CAAI Transactions on Intelligence Technology

Research Article

Multi-robot path planning based on a deep reinforcement learning DQN algorithm

ISSN 2468-2322
Received on 18th February 2020
Revised on 29th April 2020
Accepted on 18th June 2020
doi: 10.1049/trit.2020.0024
www.ietdl.org

Yang Yang[1], Li Juntao[2] ✉, Peng Lingling[2]

[1]*School of Management, China University of Mining and Technology (Beijing), Beijing, People's Republic of China*
[2]*School of Information, Beijing Wuzi University, Beijing, People's Republic of China*
✉ *E-mail: ljtletter@126.com*

Abstract: The unmanned warehouse dispatching system of the 'goods to people' model uses a structure mainly based on a handling robot, which saves considerable manpower and improves the efficiency of the warehouse picking operation. However, the optimal performance of the scheduling system algorithm has high requirements. This study uses a deep Q-network (DQN) algorithm in a deep reinforcement learning algorithm, which combines the Q-learning algorithm, an empirical playback mechanism, and the volume-based technology of productive neural networks to generate target *Q*-values to solve the problem of multi-robot path planning. The aim of the Q-learning algorithm in deep reinforcement learning is to address two shortcomings of the robot path-planning problem: slow convergence and excessive randomness. Preceding the start of the algorithmic process, prior knowledge and prior rules are used to improve the DQN algorithm. Simulation results show that the improved DQN algorithm converges faster than the classic deep reinforcement learning algorithm and can more quickly learn the solutions to path-planning problems. This improves the efficiency of multi-robot path planning.

1 Introduction

Currently, the most popular direction of development in intelligent storage systems is the unmanned storage system based on the 'goods to people' model [1, 2]. In the picking mode of this model, the unmanned warehouses include automated intelligent three-dimensional warehouses where goods are transported by aisle stackers or shuttles or by automated guided robots based on movable shelves. The Kiva system, for example, was acquired by Amazon in 2012 [3–5]. The intelligent robot system produced by the company is a typical unmanned warehouse system, as shown in Fig. 1. This system contains thousands of movable storage shelves and hundreds of handling robots, which are movable according to the order task. The rack is lifted and transported to the picking station and handed to the staff for picking, sorting, and packaging. This technology helps retailers complete online order processing quickly and with less manpower, thereby greatly improving the execution of order tasks. The success of Kiva robots has shown the world that the deployment of handling robots and reasonable scheduling [6, 7] in warehouses produces efficiency improvements and has driven a contemporary round of new warehouse logistics technology [8] that is beginning to rapidly advance the process of warehouse automation [9]. A large number of handling robots perform tasks in the warehouse in an orderly manner with the support of an unmanned warehouse scheduling system. The research background of this study is based on movable shelves in basic intelligent warehouses where goods are transported through logistics-handling robots. The intelligent mobile robots [automated guided vehicles (AGVs)] with load capacity used in unmanned storage systems transform the traditional 'people-to-goods' picking mode into the 'cargo-to-person' model. These AGVs can optimise the path planning and execution of order tasks through optimisation algorithms, greatly improving the service efficiency and quality of the storage industry, and can have a profound effect on the operating efficiency and economic benefits of the entire storage system.

Relevant domestic scholars and companies have carried out much research on mobile robot path planning under different environments and different problems. In these studies, the process of calculating path planning considers coordination and cooperation between robots and can be roughly divided into two types of solutions: centralised and distributed. Centralised scheduling is the overall path planning of all robots by a server to achieve global optimisation, with emphasis placed on the autonomous learning and decision-making of robots and robot interaction and cooperation to achieve a local optimum. Commonly used robot path-planning methods [10, 11] mainly include the artificial potential field method [12], the path coding-based genetic algorithm [13], the random search ant colony algorithm [14], neural network and reinforcement learning methods based on learning training [15], grid maps [16], particle swarms [17], the A* algorithm [18], and models that mix these methods. For this study, we use the reinforcement learning method based on the reinforcement learning of a single robot. The ideas and algorithms in this method are combined by states, actions, and strategy allocation to achieve a multi-robot continuation and expansion of learning training. In the path-planning problem of robots in unknown environments, Hu and Jun [19] and others added the rolling Q-learning method to improve the algorithm to a certain extent when a large state space causes the curse of dimensionality. Fang and Li [20] proposed a heuristic reinforcement learning method based on state backtracking for the time-consuming problem of reinforcement learning algorithm strategy selection; this method improved action selection strategies, introduced cost functions, and integrated heuristic functions. The reinforcement learning method of cost learning makes the reward and the cost reach a balanced state and applies the balance to the robot path-planning experiment, thus verifying the feasibility and efficiency of the algorithm. The learning algorithm is combined with [21], and the combined algorithm is applied to the trajectory tracking problem of the robot in a real environment. Similarly, Maeda *et al.* also used the fitting ability of neural networks and the decision-making ability of reinforcement learning to conduct robot path planning in specific environments [22]. The Q-learning algorithm in reinforcement learning does not require prior knowledge of the environment, and thus, the agent can autonomously build a complex dynamic environment and an interactive relationship; in short, path

Fig. 1 *Kiva system intelligent storage system*

planning is becoming an important research focus in the field of algorithms [23, 24].

