Construction of an Intelligent Robot Path Recognition System Supported by Deep Learning Network Algorithms

Jiong Chen*
Department of Artificial Intelligence, Shanxi Polytechnic College, Taiyuan, 030006, China

Abstract*—In recent years, intelligent robots have been widely used in fields such as express transportation, industrial automation, and healthcare, bringing great convenience to people's lives. As one of the core technologies of intelligent robots, path planning technology has become a research highlight in the field of robotics. To achieve path planning in unknown environments, a path planning algorithm based on an improved dual depth Q-network is proposed. In both simple and complex grid environments, the planned path inflection points for the improved dual depth Q-network is 4 and 9, respectively, with path lengths of 27.21m and 28.63m, respectively. Both are less than double depth Q network and adaptive Ant colony optimization algorithms. The average reward values of the improved dual depth Q network in simple and complex environments are 1.12 and 1.02, respectively, which are higher than those of the dual depth Q network. In a random environment, the lowest probability of the improved dual depth Q network successfully reaching the destination without colliding with obstacles is 95.1%, which is higher than the other two algorithms. In the Gazebo environment, when the number of iterations reaches 2000, the average cumulative reward value is positive. The average cumulative reward value in the range of iterations from 3500 to 4000 and iterations from 4000 to 4500 exceeds 500. The average cumulative reward value of the dual depth Q network is only positive within the two intervals of iterations 2500-3000 and 3000-3500. The average cumulative reward value does not exceed 100. According to the findings, the path planning ability of the improved dual depth Q network is better than that of the dual depth Q network and the adaptive Ant colony optimization algorithms.

Keywords*—*Deep learning; reinforcement learning; intelligent robots; path planning

I. INTRODUCTION

With the progress of computer technology, intelligent robots have gradually entered people's lives. They play an important role in their respective fields. At the same time, due to the technological reform of intelligent robots, humans have gradually raised their requirements for the mobility of intelligent robots. Intelligent robots are expected to quickly plan a route to the destination in unfamiliar environments. The path planning of robots consists of global path planning and local path planning based on their mastery of environmental information. In local path planning, the motion trajectory of the robot has uncertainty. It needs to constantly obtain information from the environment to determine the next step. At the same time, it is necessary to obtain data information on obstacles to avoid unknown obstacles [1]. Reinforcement learning (RL) is a good solution to how to avoid unknown obstacles. However, in a complex environment, RL will have an exponential growth problem of state-action collection, resulting in a "Curse of dimensionality" [2]. The deep reinforcement learning (DRL) obtained from deep learning (DL) combined with Reinforcement learning can effectively improve the above problems. The deep learning in Deep reinforcement learning can extract the data information characteristics of the environment through neural network, realizing the fitting between the state-action value function (SAVF) and the environment [3]. At present, DRL has become a popular algorithm in the research of robot path planning (RPP) for unknown environments. However, due to the overestimation of DL, the output state-action values in DRL applications are higher than the true values. Therefore, to realize the path planning of the robot in an unknown environment, the influence of overestimation on the robot action selection is reduced. A RPP based on Improved Double Deep Q-Network (IDDQN) is proposed in the research. This algorithm effectively avoids the "dimension disaster" problem, ensures the action speed and accuracy of the robot, enriches the application scenarios of the robot, and provides strong support for the future intelligence of the robot.

The article consists of four sections. Section I is the introduction. Section II deals with related works. Section III deals with RPP algorithm based on DRL. Section IV is the simulation experiments and result analysis and Section V concluded the whole study.

II. RELATED WORKS

In the current research on various technologies of intelligent robots, path planning is a hot research direction. The way and quality of path planning determine whether a robot can safely and quickly reach the destination. Rath A K and his team proposed a navigation control algorithm based on genetic algorithm (GA) and neural network for robot navigation problems in complex environments. The GA controller is used to generate the initial turning angle of the robot. Then, the GA controller and neural network controller are mixed to generate the final steering angle. After testing, the navigation parameter error of this algorithm is relatively small [4]. Nie B et al. proposed a path planning method based on value iteration for intelligent agents with complex kinematics. The state-action transition probability is used to encode the ability of the agent. According to the findings, it has higher precision, faster convergence rate and lower random seed sensitivity [5]. Wang G and other scholars proposed a dynamic path planning method based on fuzzy neural networks for intelligent robots. Compared with the traditional Particle swarm optimization (PSO), it can significantly improve the control accuracy and robustness of the model [6]. Oultiligh A and his team proposed a trajectory planning method based on PSO and gray wolf optimization for mobile robot trajectory optimization. This algorithm can effectively balance global and local search capabilities. After testing, the optimal trajectory search ability of PSO-GWO has significantly improved [7]. Raheem F A et al. proposed a RPP method based on probabilistic landmarks and artificial potential fields. The PSO is applied to obtain the optimization weight required by each control point participating in the formation of the spline curve. After testing, this method can ensure the feasibility and rationality of the path [8].

As an artificial intelligence method close to human thinking mode, DRL is an effective way to solve complex perception problems. Therefore, it is widely used in automatic driving, robotics, game control, machine translation and other fields. Guang zheng W et al. proposed a DRL based collision detection and avoidance method for distributed multiple unmanned aerial vehicles. Human experience is also integrated into training. Compared to traditional DRL methods, DRL that integrates human experience has significant improvements in multi drone collision detection and avoidance. In addition, the flight safety brought by the hybrid control method has also been verified [9]. Cao D and his team proposed an analysis method based on MDP and PPO to analyze the optimal power flow problem of distribution networks containing renewable energy and energy storage devices. This method obtains knowledge from historical data through neural networks and provides online decision-making based on the real-time status of the power grid. The experimental results show that the real-time control strategy proposed by this method is more flexible, which has better performance [10]. Zhao J et al. proposed a dynamic multi micro grid formation method based on CNN and DDQN to develop a multi micro grid formation plan. In this method, the dynamic micro grid formation problem is transformed into a Markov decision process. The topology changeable micro grid is designed through the DRL framework. According to the findings, this method has strong elasticity, which can respond to changing system conditions in a timely manner [11]. Zhang D et al. proposed a two-stage deep Q-learning algorithm based on pre-exploration for intelligent train control. After testing, this algorithm smoothes the acceleration curve. It can effectively complete train control tasks in multi train tracking scenarios [12]. Shihab S A M and Wei P proposed a strategy formulation method based on DRL for developing optimal seat inventory control strategies. In this method, DNN is used to calculate the expected optimal return for all possible state action combinations. Various factors such as random demand, passenger arrival, and booking cancellation have been fully considered. According to the findings, interacting with the market can learn the optimal airline revenue management strategy [13].

