ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 3 - 2026

A Real-Time Control Framework Combining Edge Computing and Deep Reinforcement Learning in Industry 4.0 Environments

Qianwen Wang[1,*]

[1] JSNU SPBPU Institute of Engineering—Sino-Russian Institute, Jiangsu Normal University, Xuzhou, Jiangsu, 221000, China

SUMMARY: *Industry 4.0 environments are characterized by an explosion in the number of mobile devices, generating massive data and computation demands. Edge computing allows IoT devices to offload tasks to the edge environment for execution in order to fulfill the task's demand for resources. To achieve real-time control of the edge computing environment, this study proposes a deep reinforcement learning-based task offloading and resource allocation method. The method uses the Lagrange multiplier method to solve for the optimal allocation of resources and the DDPG algorithm to search for the optimal offloading decision method. Experiments show that the method in this paper outperforms the baseline algorithm in terms of cumulative rewards and task discarding rate, meanwhile, it can significantly improve the throughput of the system while reducing the latency and energy consumption, with the former improved by 9.98%~28.61% and the latter two reduced by 8.05%~22.48% and 4.45%~16.05%, which can effectively improve the efficiency of task offloading and resource allocation in edge environments,. It has high real-time performance and good dynamic environment adaptability.*

KEYWORDS: *edge computing; deep reinforcement learning; Lagrange multiplier method; DDPG algorithm; task offloading; resource allocation*

1 Introduction

With the rapid development of industrial technology and intelligent informatization, Germany has put forward the Industry 4.0 strategy, which responds to the current situation of world scientific and technological development. At the same time, the processing of big data by artificial intelligence allows industrial production to reach a higher level. This is because AI has the ability of deep learning, and can control, monitor, and predict the state of equipment through the acquired data, and achieve the purpose of self-learning through the establishment of neural networks [1, 2]. On this basis, industrial production can be based on artificial intelligence to achieve advanced intelligent requirements [3].

With the rapid development of automation control technology in the industrial field, higher requirements for production efficiency, product quality and production safety have been put forward [4]. As the key to the automation control field, the industrial production machine is widely used in many industries such as automobile manufacturing, electronics, machining and so on with the advantages of high precision, high speed and high reliability [5-7]. It can replace manual labor to complete repetitive and dangerous work tasks and realize the automated control of complex production processes [8, 9]. An in-depth study of the

*Chiwww111@163.com
https://doi.org/10.65102/is20261147 Published: 10 June, 2026

automation control framework for industrial production machines in the context of Industry 4.0 is of great practical significance for promoting the transformation and upgrading of Chinese industry and improving the competitiveness of enterprises [10].

In the future, the intelligence of industrial production will become the road to production development, for which many scholars have proposed the development of intelligent technology to provide protection for the realization of Industry 4.0 and even higher-level requirements of industrial production. Xu, W. et al. proposed an industrial cloud robot control framework based on digital twin technology, which effectively solves the control inaccuracy problem of traditional industrial robots due to the delayed data update from the cloud server with the help of the high-fidelity digital model and sensing data capability of digital twin technology, which not only realizes the fine sensing control, but also has a high degree of flexibility and scalability [11]. Liu, X. et al. designed a fractional-order sensor based on an improved particle swarm optimization algorithm, which can further improve the real-time monitoring and control performance of the system by applying it to an industrial robot control framework supported by digital twin technology [12]. Righettini, P. et al. used a neural network model to optimize the time planning and scheduling strategies for industrial robots performing complex tasks, which substantially improved the productivity of industrial robots through real-time control and large-scale motion trajectory training [13]. Lu, Y. et al. showed that accurate robot dynamics studies are beneficial to improve the control performance of machine models by establishing a unified framework for in-situ calibration and simultaneous identification of industrial robots based on composite sensing to identify production machines in different scenarios in order to enhance the perception and control of production machines [14]. Chuang, W. L. et al. constructed an industrial robot control system based on Robot Operating System (ROS) and EtherCAT technology, which supports the real-time exchange of data between different kinematic kernels and servo-driven systems for efficient decision-making of the robot in response to user commands [15]. Although the above research has achieved certain results in the corresponding fields, the development of most intelligent technologies in the industrial field is still in its infancy, and further research is needed for the real-time control technology to promote industrial production in the context of Industry 4.0.

