Complex & Intelligent Systems (2023) 9:4287–4300
https://doi.org/10.1007/s40747-022-00948-7

ORIGINAL ARTICLE

DM-DQN: Dueling Munchausen deep Q network for robot path planning

Yuwan Gu[1,2] · Zhitao Zhu[1,2] · Jidong Lv[1,2] · Lin Shi[1,2] · Zhenjie Hou[1,2] · Shoukun Xu[1,2]

Received: 1 April 2022 / Accepted: 3 December 2022 / Published online: 30 December 2022

Abstract

In order to achieve collision-free path planning in complex environment, Munchausen deep Q-learning network (M-DQN) is applied to mobile robot to learn the best decision. On the basis of Soft-DQN, M-DQN adds the scaled log-policy to the immediate reward. The method allows agent to do more exploration. However, the M-DQN algorithm has the problem of slow convergence. A new and improved M-DQN algorithm (DM-DQN) is proposed in the paper to address the problem. First, its network structure was improved on the basis of M-DQN by decomposing the network structure into a value function and an advantage function, thus decoupling action selection and action evaluation and speeding up its convergence, giving it better generalization performance and enabling it to learn the best decision faster. Second, to address the problem of the robot's trajectory being too close to the edge of the obstacle, a method of using an artificial potential field to set a reward function is proposed to drive the robot's trajectory away from the vicinity of the obstacle. The result of simulation experiment shows that the method learns more efficiently and converges faster than DQN, Dueling DQN and M-DQN in both static and dynamic environments, and is able to plan collision-free paths away from obstacles.

Keywords Deep reinforcement learning · DM-DQN · Path planning · Dueling network

Introduction

With the development trend of artificial intelligence, the robot industry is also developing towards the intelligent direction of self-learning and self-exploration [1]. The path planning of mobile robot is the core problem of robot motion, and its aim is to find an optimal or suboptimal path without collision from the starting point to the ending point. With the development of science and technology, robots face more and more complex environment, and in the unknown environment, we cannot get the information of the whole environment. Therefore, the traditional path planning algorithm cannot meet the needs of people, such as artificial potential field algorithm [2, 3], ant colony algorithm [4], genetic algorithm [5], and particle swarm algorithm [6].

For the problem, deep reinforcement learning (DRL) is proposed [7, 8]. DRL combines deep learning (DL) [9] with reinforcement learning (RL) [10]. Deep learning focuses on the extraction of features from the input unknown environmental states by means of neural network to achieve a fit between the environmental states and the action value function. Reinforcement learning then completes the decision based on the output of the deep neural network and the exploration strategy, thus enabling the mapping of states to actions. The combination of deep learning and reinforcement learning solves the dimensional catastrophe problem posed by state-to-action mapping [11] and better meet the needs of robot movement in complex environment.

✉ Shoukun Xu
xsk@cczu.edu.cn

Yuwan Gu
guyuwan@cczu.edu.cn

Zhitao Zhu
zhuzhitao1999@163.com

Jidong Lv
ljd@cczu.edu.cn

Lin Shi
slcczu@cczu.edu.cn

Zhenjie Hou
houzj@cczu.edu.cn

1 School of Computer Science and Artificial Intelligence, Changzhou University, Changzhou 213164, China

2 Digital Twin Technology Engineering Research Center for Key Equipment of Petrochemical Process, Changzhou University, Changzhou 213164, China

Mnih et al. [12] proposed deep Q-learning Network (DQN) in GoogleDeepMind, a method that combines deep neural network with Q-learning in reinforcement learning [13] and validates its superiority on Atari 2600. Wang et al. [14] divided the entire network structure into two parts, one for the state value function and one for the dominance function, by altering the structure of the neural network in DQN. The improvement can significantly improve the learning effect and accelerate the convergence. Haarnoja et al. [15] introduced maximum entropy into reinforcement learning. The introduction of entropy regularization makes the strategy more random and so adds more exploration, which can speed up subsequent learning. Vieillard et al. [16] add scaled log-policy to the immediate reward, based on maximum entropy reinforcement learning, to maximize the entropy of the expected payoff and the resulting strategy, the algorithm (M-DQN) is also the first to outperform distributed reinforcement learning [17] without the use of distributed reinforcement learning.

Some progress has been made in applying deep reinforcement learning to path planning for agent. Dong et al. [18] combined Double DQN and average DQN to train the network parameters to reduce the problem of overestimation [19] of robot action selection. Huang et al. [20] solved the problem that relative motion between a moving obstacle and a robot may lead to anomalous rewards by modifying the reward function and validating it on the DQN and Dueling DQN algorithms. Lou et al. [21] combined a deep reinforcement learning approach with DQN and prior knowledge to reduce training time and improve generalization. Yan et al. [22] used a long short-term memory (LSTM) network and combined it with Double DQN to enhance the unmanned ground vehicle's ability to remember its environment. Yan et al. [23] used a combined prioritized experience replay (PER) and Double DQN algorithm to solve the UAV trajectory planning problem with global situational information by combining an epsilon greedy strategy with heuristic search rules to select actions. Hu et al. [24] presented a novel method called covariance matrix adaptation-evolution strategies (CMA-ES) for learning complex and high-dimensional motor skills to improve the safety and adaptiveness of robots in performing complex movement tasks. Hu et al. [25] proposed a learning scheme with nonlinear model predictive control (NMPC) for the problem of mobile robot path tracking.

In summary, an improved M-DQN algorithm (DM-DQN) is proposed in the paper, the method introduces maximum entropy and implicitly exploits the Kullback–Leibler divergence between successive strategies, thus outperforming distributed reinforcement learning algorithm. In addition, due to the introduction of the competing network structure, the convergence speed is significantly improved compared to that of M-DQN. By designing a reward function approach based on an artificial potential field, the robot's path planning is kept away from the vicinity of the obstacle. Moving obstacle in unknown environments can be a huge challenge for mobile robot as they can negatively affect the range of the sensors. Therefore, not only the path planning problem in the static obstacle environment is studied, but also the path planning problem in the dynamic and static obstacle environment is considered. Finally, the DM-DQN algorithm is applied to mobile robot path planning and is compared with the DQN, Dueling DQN and M-DQN algorithms.

A summary of the key contributions of the paper are as follows:

- A virtual simulation environment has been constructed using the Gazebo physical simulation platform, replacing the traditional raster map. The physical simulation platform is a simplified model of the real world that is closer to the real environment than a raster map, reducing the gap between the virtual and real environment and reflecting whether the strategies learned by the agent will ultimately be of value to the real robot problem.
- The network structure of the M-DQN is decomposed into a value function and an advantage function, thus decoupling action selection and action evaluation, so that the state no longer depends entirely on the value of the action to make a judgment, allowing for separate value prediction. By removing the influence of state on decision making, the nuances between actions are brought out more, allowing for faster convergence and better generalization of the model.
- The negative impact of obstacle is considered and an artificial potential field is used to set up a reward function to balance obstacle avoidance and approach to the target, allowing the robot to plan a path away from the vicinity of the obstacle.

The structure of the paper as follows: "Theoretical background" introduces the mobile robot model; "Proposed algorithm" introduces the proposed DM-DQN algorithm in detail; "Materials and methods" describes the simulation environment and performs an experimental comparison; and "Experiments and results" concludes the paper.

Theoretical background

M-DQN

Reinforcement learning is the use of Markov decision process (MDP) [26] to simplify modeling, and the Markov decision process can be represented as a tuple $M = \{, , , r, \gamma\}$, where denotes the state, denotes the action, denotes the state transfer

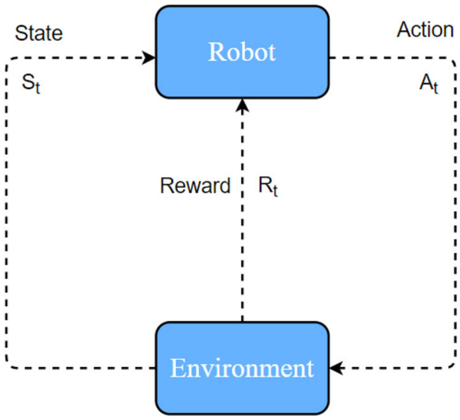


Fig. 1 The process of reinforcement learning

matrix, r denotes the reward function, and γ denotes the discount factor. The process of reinforcement learning is shown in Fig. 1; the whole process includes environment, agent, state, action and reward.

