Adaptive Path-Planning for Autonomous Robots: A UCH-Enhanced Q-Learning Approach

Wei Liu[1,2,3*], Ruiyang Wang[1†], Haonan Wang[4†], Guangwei Liu[5†]

[1]College of Science, Liaoning Technical University, Fuxin, 123000, Liaoning, China.

[2]Institute of Mathematics and Systems Science, Liaoning Technical University, Fuxin, 123000, Liaoning, China.

[3]Institute of Intelligent Engineering and Mathematics, Liaoning Technical University, Fuxin, 123000, Liaoning, China.

[4]Whiting School of Engineering, Johns Hopkins University, Baltimore, 21218, Maryland, USA.

[5]College of Mines, Liaoning Technical University, Fuxin, 123000, Liaoning, China.

*Corresponding author(s). E-mail(s): liuwei@lntu.edu.cn;
Contributing authors: 472321492@stu.lntu.edu.cn; hwang298@jh.edu; liuguangwei@lntu.edu.cn;
†These authors contributed equally to this work.

Abstract

Q-learning methods are widely used in robot path planning but often face challenges of inefficient search and slow convergence. We propose an Improved Q-learning (IQL) framework that enhances standard Q-learning in two significant ways. First, we introduce the Path Adaptive Collaborative Optimization (PACO) algorithm to optimize Q-table initialization, providing better initial estimates and accelerating learning. Second, we incorporate a Utility-Controlled Heuristic (UCH) mechanism with dynamically tuned parameters to optimize the reward function, enhancing the algorithm's accuracy and effectiveness in path-planning tasks. Extensive experiments in three different raster grid environments validate the superior performance of our IQL framework. The results demonstrate that our IQL algorithm outperforms existing methods, including FIQL, PP-QL-based CPP, DFQL, and QMABC algorithms, in terms of path-planning capabilities.

Keywords: Path Planning, PACO algorithm, UCH mechanism, IQL algorithm, Robot

1 Introduction

With the rapid development of the combination of control technology and the Artificial Intelligence(AI) field, the intelligent control of mobile robots and their applications like industrial manufacturing, logistics sorting, etc. in this field is evolving towards self-learning and adaptation [1]. For example, intelligent control of mobile robots in complex environments can autonomously move in various environments without external assistance [2], which requires navigation [3] and motion planning-related technologies in practical applications. Motion planning is divided into path planning and trajectory planning [4]. Path planning often serves as the crucial step of trajectory planning, its goal is to find the optimal path from a starting point to an endpoint in a given environment. However, path planning in dynamic environments is more practical and challenging [5].

In recent years, several path-planning algorithms have been widely adopted in the field of mobile robotics, such as Dijkstra combined with octagonal search to optimize paths [6], Bellman-Ford to cope with fuzzy image environments [7], A-Star based on geometrical optimization [8], RRT combined with CNNs to improve the efficiency [9], the Ant Colony Optimization (ACO)-Artificial Potential Field (APF) fusion algorithm for UUV dynamic path planning [10], Particle Swarm Optimization (PSO) combined with higher-order Bessel curves to achieve smooth paths [11] and improved Genetic Algorithm (GA) to cope with complex maps [12]. In addition, many researchers also use Reinforcement Learning (RL) to address modeling unknowns and accelerate convergence [13], which presents G2RL for large dynamic environments and introduces MAPPER [14], a decentralized evolutionary RL for hybrid dynamics that combines DNNs and MDPs for UAV autonomy [15]. DRL is utilized in DDQN with map data to balance navigation and mission goals [16]. SAC is applied with DRL for robotic arm obstacle avoidance [17]. Two-depth Q-networks are proposed for QoS-driven actions [18]. An optimized TD3 model is offered for UAV energy-efficient paths [19]. DQN-based DRL is used to enhance USV autopilots and collision avoidance [20]. DDPG and priority sampling are leveraged in a COLREGs-based USV avoidance algorithm [21]. MADPG and Gumbel-Softmax are employed for AGV conflict-free paths [22]. DRL is proposed for real-time planning of driverless vehicles in challenging settings [23].

Although the above algorithms have achieved some results, they still face significant challenges such as difficult environment modeling, poor adaptation to complex environments, slow convergence of algorithms, easy falling into local optimums, high consumption of computational resources, and difficulty in parameter tuning. Meanwhile, researchers have extensively applied Q-learning and its enhancements to path planning in RL, achieving notable success. DFQL [24] marries Q-learning with artificial potential fields, effectively tackling UUV path planning in partly known seas. EQL [25] expedites convergence to optimal paths for mobile robots via innovative rewards, enhancing path optimization, efficiency, and safety. However, Q-table initialization [26] crucially impacts performance, prompting Hao Bing et al. [27] to optimize it with FPA for quicker path searches. Heuristic algorithms [28, 29] address complex problems but grapple with non-optimality and complexity, requiring enhancements for efficiency and learning speed. Q-learning faces credit allocation hurdles, local optima traps, dimensionality challenges, and reward function over-optimization [30–34], hindering real-time action assessment, strategy discovery, model performance, and generalization. To tackle these, researchers devised refined reward functions [35], inspired by game theory's utility theory [36] and hierarchical/model-based RL [37]. Meiyan Zhang et al. [38] proposed a Predator-Prey-based reward to boost performance. While advancements have been made, mobile robot path planning in dynamic environments still requires deeper research on adaptability and generalization [39].

Therefore, the motivation of this paper hinges on the shortcomings of traditional Q-learning algorithms, including convergence issues, local optima from under-exploration, slower progress from over-exploration, increased computational demands due to state-space dimensionality, and challenges in learning effective strategies from sparse or poorly designed reward functions.

In this work, we propose a framework for an improved Q-learning algorithm with UCH mechanism and optimized Q-table initialization is proposed to enhance path planning efficiency and selection accuracy.

Our **research contributions** are outlined as follows:

- We developed a novel framework–**IQL**, which enhances performance by dynamically adjusting pheromone volatility, thus avoiding local optima.
- Our framework improves the Q-table initialization process of the Q-learning algorithm using an enhanced ACO algorithm.
- Our framework uses a UCH reward mechanism is introduced that dynamically adjusts reward parameters, reducing exploration dilemmas and enabling more precise reward criterion assessment, thereby enhancing algorithm performance.

The subsequent sections of this document are structured as follows: Section 2 covers RL basics, Q-learning, and env modeling for path planning. Section 3 presents PACO for Q-table init, UCH for reward tuning, IQL algorithm, and evaluation. Section 4 discusses simulations and results. Section 5 summarizes key contributions.

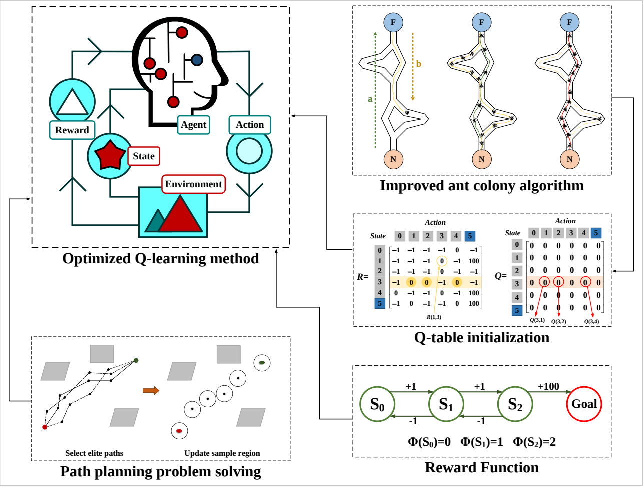


Fig. 1 Framework diagram of the improved Q-learning (IQL) algorithm

2 Background

This section covers the basics of RL and traditional Q-learning, focusing on reward functions and environment modeling essential for the improved Q-learning (IQL) algorithm proposed in this paper.