This study builds a multi-robot path-planning model based on an improved deep Q-network (DQN) algorithm. The study first notes that when multi-robot systems perform path planning, it is necessary to consider not only how a single robot can have the shortest optimal route but also how all the robots can work in overall coordination with each other. Therefore, a deep reinforcement learning (DRL) algorithm is used. However, due to the special nature of the problem at hand, the DQN algorithm suffers from the disadvantages of slow convergence and excessive randomness during training. Therefore, a priori knowledge is introduced into this algorithm, and a priori rules are formulated. The improved algorithm is more applicable to the problem addressed in this study and improves overall learning efficiency.

2 Analysis of multi-robot path-planning problems and optimisation methods

2.1 Analysis of multi-robot path-planning problems

This study considers the environment map as a concrete exemplar of the reinforcement learning problem. The first reason for the applicability of this map is that the blueprint of the warehouse has been rasterised, the locations are relatively fixed, and the map size is limited, which is consistent with the general requirements of reinforcement learning [i.e. a Markov decision process (MDP)]. The second reason is that the learned knowledge can be shared in the database, saving on the resource consumption of path calculation when the system is running. The Q-learning algorithm in reinforcement learning is used as the main learning problem because the method does not require an accurate environment model. In addition, the method is an off-policy algorithm in its use of different policies, facilitating full search without affecting the generation of optimal policies and utilising experience to a greater extent than on-policy algorithms. These two points clarify the appropriateness of the Q-learning algorithm for learning in the path-planning problem.

Since the state information of the set environment changes with the respective actions of multiple robots, it is inevitable that the complexity of the learning strategy increases exponentially as the dimensions of the state and action increase. There are two problems: one is that the computational complexity is increased, and the other is that the learning efficiency is reduced. The process of AGV path planning must consider avoidance of collisions with static obstacles (shelves, workbenches etc.) and dynamic obstacles (other AGVs) and try to get as close to the target point as possible while covering the shortest distance at the fastest speed. Therefore, this study combines the current motion information of all AGVs into an action vector and uses this combined action to ensure that the robots of the entire system reach their respective target points in a coordinated manner. To ensure that the environmental information is constantly changing, the state information of all AGVs is also incorporated in a state vector. In this way, in the entire system, the Q-value tables of each AGV are all merged into combined action vectors and combined state vectors mapped to Q-values so that the actions and states of other AGVs in the system can be taken into account when the current AGV makes decisions.

2.2 DQN algorithm principle and path-planning problem analysis

DRL, which is the product of the combination of deep learning algorithms and reinforcement learning algorithms, integrates the strong understanding of perceptual problems of deep learning algorithms with the ability to fit learning results of reinforcement learning algorithms. These features make it suitable for large-scale and complex problems in real-world scenarios. The Q-learning algorithm selects the optimal strategy by constructing a Q-value table. However, most problems occur in practical environments where the state space is excessively large and the dimensions become large, making it impossible to use tables to record and index, which leads to the curse of dimensionality, i.e. due to the weak perception ability of the Q-learning algorithm, robots or agents cannot process Q-learning inputs because the output of a control value function in a high-dimensional state is difficult to generalise to large state space. Therefore, in the spatial learning of large-scale states, a convolutional neural network is added to the algorithms feature extraction and reinforcement learning capability. This combination of decision-making capabilities, i.e. DRL algorithms and using a Q-network to fit the results and obtain the output or decision, allows the agent to perceive and establish action strategies in more complex environments, thereby improving the convergence and generalisation capabilities of the algorithm, increasing the learning speed, and enabling agents to perform good path planning in unknown and complex environments. For these reasons, this study uses the DQN algorithm in the DRL algorithm, which combines the Q-learning algorithm, an empirical playback mechanism, and the method of generating the target Q-value based on a convolutional neural network.

The DQN algorithm is a method of DRL. The rationale for using the DQN algorithm is that it can combine deep learning and reinforcement learning algorithms. The Q-learning algorithm constructs an objective function that can be used for deep learning. The convolutional neural network generates the target Q-value, evaluates the Q-value of the next state based on the target Q-value in this state, and adds an empirical playback mechanism and the target network to break the association between the data. In other words, the DQN algorithm can use the value function to approximate the Q-value. It uses a function, $f(s, a, \omega)$, to represent the Q-value, $Q(s, a) = f(s, a, \omega)$.

Here, the function f represents any type of function and represents the function, $f(s, a, \omega)$. Since the dimension is reduced to a single Q-value through matrix operations, the dimension of the state s represented by the function is irrelevant, which is the basic idea of the value function approximation.