In the classical Q-learning algorithm, the iterations of the q-function can be expressed by the following formula:

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \eta\left(r_t + \gamma \max_{a'} q_*\left(s_{t+1}, a'\right) - q(s_t, a_t)\right) \tag{1}$$

where s_t denotes the state at time t, a_t denotes the action at time t, η denotes the ratio coefficient, r_t denotes the reward at time t, s_{t+1} denotes the state at time $t + 1$, a' denotes the next action and γ denotes the discount factor. However, q_* is unknown in practice, so the value function q_t of the current strategy can only be used to replace the value function q_* of the optimal strategy, the process often referred to as bootstrapping. In short, it is leading itself to be updated to q_{t+1} by the current value of itself, q_t.

In M-DQN, a guiding signal, "log-policy", which is different from q_t, is proposed, that is, the probability value of the policy is taken as log. Since there is an argmax operation in Q-learning, all optimal strategies are determined, so the probability is 1 for each optimal strategy and 0 for all other non-optimal strategies. After taking log for the strategies, the probability of the optimal strategy becomes zero, while the probability of the remaining non-optimal strategies becomes negative infinity. This is certainly a stronger signal for strategy choice, as it suppresses all non-optimal strategies. In addition, by adding this signal value to the immediate reward, the learning process of reinforcement learning can be simplified. Although the optimal action is 0, the rest of the actions are negative infinity, so the choice of the optimal action does not change. Then, in M-DQN, the immediate reward becomes: $r_t + \alpha \ln\pi(a_t|s_t)$.

However, the value of $\ln\pi(a_t|s_t)$ is not computable in Q-learning, so the same maximum entropy strategy as in the Soft-AC algorithm is introduced in DQN, which becomes Soft-DQN. In Soft-DQN, not only the return value of the environment is maximized, but also the entropy of the strategy, and the regression objective of Soft-DQN is expressed as

$$\widehat{q}_{\text{Soft-DQN}}(r_t, s_{t+1}) = r_t + \gamma \sum_{a' \in \mathcal{A}} \pi_{\overline{\theta}}\left(a'|s_{t+1}\right) \left(q_{\overline{\theta}}(s_{t+1}, a') - \tau \ln\pi_{\overline{\theta}}(a'|s_{t+1})\right) \tag{2}$$

where s denotes the state, a denotes the action, r denotes the reward value, and γ denotes the discount factor. $\pi_{\overline{\theta}}$ satisfies $\pi_{\overline{\theta}} = \mathrm{sm}(q_{\overline{\theta}}/\tau)$, τ is the temperature parameter, which is used to control the weight of entropy, a' denotes the action at moment $t + 1$, and $\mathcal{A}$ is the action available. Since the policy chosen for Soft-DQN is softmax, which is different from the deterministic policy of argmax in Q-learning, the policy of Soft-DQN is random and it is possible to calculate the "log-policy" guiding signal in M-DQN. Therefore, M-DQN makes some simple modifications to Soft-DQN, which replaces r_t in Eq. (2) with $r_t + \alpha\tau\ln\pi_{\overline{\theta}}(a_t|s_t)$, i.e.,

$$\widehat{q}_{\text{M-DQN}}(r_t, s_{t+1}) = r_t + \alpha\tau ln\pi_{\overline{\theta}}(a_t|s_t) + \gamma \sum_{a' \in \mathcal{A}} \pi_{\overline{\theta}}(a'|s_{t+1})\left(q_{\overline{\theta}}(s_{t+1}, a') - \tau\ln\pi_{\overline{\theta}}(a'|s_{t+1})\right) \tag{3}$$

where $\pi_{\overline{\theta}} = \mathrm{sm}(q_{\overline{\theta}}/\tau)$, retrieved by setting $\alpha = 0$. M-DQN not only maximizes the environmental reward while selecting a strategy each time, but also minimizes the Kullback–Leibler divergence [27] of the old and new strategies, which is consistent with the ideas of TRPO [28] and MPO [29]. Minimizing the Kullback–Leibler divergence of the old and new policies can lead to an improvement in M-DQN performance, mainly due to the following two aspects:

- As with TRPO, using the distribution of the old strategy to estimate the distribution of the new strategy only when the Kullback–Leibler divergence of the old and new strategies are close does not lead to excessive errors between the old and new strategies. The "log-policy" guiding signal used by M-DQN dynamically limits the error caused by the large difference between the old and new policies, precisely because it implicitly exploits the Kullback–Leibler divergence.
- The problem of overestimation in DQN is described in Double DQN [30], and the impact of the overestimation problem on the performance of the algorithm is demonstrated, and solving the overestimation problem can lead to performance improvements. By the limitation of the Kullback–Leibler divergence in M-DQN, large Q values

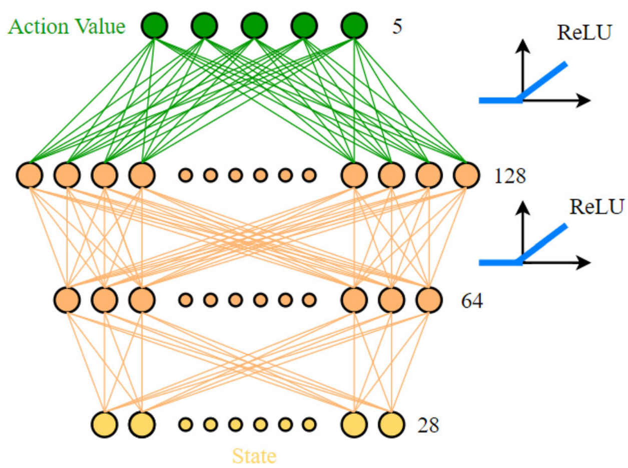


Fig. 2 The structure of M-DQN network

will be suppressed, thus reducing the negative effects of overestimation of Q values.

The M-DQN builds two neural networks, like the DQN, and they have exactly the same network structure, but the Q network is updated every iteration, while the target Q network is only updated every fixed C iterations. The target Q network is used instead of the Q network in the calculation of the target value to reduce the correlation between the target and current values. The structure of the Q network and target Q network of the M-DQN is shown in Fig. 2. This network has four layers: input states; output action values; and two hidden layers of 64 and 128, respectively.

DM-DQN

In the structure of M-DQN network, each time the Q value is updated, only the value corresponding to one of the actions is updated, while the values corresponding to the other actions remain unchanged, which leads to its inefficient updating. The competitive network structure used in DM-DQN updates the values of all other actions when the Q value is updated once. This more frequent updating of values allows for better estimation of state values and the better the competitive network structure performs when the number of action values is higher.

The structure of M-DQN network is divided into two parts, as shown in Fig. 3. The first part is only related to the state S and is called the value function, denoted as $V(s, \omega, \alpha)$; the other part is related to the state S and the action A and is called the advantage function, denoted as $A(s, a, \omega, \beta)$. Thus, the output of the network can be expressed as

$$Q(s, a, \omega, \alpha, \beta) = V(s, \omega, \alpha) + A(s, a, \omega, \beta) \tag{4}$$

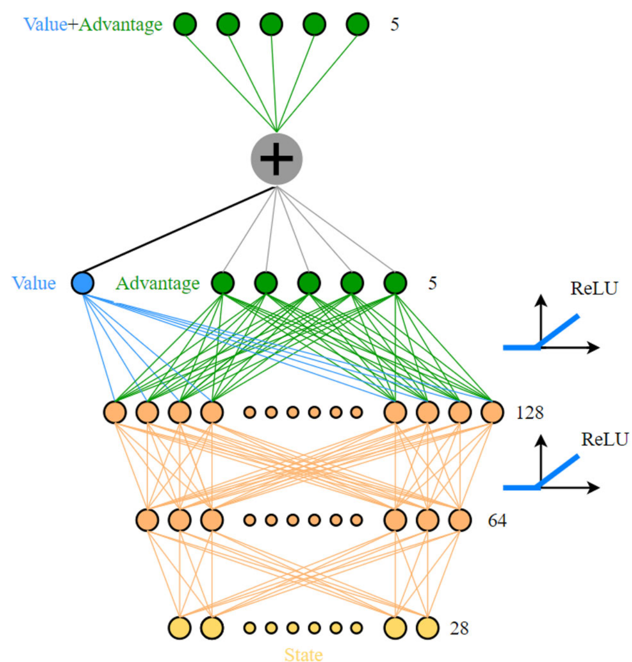


Fig. 3 DM-DQN network structure

where ω is the common parameter of V and A, s denotes the state, a denotes the action, and α and β are the parameters of V and A, respectively. The V value can be thought of as the average of the Q values in that state. The A value is limited to an average of 0, and the sum of the V and A values is the original Q value.