In summary, with the development of intelligent robot technology, RPP methods for mobile robots have become a hot research topic. There have been significant achievements in current research on RPP. In local path planning, DRL is a popular algorithm. However, due to the inherent overestimation problem of RL, the path chosen by the robot is not necessarily the optimal path. To address the above issues, a RPP method based on IDDQN is proposed, aiming to achieve local path planning in unfamiliar environments.

III. RPP Algorithm based on DRL

With the progress of science and technology, robot technology is receiving increasing attention from people. As one of the core technologies of intelligent mobile robots, path planning has become a research highlight in the robotics. To achieve safe movement of robots in unknown environments, a RPP method based on improved DDQN is proposed-IDDQN. The IDDQN algorithm effectively avoids the DDQN over estimation through more moderate Q update method and improves the utilization efficiency through the sequencing of priority playback mechanism. It effectively reduces the disadvantage of slow DDQN training speed.

A. RPP based on DDQN

The core of DRL is Reinforcement learning. Reinforcement learning guides behavior through rewards gained from interaction with the environment. However, the reinforcement learning model is more complex. Therefore, MDP is introduced into the reinforcement learning model to simplify it. In RPP, the probability of state transition and the immediate reward return function are generally unknown. Therefore, robots need to constantly interact with the environment. The state transition probability reflecting actions is displayed in Eq. (1).

$$P_{ss'}^{a} = P\left[S_{t+1} = s' \middle| S_t = s, A_t = a\right] \tag{1}$$

In Eq. (1), $P_{ss'}^{a}$ represents the execution of action a in state s. s' represents the probability that the robot can reach the state. P represents the probability of state transition. S_t represents the set of state spaces at time t. A_t stands for the set of action spaces at time t. The calculation method for cumulative reward value (CRV) is shown in Eq. (2).

$$G_t = R_{t+1} + \gamma R_{t+2} + \text{L} = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2}$$

In Eq. (2), R represents the immediate reward function. γ represents the discount factor. G_t represents the CRV. The state VF is shown in Eq. (3)

$$v_\pi(s) = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right] \tag{3}$$

In Eq. (3), $v_\pi(s)$ represents the state value function (SVF). π stands for strategy. E_π represents the expected cumulative reward return in strategy π. The SAVF is shown in Eq. (4).

$$q_\pi(s,a) = E_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right] \tag{4}$$

In Eq. (4), $q_\pi(s,a)$ represents the action SVF. The calculation of the SVF is to construct the data in the algorithm to obtain the optimal strategy. The Bellman optimal equations for the SVF and the SAVF are shown in Eq. (5).

$$\begin{cases} v^*(s) = \max_a R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a v^*(s') \\ q^*(s,a) = R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a \max_{a'} q(s',a') \end{cases} \tag{5}$$

In Eq. (5), $v^*(s)$ and $q^*(s,a)$ represent the optimal SVF and the optimal SAVF, respectively. If the optimal SVAF is already specified, the optimal strategy is determined by maximizing the optimal SVAF. At this point, the optimal strategy expression is shown in Eq. (6).

$$\pi^* = \begin{cases} 1 & a = \arg\max_{a\in A} q^*(s,a) \\ 0 & otherwise \end{cases} \tag{6}$$

Eq. (6) indicates that the SVAF is the maximum. At this point, under strategy π and state s, the probability of the robot executing action a is 1. The execution probability of other actions is 0. The value function of the DQN algorithm utilizes DNN for approximation. The network structure of DQN is illustrated in Fig. 1.

In Fig. 1, the parameters corresponding to the DQN value function represent the weight size of each layer in DNN. At this point, updating the value function means updating the network parameters. If the network structure is determined, the network parameters are the value function [14-15]. When using DNN to approximate a value function, the strong correlation between data sample tuples can easily lead to instability and non-convergence issues. Therefore, experience playback technology is used to address this issue. The experience of each time step is stored in the data pool. When updating network parameters, random samples are taken from the data pool for training. The method for updating network parameters is shown in Eq. (7).

$$\theta_{t+1} = \theta_t + \alpha\begin{bmatrix} r + \gamma \max_a Q(s',a',\theta) \\ -Q(s,a,\theta) \end{bmatrix} \nabla Q(s,a,\theta) \tag{7}$$

In Eq. (7), θ represents the network parameters. $Q(s,a,\theta)$ represents the value function of Q-Learning. α represents a parameter. r represents timely return. However, when calculating the SAVF, the parameters used are the same as those approximated by the value function, which can easily lead to unstable training. Calculating the time difference optimization objective through target network parameters can effectively solve this problem. At this point, the parameter update formula is shown in Eq. (8).

$$\theta_{t+1} = \theta_t + \alpha\begin{bmatrix} r + \gamma \max_{a'} Q(s',a',\theta^-) \\ -Q(s,a,\theta) \end{bmatrix} \nabla Q(s,a,\theta) \tag{8}$$

In Eq. (8), θ^- stands for the parameters of the target network. The training process of DQN is shown in Fig. 2.