Take state s as the input, and then output the Q-value of each action, i.e. output a $\boldsymbol{Q}$-value vector containing all actions $[(Q(s, a_1)), (Q(s, a_2)), Q(s, a_3)\ \ (Q(s, a_4)), (Q(s, a_5))]$. As long as state s is input, the Q-value including all actions can be obtained. In this way, it is more convenient to select the action and update the Q-value through Q-learning. In summary, the core idea of the DQN algorithm is that when the state and action space are high-dimensional and continuous, deep learning is used to build both a value network that solves reinforcement learning tasks and a loss function [25], which is the objective function in the reinforcement learning method that calculates labels and networks. The deviation of the output minimises the loss function. To train the Q-network, one needs to provide labelled samples for it. The updated formula of the Q-learning algorithm is (1)

$$q(S_t, A_t) \leftarrow q(S_t, A_t) + a\left[R_{t+1} + \gamma \max q(S_{t+1}, A_t) - q(S_t, A_t)\right] \quad (1)$$

CAAI Trans. Intell. Technol., 2020, Vol. 5, Iss. 3, pp. 177–183

Thus, define the loss function as (2)

$$L(\omega) = E[(R_{t+1} + \gamma \max q(S_{t+1}, A_t; \omega^-) - q(S_t; A_t; \omega)^2] \quad (2)$$

During the training of the value network, historical data are continuously collected as training labels and stored in the experience pool, and then the value network is trained on the data in the experience pool by using a small batch of random gradient descent methods to optimise the weight matrix to solve the Q-value function.

2.3 Disadvantages of the DQN algorithm

The advantage of the DQN algorithm in solving high-dimensional, large-scale, and continuous state-space problems is that it can use the results of neural networks to approximately replace the Q-value function, effectively reducing the dimensionality of the input data. However, the algorithm has two shortcomings: one is the slow convergence speed, and the other is excessive randomness.

(i) *Slow convergence*: In the process of collision-free path planning for multiple robots, to ensure that all robots perform tasks in coordination with each other, the states of all robots are combined into state vectors $\boldsymbol{s} = [s_1, s_2, s_3, \ldots, s_n]$. All actions make up an action vector $\boldsymbol{a} = [a_1, a_2, a_3, \ldots a_n]$. Therefore, in the entire system, the Q-value table of each robot is mapped to the combined action vector and combined state vector so that the robot's decision-making can take into account the actions and states of other robots in the system. However, the DQN algorithm is a combination of deep learning and reinforcement learning. It has the basic characteristics of reinforcement learning; i.e. it needs to learn from scratch in an unknown environment and explore step by step. This means each path of the path planning for the robots needs to be explored and the entire map learned incrementally from scratch, which greatly increases the amount of calculation and makes the convergence rate very slow.
(ii) *Excessive randomness*: For the Q-learning algorithm, the balance between exploration and utilisation is a very basic and important concept. Exploration is the collection of information, and utilisation is the execution of actions that can produce the maximum value. Exploration can yield the best but also the worst reward value. Utilisation involves adopting the scheme that yields the maximum reward value. The action that obtains the maximum reward value must always be performed to achieve the goal with the maximum score, but this action is also based on insufficient samples or empirical information and incomplete exploration of the environment, resulting in a suboptimal strategy. When using the DQN algorithm for multi-robot path planning, we must consider not only how to obtain the solution faster but also whether the solution is optimal. This study follows the Q-learning algorithm ε-strategy to solve the path-planning problem. This strategy is based on probability $1 - \varepsilon$ (choose the action with the largest action value [greedy action]) or probability ε (choose random movements, which generate much useless exploration).

In summary, it can be seen that when training a neural network, the training data requirements are independent and identically distributed, but there is a certain correlation between the previous state data and the latter state data collected through reinforcement learning. Consequently, by using these related data for training, it is inevitable that the neural network training results become unstable, making it difficult for Q-learning algorithms to converge, and that the loss values fluctuate. To solve the problem of convergence, the DQN algorithm introduces empirical playback and the method of establishing a double-layer network structure to train the reinforcement learning process to better combine these two algorithms. Therefore, the method employed in this study combines the Q-learning method with a convolutional neural network, i.e. the DQN algorithm model, using the strong fitting ability of a convolutional neural network to process high-dimensional input information, perform the extraction, effectively reduce the input dimensions, add experience playback methods, and then decide the output results to enable collaborative learning between robots.

3 Multi-robot path-planning model based on the improved DQN algorithm

3.1 DQN algorithm improvement ideas

To address the two problems that occur during the application of the DQN algorithm to robot path planning, this study uses the introduction of prior knowledge to initialise the Q-value table. Prior knowledge is the knowledge that precedes the experience of the agent. At the same time, it avoids the exploration of the system from scratch and reduces useless exploration due to randomness. Therefore, when performing multi-robot path planning, first let the AGV run the A* algorithm with a single robot in a static environment to calculate the collision-free path from the initial point to the target point, $s^i = (s_1^i, s_2^i, s_3^i, \ldots, s_d^i)$, i.e. the static obstacle avoidance path on behalf of the ith AGV. The information obtained is used to guide subsequent path planning. This allows the AGV to have a certain understanding of the environment before learning and reduces trial and error from scratch. The computational complexity of each AGV involves choosing the optimal strategy for avoiding collisions with static obstacles using prior knowledge in the next action. When there is a conflict with other AGV resources, the corresponding Q-value is updated according to the DQN algorithm to shorten the entire learning time, and the speed of convergence significantly improves the efficiency of the algorithm.

