Mathematical Biosciences and Engineering

MBE, 20(2): 1903–1918.
DOI: 10.3934/mbe.2023087
Received: 04 September 2022
Revised: 01 November 2022
Accepted: 02 November 2022
Published: 08 November 2022

http://www.aimspress.com/journal/MBE

Research article

Multi-robot task allocation in e-commerce RMFS based on deep reinforcement learning

Ruiping Yuan[1,2], Jiangtao Dou[1], Juntao Li[1,2,*], Wei Wang[1] and Yingfan Jiang[1]

[1] School of Information, Beijing Wuzi University, Beijing 101149, China
[2] Beijing Key Laboratory of Intelligent Logistics System, Beijing 101149, China

* **Correspondence:** Email: ljtletter@126.com.

Abstract: A Robotic Mobile Fulfillment System (RMFS) is a new type of parts-to-picker order fulfillment system where multiple robots coordinate to complete a large number of order picking tasks. The multi-robot task allocation (MRTA) problem in RMFS is complex and dynamic, and it cannot be well solved by traditional MRTA methods. This paper proposes a task allocation method for multiple mobile robots based on multi-agent deep reinforcement learning, which not only has the advantage of reinforcement learning in dealing with dynamic environment but also can solve the task allocation problem of large state space and high complexity utilizing deep learning. First, a multi-agent framework based on cooperative structure is proposed according to the characteristics of RMFS. Then, a multi agent task allocation model is constructed based on Markov Decision Process. In order to avoid inconsistent information among agents and improve the convergence speed of traditional Deep Q Network (DQN), an improved DQN algorithm based on a shared utilitarian selection mechanism and priority empirical sample sampling is proposed to solve the task allocation model. Simulation results show that the task allocation algorithm based on deep reinforcement learning is more efficient than that based on a market mechanism, and the convergence speed of the improved DQN algorithm is much faster than that of the original DQN algorithm.

Keywords: multi robot task allocation; deep reinforcement learning; improved DQN algorithm; multi-agent modeling; Markov Decision Process

1. Introduction

The spread of Covid-19 promotes the demand for online shopping. According to the data of the National Bureau of Statistics, China's online retail sales reached 13.1 trillion yuan in 2021. A large amount of orders has brought great pressure to the distributions of e-commerce enterprises. In order to meet distribution time limits, e-commerce enterprises urgently need to improve order picking efficiency. A Robotic Mobile Fulfillment System (RMFS) is a new type of parts-to-picker order fulfillment system with the advantages of high picking efficiency, strong flexibility and scalability [1]. It is especially suitable for e-commerce order picking with large demand fluctuation and strong timeliness [2]. RMFS was brought into the market by Amazon in 2012 [3]. In recent years, Chinese e-commerce enterprises such as JD, Ali and Suning have also begun to pilot RMFS. However, the system has not been applied on a large scale in China, mainly due to the difficulty of cooperative scheduling of large-scale robots. In large-scale RMFS, there are usually thousands of movable shelves and mobile robots, which have to coordinate to complete a large number of order picking tasks. So, the task allocation of robots in RMFS is a complex multi-robot task allocation (MRTA) problem in dynamic environment.

Multi-robot Task Allocation (MRTA) is as follows: Given a multi-robot system, a task set and system performance evaluation indicators, find a suitable robot for each task to execute. When robots complete all tasks in the task set, it is guaranteed that the overall benefit achieved is maximized [4,5]. Most of the literature takes the minimum cost of robots to complete tasks, the maximum utilization rate or the maximum output rate as an optimization goal and uses behavior-based [6], emotion [7,8], market [9], clustering [10], simulation [11] and other methods to solve the MRTA problem. The MRTA method used in RMFS can be mainly divided into four classes.

1) Methods based on market mechanism. Heap [12] designed a repetitive sequential single clustering auction algorithm for multi-robot dynamic task assignment. Zhao [13] proposed an improved market method suitable for task assignment in an unknown environment. Yuan [14] proposed a balanced heuristic auction algorithm to solve the task allocation problem in RMFS considering the picking time balance of picking stations and the load balance of robots.

2) Methods based on Queuing theory. Zou [15] built a queuing network model to study the RMFS task allocation rules and designed a neighborhood search method to find the optimal allocation rules. The results showed that the allocation rules based on order processing speed are better than random allocation. On this basis, Roy [16] studied the robot allocation strategy of RMFS in multi-partition storage mode and constructed a two-stage queuing network stochastic model. The results show that the shortest queue allocation method is more efficient in multi-partition storage. Yuan [17] used a queuing network and sharing protocol to study the task sharing and distribution of RMFS robots.

3) Heuristic rules and simulation methods. Ghassemi [18] used the simulation method to study the task allocation problem of RMFS and compared the two task allocation modes of decentralized and centralized allocation. The results showed that the decentralized allocation is faster than the centralized algorithm while ensuring the quality of the solution. Gue [19] studied the control method of the robot system by using the simulation method and designed the multi-robot decentralized control algorithm. The results show that the decentralized control can avoid the deadlock of the robot better than the centralized control method.

4) Intelligent optimization algorithms. Shen [20] used the intelligent scheduling algorithm to study the task allocation and scheduling problem of RMFS, but the allocation and processing of orders

was too ideal. Yuan [21] further studied the task scheduling problem of RMFS and designed an improved co-evolution genetic algorithm to solve the problem. Zhang [22] used an improved genetic algorithm to study the robot allocation problem of RMFS. Compared with the traditional rule-based scheduling method, the improved genetic algorithm is more effective.