2.1 Reinforcement Learning

In the standard RL model, its fundamental framework is composed of three core elements: state, action, and reward. Agent [40] acquires the current state s_t of the environment through observation at moment t, and then performs a specific behavior a from all feasible behaviors according to an established strategy and receives a reward r_t for evaluating the merit of the behavior. Subsequently, the intelligent body transfers to a new state s_{t+1} based on the performed behavior and continues the process until the training is complete. The ultimate goal of the intelligent body is to learn a strategy $S \to A$ that results in maximizing the desired cumulative reward r. In this process, the set of states of the environment is denoted as $S = \{s_1, s_2, \ldots, s_{t+1}, \ldots\}$, while all the possible behaviors constitute the set $A = \{a_1, a_2, \ldots, a_n\}$. Each behavior has a corresponding Q-value by which the intelligent body decides its behavioral choices.

2.2 Basic Q-learning algorithms

Q-learning algorithm [41], introduced in 1989, is a model-free reinforcement learning method that guides an agent's actions across various states. As a classic algorithm, it doesn't require model construction and consists of three main components [42]: Q-table initialization, action selection strategy, and Q-table update. Typically, the Q-table is initialized with constant values, and the ϵ-greedy strategy is used for action selection. The updated formula for the value function is shown in equation (1).

$$Q(s,a) = Q(s,a) + \alpha \left(r(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \tag{1}$$

In addition to the three essential components, defining the reward function and collision handling is crucial for the path planning problem. The agent receives a penalty for each step taken, which is related to the length of the path from the previous step. When the agent hits an obstacle, it remains in place and seeks actions to avoid the obstacle until reaching the endpoint. The reward function can be defined as shown in equation (2).

$$r(s,a) = -\sqrt{(x_s - x''_s)^2 + (y_s - y''_s)^2} \tag{2}$$

where s is the current agent state, and x_z, y_z is the coordinates of the current state corresponding to the raster; s'' is the previous state of the agent, and (x''_z, y''_z) is the coordinates of the raster corresponding to the previous state.

Q-learning is widely used in gaming, robotics, and resource management due to its simplicity and effectiveness, providing a solid foundation for reinforcement learning. However, traditional Q-learning faces challenges such as slow iteration speed, difficulty in environment comprehension during Q-table initialization, and a tendency to fall into local optima.

The basic flow of the Q-learning algorithm is depicted in Fig. 2.

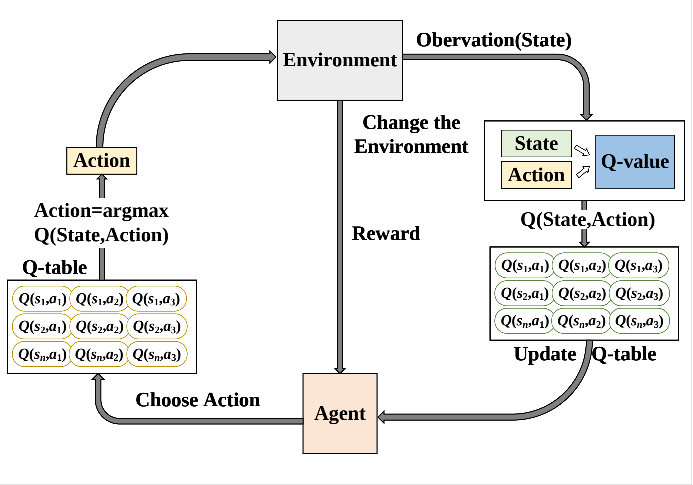
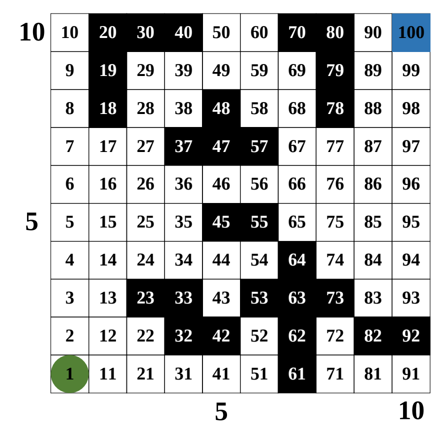

Environment
Obervation(State)
Change the Environment
Reward
State
Action
Q-value
Q(State,Action)
$Q(s_1,a_1)$ $Q(s_1,a_2)$ $Q(s_1,a_3)$
$Q(s_2,a_1)$ $Q(s_2,a_2)$ $Q(s_2,a_3)$
$Q(s_n,a_1)$ $Q(s_n,a_2)$ $Q(s_n,a_3)$
Update Q-table
Action
Action=argmax Q(State,Action)
Q-table
$Q(s_1,a_1)$ $Q(s_1,a_2)$ $Q(s_1,a_3)$
$Q(s_2,a_1)$ $Q(s_2,a_2)$ $Q(s_2,a_3)$
$Q(s_n,a_1)$ $Q(s_n,a_2)$ $Q(s_n,a_3)$
Choose Action
Agent

Fig. 2 Interplay process of the Q-learning algorithm and the surrounding environment

2.3 Environmental modelling for path planning

This paper uses the raster method to model the environment, which includes static obstacles[43]. A grid cell that contains an obstacle is marked as ”1” and represented by a black grid in the simulation drawing, regardless of whether the obstacle completely covers the cell. Conversely, a grid cell that has no obstacle is marked as ”0” and represented by a white grid in the simulation drawing.



10	10	20	30	40	50	60	70	80	90	100
	9	19	29	39	49	59	69	79	89	99
	8	18	28	38	48	58	68	78	88	98
	7	17	27	37	47	57	67	77	87	97
	6	16	26	36	46	56	66	76	86	96
5	5	15	25	35	45	55	65	75	85	95
	4	14	24	34	44	54	64	74	84	94
	3	13	23	33	43	53	63	73	83	93
	2	12	22	32	42	52	62	72	82	92
	1	11	21	31	41	51	61	71	81	91
					5					10

Fig. 3 Raster map environment and serial number encoding

Fig. 3 is used as an example to illustrate the relationship between raster coding and horizontal and vertical coordinates [44]. Firstly, a Cartesian coordinate method is used to number grids: horizontal number of grid is used as the abscissa of grid, and vertical number of grid is used as the ordinate of grid. In Fig. 3, the maximum h of the raster abscissa is 10, the maximum v of the raster ordinate is 10, and the total number of rasters is 100. Grids are numbered $1, 2, \cdots, h \times v$ from bottom to top and left to right. The relationship between the grid coordinates and the grid number is expressed as the following formula. Table 1 shows the symbols and their meanings in (3).

$$\begin{cases} x_c = ceil(c/h); \\ y_c = \begin{cases} \mathrm{mod}(c/h), \mathrm{mod}(c/h) \neq 0; \\ h \qquad\quad , \mathrm{mod}(c/h) = 0. \end{cases} \end{cases} \tag{3}$$

Table 1 Explanation of the formula symbols for the relationship between grid coordinates and grid numbers

Symbols	Symbolic meanings
x_c	Grid abscissa
y_c	Grid ordinates
ceil(x)	Take an integer operation that is not less than x
mod(x,y)	Take the remainder of x/y
c	The current number of the raster

Path planning algorithm, which adopts an obstacle avoidance strategy, ensures that the planned path neither passes through obstacles nor collides with them. Fig. 4, which illustrates the agent's movement from grid c to grid n, shows the path options in 8 directions and the correct detour path.