When an AGV performs path planning, certain rules or priorities need to be set; i.e. in some special situations, when the resources of two AGVs conflict, which AGV waits and which AGV moves forward is determined by an a priori rule. In this study, some a priori rules are developed to influence the robot's decisions in some special situations because when the ε-greedy strategy is used to explore the environment, a large exploration factor, ε, is set since the environment is unknown at the beginning. This in turn generates useless exploration processes and increases training time, while prior rules can circumvent this useless learning in some cases and improve the overall efficiency of the algorithm.

The prior knowledge is set as follows:

(i) Suppose there are n AGVs in the multi-robot storage system, and the combined state vector is $\boldsymbol{s} = [s_1, s_2, s_3, \ldots, s_n]$.
(ii) The combined action vectors of n AGVs are $\boldsymbol{a} = [a_1, a_2, a_3, \ldots, a_n]$.
(iii) Let m be special state sequences in the prior rule be $\boldsymbol{j} = [j_1, j_2, j_3, \ldots, j_m]$. When the state jk in the special sequence appears, the corresponding action strategy ak' is taken. $\boldsymbol{s}$ and $\boldsymbol{j}$ both serve as the state inputs of the neural network and satisfy $\boldsymbol{s} \cap \boldsymbol{j} = \emptyset$.
(iv) The combination vector corresponding to the Q-value of the prior rule sequence is defined as $\boldsymbol{Q} = [Q_{ja1}, Q_{ja2}, Q_{ja3}, \ldots, Q_{jan}]$, where Q_{jak} is the prior Q-value of action ak'.

The set of a priori rules does not affect the learning results of system $\boldsymbol{j}$. The states that appear in the sequence affect or control the robot's action choices and do not affect the original action decisions of the DQN algorithm. According to the above rules, the state jk appears $\in \boldsymbol{j}$, according to the combined vector of corresponding $\boldsymbol{Q}$-values $\boldsymbol{Q} = [Q_{ja1}, Q_{ja2}, Q_{ja3}, \ldots, Q_{jan}]$. To select the action, if the sequence does not appear in the prior rule, the training process makes a normal decision with the ε-greedy strategy. Conversely, when the sequence jk appears in the a priori rule, the behaviour of the robot can be controlled according to the action strategy ak' to reduce the occurrence of the phenomena of useless unnecessary exploration phenomena. The ε-greedy strategy fully explores the environment without affecting the original DQN learning process. The improved algorithm flow is shown in Fig. 2.

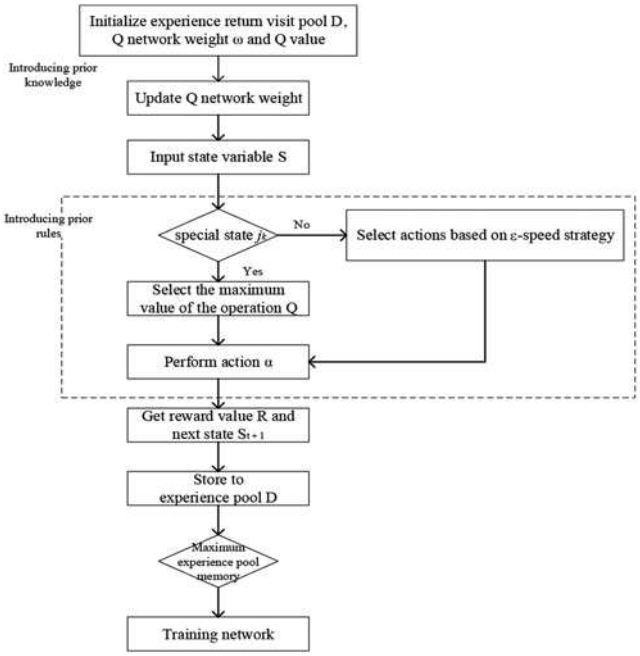

Fig. 2 *Improved algorithm flow*

4 Improved DQN algorithm flow

The following DQN algorithm flow introduces prior knowledge and prior rules:

(i) Input $\boldsymbol{s}$, $\boldsymbol{a}$, $\boldsymbol{j}$, and $\boldsymbol{Q}$. Initialise $Q(s, a)$, given parameters α, γ, and ε.
(ii) Initialise the experience playback pool d; the amount of data it can hold is n.
(iii) Initialise the Q-network and randomly generate weights ω.
(iv) Initialise the Q-target network with a weight of $\omega^{-} = \omega$.
(v) Repeat initialisation state s.

Repeat: If jk appears $\in \boldsymbol{j}$.

Select the combination vector $\boldsymbol{Q} = [Q_{ja1}, Q_{ja2}, Q_{ja3}, \ldots, Q_{jan}]$ directly corresponding to the Q-value in Ak′.

Otherwise, use the ε-greedy strategy to select action a in state s to obtain reward value R and the next state s_{t+1}. Store experience samples (s, a, R, s_{t+1}) to experience pool D. Randomly sample a small batch of stored samples (s_t, a_t, R_t, s_{t+1}) from the experience pool D.

