhang, Y.-Y. Cui, J.-X. Gao, X.-H. Liu, T. Zhang, and Y.-M. Tang, "Novel reliable routing method for engineering of Internet of vehicles based on graph theory," *Eng. Comput.*, vol. 36, no. 1, pp. 226–247, 2019.
- [7] A. S. Ghiduk, "Automatic generation of basis test paths using variable length genetic algorithm," *Inf. Process. Lett.*, vol. 114, no. 6, pp. 304–316, Jun. 2014.
- [8] D.-G. Zhang, S. Liu, T. Zhang, Y.-Y. Cui, and X.-H. Liu, "Novel dynamic source routing protocol (DSR) based on genetic algorithm-bacterial foraging optimization (GA-BFO)," *Int. J. Commun. Syst.*, vol. 31, no.18, pp. 1–20, 2018. doi: [10.1002/dac.3824](https://doi.org/10.1002/dac.3824).
- [9] D.-G. Zhang, T. Zhang, and Y. Dong, "Novel optimized link state routing protocol based on quantum genetic strategy for mobile learning," *J. Neww. Comput. Appl.*, vol. 2018, no. 122, pp. 37–49, Jul. 2018. doi: [10.1016/j.jnca.2018.07.018](https://doi.org/10.1016/j.jnca.2018.07.018).
- [10] D.-G. Zhang, C. Chen, Y. Y. Cui, and T. Zhang, "New method of energy efficient subcarrier allocation based on evolutionary game theory," *Mobile Neww. Appl.*, vol. 24, no. 10, pp. 1930–1945, Oct. 2019. doi: [10.1007/s11036-018-1123-y](https://doi.org/10.1007/s11036-018-1123-y).
- [11] D.-G. Zhang, T. Zhang, D.-X. Zhao, J.-X. Gao, and X.-H. Liu, "Novel approach of distributed & adaptive trust metrics for MANET," *Wireless Netw.*, vol. 25, no. 6, pp. 3587–3603, Aug. 2019. doi: [10.1007/s11276-019-01955-2](https://doi.org/10.1007/s11276-019-01955-2).
- [12] Z. Y. Ren, "A summary of research on autonomous obstacle path planning for intelligent vehicles," *Softw. Guide*, vol. 16, no. 10, pp. 209–212, 2017.
- [13] D.-G. Zhang, T. Zhang, Y.-Y. Cui, X. Liu, G. Mao, and H. Ge, "New multi-hop clustering algorithm for vehicular Ad hoc networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 4, pp. 1517–1530, Apr. 2019.
- [14] H. Wu, H. Yu, Y. Cheng, and D. X. Zhou, "Distribution majorization of corner points by reinforcement learning for moving object detection," *Proc. SPIE*, vol. 10696, Apr. 2018, Art. no. 106960C.
- [15] D.-G. Zhang, S. Zhou, and Y.-M. Tang, "A low duty cycle efficient MAC protocol based on self-adaption and predictive strategy," *Mobile Neww. Appl.*, vol. 23, no. 4, pp. 828–839, Aug. 2018.
- [16] C. Cheng, Z. Zhu, B. Xin, and C. Chen, "A multi-agent reinforcement learning algorithm based on stackelberg game," in *Proc. IEEE 6th Data Driven Control Learn. Syst. Conf.*, May 2017, pp. 727–732.
- [17] D.-G. Zhang and W. B. Li, "Novel ID-based anti-collision approach for RFID," *Enterprise Inf. Syst.*, vol. 10, no. 7, pp. 771–789, 2016.
- [18] D.-G. Zhang, J. Wang, and X. J. Kang, "A novel image de-noising method based on spherical coordinates system," *EURASIP J. Adv. Signal Process.*, vol. 2012, no. 1, p. 110, Dec. 2012. doi: [10.1186/1687-6180-2012-110](https://doi.org/10.1186/1687-6180-2012-110).
- [19] Y. B. Zheng and N. Li, "Multi-path planning method based on hierarchical reinforcement learning and artificial potential field," *Comput. Appl.*, vol. 35, no. 12, pp. 3491–3496, 2015.
- [20] M. Brezak and I. Petrović, "Path smoothing using clothoids for differential drive mobile robots," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 1133–1138, Jan. 2011.
- [21] D.-G. Zhang, Y.-N. Zhu, C.-P. Zhao, and W.-B. Dai, "A new constructing approach for a weighted topology of wireless sensor networks based on local-world theory for the Internet of Things (IOT)," *Comput. Math. Appl.*, vol. 64, no. 5, pp. 1044–1055, Sep. 2012.
- [22] D.-G. Zhang, "A new approach and system for attentive mobile learning based on seamless migration," *Appl. Intell.*, vol. 36, no. 1, pp. 75–89, Jan. 2012.
- [23] K. Martin, B. Slavomír, and K. Michal, "An optimal path planning problem for heterogeneous multi-vehicle systems," *Int. J. Appl. Math. Comput. Sci.*, vol. 26, no. 2, pp. 297–308, Jun. 2016.
- [24] D.-G. Zhang and X.-D. Zhang, "Design and implementation of embedded un-interruptible power supply system (EUPSS) for Web-based mobile application," *Enterprise Inf. Syst.*, vol. 6, no. 4, pp. 473–489, 2012.
- [25] D.-G. Zhang, C.-P. Zhao, Y.-P. Liang, and Z.-P. Liu, "A new medium access control protocol based on perceived data reliability and spatial correlation in wireless sensor network," *Comput. Elect. Eng.*, vol. 38, no. 3, pp. 694–702, May 2012.