Aiming at the resource allocation problem of resource constraints in edge computing environment, deep reinforcement learning method is utilized to control it in real time. A system model including task model, local computing model, edge computing model and emergency factor model is constructed, a multi-level weight regulation method for different priority tasks is proposed, and a user-fairness oriented delay cost minimization optimization model is established. The solved mixed integer nonlinear programming problem is decomposed into resource allocation and offloading decision subproblems, and the Lagrange multiplier method is utilized as the solution method for optimal resource allocation, and the DDPG deep reinforcement learning algorithm is utilized for task offloading decision. The Java-based CloudSim simulation tool is used to simulate the edge environment to carry out experiments, compare the cumulative rewards and task discard rates of this paper's method and the three baseline algorithms in training and testing experiments, and then explore the system energy consumption with different numbers of devices and different task sizes, and finally examine the proposed method's effectiveness on the edge environment in terms of the three metrics, namely, throughput, average task latency, and energy consumption of the edge servers. The overall performance of the proposed method for real-time control of task offloading and resource scheduling in edge environments is examined in terms of three metrics: throughput, average task latency, and energy consumption of edge servers.

2 Real-time control framework combining edge computing and deep reinforcement learning

In the era of rapid development of information, the limited computing power of user devices is difficult to match their rapidly growing computing needs. With the emergence of Mobile Edge Computing (MEC), users can offload computing tasks to edge servers for processing to reduce computing costs. With more and more application scenarios empowered by mobile edge computing, such as autonomous driving, industrial manufacturing, and smart cities, more and more terminal devices tend to improve their service experience with the help of computation offloading. In this paper, we address the problem of real-time control of computation offloading in mobile edge computing scenarios, and utilize deep reinforcement learning algorithms to design offloading decision-making and resource allocation methods based on deep reinforcement learning, so as to realize real-time control of resource allocation in resource-constrained edge computing systems.

2.1 System model

The edge computing system is composed of an edge computing node with its serving M indoor terminal devices.The MEC server has a computing power of Fe and is connected to its neighboring macro base station via optical fiber so that the data transmission delay between them is negligible. The access bandwidth of the base station is B, and the terminal devices request access to the base station and obtain offloading services by orthogonal frequency division multiple access.

2.1.1 Task model

It is assumed that the tasks requested by the user in each scheduling cycle can be categorized into three types according to their business volume and priority i.e., urgent tasks with small business volume, ordinary tasks with regular business volume, and non-urgent tasks with large business volume. The computational task of end user m is represented by a triad $o_m = (e_m, \ell_m, \rho_m)$. Where e_m is the urgency factor, whose value reflects the urgency of the task, and the higher the priority of the task, the larger its value. ℓ_m denotes the size of the task's input data volume (in bits), and ρ_m denotes the number of CPU cycles required for every 1 bit of data processed for the task. Since the computational resource requirement here is positively correlated with the size of the task's input data volume, the difference in the business volume of different tasks is then mainly reflected in the size of their input data volume. It is worth mentioning that although the urgency factor e_m is described as an intrinsic part of the task's characteristics, it can also be defined as a configurable parameter managed by the edge nodes.

$\bar{o} = (o_1, o_2, \cdots, o_M)$ contains task information from all end-users requesting computational offloading. In this paper, we consider a binary task processing model instead of an arbitrarily divisible task processing model. The x_m is utilized to indicate whether the task is computed locally $(x_m = 0)$ or at the edge $(x_m = 1)$. Once $x_m = 1$, the base station will have to allocate $\alpha_m (0 < \alpha_m \le 1)$ of radio bandwidth and computational resources $\beta_m (0 < \beta_m \le 1)$ to make sure that the task can be processed efficiently.

2.1.2 Local computational models

When the task o_n is processed on its local device, i.e., $x_n = 0$, the end device n will choose the appropriate processing speed $f_n (f_n \le Fc_n)$ to process the task based on the available computational resources Fc_n of the local device. In this case, the processing delay when the task is processed locally can be expressed as:

$$t_n^{loc} = \frac{\rho_n \ell_n}{f_n} \tag{1}$$

2.1.3 Edge Computing Model

For a user request with $x_m = 1$, the end device m will utilize its share of the wireless bandwidth resource α_m to upload its task-related data, and the maximum transmission rate between it and the base station can be expressed as:

$$r_m^{up} = \alpha_m B \log_2 \left(1 + \frac{p_m h_m^2}{\sigma^2} \right) = \alpha_m R_m \tag{2}$$

where p_m is the uplink transmission power of the end device m which is usually determined by the control scheme of the base station to which it is coupled, h_m denotes the immediate channel gain between the end device m and the base station, and σ^2 is the Gaussian noise power.