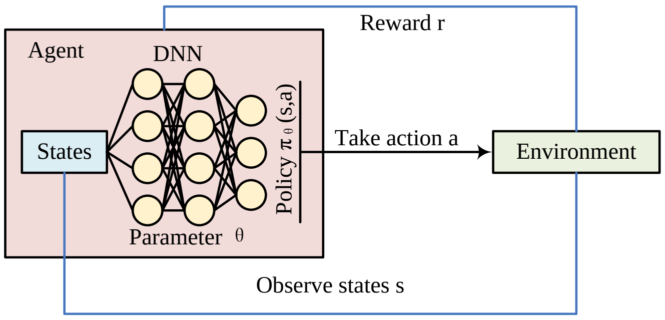


Fig. 1. Network structure of DQN.

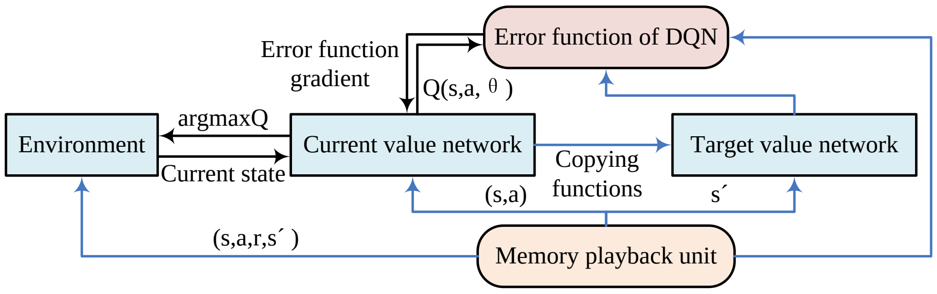


Fig. 2. Training process of DQN.

In Fig. 2, for each step reached, the parameters of the current value network are assigned to the target value network. In the playback memory unit, several samples are selected and their states are fed into the current value network. In the network output results, the Q value corresponding to the sample action is extracted and the target value is calculated. The loss function of Q value and target value is calculated. The current value network is updated by back-propagation. Although the DQN algorithm uses DNN instead of Q-table to approximate the SAVF, it still has overestimation issues [16-18]. The selection and evaluation of actions in network parameter updates are separated. Different value function networks are used to express action selection and evaluation. This can alleviate the problem, which is known as the DDQN algorithm. At this point, the calculation of the time difference optimization objective is shown in Eq. (9).

$$Y_t^{DDQN} = r + \gamma Q\left(s', \arg\max Q\left(s', a', \theta\right), \theta^-\right) \tag{9}$$

In Eq. (9), Y_t^{DDQN} represents the optimization objective. θ and θ^- represent estimated network parameters and target network parameters, respectively.

B. RPP based on Improved DDQN

In DDQN, the overestimation problem has a negative impact on the selection of the optimal action for robots, making it difficult to find the optimal action strategy and path. At the same time, as the interaction between robots and the environment deepens, the playback proportion of important samples decreases when playing back experience samples, resulting in the decline of robot learning effect. To address the above issues, an improved DDQN for RPP is proposed. The RPP training model is illustrated in Fig. 3.

From Fig. 3, the sample data is fed into the memory cache unit based on the current environment, the next state, and the obtained instant returns. Then, samples are selected from memory buffer units to train the parameters of the IDDQN. Then, based on the network output and improved exploration strategy, the optimal action is selected. The robot is sent to the next state [19-20]. In DDQN, there is a situation where the absolute error values of the value functions of the optimal and suboptimal actions are equal, resulting in the robot selecting a suboptimal action. To avoid the above issues, the range of error values can be reduced. To minimize the error, the ε-Greedy strategy is introduced into DDQN. The improved optimization objective is shown in Eq. (10).

$$Y_t^{IDDQN} = \begin{cases} r + \gamma Q\left(s', \arg\max Q\left(s', a', \theta\right), \theta^-\right) & \text{Probability=1-}\varepsilon' \\ r + \gamma Q\left(s', a', \theta^-\right) & \text{Probability} = \varepsilon' \end{cases} \tag{10}$$

In Eq. (10), ε' represents a parameter, which is a fixed value, with a value range of $(0,1)$. a' represents a random action. To better select actions, a ε-Greedy action selection strategy based on prior knowledge is proposed. The probability of action selection for strategy ε-Greedy is shown in Eq. (11).

$$\pi\left(a|s\right) = \begin{cases} 1 - \varepsilon + \dfrac{\varepsilon}{\left|A(s)\right|} & a = \arg\max_a Q(s,a) \\ \dfrac{\varepsilon}{\left|A(s)\right|} & a \neq \arg\max_a Q(s,a) \end{cases} \tag{11}$$

In Eq. (11), $A(s)$ represents the set of actions in state s. ε represents the exploration factor. When the robot selects actions, a random number is generated. If the random number is less than the exploration factor, the robot randomly selects actions. Otherwise, the average Q value of previous generations will be calculated. Based on improved ε- Greedy strategy, corresponding actions are selected and executed. The average Q value is calculated as Eq. (12).

$$Q^A(s,a) = \frac{1}{K}\sum_{k=1}^{K} Q\left(s, a, \theta_{t-k}\right) \tag{12}$$

In Eq. (12), K represents the algebraic difference between the current parameter and the previous parameter. $Q^A(s,a)$ represents the average Q value of the previous K -generations. In model training, the playback frequency of experience fragments with low time difference error values is low due to the influence of environmental noise. The lack of diversity in training samples leads to over fitting issues. Combining greedy finite and uniform sampling can ensure the playback probability of experience fragments with low time difference error values. The calculation formula for playback probability is shown in Eq. (13).