In M-DQN, when we need to update the Q value of an action, we update the Q network directly so that the Q value of the action is raised. The Q network of the M-DQN can be understood as fitting a curve to the Q value of the Q-table. A cross-section can be taken that represents the Q value of each action in the current state. For example, as shown in Fig. 4a, when the M-DQN is updating the value of action 2 in the state, it will only update the action. In the DM-DQN, the network gives priority to updating the V value because of the restriction that the sum of the A values must be zero. The V value is the average of the Q values and the adjustment of the average is equivalent to updating all the Q values in that state at once. Therefore, when the network is updated, it not only updates the Q value of a particular action, but adjusts the Q values of all actions in that state, all at once. In Fig. 4b, when action 2 in the state is updated, the V value is first updated, and because the average value is updated, the rest of the actions in the state follow. As a result, it is possible to have more values updated less often, resulting in faster convergence and the ability to learn the best decisions faster.

The DM-DQN is applied to robot path planning, and the value function is to learn the situation where the robot does not detect an obstacle, while the advantage function is to

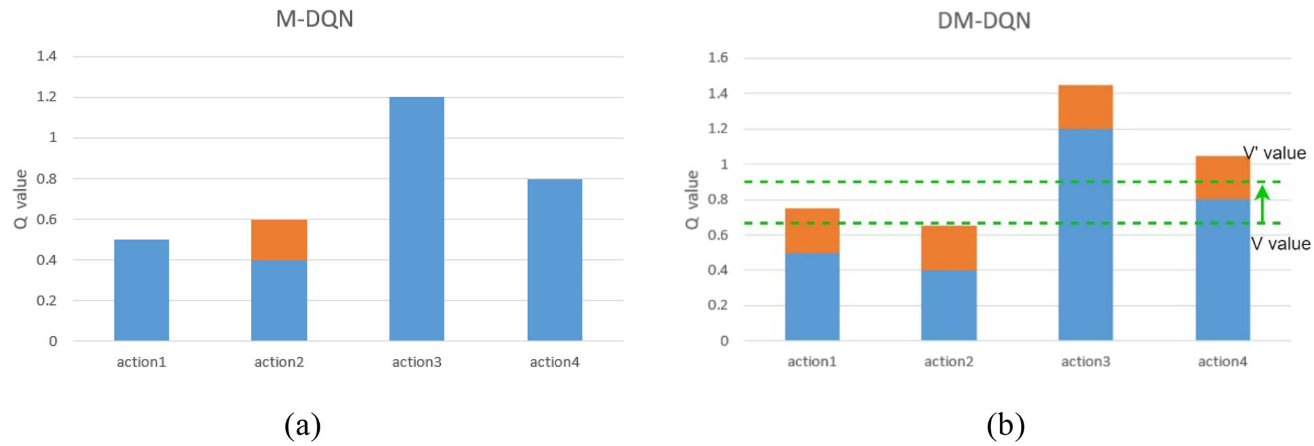


Fig. 4 Q value update based on two algorithms

understand that the robot detects an obstacle. To solve the identifiability problem, the advantage function is centralized:

$$Q(s, a, \omega, \alpha, \beta) = V(s, \omega, \alpha) + (A(s, a, \omega, \beta) - \frac{1}{A}\sum_{a' \in A} A\left(s, a', \omega, \beta\right) \quad (5)$$

where A is the optional action, ω is the common parameter of V and A, s denotes the state, a denotes the action, a' denotes the next action, and α and β are the parameters of V and A, respectively.

Proposed algorithm

The process of autonomous path planning

The design of the path planning process for a mobile robot under the DM-DQN algorithm is shown in Fig. 5. First, path planning model for the mobile robot based on DM-DQN is established to describe the mobile robot path planning problem as a Markov decision processes. Second, the robot acquires environmental information from sensors, calculates the direction and distance of obstacles and targets, and designs a reward function based on the artificial potential field. The mobile robot selects the appropriate action value from the reply buffer and the reward function based on the artificial potential field. Action value and state are first passed through the fully connected layer, followed by a value function and an advantage function to output Q values to minimize the loss function, respectively, and finally fed back to the neural network to update the network's values. If it is the end state, the environment is reset and restarted, otherwise it continues to learn in the environment; if it is the arrival state, it continues to determine if the algorithm converges, if it converges, the program ends, otherwise it continues to generate target endpoints and interact with the environment until it ends. Finally, the mobile robot interacts with the environment to obtain training data, and the sampled data are trained so that the mobile robot completes collision-free path planning.

The design of reward function based on artificial potential field method

The method of artificial potential field

The artificial potential field method is a virtual force method that treats the motion of a robot in its environment as a motion under a virtual artificial force field [31]. As shown in Fig. 6, the target point will exert a gravitational force on the robot, while the obstacle will exert a repulsive force on the robot. The resultant force of these two forces is the controlling force for the robot's motion, and with the controlling force, a collision-free path to the target point can be planned. The gravitational force on the robot becomes greater as it approaches the target point, and the repulsive force increases as it approaches the obstacle.

In the artificial potential field, the potential function U is used to create the artificial potential field, where the gravitational potential function is expressed as follows:

$$U_{\mathrm{att}}(q) = \tfrac{1}{2}\zeta d^2\left(q, q_{\mathrm{goal}}\right) \quad (6)$$

In Eq. (6), ζ denotes the gravitational potential field constant and $d(q, q_{\mathrm{goal}})$ denotes the distance between the current point q and the target point q_{goal}.

The expression for the repulsive potential function is as follows:

$$U_{\mathrm{rep}}(q) = \begin{cases} \tfrac{1}{2}\eta\left(\tfrac{1}{D(q)} - \tfrac{1}{Q^*}\right)^2, & D(q) < Q^* \\ 0, & D(q) \geq Q^* \end{cases} \quad (7)$$