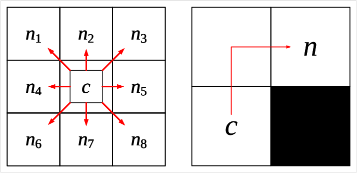


Fig. 4 The selected path and the correct detour path

Fig. 5 illustrates two common error paths, which collide with obstacles and do not meet basic requirements of conditional path constraints.

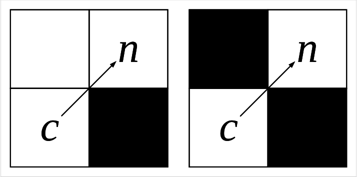


Fig. 5 Two error paths

3 Methods

The IQL algorithm introduced in this paper enhances Q-learning for path planning by addressing the inefficiencies of traditional Q-tables. Leveraging the Path Adaptive Collaborative Optimization (PACO) algorithm for Q-table initialization, IQL enables quicker adaptation and more efficient path selection, improving both solution efficiency and accuracy. ACO was chosen over alternatives like Genetic Algorithms and Particle Swarm Optimization for its superior global optimization, adaptability to dynamic environments, and balanced exploration-exploitation. Additionally, ACO's effectiveness in combinatorial optimization and scalability make it ideal for complex path-planning.

Furthermore, the UCH mechanism is implemented to handle sparse reward functions, reducing exploration risks and enhancing the agent's ability to identify reward criteria. These combined optimizations significantly enhance the IQL algorithm's performance, leading to faster convergence, higher path quality, and improved obstacle avoidance, making it the preferred choice for efficient and adaptable path planning.

Overall, the IQL algorithm, which effectively adapts to the comprehensive needs of path planning efficiency, accuracy, and adaptability, accelerates convergence speed, improves path quality, and strengthens obstacle avoidance.

3.1 Q-table initialization optimization strategy

3.1.1 PACO algorithm

Ant colony optimization (ACO) [45] simulates the foraging behavior of real ants. During foraging, ants discharge a volatile substance called pheromones along their path, using its presence and quantity to guide their movement direction. Generally, ants tend to choose paths with more pheromones, forming a positive feedback mechanism: the pheromones on the optimal path increase, while those on other paths gradually decay over time. The ACO algorithm, which simulates this foraging action [46], is a heuristic optimization algorithm with certain advantages in solving combinatorial optimization problems. However, it also has disadvantages such as slow convergence speed, sensitive parameter settings, and a tendency to fall into local optima, which can affect the algorithm's performance and applicability.

To address these issues, this paper introduces the Path Adaptive Collaborative Optimization (PACO) algorithm to improve ACO algorithm performance. The PACO algorithm is used as the optimization strategy for Q-table initialization, where the best solution obtained by the ACO algorithm serves as the Q-table initialization strategy.

PACO algorithm model is briefly described with the help of n random regions. Let m ants be placed on n random regions, where n is the size of the aggregation point; m denotes ants' number; c is a set of random areas of this problem; $\tau_{ij}(t)$ is the pheromone on the path region of the random region i and j at time t.

Step1: State Transition Guidelines.

Each ant independently selects the next random region to transfer to based on the pheromones in each path and records the random region that ant k has traveled in the $tabu_k$ table. At time t, the likelihood of ant k moving from random region i to random region j, denoted as $p_{ij}^k(t)$, is as (4).

$$p_{ij}^k(t) = \begin{cases} allowed_k = \{C - tabu_k\} \\ \frac{[\tau_{ij}(t)]^\alpha g[\eta_{ik}(t)]^\beta}{\sum\limits_{s \in allowed_k} [\tau_{is}(t)]^\alpha g[\eta_{is}(t)]^\beta} & j \in allowed_k \\ 0 & j \notin allowed_k \end{cases} \tag{4}$$

Among them, α is the information heuristic, reflecting the influence of pheromones on the path region of the ant, and β is the expectation heuristic, indicating the impact of the path area on the ant. $allowed_k$ is the random region set that can be selected when the ant k moves next.

Step2: Pheromone updates.

In the PACO algorithm, the probability of choosing a path is proportional to the pheromone concentration on that path. Over time, pheromone levels decrease according to a defined volatility coefficient. Traditional ant colony algorithms use a fixed value for this coefficient. If set incorrectly, the colony may prematurely converge on certain paths, resulting in local optima, thereby hindering the algorithm's ability to find the global best solution and slowing down convergence.

To address this limitation, the PACO algorithm incorporates a mechanism for dynamically adjusting the pheromone volatility factor. Instead of being a static constant, the pheromone volatility factor ρ decreases gradually with each iteration. This adjustment maintains path diversity and enhances convergence speed, as demonstrated in (5).

$$\rho(t) = \frac{\lambda_1}{(1 + e^{\frac{\lambda t}{3m}})} \tag{5}$$

where λ_1 represents the adjustment coefficient of the pheromone volatilization factor, Nc represents the number of the current iteration, and m is the ant's number.

The pheromone update formula in the PACO algorithm is shown in (6).

$$\begin{cases} \tau_{ij}(t+1) = (1 - \rho(t)) \bullet \tau_{ij}(t) + \Delta\tau_{ij}(t) \\ \Delta\tau_{ij}(t) = \sum\limits_{k=1}^{m} \Delta\tau_{ij}^k(t), \Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{L_k}, (i,j) \in L_k \\ 0 \quad , others \end{cases} \end{cases} \tag{6}$$

Table 2 shows the symbols and their meanings in (6).

Table 2 Symbols of pheromone update formulas in the PACO algorithm

Symbols	Symbolic meanings
$\Delta\tau_{ij}(t)$	The pheromone increment on path (i,j) at time t
k	The k-th ant
m	Total ants number
Q	Initial intensity of pheromone increment (fixed constant)
L_k	The k-th ant searches for the total length of the path at time t

The PACO algorithm draws on the feeding behavior characteristics of ants in nature to form a distributed intelligence system, and the pheromone update mechanism is improved by introducing the volatilization coefficient of the pheromone to avoid the problem of ants falling into a local optimum, as shown in Fig. 6.