Traditional MRTA methods cannot well solve the problems in a dynamic environment. For dynamic and changeable environments, the reinforcement learning method has advantages that other methods do not have [23]. It can realize its learning by interacting with the environment, so scholars introduce reinforcement learning to solve the MRTA problem. For example, in order to solve the problem of MRTA assignment in a dynamic environment, Jiajia [24] mixed reinforcement learning and a genetic algorithm to solve the scheduling problem of a multi-robot system in an intelligent warehouse.

However, reinforcement learning cannot handle the MRTA problems of high complexity with a large space state. Deep learning can make up for this shortcoming. Chen [25] proposed a learning-based algorithm for dynamic spectrum access, which keeps stable operation in a highly dynamic environment, and the Dueling Deep Q-Network with prioritized experience replay combined with a recurrent neural network is used to improve the convergence speed. Chen [26] proposed a multi-agent deep reinforcement learning game for anti-jamming secure computing in MEC network, using a post-decision state to deal with dynamic and unknown information. Bumjin Park [27] proposed a deep reinforcement learning method to find the best allocation schedule for multiple robots and showed that the proposed method finds better solutions than meta-heuristic methods, especially when solving large-scale allocation problems.

This paper uses deep reinforcement learning to solve the MRTA problems in RMFS. The main contributions of the paper are as follows: First, a multi-agent cooperative model of RMFS is established, which enables agents to learn autonomously in the system and realize communication between each agent. Second, a multi-agent task allocation model is constructed based on Markov Decision Process (MDP). Finally, an improved Deep Q Network (DQN) algorithm based on a shared utilitarian selection mechanism and priority empirical sample sampling is proposed to solve the task allocation model. The improved algorithm can avoid inconsistent information among agents and improve the convergence speed of the traditional DQN, which is a useful innovation and can be used to solve relevant problems in other fields.

The remainder of this paper is organized as follows. In Section 2, the multi-agent framework based on cooperative structure is described in detail. In Section 3, a multi-agent task allocation model is constructed based on the Markov Decision Process. In Section 4, an improved DQN algorithm is proposed to solve the task allocation model. In Section 5, simulation experiments are conducted, and the results are analyzed to verify the proposed model and algorithm. Section 6 concludes the paper and presents the limitations and prospects of the research.

2. Multi-agent framework of RMFS based on cooperative architecture

A RMFS consists of picking workstations, movable shelves, mobile robots and other equipment. The typical layout of RMFS is shown in Figure 1. Customer orders are divided into many picking tasks, which are allocated to mobile robots according to certain rules. Then, robots carry shelves to picking stations, where pickers pick the required goods off shelves to complete their tasks. Shelves, robots and picking stations need to interact with the environment and cooperate to complete the task of picking. In order to make the interaction among these elements more intelligent, multi-agent modeling is used

to describe the RMFS, in which each element is abstracted into an agent.

According to the actual situation of RMFS, seven kinds of agents are abstracted, which can interact and cooperate to work in the whole system. Considering the influence of environment change on other kinds of agents, the environment in the system is also abstracted as an agent. Then, the cooperative multi-agent system architecture is used to model RMFS. In the collaborative structure, not only are there management layers to manage agents in the system, but also agents can communicate and interact with each other.

The multi-agent model is divided into three layers: processing layer, scheduling layer and execution layer, as shown in Figure 2. The functions of each layer are as follows.

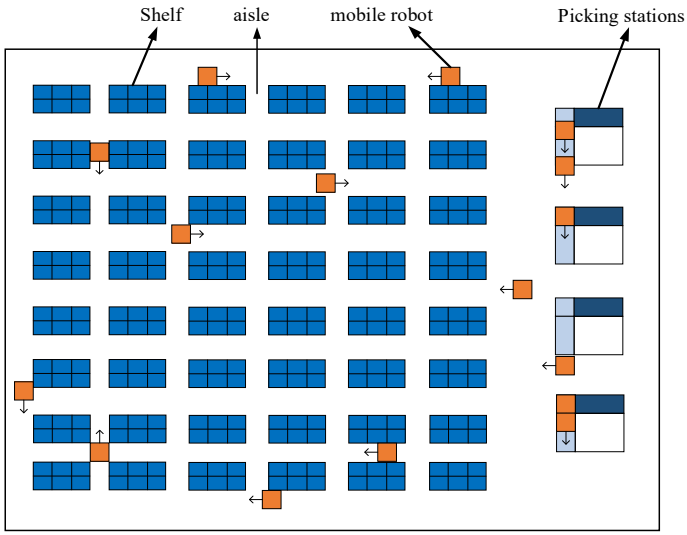

Figure 1. The typical layout of a RMFS.

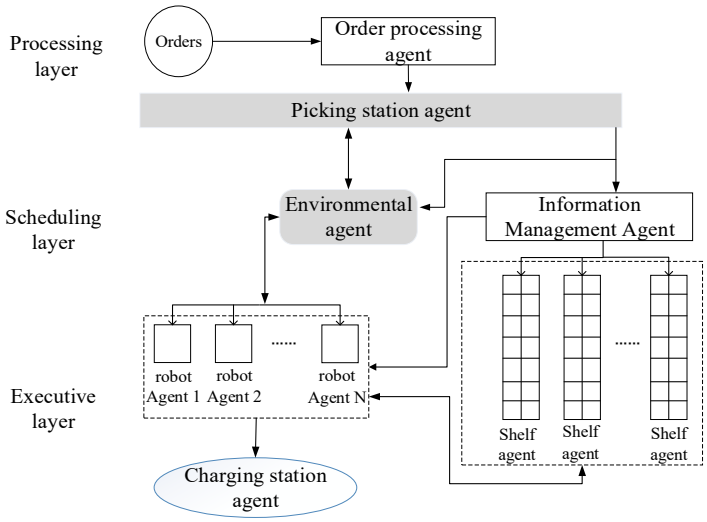

Figure 2. Collaborative multi-agent framework of RMFS.