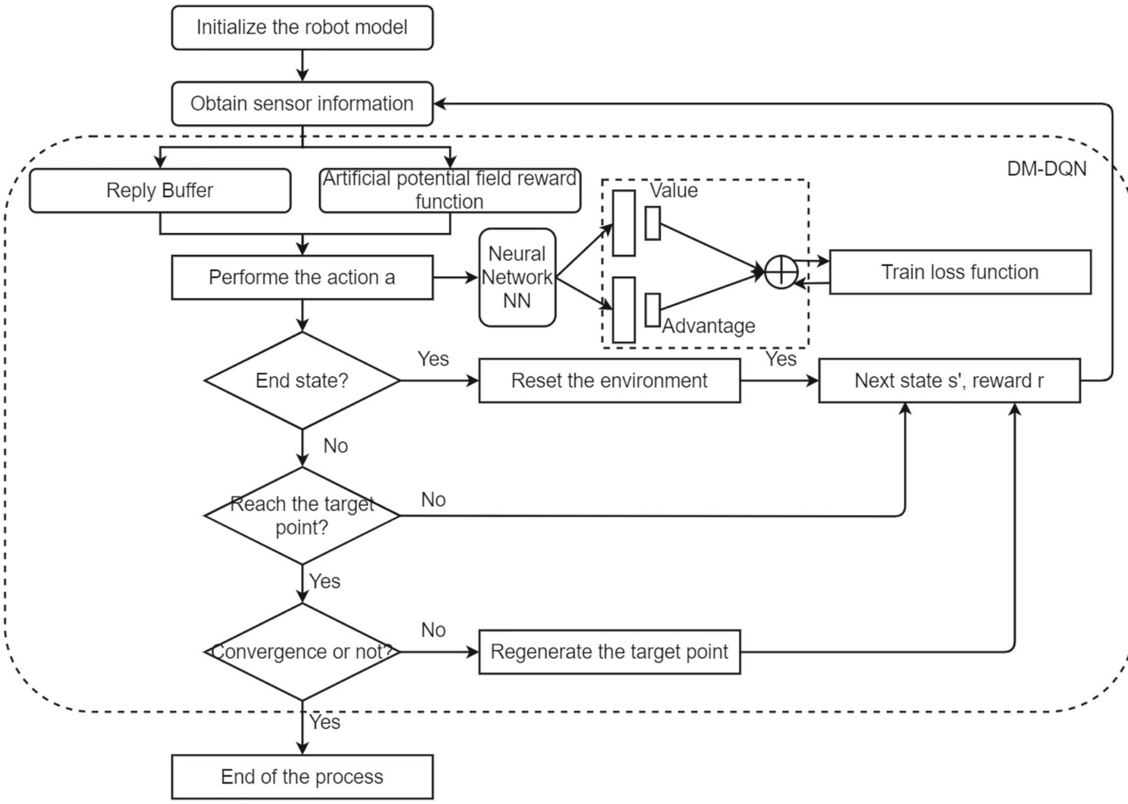


Fig. 5 The process of autonomous path planning

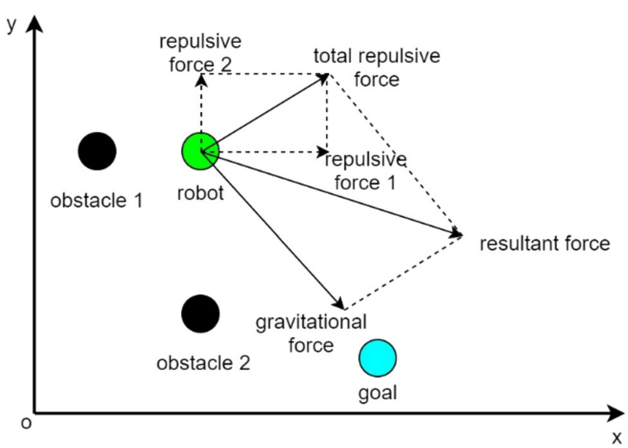


Fig. 6 Artificial potential field

In Eq. (7), η denotes the repulsive field constant, $D(q)$ denotes the distance from the current point q to the nearest obstacle, and Q^* denotes the threshold at which the obstacle generates a repulsive force, which is less than this threshold before a repulsive force is generated.

The expression for the combined potential force on a mobile robot in an artificial potential field is as follows:

$$U_q = U_{\text{att}}(q) + U_{\text{rep}}(q) \tag{8}$$

The potential function U_q of the mobile robot at point q represents the magnitude of the energy at that point, and the force vector at that point is represented by the gradient $\nabla U(q)$, which is defined as

$$\nabla U(q) = DU(q)^T = \left[\frac{\partial U}{\partial q_1}(q), \ldots, \frac{\partial U}{\partial q_m}(q)\right]^T \tag{9}$$

The gravitational function at point q can be obtained by finding the negative derivative of Eq. (6) and its expression is expressed as

$$F_{\text{att}}(q) = -\nabla U_{\text{att}}(q) = -\zeta d\left(q, q_{\text{goal}}\right) \tag{10}$$

The repulsive function at point q can be obtained by finding the negative derivative of Eq. (7) and its expression is expressed as

$$F_{\text{rep}}(q) = \begin{cases} \eta\left[\frac{1}{D(q)} - \frac{1}{Q^*}\right]\frac{1}{D^2(q)}\frac{\partial D(q)}{\partial x},\ D(q) < Q^* \\ 0,\ D(q) \geq Q^* \end{cases} \tag{11}$$

The combined forces on a mobile robot in an artificial potential field can be expressed as

$$F_q = F_{\text{att}}(q) + F_{\text{rep}}(q) \tag{12}$$

The design of reward function

When the mobile robot moves towards the target point, according to the idea of artificial potential field, the reward function is decomposed into two parts: the first part is the position reward function, including the target reward function and the obstacle avoidance reward function, the target reward function is to guide the robot to reach the target point quickly, the obstacle avoidance reward function is to make the robot keep a certain distance from the obstacles. The second part is the direction reward function. The current orientation of the robot plays a key role in reasonable navigation, and given that the direction of the combined forces on the robot in the artificial potential field can fit well with the direction of the robot's movement, the direction reward function is designed to guide the robot towards the target point.

In the position reward function, the target reward function is first constructed using the gravitational potential field function:

$$\text{reward}_{\text{att}} = \tfrac{1}{2}\zeta d_{\text{goal}}^2 \tag{13}$$

where ζ denotes the constant of gravitational reward function and d_{goal} denotes the distance between the current position and the target point.

The obstacle avoidance reward function is partially constructed using a repulsive potential field function with a negative reward that decreases as the robot's distance from the obstacle decreases:

$$\text{reward}_{\text{rep}} = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{d_{\text{obs}}} - \frac{1}{d_{\max}}\right)^2, & d_{\text{obs}} < d_{\max} \\ 0, & d_{\text{obs}} \geq d_{\max} \end{cases} \tag{14}$$

where η denotes the constant of repulsive reward function, d_{obs} denotes the distance between the current position and the obstacle and $d_{\max}$ denotes the maximum influence distance of the obstacle.

In the direction reward function, Eq. (12) represents the combined force on the robot, which coincides with the expected direction. The angular difference between the expected and actual directions of the robot is expressed as

$$\varphi = arccos\frac{\boldsymbol{F}_q \cdot \boldsymbol{F}_a}{|F_q||F_a|} \tag{15}$$

where $\boldsymbol{F}_q$ denotes the expected direction, $\boldsymbol{F}_a$ denotes the actual direction and φ denotes the angle between the expected direction and the actual direction. The direction reward function can, therefore, be expressed as

$$\text{reward}_{\text{yaw}} = \frac{\varphi\%(2\pi)}{\pi} \tag{16}$$

Combining Eqs. (13), (14) and (16), the total reward function can be expressed as

$$\begin{aligned} \text{reward} &= \text{reward}_{\text{att}} + \text{reward}_{\text{rep}} + \text{reward}_{\text{yaw}} \\ &= \frac{1}{2}\zeta d_{\text{goal}}^2 - \frac{1}{2}\eta\left(\frac{1}{d_{\text{obs}}} - \frac{1}{d_{\max}}\right)^2 + \frac{\varphi\%(2\pi)}{\pi} \end{aligned} \tag{17}$$

Therefore, the reward function for the mobile robot is expressed as a whole as

$$R = \frac{1}{2}\zeta d_{\text{goal}}^2 \begin{cases} 1000 & d_{\text{goal}} < r_{\text{goal}} \\ -\frac{1}{2}\zeta\left(\frac{1}{d_{\text{obs}}} - \frac{1}{d_{\max}}\right)^2 + \frac{\varphi\%(2\pi)}{\pi} & d_{\text{goal}} > r_{\text{goal}} \&\& d_{\text{obs}} > r_{\text{obs}} \\ -500 & d_{\text{obs}} < r_{\text{obs}} \end{cases} \tag{18}$$

where r_{goal} denotes the radius of the target area centered on the target point and r_{obs} denotes the radius of the collision area centered on the obstacle.

In order to verify the effectiveness of the reward function setting based on the artificial potential field, only the distance information between the mobile robot and the target point will be considered as the reward function setting for comparison, and its reward function setting is shown as follows:

$$\text{reward} = \left(\frac{d_{\text{goal}}}{k}\right)^2 \tag{19}$$

The process of path planning algorithm based on DM-DQN

The algorithm proposed in the paper first estimates the Q value through an online Dueling Q network with weight $\boldsymbol{\theta}$, and weight $\boldsymbol{\theta}$ is replicated in a target network with weight $\widehat{\boldsymbol{\theta}}$ at each passing C steps. Second, by interacting with the environment using a ε-greedy strategy, the robot obtains reward and the next state according to the reward function based on artificial potential field. Finally, the transitions ($\boldsymbol{s_t}$, $\boldsymbol{a_t}$, $\boldsymbol{r_t}$, $\boldsymbol{s_{t+1}}$) are stored in a fixed size FIFO replay buffer and with each F steps, DM-DQN randomly draws a batch of $\boldsymbol{D_t}$ from the replay buffer $\boldsymbol{D}$ and minimizes the following losses according to the regression objective of Eq. (8). The complete algorithm process is shown in Algorithm 1.