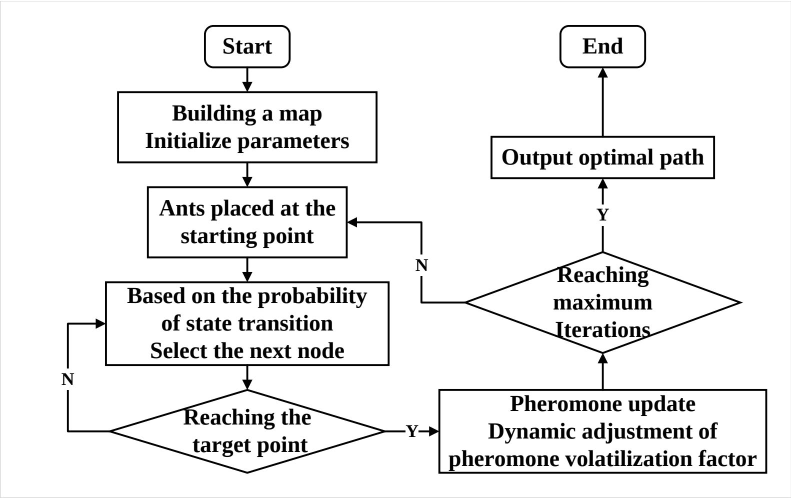
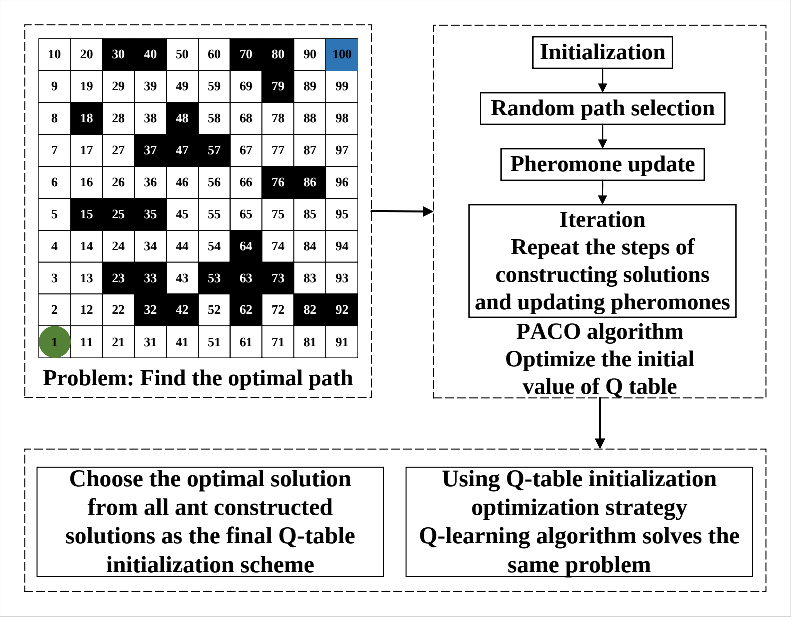

Start
Building a map
Initialize parameters
Ants placed at the starting point
Based on the probability of state transition
Select the next node
N
Reaching the target point
Y
Pheromone update
Dynamic adjustment of pheromone volatilization factor
Reaching maximum Iterations
N
Y
Output optimal path
End

Fig. 6 Flow chart of the PACO algorithm

3.1.2 Q-table initialization operation

Q-table initialization optimization strategy basis of the PACO algorithm involves these steps, which could be summarized to describe the process. Fig. 7 shows the Q-table initialization optimization strategy process based on the PACO algorithm.



10	20	30	40	50	60	70	80	90	100
9	19	29	39	49	59	69	79	89	99
8	18	28	38	48	58	68	78	88	98
7	17	27	37	47	57	67	77	87	97
6	16	26	36	46	56	66	76	86	96
5	15	25	35	45	55	65	75	85	95
4	14	24	34	44	54	64	74	84	94
3	13	23	33	43	53	63	73	83	93
2	12	22	32	42	52	62	72	82	92
1	11	21	31	41	51	61	71	81	91

Problem: Find the optimal path
Initialization
Random path selection
Pheromone update
Iteration
Repeat the steps of constructing solutions and updating pheromones
PACO algorithm
Optimize the initial value of Q table
Choose the optimal solution from all ant constructed solutions as the final Q-table initialization scheme
Using Q-table initialization optimization strategy
Q-learning algorithm solves the same problem

Fig. 7 Initialization optimization policy process of the Q-table

Step1: Problem definition. Identify the problem of finding the optimal path that needs to be solved.

Step2: PACO algorithm parameter settings. Set the parameters of the PACO algorithm, such as the total number of ants, the initial intensity of pheromone increment, the expected heuristic, and the pheromone volatilization rate.

Step3: Colony initialization. Randomly place ants at the start of the map, with each ant representing a potential solution.

Step4: Iterative process. Select the next node of the path based on the state transition probability and determine whether the target point is reached.

Step5: Pheromone updates. The path diversity is ensured by introducing the mechanism of dynamic regulation of pheromone volatilization factors, and the convergence rate is improved.

Step6: Output the result. After the algorithm is terminated, the best solution found is output as the initial value of the Q table.

Step7: Q-table initialization. The initial utility value of the ant taking different actions in each state is the IQL algorithm's initial value. This step is a vital part of the optimization strategy.

Step8: Algorithm performance evaluation. The results are evaluated to evidence the effectiveness of the optimization strategy.

3.2 Reward function optimization

The reward function, a crucial component of Q-learning algorithms, determines the reward an agent receives from interacting with the environment. By optimizing the reward function, the agent can learn an effective strategy more quickly. In this section, we optimize the reward function by introducing a Utility-Controlled Heuristic (UCH) mechanism, which significantly enhances the algorithm's performance.

The UCH mechanism dynamically adjusts the reward parameter and changes the distance calculation method. This mechanism helps the reward function develop an effective strategy faster and in real time. Table 3 uses Euclidean distance as an example to compare the optimized and original reward functions.

Table 3 Comparison between the original reward function and the optimized reward function (distance calculation method: Euclidean distance)

Meaning	Formula
Raw reward function	$r(s,a) = -\sqrt{(x_s - x_s'')^2 + (y_s - y_s'')^2}$
Parameter formulas	$\mu(t) = \mu_0 \cdot \frac{1}{\pi + \pi \cdot e^{-t}}$
Optimized reward function	$r^O(s,a) = -\mu(t)\, r(s,a)$

In this paper, we compare two distance calculation methods with the Euclidean distance method used in the original algorithm.

(1) Chebyshev distance

In mathematics, Chebyshev distance [47], also known as the $L\infty$ norm, is regarded as a measure within a vector space. The distance is determined by calculating the absolute difference between two points in each coordinate dimension and taking the maximum value from these differences.

Chebyshev distance is based on the concept of a consistent norm (or supremum norm) and is classified as an injective metric space.

This paper only discusses the Manhattan distance in a two-dimensional plane. Let the Chebyshev distance between the two points on the plane be $A(x_s, y_s)$ and $B(x''_s, y''_s)$, and the Chebyshev distance of the two points AB as (7).

$$d^Q_{AB} = \max(|x_s - x''_s|, |y_s - y''_s|) \tag{7}$$

Table 4 compares the optimized reward function with the original reward function using the Chebyshev distance as an example.

Table 4 Comparison between the original reward function and the optimized reward function (distance calculation method: Chebyshev distance)

Meaning	Formula
Raw reward function	$r_q(s,a) = -\max(\lvert x_s - x''_s\rvert, \lvert y_s - y''_s\rvert)$
Parameter formulas	$\mu(t) = \mu_0 \cdot \frac{1}{\pi + \pi \cdot e^{-t}}$
Optimized reward function	$r^Q(s,a) = \rho(t) \cdot\ r_q(s,a)$

(2) Manhattan distance

In Manhattan neighborhoods, the driving distance from one intersection to another is not a straight-line distance between two points, and this actual distance is the ”Manhattan distance [48]”. For this reason, Manhattan distance is also known as ”taxi distance” or ”city block distance”.

This paper only discusses the Manhattan distance in a two-dimensional plane. Let the two points on the plane be $A(x_s, y_s)$ and $B(x''_s, y''_s)$, and the Manhattan distance between the two points AB as (8).

$$d^M_{AB} = |x_z - x''_z| + |y_z - y''_z| \tag{8}$$

Table 5 below takes the Manhattan distance as an example to show the comparison between the optimized reward function and the original reward function.

Table 5 Comparison between the original reward function and the optimized reward function (distance calculation method: Manhattan distance)

Meaning	Formula
Raw reward function	$r_m(s,a) = -\max(\lvert x_s - x''_s\rvert, \lvert y_s - y''_s\rvert)$
Parameter formulas	$\mu(t) = \mu_0 \cdot \frac{1}{\pi + \pi \cdot e^{-t}}$
Optimized reward function	$r^M(s,a) = \rho(t) \cdot\ r_m(s,a)$

Fig. 8 below illustrates the difference and connection between the Euclidean distance, the Manhattan distance, and the equivalent Manhattan distance.