Assume

$$y_i = \begin{cases} R_i, & \text{Experience trajectory ends at step } i+1 \\ R_i + \gamma \max Q\left(S_i + 1, a', \omega^{-}\right), & \text{Non-terminating step} \end{cases}$$

Update the loss function using the gradient descent method $y_i - Q\left(S_t, a_t, \omega\right)^2$ with the middle ω.

Update every c steps ω^{-}, $\omega^{-} = \omega$.

5 Simulation experiment of multi-robot path planning based on the improved DQN algorithm

When performing AGV path planning for a multi-agent-based unmanned warehouse scheduling system, this study makes the following assumptions about the overall environment, AGVs etc. (i) For AGVs, the state of the storage environment is semi-known and there are structures of static obstacles (such as shelves and workbenches) as well as mobile obstacles (other AGVs). (ii) AGVs have sensors that sense environmental information. (iii) A random task order provides cargo information and only one AGV extracts the corresponding goods according to the information of the order task. Finally, (iv) the model, specifications, and various data of each AGV in the system are consistent. In the logistics storage system, multiple AGVs can operate simultaneously, each of the shelves stores the corresponding items, and the task scheduling agent contains a large number of pending order tasks. The order-picking process of an AGV is defined by first processing and assigning outstanding orders through the task scheduling agent so that each AGV can receive one or more order tasks. Then, the AGV uses the intelligent path-planning algorithm to move the shelf of the target goods to the picking table agent according to the order, and the staff select the target goods and place them in the corresponding packing and packaging processing. Then, the AGV returns the shelves to the original position and continues to move the shelves specified by the next task order, completing all assigned tasks in turn.

5.1 MDP tuple

In this study, the Markov process is used to model the improved DQN algorithm. The MDP consists of tuples (s, a, p, r), which are defined as follows:

(i) *State space*: In a rasterised warehouse map, the coordinates of each grid represent a state of the current system, so the state space $S = ((1,1), (1,2), \ldots, (30, 30))$, i.e. $s_t = (x_t, y_t)$. The state vector $\boldsymbol{s}_t = \left(s_t^1, s_t^2, \ldots, s_t^n\right)$, $s_t^i \in S$, s_t^i represents the state of the ith AGV at time t.
(ii) *Action space*: The actions that each AGV can take are up, down, left, right, and stationary (for low-priority AGVs, to avoid resource conflicts), so the action space $a = \{1, 2, 3, 4, 0\}$, the action vector is $\boldsymbol{a}_t = \left(a_t^1, a_t^2, \ldots, a_t^n\right)$, $a_t^i \in A$, and a_t^i represents the action taken at the tth time step by the ith AGV.
(iii) *Transfer function*: When the AGV selects an action before it executes, the transfer function moves to the next grid

$$f\left(x_t^i, y_t^i, 0, x_t^i, y_t^i + 1\right) = \begin{cases} 1, & \text{The next grid space does not contain} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$f\left(x_t^i, y_t^i, 0, x_t^i, y_t^i - 1\right) = \begin{cases} 1, & \text{The next grid space does not contain} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$f\left(x_t^i, y_t^i, 0, x_t^i + 1, y_t^i\right) = \begin{cases} 1, & \text{The next grid space does not contain} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$f\left(x_t^i, y_t^i, 0, x_t^i - 1, y_t^i\right) = \begin{cases} 1, & \text{The next grid space does not contain} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

(iv) *Reward-and-punishment function*: The reward-and-punishment function that guides optimal collision-free path planning for multiple robots is immediate feedback from the environment to the mobile robot, and the function is also an evaluation of the actions performed by the robot in the previous step. Therefore, this study defines that when the AGV moves in each space on the grid, it obtains a penalty value of -1. When it reaches the endpoint, the AGV obtains the maximum reward value of 10. If the AGV encounters an obstacle or another AGV, it obtains the maximum penalty value of -10. During the entire process, the AGV chooses the reward. High-value actions can make the AGV

CAAI Trans. Intell. Technol., 2020, Vol. 5, Iss. 3, pp. 177–183

reach the endpoint faster and eventually maximise the total reward value.

$$R(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) = \begin{cases} 10, & \boldsymbol{s}_{t+1} = \text{Target} \\ -10 & \text{Encounter an obstacle or robot collision} \\ -1 & \text{otherwise} \end{cases} \tag{7}$$

5.2 Setting the initial Q-value table

Based on the improved DQN algorithm steps, a priori knowledge is introduced to initialise the Q-value table. In the experiment, the system is set with eight AGVs, and the collision-free path of each robot in the static storage system is $s^i = (s_1^i, s_2^i, \ldots, s_n^i)$. First, initialise the Q-value table to 0. When the robot is in a transition state, the action-value function is set to a reasonable number >0 so that the robot first considers a static collision-free path when exploring the route, reducing random unnecessary exploration.