Algorithm 1 DM-DQN with reward function based on artificial potential field method

Require: $\mathbf{T} \in \mathbb{N}^*$ is the number of environment steps; $C \in \mathbb{N}^*$ is the update period, $F \in \mathbb{N}^*$ is the interaction period.

Initialize replay buffer D to capacity N
Initialize dueling Q-network with parameters θ
Initialize target dueling Q-network with parameters $\hat{\theta} = \theta$

Initialize $r_t = \begin{cases} 1000 & d_{goal} < r_{goal} \\ \frac{1}{2}\zeta d_{goal}^2 - \frac{1}{2}\eta\left(\frac{1}{d_{obs}} - \frac{1}{d_{max}}\right)^2 + \frac{\varphi\%(2\pi)}{\pi} & r_{goal} < d_{goal} \&\& \, d_{obs} > r_{obs} \\ -500 & d_{obs} < r_{obs} \end{cases}$

for t = 1 **to** T **do**
  The robot in state s_t, with probability ε selects a random action a_t
  Otherwise selects $a_t = argmax_a Q(s_t, a)$
  The robot executes action a_t and observes reward r_t and state s_{t+1}
  Store transition (s_t, a_t, r_t, s_{t+1}) in D
  if t mod F == 0 **then**
    On a random batch of transitions $D_t \subset D$, update θ with one step of SGD on L_{DM-DQN}.
  end if
  if k mod C == 0 **then**
    $\hat{\theta} = \theta$
  end if
end for

Materials and methods

Experimental platform

The experimental platform is Windows 10.1 + tensorflow1.13.1 + cuda10.0 and the hardware is Intel i7-8550U@1.6 GHz processor. The simulation platform was Gazebo simulation platform. The DQN algorithm, Dueling DQN algorithm, M-DQN algorithm and DM-DQN algorithm were trained for 320 rounds, respectively, and the effectiveness of the algorithm's obstacle avoidance strategy, the relationship between the success rate of obstacle avoidance, the effectiveness of obstacle avoidance and the number of training rounds were analyzed.

Experimental environment setup

Gazebo is a physical simulation platform model that supports a variety of robotics, sensors and environmental simulations. The Gazebo simulator is used to create a virtual simulation environment and the robot is modeled by Gazebo to implement the corresponding path planning tasks. In addition, it uses python to implement the path planning algorithm and calls the built-in Gazebo simulator to control the robot's movements and obtain robot sensory information. In all experiments, the robot model parameters are kept consistent and do not require any a priori knowledge of the environment. The parameters shown in Table 1 were used throughout the experiment.

Table 1 Training parameters and values

Parameters	Values
Learning rate	0.01
Discount factor	0.9
Pre-training steps	800
Mini-batch size	128
Replay memory size	20,000
Network update frequency	30

Experimental environments used in the experiments

Robot operating system

This paper uses the ROS (robot operating system) software platform, which is a robot software development platform that includes system software to manage computer hardware and software resources and provide services. ROS uses a cross-platform modular communication mechanism, which is a distributed framework (Nodes) and largely reduces the code reuse rate. ROS is also highly compatible and open, providing a number of functional packages, debugging tools and visualization tools.

The ROS node graph for the experiments is shown in Fig. 7. Gazebo will publish information on a range of topics such as odometry and LIDAR. In addition, the DQN algorithm communicates with these topics, the feedback from the environment can be obtained, and the strategy can be learned to output the actions to be performed and passed to the gazebo environment. This allows the algorithm to interact with the simulation environment.

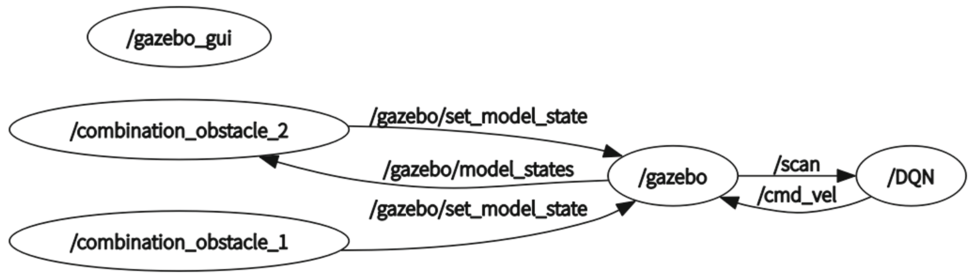


Fig. 7 The ROS node graph

Gazebo simulation environment

Gazebo is a 3D physics simulation environment based on the ROS platform, open source and powerful, providing many types of robot, sensor and environment models, and providing a variety of high-performance physics engines such as ODE, Bullet, Sim body, DART and others to achieve dynamics simulation. Gazebo has the following main advantages:

- Editing environment: Gazebo provides the basic physical model, making it easier to add robot and sensor models.
- Sensor simulation: Supports data simulation of sensors such as LiDAR, 2D/3D/depth cameras and can add noise to the data.
- 3D visualization of scenes: Effects such as light, texture and shadow can be set to increase the realistic display of the scene.

Experiments and results

Static environment

To verify the planning performance of DRL in different scenarios, we first validate the reliability of the algorithm on a static environment. At the beginning, we randomly generated seven target points to test the effect of DQN, M-DQN, Dueling DQN and DM-DQN in the environment. The environment model is shown in Fig. 8, where the black object is mobile robot and the brown objects are static obstacles.

In static environment, the DQN algorithm, the Dueling DQN algorithm, the M-DQN algorithm and the DM-DQN algorithm are used for the path planning task and their convergence rates are compared. Each model was trained 320 times. Figure 9 records the cumulative reward for each round and the average reward for the agent, where each dot indicates a round and the black curve indicates the average reward. As shown in Fig. 9, the intelligences lacked experience of how to reach the goal in the early stages and spent most of the first 100 rounds exploring the environment, so the intelligences received low rewards. However, by comparing these

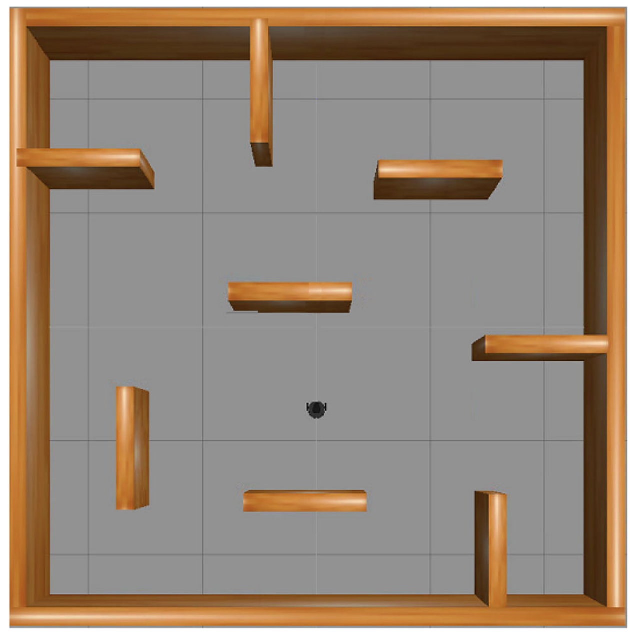


Fig. 8 The map of static environment

four algorithms, as shown in Fig. 10, we find that the remaining three algorithms all rise faster than DQN after 100 rounds, which is because the network structure adopted by Dueling DQN can update multiple Q values at once; while M-DQN is due to the introduction of maximum entropy, the addition of maximum entropy makes the strategy more random, so it will add more exploration, thus can speed up subsequent learning; DM-DQN adopts a competitive network structure compared to M-DQN, decoupling action selection and action evaluation makes it have a faster learning rate, so it can make fuller use of the experience of exploring the environment in the early stage, and thus obtain a greater reward. As can be seen in Fig. 10, the reward obtained by DQN converges to 2000, the reward of Dueling DQN and M-DQN converges to 4000, while the reward value of DM-DQN converges to 7000. Therefore, the DM-DQN proposed in this paper is able to obtain a larger reward value compared to the remaining three algorithms, which means that more target points can be reached.