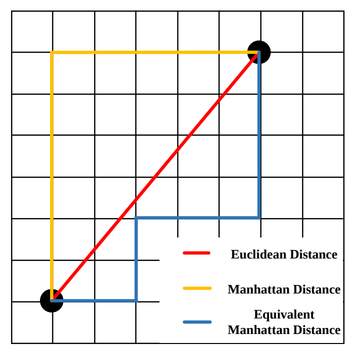


Fig. 8 Euclidean distance, Manhattan distance, and equivalent Manhattan distance

3.3 Evaluation indicators

Algorithm evaluation metrics quantitatively measure performance, including learning speed, stability, and final outcomes. These metrics provide a comprehensive assessment of the algorithm's practical effectiveness. Therefore, three evaluation indicators are used to assess the IQL algorithms.

(1) Trials number with the same number of iterations X. ($\eta(t)$)

With the same number of iterations, the algorithm requires fewer experiments to reach a steady state, indicating improved performance.

(2) The number of trials with a standard deviation of 0. ($d(t)$)

$d(t)$ is a statistic that measures the degree of dispersion in a set of numerical distributions. If the optimized algorithm achieves a standard deviation of 0 with fewer tests, it indicates improved measurement stability and reduced systematic error, and enhanced data reliability.

(3) The expected return after the algorithm's convergence. ($e(t)$)

$e(t)$ refers to the average cumulative return an agent receives from the initial state when following a strategy. The agent can better utilize the learned strategy to maximize cumulative rewards if the algorithm provides a higher $e(t)$ in the initial state.

Therefore, the formula $J(t)$ that defines the performance evaluation index of the experiment in this paper is as (9).

$$J(t) = \frac{\frac{\eta(1)-\eta(2)}{\eta(1)} + \frac{d(1)-d(2)}{d(1)} + \frac{e(1)-e(2)}{e(2)}}{3} \times 100\% \tag{9}$$

Table 6 illustrates the meaning of the symbols in (9). The i's in Table 6 are 1 or 2.

Table 6 Description of the symbol of $J(t)$

Symbols	Symbolic meaning
1	The original algorithm
2	The improved algorithm
$\eta(i)$	Trials number for the algorithm to reach steady state
$d(i)$	The algorithm's trials number with standard deviation to 0
$e(i)$	The algorithm accumulates expected returns

4 Experiments

4.1 Simulation environment

Below are three types of grid environments used in this article.

The first raster map, shown in Fig. 9 (S10), is a low-dimensional, simple environment with a few obstacles, where the robot can quickly find the optimal route between the start and endpoints. This setup is used to test the basic functionality and learning efficiency of the IQL algorithm.

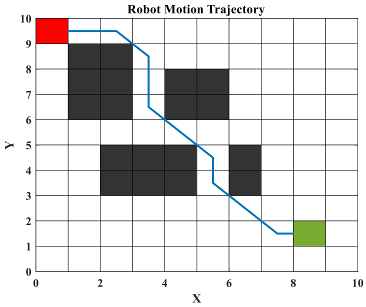


Fig. 9 10*10 raster map environment (S10)

The second environment, i.e., the map of Fig. 10 (S20), is of medium dimensionality and has a more complex structure, including more black grids. This environment allows for the testing of the route planning ability and computational efficiency of the IQL algorithms in the face of more complex situations.

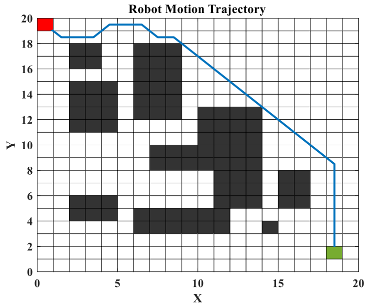


Fig. 10 20*20 raster map environment (S20)

The third environment, the map in Fig. 11 (S30), is a high-dimensional complex map designed with a large number of obstacles. This environment simulates a challenging navigation task in the real world. It is used to estimate the performance of the IQL algorithm in dealing with elaborate environments, especially in planning speed and obstacle avoidance.

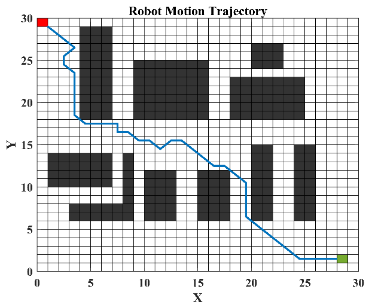


Fig. 11 30*30 raster map environment (S30)

Three MATLAB-designed raster map environments of varying complexity were implemented to evaluate the IQL algorithm's performance in route planning. These environments simulate real-world robot path planning, with each cell representing a potential robot position. Black grids denote obstacles, while white grids represent navigable paths, providing a clear framework for algorithm development and testing.

A series of simulation experiments in three different dimensional raster map environments were conducted to compare the performance of the IQL algorithm with traditional Q-learning and other algorithms. This comprehensive evaluation highlights the practical effects and potential application value of the proposed optimization methodology for path planning problems. Basic parameter settings chosen for this experiment are presented in Table 7.

Table 7 IQL algorithms parameter settings

Name of the parameters involved	Value or range
Learning rate (α)	0.3
Discount factor (γ)	0.95
Q-Table Initializations	4.5598/11.5598/9.3397/7.5598
$\rho_0$in the optimized reward function	0.156/0.0156/0.016/0.01056
Convergence target	0.25
Convergence Iterations Average	10

The algorithm runtime environment and computer configuration in this document are shown in Table 8.

Table 8 The algorithm runtime environment and computer configuration

Parameter	Configuration
Device name	LAPTOP-A3S8EAVD
Processor	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz
With RAM	8.00 GB (7.85 GB available)
System type	64-bit operating system, x64-based processor
Simulation software	MATLAB R2022a

4.2 Algorithm validation

In this section, the following four algorithms are tested in each of the above three environments:

a). Q-learning algorithm.

b). A Q-learning algorithm for optimizing the initial values of Q-tables.

c). Q-learning algorithm with improved reward function.

d). IQL algorithm.

After preliminary experiments and parameter tuning, the IQL algorithm achieved optimal learning and path planning with a Q-table initialized at 9.3397, ρ_0 set to 0.016 in the optimized reward function, and Chebyshev distance for computation. These parameters were used in all subsequent experiments. The following tables (Table 9, Table 10 and Table 11) present the results for a Q-table initialized at 9.3397, ρ_0 set to 0.016, comparing Chebyshev and Manhattan distance calculations.

Table 9 Experimental results of four algorithms in a 10*10 raster map environment

Algorithm name	Distance calculation	$\eta(t)$	$d(t)$	$e(t)$
a)	Euclidean distance	157	147	2543.26
b)	Euclidean distance	143	133	2680.80
c)	Chebyshev distance	137	127	2674.77
c)	Manhattan distance	145	135	2666.29
d)	Chebyshev distance	133	123	2726.95
d)	Manhattan distance	142	132	2686.52

Table 10 Experimental results of four algorithms in a 20*20 raster map environment

Algorithm name	Distance calculation	$\eta(t)$	$d(t)$	$e(t)$
a)	Euclidean distance	486	476	8794.61
b)	Euclidean distance	430	420	9210.92
c)	Chebyshev distance	426	416	9156.23
c)	Manhattan distance	432	422	9150.13
d)	Chebyshev distance	376	366	9594.86
d)	Manhattan distance	408	398	9395.92

Table 11 Experimental results of four algorithms in a 30*30 raster map environment

Algorithm name	Distance calculation	$\eta(t)$	$d(t)$	$e(t)$
a)	Euclidean distance	779	769	14096.71
b)	Euclidean distance	732	722	14526.66
c)	Chebyshev distance	725	715	14712.74
c)	Manhattan distance	729	719	14549.21
d)	Chebyshev distance	707	697	14849.47
d)	Manhattan distance	713	703	14743.75

Overall, the IQL algorithm performance is significantly improved for simple scenes in low dimensions and complex maps in higher dimensions.

