2024 IEEE 7th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC) | 979-8-3503-7080-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ITNEC60942.2024.10733132

Research on Multi Robot Path Planning based on Improved DQN

Xi Liu [1],Zhonghua Wang[1*]
1.School of Electrical Engineering, University of Jinan, Shandong, China
17660101985@163.com, cse_wzh@ujn.edu.cn
Corresponding Author: Zhonghua Wang Email: cse_wzh@ujn.edu.cn

Abstract—A common and important problem in the deep reinforcement learning algorithm DQN (Deep Q-learning Network) for multi robot path planning is low sample efficiency. To solve this problem, this paper proposes a SSDD-DQN (Simple and Space based Dueling Double Deep *Q-learning* Network) method for simple and sparse intelligent factory environments. Firstly, this method uses a priority based post experience replay strategy based on chaotic sequences to optimize the later training process of the model; Secondly, an improved state space balance has been introduced ε-Greedy exploration action selection strategy to deal with the equilibrium problem of exploration and utilization of intelligent agents; Finally, a consensus based congestion avoidance mechanism inspired by social consciousness was introduced. The experiment shows that this method has high sample efficiency, low solution mobility consumption, and can significantly alleviate congestion among multiple robots.

Keywords—*smart factory; Multi robot path planning; Deep Q-network; Sample efficiency*

I. INTRODUCTION

Intelligent manufacturing is the core of the Industry 4.0 framework, and smart factories are the main implementation form of intelligent manufacturing. In smart factories, AGV (Automated Guide Vehicles) can transport materials to various parts of the factory, which is the basis for meeting the production needs of small batches and multiple varieties. Path planning is one of its core issues, responsible for the efficient, accurate, and safe movement of robots in the environment, ensuring that they can follow instructions from the starting point to the designated end point.

In recent years, with the continuous acceleration of the development of artificial intelligence[1]DRL (Deep Reinforcement Learning) has shown great potential and development prospects in path planning. This method is an artificial intelligence method that simulates human thinking and can enable agents to make more accurate judgments in environmental states [2]. DQN algorithm is a type of deep reinforcement learning algorithm that combines the ability of neural networks to process data with the decision-making ability of reinforcement learning, and is currently the mainstream algorithm[3]. Xin[4] first proposed applying DQN algorithm to robot path planning. Chaul [5] used a priority based experience replay mechanism instead of equal probability sampling to improve the utilization of valuable samples when training DQN, but it requires a large storage space. Li Hui[6] improved the algorithm's generalization by designing a four layer convolutional neural network based on the traditional DQN algorithm, replacing the traditional *Q-value table* with the network's output.

The above research has improved the performance of DQN algorithm to a certain extent, but there is still a problem of low sample efficiency [7]. Sample efficiency refers to the size of training samples required to achieve a specified effect. How to improve sample efficiency and reduce the training cost of reinforcement learning based multi robot path planning is a worthwhile research topic. Therefore, this article proposes an SSDD-DQN method to handle the multi robot path planning problem in a simple sparse intelligent factory environment. This method is based on the DQN algorithm in DRL as a whole. During the training process, the value of the samples was taken into account to optimize the agent's network update exploration actions, taking into account the repetitiveness of multi-objective tasks. A post experience replay mechanism was introduced to optimize the exploration process of later tasks, resulting in overall high sample efficiency and excellent testability. Enable a single robot to achieve autonomous navigation, while multiple heterogeneous robots can independently complete path planning goals while avoiding conflicts.

II. Model description of SSDD-DQN algorithm

A. Neural Network Structure

This method learns multi robot path planning problems by establishing a DQN network model. This method adopts a DD-DQN (Dueling Double DQN) neural network structure to handle the DRL multi robot path planning problem. Dueling Double DQN has two structurally consistent neural networks: Model and Next Model. The Model is the main model used to guide the actions of the agent, while the Next model is the target model used to obtain the objectives during the training process. The output Q values of both the Model and Next models consist of a matrix of state evaluation values A and a vector of action advantage values V. The schematic diagram of network structure 1 is as follows:

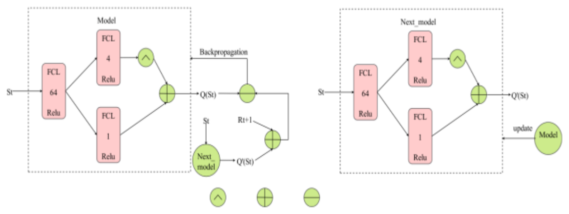

Fig.1 Schematic diagram of DD-DQN network structure

Firstly, the outputs in the Model and Next models are divided into a matrix of state evaluation values A and a vector of action advantage values V. So as to convert the target Q value into two values and improve the efficiency of learning. That is to say, converting the value function Q (s) into a state evaluation value function V (s) and comparing it with the advantage function A (s, a):

$$Q(s) = V(s) + A(s,a) \tag{1}$$

B. DRL Environment Settings

The intelligent agent environment setting is based on the problem modeling. In order to ensure that tasks do not affect each other and improve the robustness of the method, a separate neural network is trained for each possible target point. Therefore, the DRL process can be regarded as having a fixed target point, and the model does not need to consider additional target point coordinate inputs.