Seven points were designated for navigation in the test environment, and the robot was expected to explore this

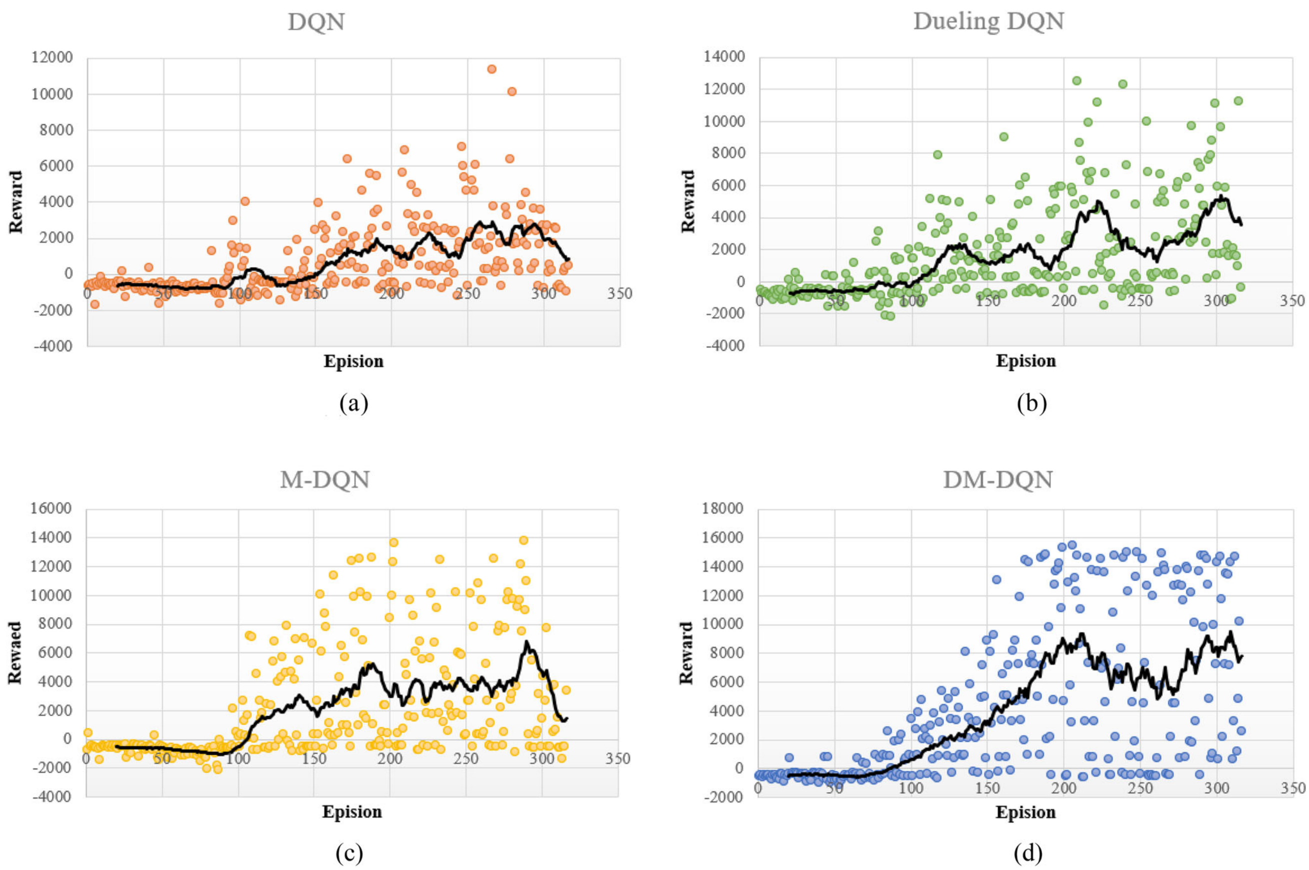


Fig. 9 The robot's reward for each epision based on four algorithms

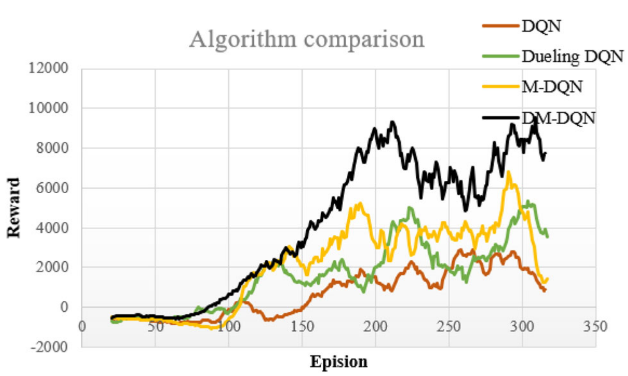


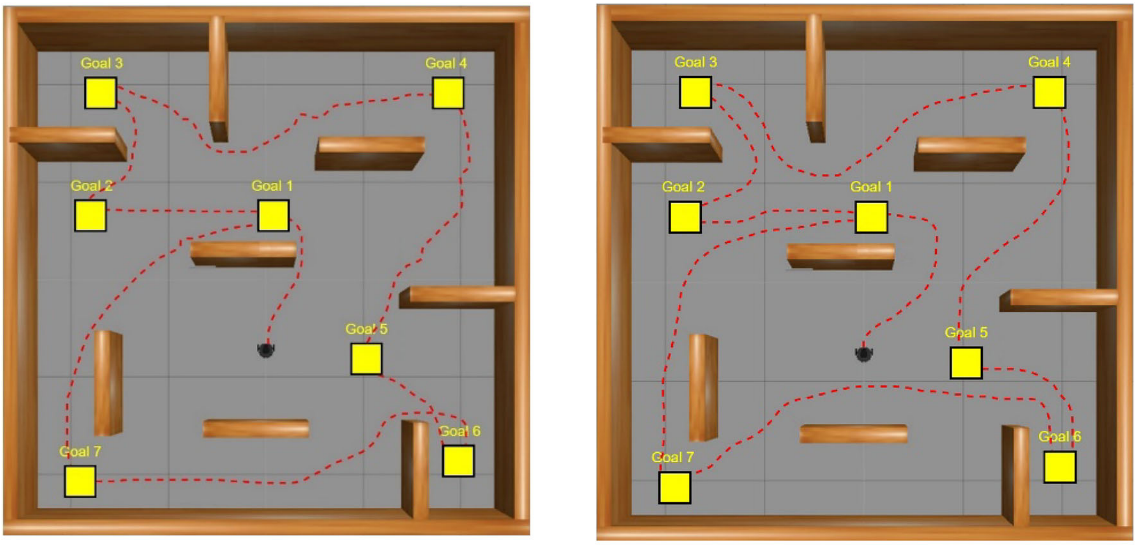
Fig. 10 Comparison of the four algorithms

unknown environment by autonomously moving from position 1 to positions 2 to 7 and back to position 1 in a collision-free sequence, as shown in Fig. 11. Table 2 shows the average number of moves made by the four algorithms to reach a target point in 320 rounds; the number of successful moves to the target point; and the success rate of reaching the target point. The table shows that the DM-DQN algorithm has a lower average number of moves compared to the rest of the algorithms and an 18.3% improvement in success rate compared to the DQN algorithm; a 3.3% improvement compared to the Dueling DQN algorithm; and a 17.2% improvement compared to the M-DQN. In Table 2, the convergence rates of the algorithms are also compared and it can be seen that DQN took 294 min to obtain a reward of 8000, Dueling DQN took 148 min, M-DQN took 127 min and DM-DQN took 112 min. DM-DQN converged faster than the other algorithms and took less time to reach the target point.

Figure 11 shows the effect of two different reward functions for path planning, where the reward function in (a) only considers the distance between the robot and the target point; (b) is the reward function proposed in this paper. From the figure, we can see that the paths in (b) are smoother and the planned paths are farther away from obstacles, which greatly reduces the probability of collision for the robot in a real environment.