In the 10*10 and 20*20 raster maps, the IQL algorithm using Chebyshev distance significantly reduces the trials needed for convergence—by 15.29% and 22.63%, respectively—while also increasing the expected return by 7.23% and 9.11%. The algorithm performs even better in the 20*20 environment, demonstrating its effectiveness in handling larger, more complex problems. In the 30*30 map, although improvements are smaller, the algorithm still reduces trials by 9.24% and increases expected returns by 5.34%. Table 12 shows the optimized algorithm's performance compared to the original algorithm.

Table 12 Performance comparison between the optimized algorithm and the original algorithm

Algorithm	Distance	Evaluation	10*10	20*20	30*30
$d)$	Chebyshev	$\eta(t)$	15.29%	22.63%	9.24%
$d)$	Chebyshev	$d(t)$	16.33%	23.11%	9.36%
$d)$	Chebyshev	$e(t)$	7.23%	9.11%	5.34%
$d)$	Manhattan	$\eta(t)$	9.55%	16.05%	8.47%
$d)$	Manhattan	$d(t)$	10.21%	16.39%	8.58%
$d)$	Manhattan	$e(t)$	5.63%	6.84%	4.59%

Chebyshev distance is chosen over Manhattan distance for its more accurate modeling of movement costs in grid-based environments, where diagonal movements are equally important as horizontal or vertical ones. This choice enhances the efficiency and accuracy of the IQL algorithm across various map dimensions, with observed performance improvements further validating its effectiveness.

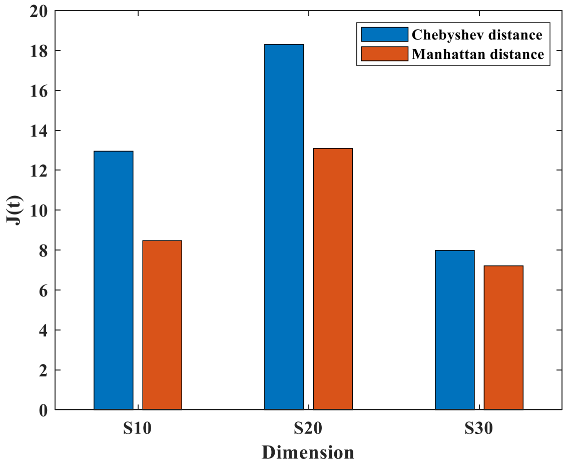


Fig. 12 Comparison of Algorithm Performance Using Different Distance Calculation Methods

4.3 Algorithm comparison

This section highlights the advantages of the IQL algorithm by comparing it with the FIQL, PP-Q-Learning-based CPP (PP-QL-based CPP), DFQL, and QMABC algorithms. The comparison follows these steps.

Step1: Clarify the purpose of the comparison.

This program aims to compare the performance of different algorithms in 10*10 and 20*20 raster environments. By examining these scenarios, we can understand the variation in efficiency and effectiveness of three algorithms across different raster map sizes.

Step2: Selection of benchmarking algorithms.

This section compares the IQL algorithm with the FIQL, PP-QL-based CPP, DFQL, and QMABC algorithms. These comparison algorithms are all variants of Q-learning that use different approaches to enhance learning speed and performance. This comparison demonstrates the superiority of the IQL algorithm.

Step3: Algorithm performance evaluation metrics.

Evaluation metrics are crucial for measuring algorithm performance. We will use three metrics from Section Methods: the number of trials X under consistent iterations ($\eta(t)$), the number of trials where the standard deviation reaches 0 ($d(t)$), and the expected return after convergence ($e(t)$). All algorithms must be run under the same conditions and tested on the same dataset. Additionally, the performance of each metric should be recorded and statistically analyzed to identify the best-performing algorithm for the given scenario.

Step4: Comparative analysis of algorithms.

Table 13 and Table 14 compared the experimental results of the Q-learning algorithm with the IQL algorithm, the FIQL algorithm, the PP-QL-based CPP algorithm, the DFQL algorithm, and the QMABC algorithm.

Table 13 The experimental results of the three algorithms are shown and compared (10*10)

Algorithm name	$\eta(t)$	$d(t)$	$e(t)$
a). Q-learning algorithm	165	155	2543.26
d). IQL algorithm	133	123	2726.95
FIQL algorithm	144	134	2688.18
PP-QL-based CPP algorithm	141	131	2694.95
DFQL algorithm	149	139	2597.78
QMABC algorithm	146	136	2622.12

In the 10*10 and 20*20 raster map environments, the IQL algorithm demonstrated significant performance advantages. For example, in the 10*10 grid, compared to traditional Q-learning, the trials required to reach a steady state with a standard deviation of 0 were reduced by 20.65%, 13.55%, 15.48%, 10.32%, and 12.26% for the IQL, FIQL, PP-QL-based CPP, DFQL, and QMABC algorithms, respectively. In the 20*20 grid, the reductions were 20.11%, 16.81%, 19.75%, 12.39%, and 14.92%, respectively. Among the five algorithms, IQL exhibited the fastest convergence speed and the

Table 14 The experimental results of the three algorithms are shown and compared (20*20)

Algorithm name	$\eta(t)$	$d(t)$	$e(t)$
a) Q-learning algorithm	486	476	8794.61
d) IQL algorithm	376	366	9594.86
FIQL algorithm	406	396	9297.78
PP-QL-based CPP algorithm	392	382	9322.12
DFQL algorithm	427	417	9367.78
QMABC algorithm	415	405	9372.12

highest stability, enabling more effective task learning and execution, and achieving optimal performance.

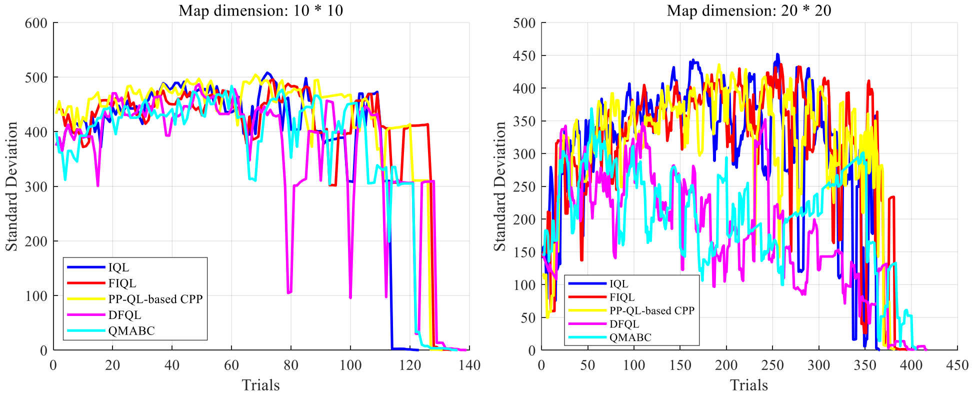


Fig. 13 Comparison of the number of trials in which the standard deviation of the five different algorithms reaches 0 for two types of raster environments

The graphs illustrate the results of experiments in two different raster environments, comparing the FIQL algorithm, the PP-QL-based CPP algorithm, and the improved IQL algorithm. This section focuses on the number of trials where the standard deviation reaches 0, a key stability and convergence speed metric. A smaller standard deviation indicates more stable data; a standard deviation 0 means complete stability. As shown in Fig. 13, the IQL algorithm converges faster than the other two algorithms in both raster environments. This quick convergence is valuable for practical applications as it reduces the time needed for learning and planning. The IQL algorithm not only converges faster but also achieves better final performance.