1) Processing layer

When a order arrives at the system, the order processing agent processes and integrates the order

and then distributes the order to the picking table according to a certain order batching method.

2) Scheduling layer

In the scheduling layer, according to the received order data, the picking agent uses a deep reinforcement learning method to assign tasks to each robot agent, which is the focus of this paper. The information management agent is mainly responsible for managing information. All the information of picking station agents, shelf agents, robot agents, etc. is supervised by it.

This paper also abstracts a very important environment agent to manage the environment information, which mainly includes road information, congestion information and the location information of each agent. Other agents in the system interact with the environmental agents almost every moment, obtain the information of the environment and influence the environment. At the same time, the environmental agents respond to this influence and change their state by constantly communicating and interacting with all kinds of agents in the environment.

3) Executive layer

The execution layer includes robot agents, shelf agents and charging station agents, which have equal status. They can interact and cooperate with each other to complete the task of picking.

3. Multi-robot task allocation model based on Markov Decision Process

In the dynamic multi-robot task allocation process, information about future tasks is uncertain. At the current moment, decisions being made need to take into account both the tasks that have been identified and possible tasks to be generated in the future. Markov Decision Process (MDP) has the advantage of describing such sequential decision optimization problems in an uncertain, information-imperfect environment. Therefore, in this paper, the MRTA problem in RMFS is described as a multi-stage MDP under the condition of cooperation, which provides a theoretical basis for solving the MRTA problem using deep reinforcement learning.

MDP has the Markov property, and the task allocation problem under distributed conditions also has the Markov property. So, the overall profit of task allocation and the next moment state of the whole system are only related to the current state and the current decision of all agents in the system.

3.1. Parameter definitions

The relevant definitions are given below:

- Time period. The allocation time period is defined as t. It is assumed that the allocation in this time period is not affected by other time periods, and it does not affect the allocation of other time periods. One allocation time segment is called a round. The total number of time period is T.
- Robot Agents and their ability. Define the set of robot agents as $A = \{A_1, A_2, \dots, A_I\}$. Among them, A_i represents the i-th agent. The ability of the agent in this paper mainly refers to the current remaining power and power consumption of the agent. The current remaining power represents how long the agent can perform tasks, and the power consumption represents the power consumed by the agent in the process of performing tasks. Define B_i as the ability space of agent i. For each agent, its maximum ability is defined as B^{Max}.
- Tasks. In the process of multi-robot task allocation, there is a series of tasks that arrive randomly. The k-th task arrived is denoted as task k, $k = 1, \dots, N$, where N is the total number of arrival tasks.

- Other variables. Define ω_k as the execution revenue of the k-th task, and the capacity consumption of completing the k-th task is G_k. The paper assumes that all tasks can be successfully completed, and benefits can be obtained without breaking the constraints.

3.2. Constraints

In the process of multi-robot task assignment, the following constraints need to be observed:

Whether the agent A_i performs task k at time t is defined as Eq (1):

$$C_i(k,t) = \begin{cases} 1, & Accept\ task\ k\ , \\ 0, & Refuse\ task\ k\ . \end{cases} \tag{1}$$

The same agent can only accept one task at the same time, namely:

$$\sum_k C_i(k,t) \le 1\ ,\ \forall\ A_i, t. \tag{2}$$

Task k can only be executed once:

$$\sum_t \sum_i C_i(k,t) = 1,\ \forall\ k. \tag{3}$$

Limited by the ability of agents, the energy consumed by an agent to perform tasks cannot be greater than its maximum energy. For an agent, its ability constraints can be defined as Eq (4):

$$\sum_t \sum_k G_k C_i(k,t) \le B^{Max}\ ,\ \forall\ A_i. \tag{4}$$

3.3. MDP tuple

1) State space

Define f_i as the working state of the i-th agent, whose value is 1 when the robot is free and 0 when it is not. The state of the i-th agent at moment t is $s_i = [f_i, B_i]$. The state of the system is the set of states of all agents, denoted as $s = [s_1, \dots, s_i, \dots, s_I]$.

2) Action space

The action of an agent is defined in Eq (5).

$$a_i = \begin{cases} 1,\ Accept\ task\ k, \\ 0,\ Refuse\ task\ k. \end{cases} \tag{5}$$

The set of actions of all agents is defined in Eq (6).

$$a = (a_1, \dots, a_i, \dots, a_I). \tag{6}$$

3) Reward function

The reward function at moment t is defined in Eq (7).

$$r_t^i = \begin{cases} \omega_k \alpha - \dfrac{G_k}{\omega_k}\beta,\ \ if\ a_i = 1, f_i = 1\ and\ B_i \ge\ G_k\ , \\ 0, \qquad\qquad otherwise\ . \end{cases} \tag{7}$$

ω_k is the execution revenue of the k-th task; α and β are the normalized indexes of gain and loss, respectively.

4) Strategy and Objective function

π_i is the allocation strategy of the i-th agent, $\pi_i(s_i)$ is the allocation strategy of the i-th agent under state s_i, indicating how the agent acts under state s_i to maximize the expected cumulative reward.

$$\pi_i(s_i) = \underset{a_i}{\operatorname{argmax}}\, Q_i\,(s_i, a_i). \tag{8}$$

The action value function $Q_i\,(s_i, a_i)$ denotes the cumulative reward obtained by performing action a_i under state s_i. It is the total expected discounted reward, as defined in Eq (9).

$$Q_i(s_i, a_i) = E\{\sum_{t=1}^{T} \gamma^t r_t^i | s_i, a_i\}. \tag{9}$$

Among them, γ is the discount coefficient. Generally, the value of γ is between 0 and 1. When the value is 0, only immediate reward is concerned. When the value is 1, there is no discount, and the future reward is the same as the immediate reward.