Based on the above transmission rate, the transmission delay for task o_m to upload its task input data to the edge node can be expressed as:

$$t_m^{up} = \frac{\ell_m}{r_m^{up}} = \frac{\ell_m}{\alpha_m R_m} \tag{3}$$

For the offloading task o_m, the edge computing node allocates a portion of computing resources β_m for it, and the computational delay of the task o_m computed at the edge node can be denoted as:

$$t_m^{com} = \frac{\rho_m \ell_m}{\beta_m Fe} \tag{4}$$

Neglecting the transmission delay in the process of returning the computation results of the task and considering only the two phases of uplink transmission and edge processing, the processing delay of the offloading task can be expressed as follows:

$$t_m^e = t_m^{up} + t_m^{com} = \frac{\ell_m}{\alpha_m R_m} + \frac{\rho_m \ell_m}{\beta_m Fe} \tag{5}$$

2.1.4 Emergency factor model

The computing tasks are divided into three categories based on the data volume of the tasks,

and different priorities are set for computing tasks belonging to different categories: for task requests with small data volume, their priority is set to urgent to support the competition of small data volume computing tasks for system resources. For task requests with normal data volume, its priority is set to normal. For large data-volume task requests, the priority is set to non-urgent to limit large data-volume computation tasks from grabbing too many system resources.

In particular, this paper utilizes weights to softly regulate the resource allocation for different priority tasks. For a task with a computational task data volume of ℓ_n, if it falls in the small data volume computational task interval, i.e., $\ell_n \le \ell^s_{\max}$, the urgency factor (i.e., weight) for the task is set to be $e_n = supportive \times (\overline{\ell^{re}} / \ell_n)^{\xi}$, where $\overline{\ell^{re}}$ denotes the average size of a general data volume computation task (i.e., a normal priority task). If it falls in the large data volume computation task interval, i.e., $\ell_n \ge \ell^L_{\min}$, the urgency factor for this non-urgent task is set to $e_n = 1/\ penalty \times (\overline{\ell^{re}} / \ell_n)^{\xi}$. where supportive and penalty are the reinforcement factor and penalty factor, respectively. ξ is the thermal coefficient, the larger the value is, the more significant the regulation effect is. For normal priority business requests, the urgency factor is not modified and its default value is set to 1.

2.2 Optimization problem description

For computing tasks with different data volumes, multi-level weights are used to regulate the offloading behaviors among users on the basis of dividing the priority levels to guarantee the fairness of multi-user offloading decisions and resource allocation. Based on the above model, the user-fairness-oriented delay cost minimization offloading decision and resource allocation problem can be expressed as:

$$\begin{aligned} &Minimize: \sum_{m=1}^{M} x_m e_m t_m^e + (1-x_m) e_m t_m^{loc} \\ &S.T.C1: \sum_{m=1}^{M} x_m \alpha_m \le 1 \\ &C2: \sum_{m=1}^{M} x_m \beta_m \le 1 \\ &C3:\ f_m \le Fc_m, \forall m \in \mathcal{M} \\ &C4: x_m \in \{0,1\} \end{aligned} \tag{6}$$

where the decision vector $\bar{x} = (x_1, x_2, \cdots, x_m, \cdots, x_M)$ represents whether the computation task is processed locally or offloaded to the edge node for processing, and the vector $\bar{f} = (f_1, f_2, \cdots, f_m, \cdots, f_M)$ then denotes the local computation rate when the task is processed locally, and for the offloading task o_m, set its $f_m = 0$, the vector $\bar{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_m, \cdots, \alpha_M)$ denotes the bandwidth allocated by the base station for each offloaded user, while the vector $\bar{\beta} = (\beta_1, \beta_2, \cdots, \beta_m, \cdots, \beta_M)$ then denotes the computational resources allocated by the base station for each offloaded user. Constraint C1 indicates that the bandwidth resources occupied by the offloading users do not Constraint C1 indicates that the bandwidth resources occupied by the offloading users cannot exceed the bandwidth capacity of the system. Constraint C2 indicates that the computing resources occupied by the offloaded users cannot exceed the

computing resource capacity of the system. Constraint C3 indicates that the computational resources occupied by a locally computed task cannot exceed the computational capacity of its own device. Constraint C4 indicates that the task can only be computed locally or at an edge node.