Dynamic and static environment

In the dynamic and static environment, we still randomly generated seven target points to test the effect of DQN, M-DQN, Dueling DQN and DM-DQN in the environment. Compared to the static environment with two moving obstacles, the



(a) (b)

Fig. 11 Generated paths in a static environment

Table 2 The result of algorithm comparison

Model	Average moving step	Number of success	Success rate	Convergence time
DQN	27.08	148	49.3	294
Dueling DQN	24.14	193	64.3	148
M-DQN	22.02	164	50.4	127
DM-DQN	**20.81**	**203**	**67.6**	**112**

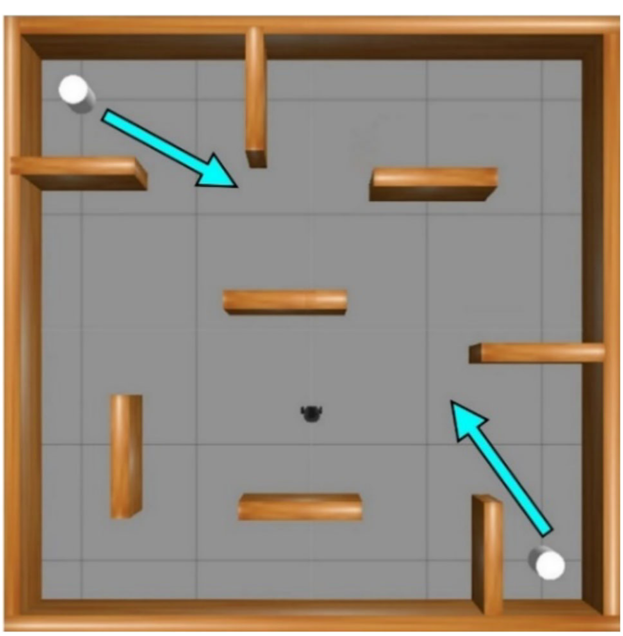


Fig. 12 Dynamic and static environment map

dynamic obstacles move in a randomized direction. The environment model is shown in Fig. 12, where the black object is the moving robot, the brown objects are the static obstacles and the white cylinders are the moving obstacles, which move in a randomized direction.

The DQN algorithm, the Dueling DQN algorithm, the M-DQN algorithm and the DM-DQN algorithm were also used for the path planning task in a dynamic and static environment and their convergence rates were compared. The cumulative rewards for each round and the average rewards for the agent are recorded in Fig. 13, and Fig. 14 compares the four algorithms. Unlike the static environment, the reward values of the DQN, Dueling DQN, and M-DQN algorithms did not rise significantly after 100 rounds. The upward trend occurs at round 150, which is caused by the inclusion of dynamic obstacles, while the DM-DQN proposed in the paper still starts to converge at around round 120, indicating its good generalization ability compared to the other algorithms.

Table 3 also compares the average number of moves to reach a target point, the number of successful arrivals and the success rate of reaching the target point for 320 rounds, as the performance of all four algorithms decreases with the inclusion of dynamic obstacles. The table shows that DM-DQN still has the lowest average number of moves, with

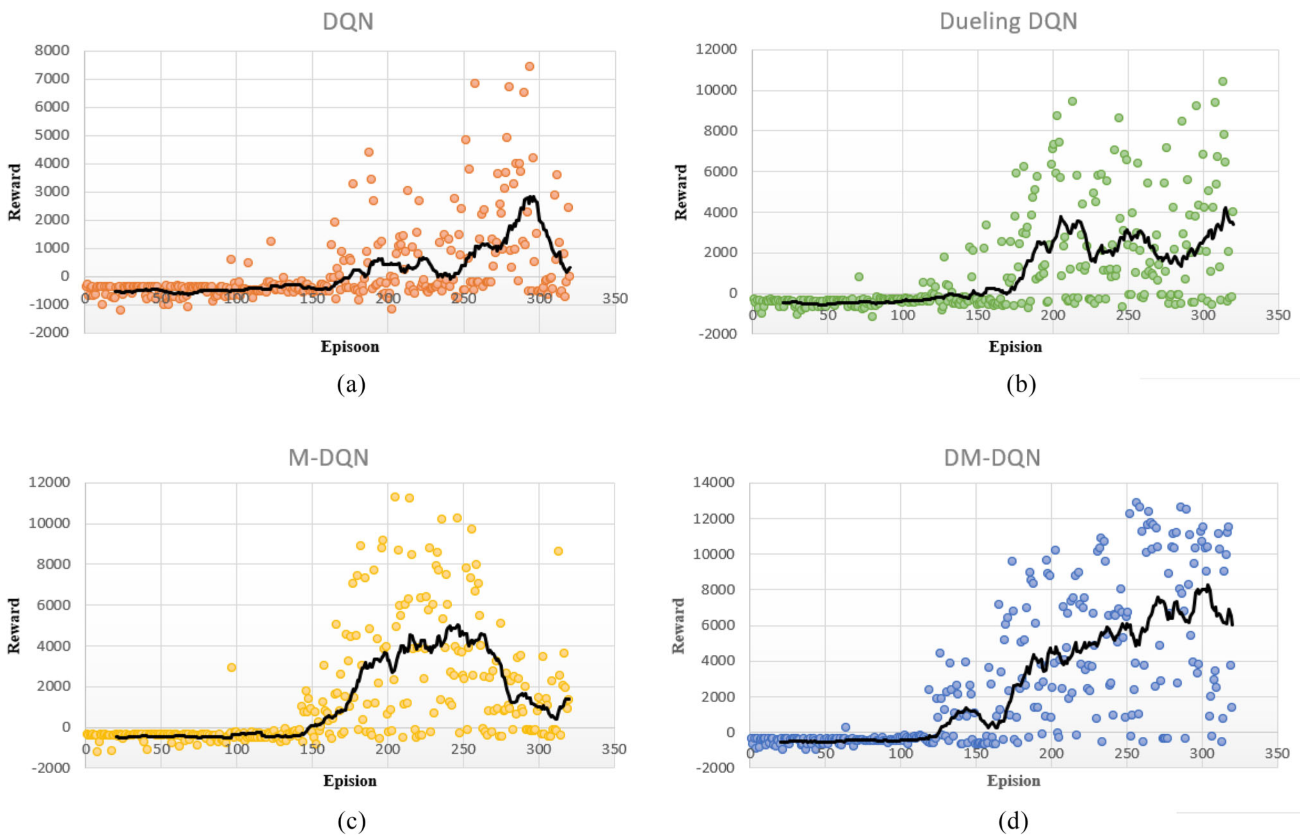


Fig. 13 The robot's reward for each epision base on four algorithms

a 27.3% improvement in success rate compared to DQN, a 12.6% improvement compared to Dueling DQN, and a 9.3% improvement compared to M-DQN. In Table 3, which also compares the convergence speed of each algorithm, it can be seen that DQN took 261 min to obtain the 8000 reward, Dueling DQN took 186 min, M-DQN took 150 min and DM-DQN took 131 min. DM-DQN converged faster than the other algorithms in the dynamic and static environment and took less time to reach the target point. The time taken to reach the target point was shorter.

Figure 15 shows the path planning effect of two different reward functions in dynamic and static environments, where the reward function in (a) only considers the distance between the robot and the target point; (b) the same reward function is proposed in the paper. The addition of dynamic obstacles places higher demands on path planning. A comparison of the two figures shows that the reward function setting proposed in the paper can effectively solve the problem of dynamic obstacles, because the reward function in the paper takes into account the distance from the obstacles, which enables the