For the entire system, its total revenue function is defined in Eq (10).

$$Q(s) = \sum_{i=1}^{I} Q_i\big(s_i, \pi_i(s_i)\big). \tag{10}$$

The ultimate goal of multi-robot task allocation is to maximize the overall revenue of all robots, as shown in Eq (11):

$$max\big(Q(s)\big) = max\big[\sum_{i=1}^{I} Q_i\big(s_i, \pi_i(s_i)\big)\big]. \tag{11}$$

According to Eq (11), each agent must continuously adjust its own decision-making plan to minimize the contradiction and maximize the overall total revenue.

4. Algorithm design

One of the most popular reinforcement learning methods is Q-Learning, which chooses its actions based on Q-values. Q-Learning uses a table to store all possible {state, action} pairs. Although effective, it does not scale with the number of states since the number of pairs {state, action} increases exponentially, leading to a very large Q-table and so requiring a large amount of memory. To overcome this problem, the Deep Q Network (DQN) method has been introduced.

DQN combines deep learning with reinforcement learning, which uses a deep neural network (DNN) as a function approximator to predict Q-values. A DQN consists of two neural networks: Q-network and target network. The Q-network is used to predict the optimal Q-value, while the target network is used to forecast the next state based on the sample data and the optimal Q-value from all potential actions in the next state. In addition, DQN has a component called Experience Replay (ER) that stores and generates training data for the Q-Network.

DQN utilizes the powerful representation ability of neural networks and the dynamic learning ability of Q-learning to get the Q value corresponding to each action. By comparing the Q values, the agent can execute the action with the largest Q value. DQN uses experience playback and dual network structure to improve the stability of the algorithm, and it can perform well in solving complex multi agent task allocation. Therefore, this paper intends to use the DQN algorithm to solve the multi-robot task allocation model in RMFS.

4.1. Limitations of DQN algorithm when used in solving the MRTA model of RMFS

Although the DQN algorithm combines the powerful representation capabilities of neural

networks and the powerful decision-making capabilities of reinforcement learning, it may encounter some problems when used to solve the proposed MRTA model in the previous section.

1) The information among agents is inconsistent. The distributed task assignment enables each agent to use the DQN method to make independent decisions, which may cause multiple agents to choose the same task after learning. In order to coordinate the behavior of each agent and avoid the waste of resources, it is necessary for each agent to share decision-making strategies and keep the information consistent.

2) The learning and convergence speed is slow. The DQN algorithm implements an experience playback mechanism. When the agent interacts with the environment, sample data will be generated at every moment. DQN establishes an experience pool and stores the generated sample data in the experience pool. When training the network, it will randomly select a small part of the data from the experience pool. When the reward is very small, because each time a small part of sample data is randomly extracted, the algorithm will produce a lot of useless trainings, and the learning speed will also slow down, resulting in a longer overall task allocation time.

4.2. Improved DQN algorithm

Aiming at solving the MRTA model of RMFS, this paper adopts a shared decision-making strategy based on a utilitarian selection mechanism and the priority empirical sample sampling method to avoid the problems of inconsistent information and slow convergence speed.

1) Shared decision-making strategy

For the problem of inconsistency of information among agents, the paper chooses a shared decision-making strategy for task assignment under distributed conditions. Strategy sharing means that each agent in the system shares its own decision-making plan, which can be achieved by formulating a strategy selection mechanism in advance, so that the agents can agree on the current decision environment. Specifying such a decision strategy can reduce the communication between agents and at the same time enable information sharing in the entire system.

Generally speaking, there are four common strategy selection mechanisms: utilitarianism, egalitarianism, plutocracy and dictatorship [28]. This paper chooses to use the utilitarian selection mechanism *f*, which means that the goal is to maximize the sum of the rewards of all agents. In state s,

$$\max_{\pi_s \in D(s)} \sum_{a \in (A(s))} P(a|s) Q(s,a). \tag{12}$$

where π_s is the strategy in state s, $D(s)$ represents the strategy set of all agents, $A(s)$ represents the action set of all agents, $P(a|s)$ represents the possibility of choosing action a under state s and $Q(s,a)$ represents the Q value of agents taking action a in state s.

2) Sampling based empirical sample priority

In order to accelerate the learning and convergence speed of DQN, this paper proposes a sampling method of empirical samples based on priority. That is, the sample in the experience playback pool is set to a priority. When extracting small samples of data, the samples with high priority are selected first, so that it can extract the sample data that can be learned quickly and effectively.

The sample priority in the experience pool is mainly calculated by the time difference error in the DQN algorithm, that is, the target Q value minus the predicted Q value. It is assumed that $\langle s, a, r, s' \rangle$ is a sequence of experience sample j, and s' indicates the next state. The time difference error δ_j of sample j is

$$\delta_j = r_j + \gamma \max_a \hat{Q}(s', a) - Q(s, a). \tag{13}$$

Among them, $Q(s, a)$ represents the Q value of agent taking action a in state s. $\max_a \hat{Q}(s', a)$ represents the optimal action value function for the next state, and γ is the discount coefficient. r_j represents rewards under state s.