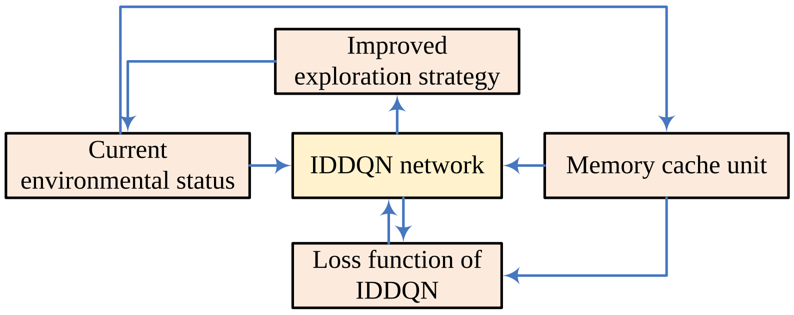


Fig. 3. Training model for robot path planning.

$$P_i = \frac{p_i^{\alpha}}{\sum_k p_k^{\alpha}} \tag{13}$$

In Eq. (13), $P(i)$ represents the playback probability of the i-th segment. p_i represents the priority of the i-th segment. α represents the degree of control priority. The priority calculation formula is shown in Eq. (14).

$$p_i = 1/rank_i \tag{14}$$

In Eq. (14), $rank_i$ represents the sequence number sorted by the absolute value of time difference error. To prevent over fitting and ensure the diversity of experience fragments, weight is introduced for adjustment. The weight calculation formula is shown in Eq. (15).

$$w_i = 1\bigg/\left(\frac{p_i}{p_{\min}}\right)^{\beta} \tag{15}$$

In Eq. (15), $p_{\min}$ represents the minimum probability of the experience segment. β represents the correction degree. At this time, the expression of loss function is shown in Eq. (16).

$$L = \sum w(t)\left(Y_t^{IDDQN} - Q(s,a,\theta)\right)^2 \tag{16}$$

In Eq. (16), $Q(s,a,\theta)$ stands for the output Q of the estimated network. The IDDQN algorithm process is shown in Fig. 4.

From Fig. 4, the IDDQN algorithm first initializes the estimated network parameters, target network parameters, and experience playback pool. The event is cycled. Then the state of each cycle is initialized and cycled throughout the time cycle. Actions are randomly selected based on probability. If a small probability event does not occur, the action with the highest current value function is selected and executed. Next, based on the reward feedback of the environment and the transferred state, the time difference error value is calculated. The calculation results are arranged in order of size. According to the sequence number, priority is calculated. Based on priority, the sampling probability is obtained. Then the correction weight is calculated by sampling probability, and the experience fragments are put into the experience playback pool according to the probability. Next, samples are selected based on probability from the experience replay pool and optimization objectives are calculated. Then, the estimated parameters are updated according to the calculation results of the loss function. The target parameters are replaced by these parameters. The application process of IDDQN algorithm in robot path planning is shown in Fig. 5.

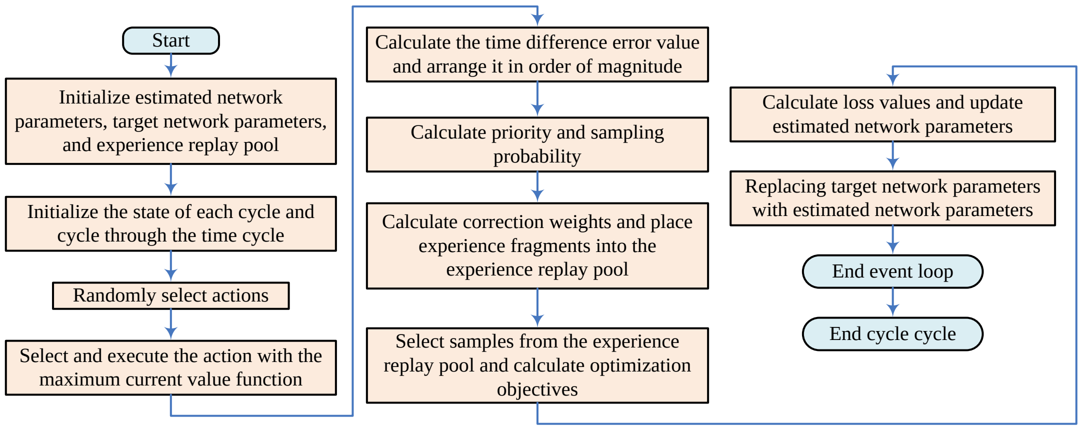

Start
Initialize estimated network parameters, target network parameters, and experience replay pool
Initialize the state of each cycle and cycle through the time cycle
Randomly select actions
Select and execute the action with the maximum current value function
Calculate the time difference error value and arrange it in order of magnitude
Calculate priority and sampling probability
Calculate correction weights and place experience fragments into the experience replay pool
Select samples from the experience replay pool and calculate optimization objectives
Calculate loss values and update estimated network parameters
Replacing target network parameters with estimated network parameters
End event loop
End cycle cycle

Fig. 4. Flow of IDDQN algorithm.

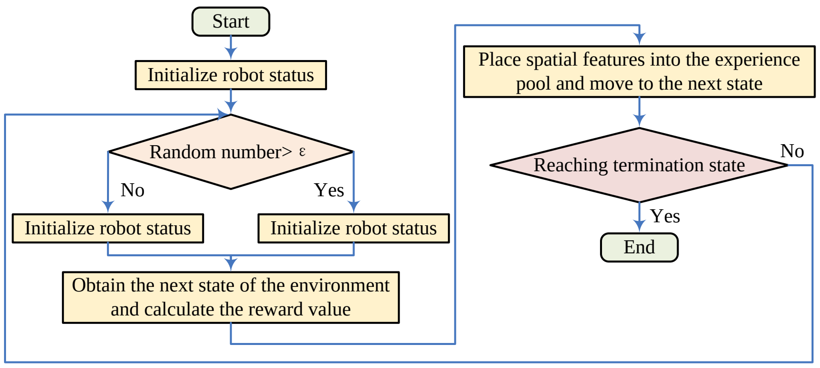

Start
Initialize robot status
Random number> ε
No
Yes
Initialize robot status
Initialize robot status
Obtain the next state of the environment and calculate the reward value
Place spatial features into the experience pool and move to the next state
Reaching termination state
No
Yes
End

Fig. 5. Application process of IDDQN in RPP.