The IQL algorithm enhances learning efficiency and task performance within the same environment, boosting overall performance, which is crucial for tackling more complex and larger-scale problems. In summary, its performance in a raster environment demonstrates its effectiveness in improving learning efficiency, stability, and outcomes, offering valuable insights for future algorithm design and optimization in similar settings.

4.4 Discussion

Our work presents an optimized algorithm that enhances Q-table initialization and refines reward functions, enabling faster and more efficient strategy development in complex environments by minimizing unnecessary trial and error. This approach shows broad applicability across various raster map environments, consistently improving the IQL algorithm's ability to navigate high-dimensional maps with numerous obstacles. Experimental results confirm the algorithm's stability and reliability, validating its effectiveness in path planning through both theoretical and empirical analysis.

The IQL algorithm shows enhanced performance across all map dimensions. In low-dimensional maps, it efficiently guides the agent to avoid obstacles and find the optimal path, significantly outperforming traditional Q-learning. Even in more complex 30*30 maps, the algorithm reduces trials needed for convergence and increases expected returns, consistently providing better stability, faster convergence, and more effective pathfinding across all tested dimensions.

These findings have significant implications for practical applications, particularly in the operation of autonomous robots or vehicles in complex and dynamic environments. Although the IQL algorithm demonstrates superior stability and convergence speed under certain conditions, further optimization is needed to enhance its adaptability and decision-making speed in highly complex, dynamic environments. Future research should focus on improving the algorithm's robustness in such environments to ensure that autonomous systems can achieve efficient and stable path planning across various real-world scenarios. This will not only increase the practical value of the algorithm but also advance the development of autonomous robotics and vehicle technologies.

5 Conclusion

We propose an Improved Q-Learning (IQL) framework that leverages the Path Adaptive Collaborative Optimization (PACO) algorithm to optimize the initial values of the Q-table. This optimization enables the agent to quickly adapt and select optimal paths, effectively addressing key challenges in path planning within complex environments, such as slow convergence, susceptibility to local optima, and the difficulty of precise environment modeling. Furthermore, we incorporate a Utility-Controlled Heuristic (UCH) reward function mechanism to alleviate reward function sparsity, reduce exploration difficulties, and enhance reward evaluation. Compared to traditional methods, our IQL algorithm significantly improves performance, speed, and accuracy, effectively meeting the demands of path planning and obstacle avoidance. Experimental results indicate that our model demonstrates strong generalization capabilities across various scenarios.

Declarations

- Funding
  This work was supported in part by the National Natural Science Foundation of China under Grant 52374123, in part by the Basic Scientific Research Project of

the Liaoning Provincial Department of Education under Grant LJ212410147013 and Grant LJ212410147019, and in part by Liaoning Revitalization Talents Program under Grant XLYC2211085.

- Conflict of interest/Competing interests (check journal-specific guidelines for which heading to use)
  The authors declare no competing financial interests.
- Author Contributions
  Ruiyang Wang: Propose and design research, Collect and analyze data, Contribute design ideas, and Write the main parts. **Haonan Wang**: Participated in design, Provided technical support, Assisted in data analysis and Discussion. **Wei Liu** and **Guangwei Liu**: Funding acquisition, Project management, Review and Editing of writing.

References

[1] Alatise MB, Hancke GP (2020) A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods. IEEE Access. vol. 8. p. 39830–39846

[2] Rubio F, Valero F, Llopis-Albert C (2019) A review of mobile robots: Concepts, methods, theoretical framework, and applications. International Journal of Advanced Robotic Systems 16(2)

[3] Xiao X, Liu B, Warnell G, Stone P (2022) Motion planning and control for mobile robot navigation using machine learning: a survey. Auton Robot. vol. 46. p. 569–597

[4] Teng S, Hu X, Deng P, Li B, Li Y, et al (2023) Motion Planning for Autonomous Driving: The State of the Art and Future Perspectives. IEEE Transactions on Intelligent Vehicles 8(6): 3692-3711

[5] Chu Z, Wang F, Lei T, Luo C (2023) Path Planning Based on Deep Reinforcement Learning for Autonomous Underwater Vehicles Under Ocean Current Disturbance. IEEE Transactions on Intelligent Vehicles 8(1): 108-120

[6] Sun Y, Fang M, Su Y (2020) AGV Path Planning based on Improved Dijkstra Algorithm. Journal of Physics: Conference Series. vol. 1746. p. 22-23

[7] Parimala M, Broumi S, Prakash K, Topal S (2021) Bellman–Ford algorithm for solving shortest path problem of a network under picture fuzzy environment. Complex & Intelligent Systems. vol. 7. p. 2373–2381

[8] Tang G, Tang C, Claramunt C, Hu X, Zhou P (2021) Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment. IEEE Access. vol. 9. p. 59196-59210

[9] Wang J, Chi W, Li C, Wang C, Meng Q. H. (2020) Neural RRT*: Learning-Based Optimal Path Planning. IEEE Transactions on Automation Science and

Engineering 17(4): 1748-1758

[10] Chen Y, Bai G, Zhan Y, Hu X, Liu J (2021) Path Planning and Obstacle Avoiding of the USV Based on Improved ACO-APF Hybrid Algorithm With Adaptive Early-Warning. IEEE Access. vol. 9. p. 40728-40742

[11] Song B, Wang Z, Zou L (2021) An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve. Applied Soft Computing 100:106960

[12] Li Y, Dong D, Guo X (2020) Mobile Robot Path Planning based on Improved Genetic Algorithm With A-star Heuristic Method. In: 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC). p. 1306-1311

[13] Wang B, Liu Z, Li Q, Prorok A (2020) Mobile Robot Path Planning in Dynamic Environments Through Globally Guided Reinforcement Learning. IEEE Robotics and Automation Letters 5(4): 6932-6939

[14] Liu Z, Chen B, Zhou H, Koushik G, Hebert M, Zhao D (2020) MAPPER: Multi-Agent Path Planning with Evolutionary Reinforcement Learning in Mixed Dynamic Environments. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). p. 11748-11754

[15] He L, Aouf N, Song B (2021) Explainable Deep Reinforcement Learning for UAV autonomous path planning. Aerospace Science and Technology 118:107052

[16] Theile M, Bayerlein H, Nai R, Gesbert D, Caccamo M (2021) UAV Path Planning using Global and Local Map Information with Deep Reinforcement Learning. In:2021 20th International Conference on Advanced Robotics (ICAR). p. 539-546

[17] Chen P, Pei J, Lu W, Li M (2022) A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance. Neurocomputing. vol. 497. p. 64-75

[18] Liu Q, Shi L, Sun L, Li J, Ding M, Shu F (2020) Path Planning for UAV-Mounted Mobile Edge Computing With Deep Reinforcement Learning. IEEE Transactions on Vehicular Technology 69(5): 5723-5728

[19] Hong D, Lee S, Cho YH, Baek D, Kim J, Chang N (2021) Energy-Efficient Online Path Planning of Multiple Drones Using Reinforcement Learning. IEEE Transactions on Vehicular Technology 70(10): 9725-9740

[20] Li L, Wu D, Huang Y, Yuan Z (2021) A path planning strategy unified with a COLREGS collision avoidance function based on deep reinforcement learning and artificial potential field. Applied Ocean Research 113:102759