When the time difference error is large, the improvement range of prediction accuracy will increase accordingly, which shows that using this sample to learn can better improve the effect of the algorithm, and thus, the higher its priority should be set. The priority $P(j)$ of sample j is calculated as follows:

$$P(j) = \delta_j \Big/ \sum_j \delta_j . \tag{14}$$

The process of the improved DQN algorithm is shown in Figure 3. The pseudo code of the algorithm is as follows:

Improved DQN algorithm

Input: MDP quaternion, Selection mechanism f
Output: Q value, Optimal Strategy π^*, Reward value r
1 Initialize the experience pool D, the action value function Q of the prediction network with random weights θ.
2 Initialize state s, action a
3 Get the initial state s
4 When the maximum episode is not reached:
5 If exploration value $\varepsilon > u$:
6 Choose a random action a
7 else:
8 Choose $a \in f$
9 Perform actions a
10 Obtain the reward r and the next state s'
11 Store experience samples $\langle s, a, r, s' \rangle$ into experience pool D
12 Use Eq (13) to calculate the time difference error of the sample.
13 Use Eq (14) to calculate the priority of the sample.
14 The priority calculated in the above formula is to sample small batches of stored samples $\langle s, a, r, s' \rangle$ from the experience pool D.
15 If the experience trajectory ends at time step $j + 1$:
16 $y_j = r_j$
17 else:
18 $y_j = r_j - \gamma max Q(\emptyset_{j+1}, a'; \theta^{-1})$;
19 Use gradient descent algorithm to update the network model parameters θ in the loss function$\left(y_j - Q(\emptyset_j, a; \theta)\right)^2$.
20 set $s = s'$
21 Return Q value, action a, reward value r.

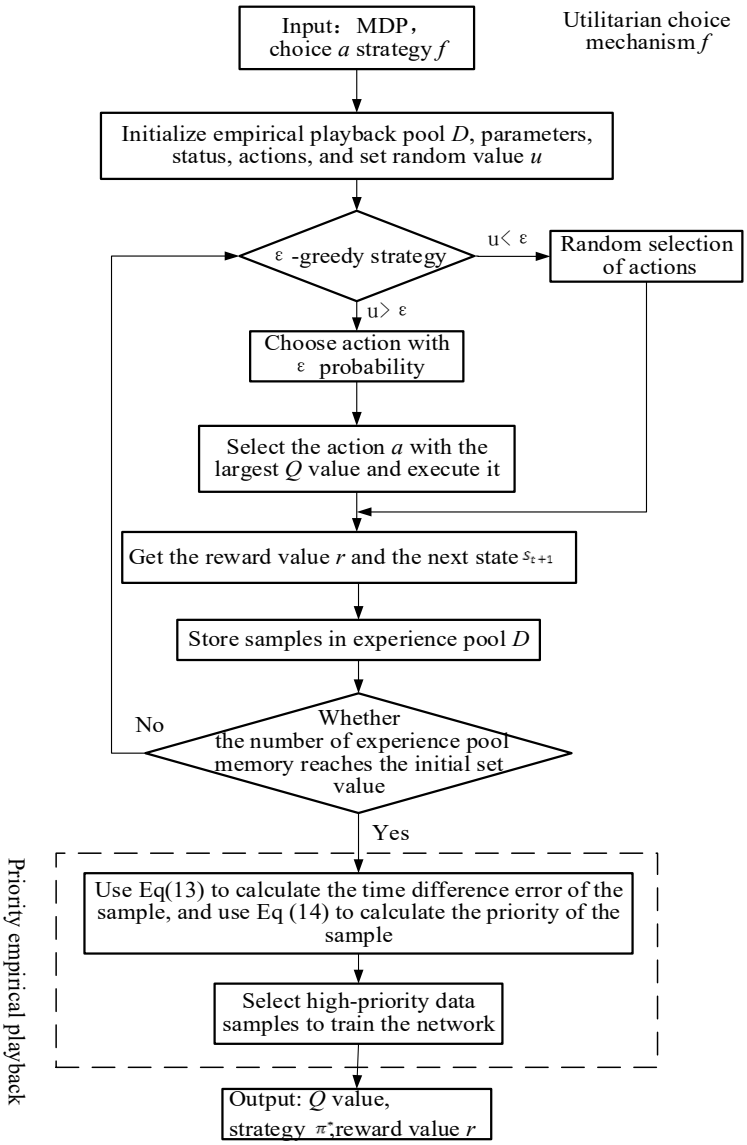

Figure 3. The process of the improved DQN algorithm.

5. Results and discussion

5.1. The simulation environment and parameter setting

The simulation is conducted using Anaconda 3.6.1 and Tensor Flow machine learning framework. Libraries such as pandas, matplotlib and gym in python are used to build the simulation environment.

A grid map is constructed in the simulation environment, as shown in Figure 4, which consist of 7 × 8 sets of shelves (Grey ones), 6 picking stations (red ones) and 10 robots (yellow ones). In this experiment, 180 order tasks are randomly generated in each allocation cycle. The starting coordinates of these robots are (26, 5), (26, 7), (26, 9), (26, 11), (26, 13), (26, 15), (26,17), (26,19), (26,21), (26,23); and the coordinates of the six picking stations are (2,37), (6,37), (10,37), (14,37), (18,37), (22,37).

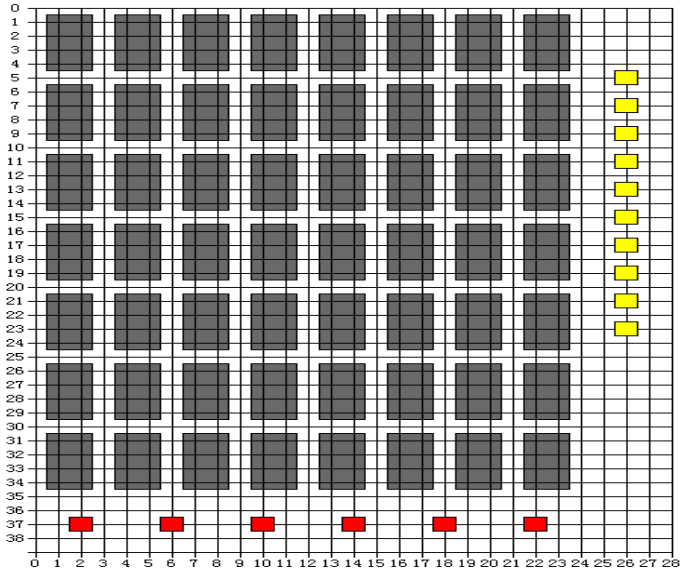

Figure 4. The simulation environment.