In Fig. 5, this algorithm first initializes the state of the robot. The current coordinates of the robot are determined. Then the generated random numbers are compared. If it is greater than the exploration factor, the average output value of the improved optimization objective network is calculated and an action is selected. Otherwise, the action is randomly selected. Then the next state of the environment is obtained and its reward value is calculated. Then the spatial features are placed in the experience pool and moved to the next state. Finally, the current state is judged. If the termination state is reached, the process ends. Otherwise, the operation is returned to the second step.

IV. Simulation Experiments and Result Analysis Based on Grid Environment and Gazebo Environment

To verify the path planning ability of the IDDQN, testing experiments are carried out in both grid and Gazebo environments. The proposed method is compared with DDQN algorithm and adaptive Ant colony optimization algorithms. The grid environment is divided into simple environment, complex environment, and random environment. The processor used in this simulation experiment is Xeon (R), the CPU is NVIDIA GeForce GTX 1080Ti, the running memory is 32 GB, and the software environment is Python and TensorFlow. The exploration factor for grid and Gazebo environments has an initial value of 1 and an end value of 0.1. The convergence of DDQN, adaptive ant colony algorithm, and IDDQN algorithm is shown in Fig. 6.

From Fig. 6, the DDQN value converges after approximately 270 iterations, with a loss value of approximately 0.24. The adaptive Ant colony optimization algorithm starts to converge after about 220 iterations, and the loss value is about 0.21. IDDQN begins to converge after approximately 180 iterations, with a loss value of approximately 0.19. The convergence rate of IDDQN is faster. The RPP results of the three methods in a simple environment are shown in Fig. 7.

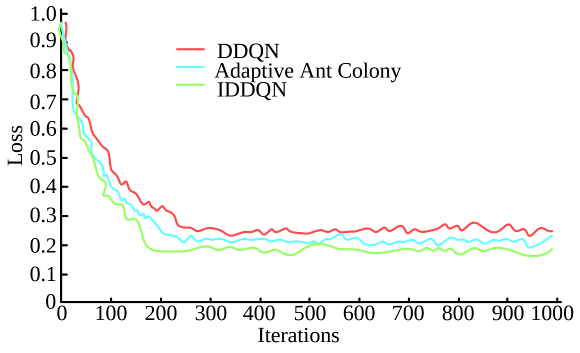
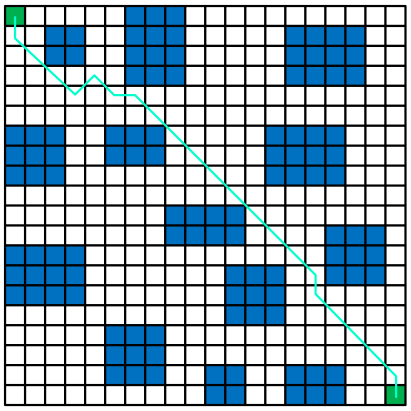
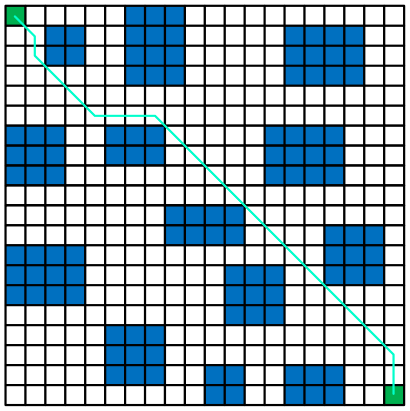


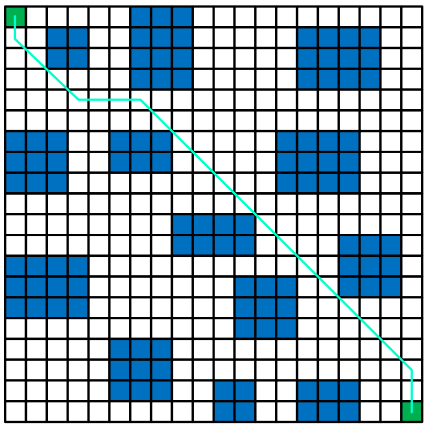
Fig. 6. Convergence of DDQN, adaptive ant colony, and IDDQN.



(a)DDQN Planning Path

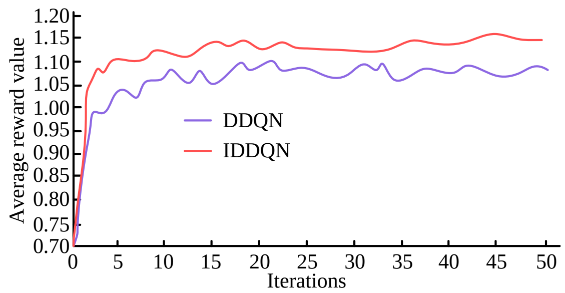


(b)Adaptive Ant Colony Planning Path

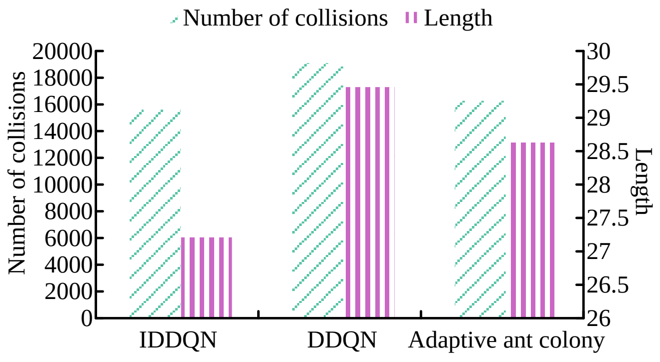


(c)IDDQN Planning Path

Fig. 7. RPP results of three methods in a simple environment.