- [26] D.-G. Zhang, Y. Dong, X.-D. Zhang, T. Zhang, and J. Zhang, "A kind of effective data aggregating method based on compressive sensing for wireless sensor network," *EURASIP J. Wireless Commun. Netw.*, vol. 2018, no. 159, p. 159, Dec. 2018. doi: [10.1186/s13638-018-1176-4](https://doi.org/10.1186/s13638-018-1176-4).
- [27] D.-G. Zhang and Y.-P. Liang, "A kind of novel method of service-aware computing for uncertain mobile applications," *Math. Comput. Model.*, vol. 57, nos. 3–4, pp. 344–356, Feb. 2013.
- [28] D.-G. Zhang, H.-L. Niu, and S. Liu, "Novel PEECR-based clustering routing approach," *Soft Comput.*, vol. 21, no. 24, pp. 7313–7323, Dec. 2017.
- [29] Y. W. Liu and H. Q. Wu, "Path planning based on theoretical shortest distance variable weight a algorithm," *Comput. Meas. Control*, vol. 26, no. 4, pp. 175–178, 2018.
- [30] D.-G. Zhang, K. Zheng, D.-X. Zhao, X.-D. Song, and X. Wang, "Novel quick start (QS) method for optimization of TCP," *Wireless Netw.*, vol. 22, no. 1, pp. 211–222, Jan. 2016.
- [31] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proc. IEEE Int. Conf. Robot. Automat.*, Mar. 1985, vol. 3, no. 2, pp. 116–121.
- [32] D.-G. Zhang, S. Liu, Z. Liang, and T. Zhang, "Novel unequal clustering routing protocol considering energy balancing based on network partition & distance for mobile education," *J. Netw. Comput. Appl.*, vol. 88, pp. 1–9, Jun. 2017. doi: [10.1016/j.jnca.2017.03.025](https://doi.org/10.1016/j.jnca.2017.03.025).
- [33] M. P. Aghababa, M. H. Amrollahi, and M. Borjkhani, "Application of GA, PSO, and ACO algorithms to path planning of autonomous underwater vehicles," *J. Marine Sci. Appl.*, vol. 11, no. 3, pp. 378–386, Sep. 2012.
- [34] D.-G. Zhang, X. Wang, X. D. Song, T. Zhang, and Y. N. Zhu, "A new clustering routing method based on PECE for WSN," *EURASIP J. Wireless Commun. Netw.*, vol. 2015, no. 162, pp. 1–13, Dec. 2015. doi: [10.1186/s13638-015-0399-x](https://doi.org/10.1186/s13638-015-0399-x).
- [35] R. Rodríguez-Puente and M. S. Lazo-Cortés, "Algorithm for shortest path search in Geographic information systems by using reduced graphs," *SpringerPlus*, vol. 2, p. 291, Dec. 2013.
- [36] H. Zhao, J. Shao, J. Yang, and W. Zhang, "Reinforcement learning algorithm for path following control of articulated vehicle," *J. Agricult. Machinery*, vol. 48, no. 3, pp. 376–382, Mar. 2017.
- [37] D. Zhang, G. Li, X. Ming, Z.-H. Pan, and K. Zheng, "An energy-balanced routing method based on forward-aware factor for wireless sensor network," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 766–773, Feb. 2014.
- [38] S. Doostie, A. K. Hoshiar, S. Lee, H. Choi, and M. Nazarahari, "Optimal path planning of multiple nanoparticles in continuous environment using a novel adaptive genetic algorithm," *Precis. Eng.*, vol. 53, pp. 65–78, Jul. 2018.
- [39] D.-G. Zhang, X. Wang, X. Song, and D. Zhao, "A novel approach to mapped correlation of ID for RFID anti-collision," *IEEE Trans. Services Comput.*, vol. 7, no. 4, pp. 741–748, Oct. 2014.
- [40] C. Horoba and D. Sudholt, *Running Time Analysis of ACO Systems for Shortest Path Problems*, vol. 6, no. 15. Berlin, Germany: Springer, 2009, pp. 11–17.



**XIAO-HUAN LIU** (M'12) was born in 1988. She is currently pursuing the Ph.D. degree. She is also a Researcher with the Tianjin University of Technology, Tianjin, China. Her research interests include path planing, the IoT, intelligent driving, and mobile computing.



**DE-GAN ZHANG** (M'01) was born in 1970. He received the Ph.D. degree from Northeastern University, China. He is currently a Professor with the Tianjin Key Lab of Intelligent Computing and Novel software Technology, Key Lab of Computer Vision and System, Ministry of Education, Tianjin University of Technology, Tianjin, China. His research interest includes service computing.



**YU-YA CUI** (M'16) was born in 1990. He is currently pursuing the Ph.D. degree. He is also a Researcher with the Tianjin University of Technology, Tianjin, China. His research interests include WSN and mobile computing.



**HAO-RAN YAN** (M'18) was born in 1995. He is currently pursuing the Ph.D. degree. He is also a Researcher with the Tianjin University of Technology, Tianjin, China. His research interests include WSN and mobile computing.



**LU CHEN** (M'17) was born in 1992. She is currently pursuing the Ph.D. degree. She is also a Researcher with the Tianjin University of Technology, Tianjin, China. Her research interests include WSN and industrial application.

...