Table 1. Simulation parameter setting.

Parameter	Meaning	Value
I	Number of robots	10
S_num	Number of picking tables	6
N	Number of tasks	180
B_i	Power range	15–100%
ω_k	Revenue of task	1–10
α	Learning rate	0.001
D	Experience pool size	10000
ε	Exploration value	0.7–0.9995
γ	Attenuation coefficient	0.9
Sample_size	Sample size of the experience pool	128
Update_net	Frequency of parameter update of target network	200
Total_episodes	Total training episodes	6000

The basic parameters of the simulation experiment are listed in Table 1. The learning rate α is 0.001; the experience pool size D is 10,000. The initial exploration value in the greedy strategy of ε-greedy action selection is 0.7, and the final exploration value is no higher than 0.9995, which is increased by 1×10^{-5}. The attenuation coefficient γ of the Q-learning algorithm is 0.9; the revenue for each task $\omega_k \in (1,10)$. The power range B_i of the i-th robot is set to [15–100%]. If it is less than 15%, it must be charged, and no more tasks will be assigned. The sample size of random gradient descent batch sampling is 128. When the data volume of the experience pool reaches 128, the network will be trained, and the parameters will be updated. The prediction network updates its parameters at each time step, and the target network updates its parameters every 200 time steps. The total number of training

episodes is 6000.

5.2. Comparative analysis

1) Algorithm performance

In order to prove the effectiveness of the improved DQN algorithm, the convergence speeds of the loss function of the DQN algorithm before and after improvement are compared. The results are shown in Figures 5 and 6.

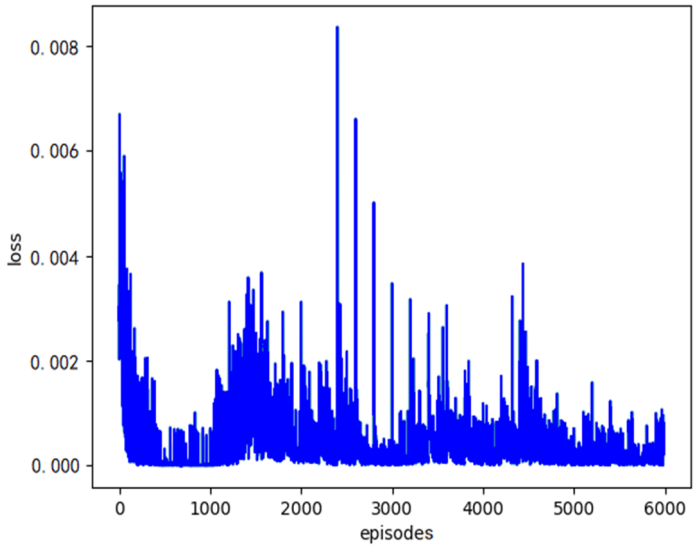

Figure 5. Convergence of loss function of DQN.

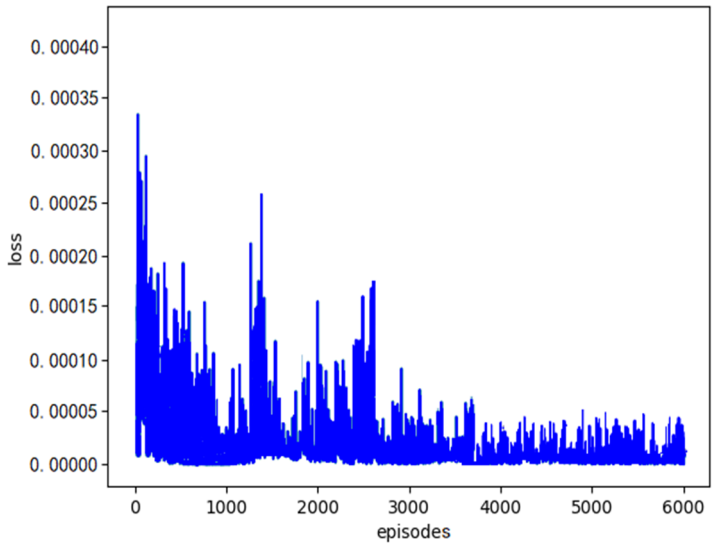

Figure 6. Convergence of loss function of improved DQN.

Figures 5 and 6 shows the loss function of the unimproved DQN algorithm has not yet converged

after 5000 rounds, while the improved DQN showed a general convergence trend after 2500 rounds. From this we can conclude that, by adding the selection mechanism and sample priority, the improved DQN reduces useless exploration, increases the learning speed and reduces the time for training, which accelerate the convergence speed of the loss function.

In order to further prove the effect of improvement, the changing trend of the total revenue is also shown in Figure 7. Figure 7 shows that the total revenue during training gradually increases and tends to converge at around 2500 rounds, which indicates that the revenue gained during the training process increases first and tends to be stable at last, which indicates that the improved algorithm learns effective decision strategy in the learning process and can ensure that the overall revenue tends to be stable.