(a)Average reward value of DDQN and IDDQN



(b)Path length and collision frequency

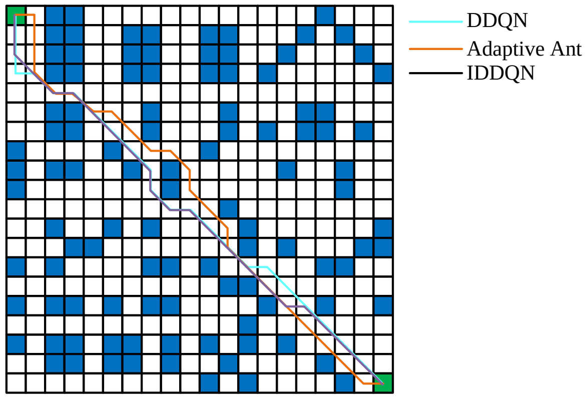
Fig. 8. The average reward values of IDDQN and DDQN, as well as the path length and collision frequency of the three algorithms.

From Fig. 7 (a), the path inflection points planned for DDQN are 8. In Fig. 7(b), the path inflection points planned by the adaptive Ant colony optimization algorithms are 5. In Fig. 7(c), the inflection points in the planned path of IDDQN are 4. IDDQN has fewer inflection points in the planned path, which can effectively reduce the movement time from the starting to the target position. The average reward values of IDDQN and DDQN, as well as the path lengths and collision times of the three algorithms, are shown in Fig. 8.

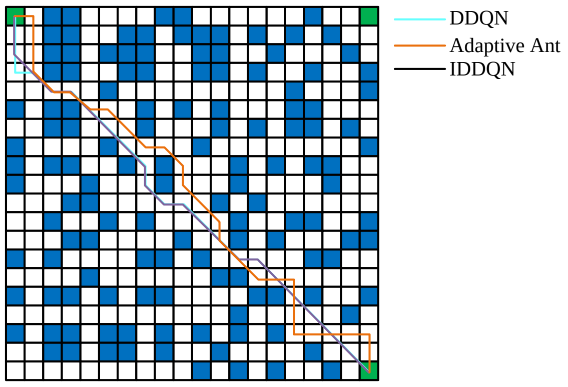
In Fig. 8(a), the average reward value of DDQN is approximately 1.03. IDDQN has an average reward value of approximately 1.12, which is higher than the DDQN algorithm. In Fig. 8(b), the path length planned by the DDQN is approximately 29.46m, and the number of obstacle collisions is 19806. The optimal path length of the adaptive Ant colony optimization algorithms is about 28.63m, and the collisions with obstacles are 16275. The optimal path length for IDDQN planning is approximately 27.21m, and the number of collisions with obstacles during training is 15613. The IDDQN algorithm not only has the shortest planned path length, but also reduces the probability of robot collision with obstacles. The RPP results of the three methods in complex environments are illustrated in Fig. 9.

In Fig. 9(a), the inflection points planned by the DDQN algorithm are 10, the inflection points of the path planned by the adaptive Ant colony optimization algorithms are 13. The path inflection points planned by IDDQN are 9. From Fig. 9(b), in complex environment 2, the path inflection points of both the ant colony algorithm and DDQN are redundant with IDDQN. The length is also longer than the IDDQN algorithm. From this, in complex environments, the IDDQN algorithm can still move to the endpoint with as few inflection points as possible. In a complex environment, the path lengths and collision times of the three algorithms, as well as the reward values of IDDQN and DDQN, are shown in Fig. 10.

From Fig. 10(a), the optimal path length for DDQN planning is approximately 29.22 m, and the number of collisions during training is 26671. The optimal path length for the adaptive Ant colony optimization algorithms is about 29.80 m, and the number of collisions is 19374. The optimal path length for IDDQN is approximately 28.63m and the number of collisions is 17389. In Fig. 10(b), the average reward values for DDQN and IDDQN are approximately 0.92 and 1.02, respectively. The average reward value of IDDQN is higher. IDDQN can still maintain a small collision probability in complex environments. The success rates and path length differences of the three algorithms in a random environment are shown in Fig. 11.



(a) Complex Environment 1



(b) Complex Environment 2

Fig. 9. RPP results in complex environments.

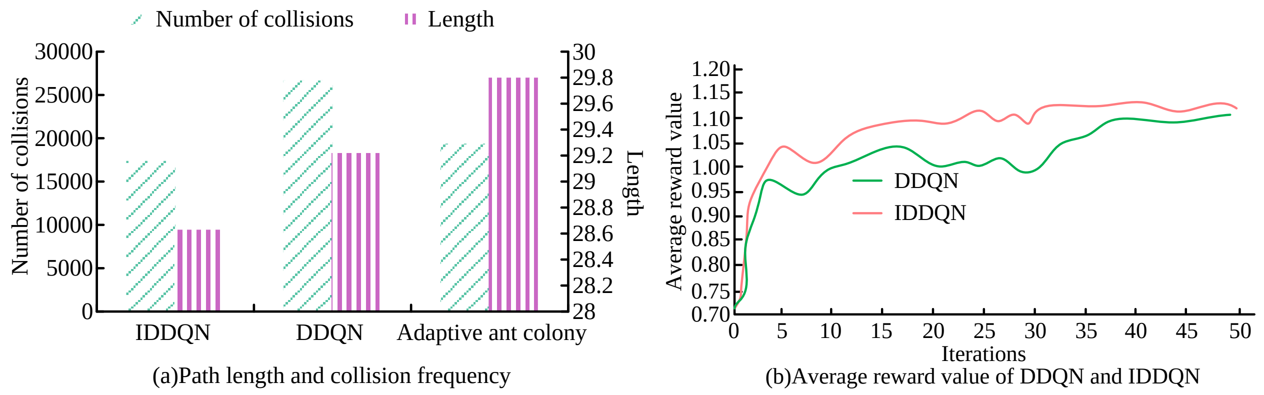


Fig. 10. The path length, collision frequency, and reward values for IDDQN and DDQN of the three algorithms are shown in the figure.