The state of the intelligent agent is represented as:

$$s = \{P_i, D_i\} \tag{2}$$

Among them, P_i Represents the current position coordinates of the robot, D_i it is the directional attribute of the robot.

C. Exploration - Utilization Strategy

This article proposes a space based ε - greedy strategy to accelerate the agent learning process of this method.

For each task, a state exploration frequency labeling matrix M *is* used to record the number of times the agent arrives at each state at the current target point.

dynamic exploration rate parameters ε calculated by the following equation:

$$\varepsilon = \varepsilon_{max} * (1 - \text{epoch} / \text{max\_epoch}) \tag{3}$$

Among them, epoch is the current number of rounds, and max _ epoch is the maximum number of rounds. ε_{max}For the maximum fixed exploration rate parameter.

This strategy can ensure that the agent has at least a certain degree of exploration at all times, while increasing exploration in the early stages of training to seek more possibilities, and increasing utilization in the later stages of training to strengthen the strategy. It also takes into account the actual arrival times of each state, balancing the agent's exploration in space.

D. Sample collection and experience replay mechanism

This method proposes a Chaos based Prioritized Hindsight Experience Replay (CPHER) strategy as a sampling learning method for neural networks. CPHER is based on an empirical sample space access table. The experience sample space access table records the number of visits for each action in each state, updates the agent's actions after each step is executed, and stores them for a long time throughout the entire DRL process.

Each sample consists of a five tuple of state, action, reward, next state, and end identifier:

$$\text{sample} = (s, a, r, s', \text{done}) \tag{4}$$

Among them, *s* and s'Represents the two states of the intelligent agent, consisting of the current position of the agent and the target position of the agent.

The higher the priority of the sample, the more likely it is to be sampled. The priority of the sample is defined as:

$$\text{Priority}\,[s] = \begin{cases} \max(\text{Priority}), & \text{if } s \text{ is new} \\ TD\_\text{Error}\,(s) & \text{, if not} \end{cases} \tag{5}$$

TD_Error Indicate the time difference error of the sample.

The sampling probability of a sample is determined by its priority, expressed as follows:

$$\text{probabilities} = \left(\text{priorities} + \frac{1}{\text{capacity}}\right)^{\alpha} \tag{6}$$

Among them, capacity refers to the capacity of the sample pool.

E. Consensus based Congestion Avoidance

According to the problem modeling description in this article, when each robot takes actions with varying intervals, collisions may occur between robots. If emergency avoidance or braking measures are taken before a collision is about to occur, it has a certain degree of danger and may cause road congestion. Therefore, a preventive mechanism is needed to address the problem of robot congestion.

Given the detection radius *r*, the number of predicted steps *s*, and the conservatism threshold *t*. Based on the equivalence of neural networks, the robot considers other robots in the front direction of the detection radius *r* at each step and predicts their trajectory. If one of the following conditions is met, the other actions are input to the model for judgment, and the optimal action is selected to proceed:

(1) At least *t* robots intersect their trajectories within *s* steps with their own trajectories within *r+s+1* steps.

(2) At least one robot, after reaching the target point within *s* steps and needing to stop, intersects its trajectory with its own trajectory within *r+s+1* steps.

III. EXPERIMENTAL RESULTS AND ANALYSIS

A. Experimental Design

Given the particularity of DRL environment settings, the actions, state space, and simulation maps of intelligent agents need to be highly customized. This article uses Python language and the Turtle library to self code and establish a visual testing environment for DRL based on a simple sparse intelligent factory environment. Based on this environment, the pytorch library is used to build and train DRL models.

The simulation map is customized based on a two-dimensional list of stored characters, and the images of robots and various road sections are drawn using draw.io software and processed through the PIL library. The virtual environment is a pixelated simulated simple sparse intelligent factory environment, as shown in Fig.2. Among them, the red arrow represents the target point, the orange part represents the road of the smart factory, and the robot is represented by a graphic of a vehicle model, which can reflect the current direction of the robot in real time. The parameter settings for SSDD-DQN are shown in Table I.