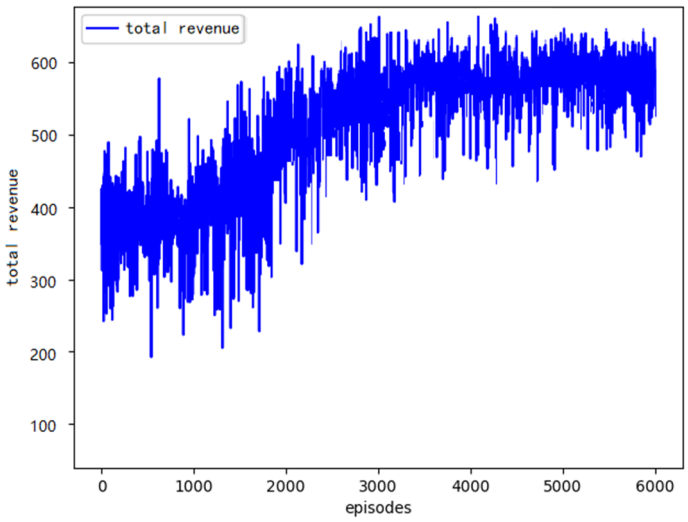

Figure 7. The total revenue of the improved DQN algorithm.

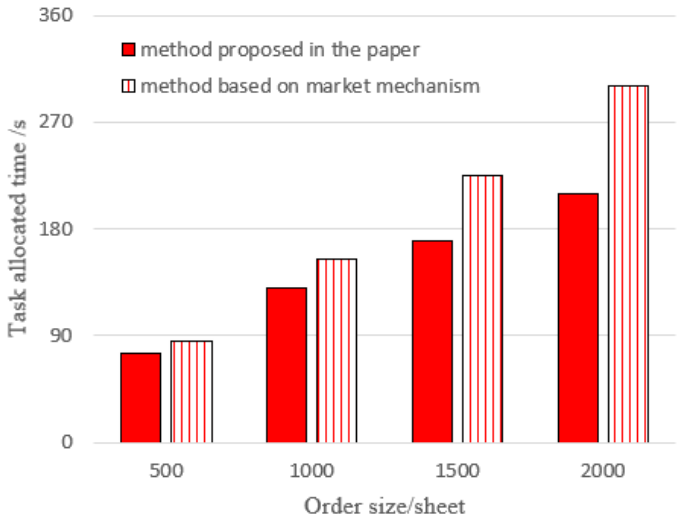

Figure 8. The comparison of task allocation times.

2) Task allocation time

The paper also proves the effectiveness of the MRTA model and algorithm proposed by comparing with the method based on the market mechanism used in [14]. The total task allocation time required for the two methods under different order sizes is shown in Figure 8. Experimental results show our proposed method takes less time to complete all the tasks than the market mechanism-based method under different sizes of tasks. The picking efficiency is improved by 14.91–29.15% under different order sizes.

6. Conclusions

In this paper, a MRTA method based on deep reinforcement learning is proposed to solve the task allocation problem in RMFS. First, this paper uses the collaborative architecture of multi-agent modeling to model the robots, picking stations and other devices in RMFS systems, and it transforms the task allocation problem of robots into a multi-agent task allocation problem under the distributed condition. Then the task allocation model of multiple agents is established based on MDP. In order to avoid inconsistent information among agents and improve the convergence speed of the traditional DQN, an improved DQN algorithm based on a shared utilitarian selection mechanism and priority empirical sample sampling is proposed to solve the task allocation model. Simulation results show that the task allocation algorithm based on deep reinforcement learning is more efficient than that based on the market mechanism, and the convergence speed of the improved DQN algorithm is much faster than that of the original algorithm.

However, there are still some problems that need to be further studied in the future: 1) The algorithm proposed in this paper aims at solving the task allocation problem under the distributed condition, which may need a lot of computing power in practice. The optimized modeling and solving algorithms based on the distributed condition need to be studied further to reduce the cost. 2) The priority of orders is not considered when designing the reward function. In follow-up research, the reward function can be optimized based on the priority of orders.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (72101033, 71831001), Beijing Natural Science Foundation Project (KZ202210037046), Canal Plan-Youth Top-notch Talent Project of Beijing Tongzhou District (YHQN2017014), Research Program of Beijing Municipal Education Commission (KM202110037003).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. N. Boysen, R. D. Koster, F. Weidinger, Warehousing in the e-commerce era: A survey, *Eur. J. Oper. Res.*, **277** (2019), 396–411. https://doi.org/10.1016/j.ejor.2018.08.023