[21] Xu X, Cai P, Ahmed Z, Yellapu VS, Zhang W (2022) Path planning and dynamic collision avoidance algorithm under COLREGs via deep reinforcement learning. Neurocomputing. vol. 468. p. 181-197

[22] Hu H, Yang X, Xiao S, Wang F (2023) Anti-Conflict AGV Path Planning in Automated Container Terminals Based on Multi-Agent Reinforcement Learning. International Journal of Production Research 61(1): 65–80

[23] Josef S, Degani A (2020) Deep Reinforcement Learning for Safe Local Planning of a Ground Vehicle in Unknown Rough Terrain. IEEE Robotics and Automation Letters 5(4): 6748-6755

[24] Hao B, Du H, Yan Z (2023) A path planning approach for unmanned surface vehicles based on dynamic and fast Q-learning. Ocean Engineering 270:113632

[25] Maoudj A, Hentout A (2020) Optimal path planning approach based on Q-learning algorithm for mobile robots. Applied Soft Computing. vol. 97. Part A. no. 106796

[26] Soong LE, Pauline O, Chun CK (2019) Solving the optimal path planning of a mobile robot using improved Q-learning. Robotics and Autonomous Systems. vol. 115. p. 143-161

[27] Hao B, Zhao J, Du H, Wang Q, Yuan Q, Zhao S (2023) A search and rescue robot search method based on flower pollination algorithm and Q-learning fusion algorithm. PLOS ONE 18(3):e0283751

[28] Ni X, Hu W, Fan Q, Cui Y, Qi C (2024) A Q-learning based multi-strategy integrated artificial bee colony algorithm with application in unmanned vehicle path planning. Expert Systems with Applications 236:121303

[29] Das PK, Behera HS, Panigrahi BK (2016) Intelligent-based multi-robot path planning inspired by improved classical Q-learning and improved particle swarm optimization with perturbed velocity. Engineering Science and Technology, an International Journal 19(1):651-669

[30] Low ES, Ong P, Low CY (2023) A modified Q-learning path planning approach using distortion concept and optimization in dynamic environment for autonomous mobile robot. Computers Industrial Engineering 181:109338

[31] Puente-Castro A, Rivero D, Pedrosa E, Pereira A, Lau N, Fernandez-Blanco E (2024) Q-Learning based system for Path Planning with Unmanned Aerial Vehicles swarms in obstacle environments. Expert Systems with Applications 235:121240

[32] Yu X, Luo W (2023) Reinforcement learning-based multi-strategy cuckoo search algorithm for 3D UAV path planning. Expert Systems with Applications 223:119910

[33] Kulathunga G (2022) A Reinforcement Learning based Path Planning Approach in 3D Environment. Procedia Computer Science. vol. 212. p. 152-160

[34] Lee GT, Kim KJ, Jang J (2023) Real-time path planning of controllable UAV by subgoals using goal-conditioned reinforcement learning. Applied Soft Computing 146:110660

[35] Petrik J, Bambach M (2023) Reinforcement learning and optimization based path planning for thin-walled structures in wire arc additive manufacturing. Journal of Manufacturing Processes. vol. 93. p. 75-89

[36] Zhao M, Lu H, Yang S, Guo Y, Guo F (2021) A fast robot path planning algorithm based on bidirectional associative learning. Computers Industrial Engineering 155:107173

[37] Ladosz P, Weng L, Kim M, Oh H (2022) Exploration in deep reinforcement learning: A survey. Information Fusion. vol. 85. p. 1-22

[38] Zhang M, Cai W, Pang L (2023) Predator-Prey Reward Based Q-Learning Coverage Path Planning for Mobile Robot. IEEE Access. vol. 11. p. 29673-29683

[39] Ding Z, Huang Y, Yuan H, Dong H (2020) Introduction to Reinforcement Learning. Deep Reinforcement Learning. Springer. p. 47-123

[40] Li W, Wang J, Li L, Peng Q, Huang W, Chen X, Li S (2021) Secure and Reliable Downlink Transmission for Energy-Efficient User-Centric Ultra-Dense Networks: An Accelerated DRL Approach. IEEE Transactions on Vehicular Technology 70(9):8978-8992

[41] Tan T, Xie H, Feng L (2024) Q-learning with heterogeneous update strategy. Information Sciences 656:119902

[42] Karimpanal TG, Le H, Abdolshah M, Rana S, Gupta S, Tran T, Venkatesh S (2023) Balanced Q-learning: Combining the influence of optimistic and pessimistic targets. Artificial Intelligence 325:104021

[43] Li X, Liang X, Wang X, Wang R, Shu L, Xu W (2023) Deep reinforcement learning for optimal rescue path planning in uncertain and complex urban pluvial flood scenarios. Applied Soft Computing 144:110543

[44] Low ES, Ong P, Low CY (2023) An empirical evaluation of Q-learning in autonomous mobile robots in static and dynamic environments using simulation. Decision Analytics Journal 8:100314

[45] García M, López N, Rodríguez I (2024) A full process algebraic representation of Ant Colony Optimization. Information Sciences 658:120025

[46] Tadaros M, Kyriakakis NA (2024) A Hybrid Clustered Ant Colony Optimization Approach for the Hierarchical Multi-Switch Multi-Echelon Vehicle Routing Problem with Service Times. Computers & Industrial Engineering 190:110040

[47] Li T, Zuo E, Chen C, Zhong J, Yan J, Lv X (2024) Gaussian distribution resampling via Chebyshev distance for food computing. Applied Soft Computing 150:111103

[48] Wang C, Yang J, Zhang B (2024) A fault diagnosis method using improved prototypical network and weighting similarity-Manhattan distance with insufficient noisy data. Measurement 226:114171

Wei Liu received a B.S. in Mathematics Education from Shenyang Normal University (2000), an M.S. in Computer Application Technology (2008), and a Ph.D. in Mining Engineering (2019), both from Liaoning Technical University. He is currently an Associate Professor and Deputy Director at the Institute of Mathematics and System Science at Liaoning Technical University. A member of several professional societies, his research focuses on machine learning, deep neural networks, and intelligent dispatching in mining. Liu has published over 30 articles in journals like the Journal of Computer Science and has participated in numerous national and provincial projects, leading four enterprise initiatives. He has received multiple awards, including several from the China Coal Industry Science and Technology Awards.

Ruiyang Wang received a B.S. degree in information and computing science from Liaoning Technical University, Liaoning, China, in 2023. She is working toward an M.S. in mathematics major with Liaoning Technical University, Liaoning, China. Her research interests include path planning, deep learning, neural networks, and deep reinforcement learning.

Haonan Wang received a B.S. degree in information and computing Science from Liaoning Technical University, Fuxin, Liaoning, China. He is currently studying M.S. degree in Computer Science at Johns Hopkins University, Maryland, USA. His research interests are the intersection of Human-computer interaction and Machine Learning/ Natural Language Processing, especially Large Language Models and Agents for Human-centered evaluation in interaction systems.

Guangwei Liu is now a professor at the College of Mines, Liaoning Technical University, Liaoning, China. He is a part-time National Engineering Research Center for Computer Software professor. He was awarded the National Coal Youth Science and Technology Prize, and May 4th Youth Medal, and selected as one of the "One Hundred Million Talents" projects in Liaoning Province. His research interests include intelligent mining in open pit mines, system optimization, and digital mine technology. He has presided over many national and provincial key projects and won 4 first prizes and 6-second prizes of provincial and ministerial level science and technology awards. He has published more than 30 papers, 4 authorized patents, and 12 software copyrights.