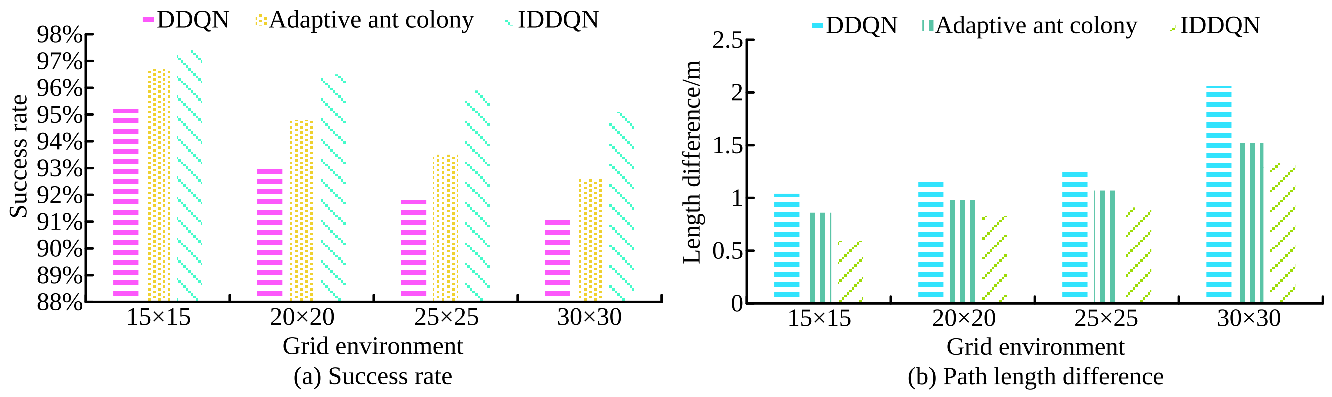


Fig. 11. Success rates and path length differences of three algorithms in a random environment.

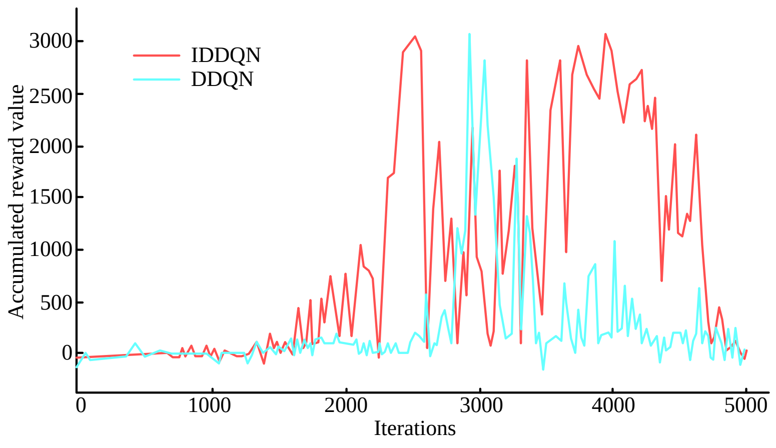


Fig. 12. Accumulated reward values of DDQN and IDDQN.

In Fig. 11 (a), in a random environment, the probability of DDQN successfully reaching the destination without colliding with obstacles is approximately 95.2%, 93.1%, 91.8%, and 91.2%, respectively. The success rate of adaptive Ant colony optimization algorithms in random environment is about 96.7%, 94.8%, 93.5% and 92.6% respectively. The success rates of IDDQN in random environments are approximately 97.4%, 96.5%, 95.9%, and 95.1%, respectively. IDDQN has the highest success rate in a random environment. In Fig. 11(b), the path length difference of DDQN in different environments is about 1.04 m, 1.17 m, 1.25 m, and 2.06 m, respectively. The path length difference of the adaptive Ant colony optimization algorithms is about 0.86 m, 0.98 m, 1.07 m and 1.52 m respectively. The difference in path length for IDDQN is approximately 0.59 m, 0.83m, 0.91m, and 1.33m, respectively. The path length difference of IDDQN is the smallest, indicating that IDDQN has a stronger ability to plan the optimal path. The CRVs of DDQN and IDDQN in the Gazebo environment are shown in Fig. 12.

In Fig. 12, the CRV of IDDQN is significantly higher than that of DDQN. The average CRV of DDQN is the highest in the range of 2500 to 3000 iterations. At this point, the average CRV is approximately 76.12. Secondly, there is an interval of

3000 to 3500 iterations, with an average CRV of approximately 29.83. The above two intervals are also the only ones with a positive average CRV. The average CRV of IDDQN is highest in the range of 3500 to 4000 iterations. The average CRV is around 748.62. Next is the interval of 4000 to 4500 iterations, with an average CRV of approximately 584.46. When the number of iterations reaches 2000, the average CRV is all positive. The average CRV between the iterations of 2000-2500 and 3000-3500 exceeds 100.

V. Conclusion

A RPP algorithm based on the improved DDQN - IDDQN algorithm is proposed for the mobile RPP problem of intelligent robots in unknown environments. Simulation experiments are conducted in both grid and Gazebo environments. According to the results, IDDQN begins to converge after approximately 180 iterations. The loss value is approximately 0.19. The convergence rate is faster than DDQN algorithm and adaptive Ant colony optimization algorithms. In a simple grid environment, the optimal path length, inflection points, and collisions during training for IDDQN are 27.21m, 4, and 15613, respectively. In a complex grid environment, the optimal path length, number of inflection points, and number of collisions during training planned by the IDDQN algorithm are 28.63m, 9, and 17389, respectively. In a random environment, the path length differences of IDDQN are 0.59m, 0.83m, 0.91m, and 1.33m, respectively. It is less than the path planned by DDQN and adaptive Ant colony optimization algorithms. The success rates of the IDDQN algorithm in random environments are approximately 97.4%, 96.5%, 95.9%, and 95.1%, respectively, which are higher than other algorithms. In the Gazebo environment, the interval with the highest average cumulative reward value for DDQN is between 2500 and 3000 iterations. At this point, the average cumulative reward value is about 76.12, and there are only two intervals where the average cumulative reward value is positive. The average cumulative reward value of DDQN is the highest in the range of 3500 to 4000 iterations. The average cumulative reward value is around 748.62. When the number of iterations reaches 2000, the average CRV is all positive. The average CRV of two intervals exceeds 100. The above results indicate that the intelligent RPP based on IDDQN can achieve optimal RPP in unfamiliar environments. Research has achieved static path planning for robots by improving DDQN. However, the proposed method does not take into account the dynamic variable environment. At the same time, when several robots cooperate to complete a certain work, the proposed algorithm cannot provide reasonable task assignment and collaboration instructions for the robot. Therefore, future work will focus on path planning, multi-robot collaborative task assignment, and path planning in dynamically variable environments.