2. R. D. Koster, T. Le-Duc, K. J. Roodbergen, Design and control of warehouse order picking: A literature review, *Eur. J. Oper. Res.*, **182** (2007), 481–501. https://doi.org/10.1016/j.ejor.2006.07.009
3. M. Wulfraat, Is Kiva systems a good fit for your distribution center? An unbiased distribution consultant evaluation, 2012. Available from: http://www.mwpvl.com /html/ kiva_systems.html.
4. H. Hu, Q. Zhang, H. Hu, J. Chen, Z. Li, Q-learning based mobile swarm intelligence task allocation algorithm, *Comput. Integr. Manuf. Syst.*, **24** (2018), 1774–1783. https://doi.org/10.13196/j.cims.2018.07.019
5. L. Mo, L. Su, X. Li, Research and development of task allocation methods in multi-robot systems, *Manuf. Autom.*, **35** (2013), 21–24+28. https://doi.org/10.3969/j.issn.1009-0134.2013.05.07
6. Z. Shi, Q. C, Cooperative task allocation for multiple UAVs based on improved multi objective quantum behaved particle swarm optimization algorithm, *J. Nanjing Univ. Sci. Technol.*, **36** (2012), 945–951. https://doi.org/10.14177/j.cnki.32-1397n.2012.06.013
7. B. H. Sun, H. Wang, B. F. Fang, Z. L. Ling, J. Lin, Task allocation in emotional robot pursuit based on self-organizing algorithm, *Robot*, **39** (2017), 680–687. https://doi.org/10.13973/j.cnki.robot.2017.0680
8. B. F. Fang, L. Chen, H. Wang, S. L. Dai, Q. B. Zhong, Research on multirobot pursuit task allocation algorithm based on emotional cooperation factor, *Sci. World J.*, (2014), 864180. https://doi.org/10.1155/2014/864180
9. L. Liu, X. C. Ji, Z. Q. Zheng, Multi-robot task allocation based on market and capability classification, *Robot*, **28** (2006), 337–343. https://doi.org/10.13973/j. cnki.robot.2006.03.019
10. F. Janati, F. Abdollahi, S. S. Ghidary, M. Jannatifar, J. Baltes, S. Sadeghnejad, Multi-robot task allocation using clustering method, in *Robot Intelligence Technology and Applications 4*, Springer, Cham, **447** (2017), 233–247. https://doi.org/10.1007/978-3-319-31293-4_19
11. A. Farinelli, L. Iocchi, D. Nardi, Distributed on-line dynamic task assignment for multi-robot patrolling, *Auton. Robots*, **41** (2017), 1321–1345. https://doi.org/10.1007/s10514-016-9579-8
12. B. Heap, M. Pagnucco, Repeated sequential single-cluster auctions with dynamic tasks for multi-robot task allocation with pickup and delivery, in *German Conference on Multiagent System Technologies*, **8076** (2013), 87–100. https://doi.org/10.1007/978-3-642-40776-5_10
13. H. Zhao, C. Shi, Exploration of the unknown environment of multi-robot collaboration based on market method, *Comput. Digital Eng.*, **45** (2017), 2085–2089. https://doi.org/10.3969/j.issn.1672-9722.2017.11.001
14. R. Yuan, J. Li, X. Wang, L. He, Multirobot task allocation in e-commerce robotic mobile fulfillment systems, *Math. Probl. Eng.*, (2021), 6308950. https://doi.org/10.1155/2021/6308950
15. B. P. Zou, Y. M. Gong, X. H. Xu, Z. Yuan, Assignment rules in robotic mobile fulfilment systems for online retailers, *Int. J. Prod. Res.*, **55** (2017), 6175–6192. https://doi.org/10.1080/00207543.2017.1331050
16. D. Roy, S. Nigam, R. D. Koster, I. Adan, J. Resing, Robot-storage zone assignment strategies in mobile fulfillment systems, *Transp. Res. Part E Logist. Transp. Rev.*, **122** (2019), 119–142. https://doi.org/10.1016/j.tre.2018.11.005
17. Z. Yuan, Y. Y. Gong, Bot-in-time delivery for robotic mobile fulfillment systems, *IEEE Trans. Eng. Manage.*, **64** (2017), 83–93. https://doi.org/10.1109/TEM.2016.2634540

18. P. Ghassemi, S. Chowdhury, Decentralized task allocation in multi-robot systems via bipartite graph matching augmented with fuzzy clustering, in *The ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2018. https://doi.org/10.1115/DETC2018-86161
19. K. R. Gue, K. Furmans, Z. Seibold, U. Onur, Grid store a puzzle-based storage system with decentralized control, *IEEE Trans. Autom. Sci. Eng.*, **11** (2014), 429–438. https://doi.org/10.1109/TASE.2013.2278252
20. B. W. Shen, N. B. Yu, J. T. Liu, Intelligent scheduling and path planning of warehouse mobile robots, *CAAI Trans. Intell. Syst.*, **9** (2014), 659–664. https://doi.org/10.3969/j.issn.1673-4785.201312048
21. R. P. Yuan, H. L. Wang, L. R. Sun, J. T. Li, Research on the task scheduling of "goods to picker" order picking system based on logistics AGV, *Oper. Res. Manage. Sci.*, **27** (2018), 133–138. https://doi.org/10.12005/orms.2018.0241
22. J. T. Zhang, F. X. Yang, X. Weng, A building-block-based genetic algorithm for solving the robots allocation problem in a robotic mobile fulfillment system, *Math. Probl. Eng.*, (2019), 6153848. https://doi.org/10.1155/2019/6153848
23. M. Zolfpour-Arokhlo, A. Selamat, S. Z. M. Hashim, H. Afkhami, Modeling of route planning system based on Q value-based dynamic programming with multi-agent reinforcement learning algorithms, *Eng. Appl. Artif. Intell.*, **29** (2014), 163–177. https://doi.org/10.1016/j.engappai.2014.01.001
24. J. Dou, C. Chen, P. Yang, Genetic scheduling and reinforcement learning in multirobot systems for intelligent warehouses, *Math. Probl. Eng.*, (2015), 597956. https://doi.org/10.1155/2015/597956
25. M. Chen, A. Liu, W. Liu, K. Ota, M. Dong, N. N. Xiong, RDRL: A Recurrent deep reinforcement learning scheme for dynamic spectrum access in reconfigurable wireless networks, *IEEE Trans. Network Sci. Eng.*, **9** (2022), 364–376. https://doi.org/10.1109/TNSE.2021.3117565
26. M. Chen, W. Liu, N. Zhang, J. Li, Y. Ren, M. Yi, et al., GPDS: A multi-agent deep reinforcement learning game for anti-jamming secure computing in MEC network, *Expert Syst. Appl.*, **210** (2022), 118394. https://doi.org/10.1016/j.eswa.2022.118394
27. B. Park, C. Kang, J. Choi, Cooperative multi-robot task allocation with reinforcement learning, *Appl. Sci.*, **12**(2022), 272–291. https://doi.org/10.3390/app12010272
28. Y. D. Wang, *Research on Multi-Object Workflow Scheduling Method Based on Deep-Q-Network Multi-Agent Reinforcement Learning*, MA.Sc thesis, Chongqing University, 2019. https://doi.org/10.27670/d.cnki.gcqdu.2019.000536

AIMS Press