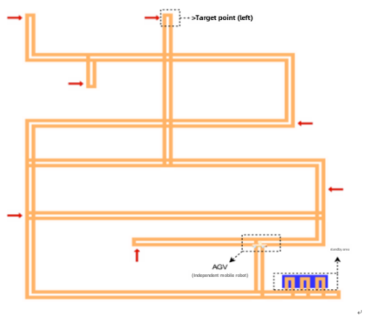

Fig.2 Example of Simulated Simple Sparse Intelligent Factory Experimental Environment

Table I. SSDD-DQN parameter settings

Parameter Name	value	explain
γ	0.98	Future reward discount rate
$\mathcal{E}_0$	0.05	Fixed exploration rate
ε_{max}	0.15	Maximum fixed exploration rate
α	0.7	Sampling priority factor
β	0.65	Sampling importance factor
$\boldsymbol{\Omega}$	0.5	Logistic map parameter
lr	0.004	Main model learning rate
r	5	Congestion avoidance detection radius
s	4	Congestion avoidance prediction steps
t	3	Congestion avoidance conservatism threshold

B. Comparative experiment

For the convenience of explanation, this section introduces classic DQN and PPO algorithms, and compares these methods with SSDD-DQN in ten different simple scenarios. For the PPO algorithm, its action space is mapped and modified to output discrete actions, and the movement consumption is calculated based on the minimum value from ten tests. Ten different maps are used to test the above method, with a maximum training round of 300. Table II lists the comparison of indicators for the above methods and records the indicator values for all tasks.

Table II. Comparison of Algorithm Index Values

case	PPO		DQN		SSDD-DQN	
	Rounds	Mobile consumption	Rounds	Mobile consumption	Rounds	Mobile consumption
1	129	53	113	53	64	53
2	79	37	62	32	43	32
3	217	46	156	42	79	39
4	189	29	143	29	83	29
5	116	32	112	32	76	32
6	152	33	134	28	98	28
7	198	37	132	39	103	34
8	135	38	93	38	81	38
9	unsolvable	unsolvable	215	56	167	56
10	unsolvable	unsolvable	unsolvable	unsolvable	179	63

From the data in the table, it can be seen that SSDD-DQN can obtain a feasible solution with a small number of samples in all cases, and the quality of the final solution is also the best. In complex cases such as Case 9 and Case 10, both PPO and DQN were unable to obtain feasible solutions, while SSDD-DQN was still able to obtain feasible solutions. SSDD-DQN has higher sample efficiency compared to PPO and DQN, and can ensure the quality of the final solution.

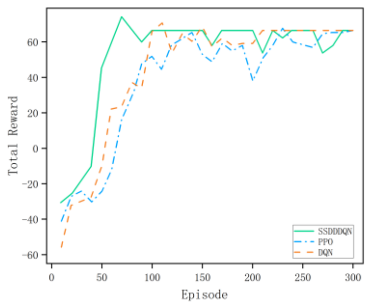

Fig. 3.Total Reward Variation Curve of Three Algorithms

Fig.3 shows the convergence curves of SSDD-DQN, DQN, and PPO for the cases in Table 4. The vertical axis in the table displays the total earnings of each round. As shown in the figure, SSDD-DQN is able to obtain higher reward values and converge faster overall compared to the other two algorithms.

IV. CONCLUSIONS

A high sample efficiency SSDD-DQN method is proposed based on the simple sparse intelligent factory environment problem in the design work for low complexity work environments. Compared to the general DQN method, this method introduces a Dueling Double structure in the network structure to optimize the learning and model updating methods; Propose special action state spaces and reward functions for the problem in environmental settings, enabling DRL methods to learn navigation tasks for each target point more accurately; Propose a spatial based exploration utilization strategy ε -Greedy strategy is used to balance the spatial distribution of agent selection actions and improve exploration efficiency; The SSDD-DQN method proposed in this article has better solution quality and significantly improved sample efficiency than some classical methods. Compared with DQN and PPO methods, it can obtain higher reward values faster and converge faster overall. This further validates the effectiveness of the improved method.

REFERENCES

[1] S. Aradi . Survey of deep reinforcement learning formotion planning of autonomous vehicles[J].IEEE Transactions on Intelligent Transportation Systems , vol.23, issue 2,pp.740-759,2020.

[2] Q.Liu,J. W.Zhai, et al. A review of deep reinforcement learning[J]. Journal of Computer Science,vol.41,issue 1,pp.1-27,2018.

[3] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with deep reinforcement learning[J] Computer Science, 2013.

[4] X.J, Z.H, L.D, et al. Application of Deep Reinforcement Learning in Mobile Robot Path Planning[J]// Proc of Chinese Automation Congress. pp.12-16,2018.

[5] S.T, Q.J, AntonoglouI, et al. Prioritized experience replay[C]/ /ICLR. February 25, 2016.

[6] H.Li, Y.M.Qi. A Robot Path Planning Method Based on Deep Reinforcement Learning in Complex Environments [J].Computer Application Research, vol.37,issue s1,pp.129-131,2020

[7] J.W.Zhang, S.A.Lv, Z.H.Zhang, et al. A review of deep reinforcement learning methods based on sample efficiency optimization [J]. Journal of Software, vol.33,issue 11,pp. 4217-4238,2022.