Acknowledgment

The research is supported by: 2022 China University Industry Education Research Innovation Fund: "Research and Practice on the Cultivation Model of Composite Talents in Artificial Intelligence Technology Application from the Perspective of Industry Education Integration (No. 2021BCE02013)".

References

[1] J. Zan. "Research on robot path perception and optimization technology based on whale optimization algorithm," J. Comput. Cognitive Eng., vol. 1, no. 4 pp. 201-208, Aug. 2022.

[2] G. Campos, N. H. El-Farra, and A. Palazoglu, "Soft actor-critic deep reinforcement learning with hybrid mixed-integer actions for demand responsive scheduling of energy systems," Ind. Eng. Chem. Res., vol. 61, no. 24, pp. 8443-8461, Apr. 2022.

[3] Y. Yang and. X. Song, "Research on face intelligent perception technology integrating deep learning under different illumination intensities," J. Comput. Cognitive Eng., vol. 1, no. 1, pp. 32-36, Aug. 2022.

[4] A. K. Rath, D. R. Parhi, H. C. Das, P. B. Kumar, and M. K. Mahto, "Design of a hybrid controller using genetic algorithm and neural network for path planning of a humanoid robot," Int. J. Intell. Unmanned Syst.,vol. 9, no. 3, pp. 169-177, May. 2021.

[5] B. Nie, Y. Gao, Y. Mei, and F. Gao, "Capability iteration network for robot path planning," Int. J. Robotics Automation, vol. 37, no. 3, pp. 266-272, Apr. 2022.

[6] G. Wang and J. Zhou, "Dynamic robot path planning system using neural network," J. Intell. Fuzzy Syst., vol. 40, no. 2, pp. 3055-3063, Feb. 2021.

[7] A. Oultiligh, H. Ayad, A. E. Kari, M. Mjahed, and N. E. L. Gmili, "A hybrid PSO-GWO algorithm for robot path planning in uncertain environments," Int. Rev. Automat. Contr., vol. 14, no, 6, pp. 360-372, Aug. 2021.

[8] F. A. Raheem and M. I. Abdulkareem, "Development of A* algorithm for robot path planning based on modified probabilistic roadmap and artificial potential field," J. Eng. Sci. Technol., vol. 15, no. 5, pp. 3034-3054, Oct. 2020.

[9] W. Guanzheng, X. Yinbo, L. Zhihong, X. Xin, W. Xiangke, and Y. Jiarun, "Integrating human experience in deep reinforcement learning for multi-UAV collision detection and avoidance," Ind. Robot, vol. 49, no. 2, pp. 256-270, Sept. 2022.

[10] D. Cao, W. Hu, X. Xu, Q. Wu, Q. Huang, Chen Z., and F. Blaabjerg, "Deep reinforcement learning based approach for optimal power flow of distribution networks embedded with renewable energy and storage devices," J. Mod. Pow. Syst. Clean Energy, vol. 9, no. 5, pp. 1101-1110, Jun. 2021.

[11] J. Zhao, F. Li, S. Mukherjee, and D. Sticht, "Deep reinforcement learning based model-free on-line dynamic multi-microgrid formation to enhance resilience," IEEE Trans. Smart Grid, vol. 13, no. 4, pp. 2557-2567, Jul. 2022.

[12] D. Zhang, J. Zhao, Y. Zhang, and Q. Zhang, "Intelligent train control for cooperative train formation: A deep reinforcement learning approach," Proc. Inst. Mech. Eng., Part I: J. Syst. Contr. Eng., vol. 236, no. 5, pp. 975-988, Dec. 2022.

[13] S. A. M. Shihab and P. Wei, "A deep reinforcement learning approach to seat inventory control for airline revenue management," J. Revenue Pricing Manage., vol. 21, no. 2, pp. 183-199, Mar. 2022.

[14] Y. Zheng, J. Tao, Q. Sun, H. Sun, M. Sun, and Z. Chen, "An intelligent course keeping active disturbance rejection controller based on double deep Q-network for towing system of unpowered cylindrical drilling platform," Int. J. Robust and Nonlinear Contr., vol. 31, no. 17, pp. 8463-8480, Aug. 2021.

[15] Y. Dai, K. D. Lee, and S. G. Lee, "A real-time HIL control system on rotary inverted pendulum hardware platform based on double deep Q-network," Measure. Contr., vol. 54, no. 3-4, pp. 417-428, Mar. 2021.

[16] X. Tao and A. S. Hafid, "Deep sensing: a novel mobile crowd sensing framework with double deep Q-network and prioritized experience replay," IEEE Internet Things J., vol. 7, no. 12, pp. 11547-11558, Sept. 2020.

[17] Q. Fang, X. Xu, and D. Tang, "Loss-based active learning via double-branch deep network," Int. J. Advan. Robotic Syst., vol. 18, no. 5, pp. 538-550, Sept. 2021.

[18] J. Zhao, T. Qu, and F. Xu, "A deep reinforcement learning approach for autonomous highway driving," IFAC-PapersOnLine, vol. 53, no. 5, pp. 542-546, May. 2020.

[19] L. Xu, M. Cao, and B. Song, "A new approach to smooth path planning of mobile robot based on quartic Bezier transition curve and improved PSO algorithm," Neurocomputing, vol. 473, no. 7, pp. 98-106, Feb. 2022.

[20] Y. Yang, Z. Lin, M. Yue, G. Chen, and J. Sun, "Path planning of mobile robot with PSO-based APF and fuzzy-based DWA subject to moving obstacles," Trans. Inst. Measure. Contr., vol. 44, no. 1, pp. 121-132, Jul. 2022.