5.3 Setting a priori rules

Set a priori rules suitable for the problem in this study. When two robot resources conflict, one of them needs to give way to the other robot, i.e. choose a stationary action, and the probability of the two robots choosing a stationary action is 50% sequence $\boldsymbol{j} = [j_1, j_2, j_3, \ldots, j_m]$ is defined as follows.

j_1: In a resource conflict between Robot 1 and Robot 2, $s_t^1 = s_t^2$, corresponding to action strategy $j_1 \rightarrow a1'$. This strategy is to make Robot 1 choose a stationary action with a probability of 50%; otherwise, Robot 2 chooses a stationary action.

j_2: In a resource conflict between Robot 1 and Robot 3, $s_t^1 = s_t^3$, corresponding to action strategy $j_1 \rightarrow a2'$. This strategy is to make Robot 1 choose a stationary action with a probability of 50%; otherwise, Robot 3 chooses a stationary action.

j_3: In a resource conflict between Robot 1 and Robot 4, $s_t^1 = s_t^4$, corresponding to action strategy $j_1 \rightarrow a3'$. This strategy is to make Robot 1 choose a stationary action with a probability of 5%; otherwise, Robot 4 chooses a stationary action. By extension, there are 28 resource conflict selection strategies; i.e. $m = 28$ and $\boldsymbol{j} = [j_1, j_2, j_3, \ldots, j_{28}]$.

5.4 Hyperparameter setting of the DQN algorithm

The hyperparameters of the DQN algorithm are set as follows: the learning rate α is 0.0025; the experience pool size n is 1000; and the choice of the greedy action in the ε-greedy strategy is ε. The initial exploration value is 1; the final exploration value is not <0.1; and the attenuation coefficient updated by the Q-learning algorithm is γ. The sample size for batch sampling under stochastic gradient descent is 32, so the training and parameter learning begin only when the number of experiences stored reaches 32. The Q-network updates the parameters at every time step; the update frequency of the Q-target network is 50 time steps; and the total number of training steps is 2500.

6 Comparative experimental results and analysis

6.1 Algorithm performance analysis

The simulation experiment was performed using TensorFlow deep learning tools and the Python language, and a 30×30 grid map was constructed as shown in Fig. 3. The positions of 45 groups of shelves (black areas) and five picking tables (orange areas) were set. Then, eight AGVs (yellow area) and 80 random order tasks (generated by the unmanned warehouse scheduling system) were added. The starting points of these AGVs were (0, 29), (2, 29), (4, 29), (6, 29), (8, 29), (10, 29), (12, 29), and (14, 29).

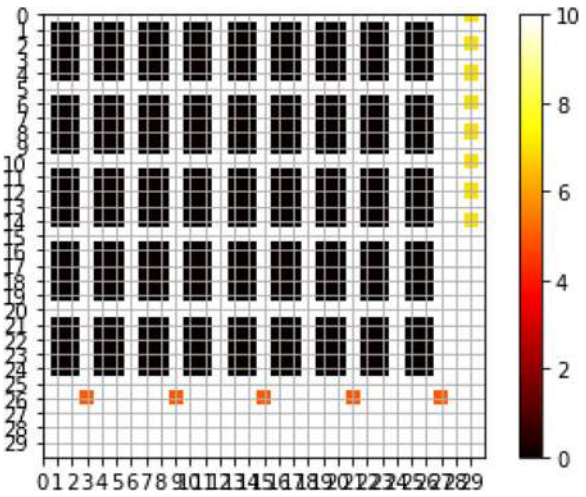

Fig. 3 *Raster map of simulation*

In this part of the experiment, there were three parts of the code. The environment model of the robot for DQN, including the action space and the state space, was written with the maze_env. py file. The network model in the DQN algorithm was written with the RL_brain.py file, and run_this.py introduced the environment model. The network model was trained and the loss function curve is drawn. The DQN algorithm is composed of a two-layer network; one layer is the eval_net estimation network, and the other is the target_net target network. The parameters of the network were approximated by the action value at each step update, and the parameters of the target network are updated every 50 steps.

For the DQN algorithm with a priori knowledge and the classic DQN algorithm, a comparison experiment was performed. To compare the convergence speed before and after the improvement of the algorithm, the training times for the loss function value convergence of the two algorithms were compared. The results are

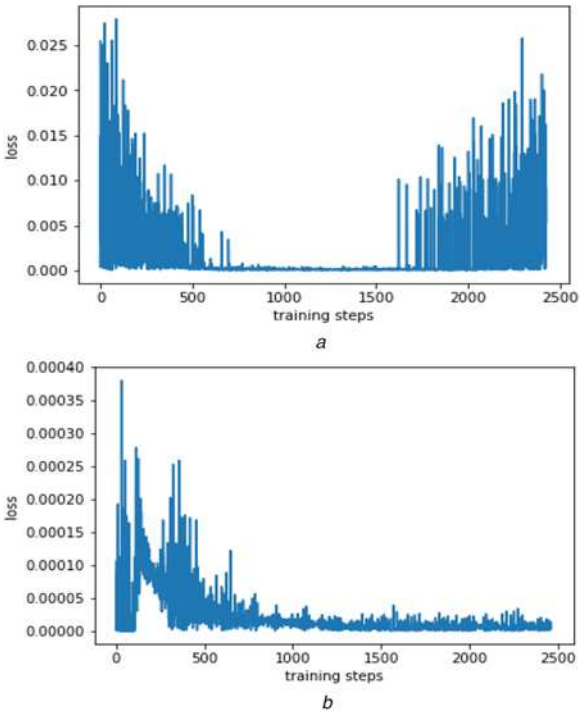

Fig. 4 *Convergence of the loss function of the classic DQN algorithm and the improved algorithm*