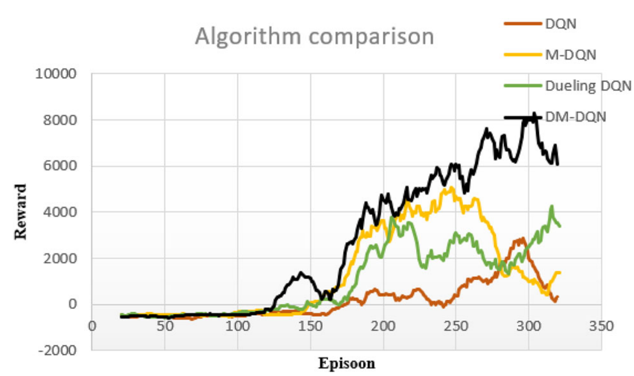
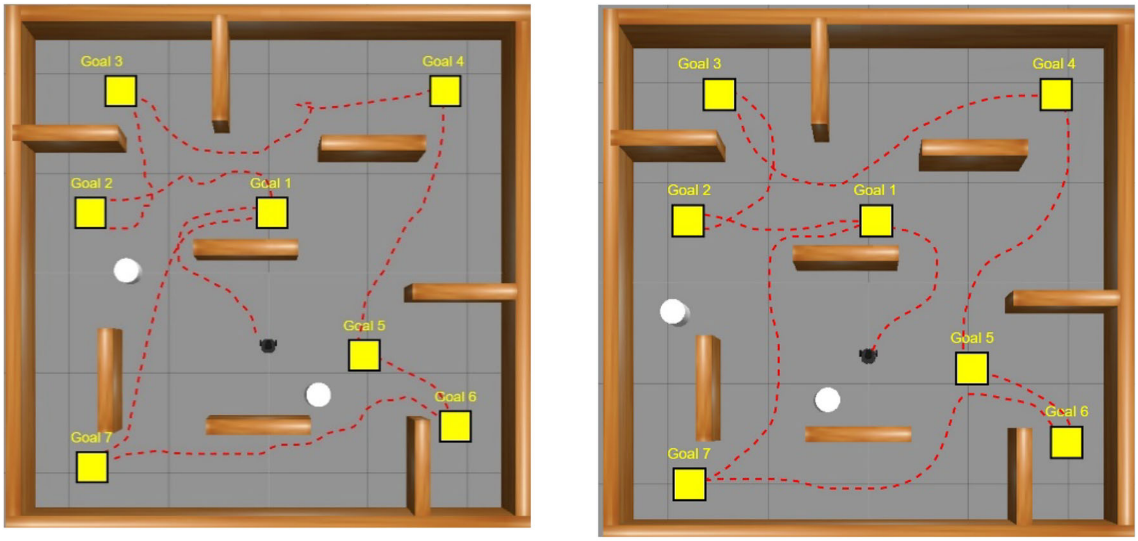


Fig. 14 Comparison of the four algorithms

Table 3 The result of algorithm comparison

Model	Average moving step	Number of success	Success rate	Convergence time
DQN	30.22	79	26.3	261
Dueling DQN	27.01	123	41.0	186
M-DQN	27.64	133	44.3	150
DM-DQN	**26.43**	**161**	**53.6**	**131**



(a) (b)

Fig. 15 Generated paths in dynamic and static environments

path planned by the robot to effectively avoid the influence of dynamic obstacles.

Conclusions

A continuous dynamic and static simulation environment is established for the path planning problem of mobile robot in complex environment. First, its network structure is improved on the basis of M-DQN, and its convergence speed is accelerated by decomposing the network structure into value and advantage functions, thus decoupling action selection and action evaluation. Second, a reward function based on an artificial potential field is designed to balance the distance of the robot from the target point and the obstacle, so that the planned path is away from the obstacle. The simulation result show that the DM-DQN proposed in the paper has a faster convergence speed compared to M-DQN, Dueling DQN and DQN, and is able to learn the best decision faster. The reward function is designed for smoother paths to be planned, while effectively avoiding the planned paths from being too close to obstacles. The future study will be made in the following areas. Reinforcement learning has a disadvantage compared to traditional algorithms in path planning over long distances, so a combination of traditional algorithms with reinforcement learning algorithms is considered to enable a breakthrough in path planning over long distances.

Acknowledgements This research is supported by National Natural Science Foundation of China (61906021) and Digital Twin Technology Engineering Research Center for Key Equipment of Petrochemical Process (DTEC202101).

References

1. Koubaa A, Bennaceur H, Chaari I et al (2018) Introduction to mobile robot path planning. Robot Path Plan Cooperation 772:3–12
2. Koren Y, Borenstein J (1991) Potential field methods and their inherent limitations for mobile robot navigation. IEEE Int Conf Robot Automation 2:1398–1404
3. Fu XL, Huang JZ, Jing ZL (2022) Complex switching dynamics and chatter alarm for aerial agents with artificial potential field method. Appl Math Model 107:637–649
4. Reshamwala A, Vinchurkar DP (2013) robot path planning using an ant colony optimization approach: a survey. Int J Adv Res Artif Intell 2(3):65–71
5. Castillo O, Leonardo T, Patricia M (2007) Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots. Soft Comput 11:269–279

6. Clerc M, Kennedy J (2002) The particle swarm explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evolution Comput 6(1):58–73
7. Boute RN, Gijsbrechts J, van Jaarsveld W et al (2022) Deep reinforcement learning for inventory control: a roadmap. Eur J Oper Res 298(2):401–412
8. Rupprecht T, Yanzhi W (2022) A survey for deep reinforcement learning in markovian cyber-physical systems: Common problems and solutions. Neural Netw Off J Int Neural Netw Soc 153:13–36
9. Halbouni A, Gunawan TS, Habaebi MH et al (2022) Machine learning and deep learning approaches for cybersecurity: a review. IEEE Access 10:19572–19585
10. Brunke L, Greeff M, Hall AW et al (2022) Safe Learning in robotics: from learning-based control to safe reinforcement learning. Annu Rev Control Robot Autonom Syst 5:411–444
11. Liu JW, Gao F, Luo XL (2019) Survey of deep reinforcement learning based on value function and policy gradient. Chin J Comput 42(6):1406–1438
12. Mnih V et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
13. Watkins CJCH, Dayan P (1992) Q-learning. Mach Learn 8:279–292
14. Wang Z, Schaul T et al (2016) Dueling network architectures for deep reinforcement learning. In: Proceedings of the 33rd international conference on international conference on machine learning. IEEE
15. Haarnoja T, Zhou A, Abbeel P, et al (2018) Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: 35th International conference on machine learning
16. Vieillard N, Pietquin O, Geist M (2020) Munchausen reinforcement learning. In: 34th advances in neural information processing systems
17. Liu SH, Zheng C, Huang YM et al (2022) Distributed reinforcement learning for privacy-preserving dynamic edge caching. IEEE J Sel Areas Commun 40(3):749–760
18. Dong Y, Yang C et al (2021) Robot path planning based on improved DQN. J Comput Des Eng 42:552–558
19. Wu HL, Zhang JW, Wang Z et al (2022) Sub-AVG: overestimation reduction for cooperative multi-agent reinforcement learning. Neurocomputing 474:94–106
20. Huang RN, Qin CX, Li JL, Lan XJ (2021) Path planning of mobile robot in unknown dynamic continuous environment using reward-modified deep Q-network. Optim Control Appl Methods. https://doi.org/10.1002/oca.2781
21. Lou P, Xu K et al (2021) Path planning in an unknown environment based on deep reinforcement learning with prior knowledge. J Intell Fuzzy Syst 41(6):5773–5789
22. Yan N, Huang SB, Kong C (2021) Reinforcement learning-based autonomous navigation and obstacle avoidance for USVS under partially observable conditions. Math Problems Eng 2021:1–13
23. Yan C, Xiang XJ, Wang C (2020) Towards real-time path planning through deep reinforcement learning for a UAV in dynamic environments. J Intell Rob Syst 98(2):297–309
24. Hu YB, Wu XY, Geng P et al (2018) Evolution strategies learning with variable impedance control for grasping under uncertainty. IEEE Trans Ind Electron 66(10):7788–7799
25. Hu YB, Su H, Fu JL et al (2020) Nonlinear model predictive control for mobile medical robot using neural optimization. IEEE Trans Ind Electron 68(12):12636–12645
26. Chades I, Pascal LV, Nicol S et al (2021) A primer on partially observable Markov decision processes. Methods Ecol Evol 12(11):2058–2072
27. Sankaran PG, Sunoj SM, Nair NU (2016) Kullback–Leibler divergence: a quantile approach. Stat Prob Lett 111:72–79
28. Schulman J, Levine S, Abbeel P, Jordan M, Moritz P (2015) Trust region policy optimization. In: 32nd International conference on machine learning
29. Abdolmaleki A, Springenberg JT, Tassa Y, Munos R, Heess N, Riedmiller M (2018) Maximum a posteriori policy optimisation. In: 8th International conference on learning representations
30. Hasselt HV, Guez A, Silver D (2016) Deep reinforcement learning with double Q-learning. In: The association for the advancement of artificial intelligence
31. Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. Int J Robot Res 5(1):90–98