a, b Convergence of the loss function of the classic

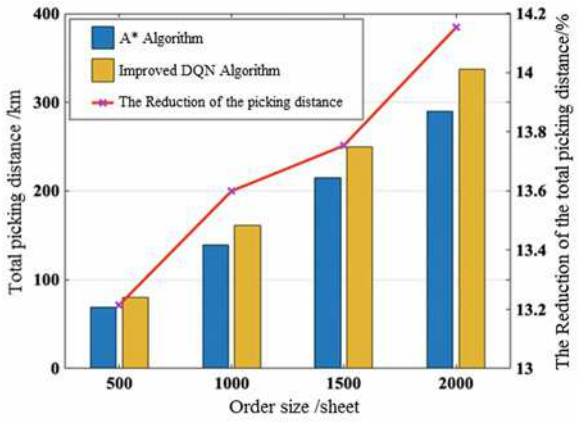

Fig. 5 *AGV total picking route*

shown in Fig. 4. Fig. 4*a* shows the results of the original algorithm and Fig. 4*b* shows the results of the improved algorithm.

6.2 Picking efficiency

The foregoing confirmed that the improved algorithm boosts the efficiency of the classic algorithm. The results of the path-planning problem were then compared in the unmanned warehouse scheduling system.

Before initialising the Q-value of the AGV, we let the AGV use the A* algorithm to plan a collision-free path in a static environment. The obtained path information is used as a priori knowledge to guide the subsequent path planning, which enables the robot system to achieve a certain understanding of the environment before learning, thereby improving the learning efficiency of the algorithm.

Therefore, regarding the analysis of picking efficiency conducted in this study, a comparison of the final optimisation result with the picking efficiency before optimisation is performed, culminating with the research conclusion that the picking efficiency is improved.

The original system used the A* algorithm for path planning for AGVs. In terms of the efficiency of the entire system, it is necessary to take the shortest total picking distance as the goal to achieve high system efficiency. Therefore, according to different order size situations, a comparative experimental analysis of the total picking distances of all AGV-completed order tasks was performed. The total picking distance was shortened by 32.7% and 34.7% under different path-planning methods as shown in Fig. 5.

7 Summary

This study considered the multi-robot path-planning problem involving high-dimensional input and analysed why the Q-learning algorithm is not applicable to this problem and why the DQN algorithm is used. Next, the improvement process of the DQN algorithm based on the problem in this study was explained. First, to address the problem of a too-slow convergence speed, an improvement of adding prior knowledge was proposed: specifically, before initialising the Q-value of the AGV, a single AGV is used in a static environment to plan a collision-free path. Then, the resulting path information is used to form a priori rules to guide subsequent path planning. This allows the multi-robot system to acquire a certain understanding of the environment before learning, thereby improving the learning efficiency of the algorithm. Second, to address the problem of excessive randomness, a certain prior rule method was formulated. When encountering resource conflicts between AGVs, a priority is set to affect the current AGV's action selection. This ensures that the multi-robot system can fully explore the map during path planning. The rule does not affect the learning process of the entire system and can avoid useless exploration and reduce the randomness of action selection to a certain extent. The above-improved method adding prior knowledge and prior rules yields a complete process for the multi-robot path-planning algorithm.

Finally, simulation experiments were carried out to improve the path-planning problem of the multi-robot system. The system was modelled from a multi-agent perspective, and the DQN algorithm was applied to the AGVs and was improved according to the actual problem, presenting a learning process that spanned the entire system. A method was designed to improve the initial Q-value table with prior knowledge that was added to allow a single AGV to learn and understand the environment before training, and special a priori rules were set in a manner equivalent to giving AGVs a certain priority and making them select a specific action strategy in these specific states. Based on a semi-known environment of a 30×30 grid map, the positions of shelves and workbenches were fixed, and the robot used the improved DQN algorithm for optimal collision-free path planning. The performance of the algorithm was compared with the classic DQN algorithm. After experimental verification, the DRL algorithm with prior knowledge and prior rules proposed in this study achieved a good performance improvement in multi-robot path-planning learning, and both the convergence speed and the learning efficiency increased. A comparison of the efficiency of the total picking distance under different order sizes was carried out in the unmanned warehouse scheduling system. The alterations described to improve the effectiveness of the method have utility in certain practical robotic applications. In addition, given that the A* algorithm has low real-time performance but is suitable for global path planning, to avoid obstacles dynamically, local path planning remains to be considered. Such inquiries are expected to be carried out in-depth in our follow-up research.

8 Acknowledgments

This research has been supported by Yueqi Youth Scholar Funding of China University of Mining and Technology (Beijing) and the Major Programme of the National Natural Science Foundation of China (No.71831001).

9 References

[1] De Koster, R., Le-Duc, T., Roodbergen, K.J.: 'Design and control of warehouse order picking: a literature review', *Eur. J. Oper. Res.*, 2017, **182**, (2), pp. 481–501
[2] Wurman, P.R., D'Andrea, R., Mountz, M.: 'Coordinating hundreds of cooperative, autonomous vehicles in warehouse', *AI Mag.*, 2016, **29**, (1), pp. 9–20
[3] Gu, J., Goetschalckx, M., McGinnis, L.F.: 'Research on warehouse operation: a comprehensive review', *Eur. J. Oper. Res.*, 2007, **177**, (1), pp. 1–21
[4] Andrea, R.D., Wurman, P.: 'Future challenges of coordinating hundreds of autonomous vehicles in distribution facilities'. IEEE Int. Conf. on Technologies for Practical Robot Applications, Woburn, MA, USA, 2008, pp. 80–83
[5] Enright, J., Wurman, P.R.: 'Optimization and coordinated autonomy in mobile fulfillment systems'. Automated Action Planning for Autonomous Mobile Robots, San Francisco, California, USA, 2011, pp. 33–38
[6] Xiaochun, W.: 'The "Cargo-to-Person" Technology of Robots in Warehouse Management–Based on the Amazon Logistics Center KIVA Orange Robot'. *Modern Economic Information*, 2017, pp. 78–80
[7] Yong, J., Yong, Z.: 'Kiva is just a kind of 'cargo to person', robots will change the logistics industry', *Robot Ind.*, 2017, **2**, pp. 57–61
[8] Jingyi, W.: 'Application analysis and prospect of Amazon warehouse Kiva robot', *Logist. Mater. Handl.*, 2015, **10**, pp. 159–164
[9] Trebilcock, B.: 'Amazon.com to acquire Kiva systems', *Mod. Mater. Handl.*, 2012, **67**, (4), p. 9
[10] Wang, X.: 'Research on path planning for multiple mobile robots in intelligent warehouse', Dalian Jiaotong University, Dalian, 2014, pp. 8–33
[11] Zhang, D., Sun, X., Fu, S., *et al.*: 'Cooperative path planning in multi-robots for intelligent warehouse', *Comput. Integr. Manuf. Syst.*, 2018, **24**, (2), pp. 410–418
[12] Huang, L., Geng, Y.: 'Robot path planning based on dynamic potential field method', *Comput. Meas. Control*, 2017, **25**, (2), pp. 164–166
[13] Wei, T., Long, C.: 'Path planning for mobile robot based on improved genetic algorithm', *J. Beijing Univ. Aeronaut. Astronaut.*, 2020, **46**, (4), pp. 703–711

[14] You, X., Liu, S., Lv, J.: 'Ant colony algorithm based on dynamic search strategy and its application on path planning of robot', *Control Decis.*, 2017, **32**, (3), pp. 552–556
[15] Mo, F.: 'Research on path planning of mobile robot based on Q learning algorithm', Beijing University of Technology School of Electronic Information and Control Engineering, Beijing, 2016
[16] Li, W., Wang, X., Zhao, Y., *et al*.: 'Unmanned crane dispatching system based on grid method in steel works', *J. Syst. Simul.*, 2020, **32**, (4), pp. 687–699
[17] Liu, Y., Chen, T., Zhang, F.: 'Mobile robot path planning based on improved particle swarm optimization', *J. Zhengzhou Univ., Nat. Sci. Ed.*, 2019, **2020**, (1), pp. 114–119
[18] Sun, W., Lv, Y., Tang, H., *et al*.: 'Mobile robot path planning based on an improved A* algorithm', *J. Hunan Univ., Nat. Sci.*, 2017, (4), pp. 94–101
[19] Hu, J., Zhu, Q.: 'Path planning of robot for unknown environment based on prior knowledge rolling Q-learning', *Control Decis.*, 2010, **29**, (9), pp. 1364–1368
[20] Fang, M., Li, H.: 'Heuristically accelerated state backtracking Q-learning based on cost analysis', *Pattern Recog. Artif. Intell.*, 2013, **35**, (9), pp. 838–844
[21] Lange, S., Riedmiller, M., Voighlander, A.: 'Autonomous reinforcement learning on raw visual input data in a real world application'. The 2012 IEEE Int. Joint Conf. on Neural Networks (IJCNN), Brisbane, Australia, 2012, pp. 1–8
[22] Maeda, Y., Watanabe, T., Moriyama, Y.: 'View-based programming with reinforcement learning for robotic manipulation'. 2011 IEEE Int. Symp. on Assembly and Manufacturing(ISAM), Tampere, Finland, 2011, pp. 69–70
[23] Tan, H., Balajee, K., Lynn, D.R.: 'Integration of evolutionary computing and reinforcement learning for robotic imitation learning'. 2014 IEEE Int. Conf. on System, Man and Cybernetics (SMC), San Diego, CA, USA, 2014, pp. 407–412
[24] Ma, L., Zhang, W., Dai, Z.: 'A review of developments in reinforcement learning for multi-robot systems', *J. Southwest Jiaotong Univ.*, 2014, pp. 1032–1044
[25] Zhou, F., Jin, L., Dong, J.: 'Review of convolutional neural network', *Chin. J. Comput.*, 2017, (6), pp. 1229–1251