The solution problem (6) is transformed into the process of solving the problem of determining the optimal resource allocation scheme under the offloading decision scheme with searching for the offloading decision scheme that minimizes the cost of system user delay. Given the offloading decision vector $\bar{x}$, the corresponding resource allocation subproblem degenerates into a convex optimization problem as shown in problem (7), which is expressed as:

$$\begin{aligned} &Minimize: \sum_{m\in\mathcal{M}} e_m t_m^e + \sum_{n\in\mathcal{M}_0} e_n t_n^{loc} \\ &S.T.C1: \sum_{m\in\mathcal{M}_1} \alpha_m \le 1 \\ &C2': \sum_{m\in\mathcal{M}_1} \beta_m \le 1 \\ &C3': f_n \le Fc_n, \forall n \in \mathcal{M}_0 \end{aligned} \tag{7}$$

where $\mathcal{M}_1$ represents the set of offloaded users and $\mathcal{M}_0$ represents the set of locally processed users.

2.3 Algorithm solving

The Lagrange multiplier method, as a common method for solving problems with constraints, eliminates complex constraints or no constraints by introducing multipliers to integrate the constraints into the Lagrange function. In this section, a closed-form solution of problem (7) is given based on this method.

2.3.1 Optimal resource allocation methodology

The Lagrange multiplier method, also known as the Lagrange multiplier method, is used to solve the function $f(x_1, x_2, ...)$ in $g(x_1, x_2, ...) = 0$ constraints on the extremes. The Lagrange multiplier method is generally divided into 3 steps: first, a new parameter λ (called the Lagrange multiplier) is introduced. Second, the constraints and the original function are associated so that the number of variables is the same as the number of equations. Finally, the solution is found for each variable in the equation. The Lagrange multiplier method has important applications in differential geometry and mathematical physics equations, as well as a wide range of applications in practical fields such as economics and engineering. The optimal resource allocation scheme proposed in this paper is solved by Lagrange multiplier method. Regarding the extremal problem of a binary function with single constraint, the relevant theorem is as follows:

For the extreme value problem of $z = f(x, y)$ under the constraint $\varphi(x, y) = 0$, where f and φ are continuously differentiable on the region D, if $(x_o, y_o) \in \operatorname{int} D$ is an extreme value point of the above function, and $\frac{\partial \varphi}{\partial y}(x_o, y_o) \neq 0$, then there exists λ_o such that (x_o, y_o, λ_o) is $L(x, y, \lambda) = f(x, y) + \lambda\varphi(x, y)$ as the stationing point of $L(x, y, \lambda) = f(x, y) + \lambda\varphi(x, y)$ which satisfies the equation:

$$\begin{cases} L_x = \dfrac{\partial f}{\partial x} + \lambda \dfrac{\partial \varphi}{\partial x} = 0 \\ L_y = \dfrac{\partial f}{\partial y} + \lambda \dfrac{\partial \varphi}{\partial y} = 0 \\ L_\lambda = \varphi(x, y) = 0 \end{cases} \tag{8}$$

where: L_x, L_y, L_λ is the derivative of $L(x, y, \lambda)$ on x, y, λ respectively.

2.3.2 Offloading decision-making methods

Deep reinforcement learning has obvious advantages in dealing with complex environments, high-dimensional state spaces, and continuous action spaces, and can better address some of the limitations and challenges in traditional reinforcement learning and deep learning, as well as expanding the scope of application areas and problem solving. Traditional reinforcement learning usually uses tabular forms or linear functions to approximate state-value functions, while deep reinforcement learning uses deep neural networks to parametrically represent value functions and strategies, which improves the ability to model high-dimensional state spaces. In this paper, on the basis of obtaining the optimal resource allocation scheme using Lagrange multipliers, the deep reinforcement learning DDPG algorithm is used to search for the optimal offloading decision that minimizes the cost of the system.

The core idea of DDPG is the same as that of the classical reinforcement learning class of algorithms, for a given optimal action-value function $Q^*(s,a)$, then for any given state s, the optimal action $a^*(s)$ for its state can be obtained by taking the maximum of the Q function:

$$a^*(s) = \arg\max_a Q^*(s,a) \tag{9}$$

The DDPG algorithm learns both a Q-function and a policy function. It uses the data of heterogeneous policies and Bellman's equation to learn the Q function, and then uses this Q function to learn the policy.DDPG interactively learns an approximate Q function $Q^*(s,a)$ and the optimal action $a^*(s)$ for a given state. The policy function acts as a participant to generate the action. The q-value function acts as a critic to evaluate the actor's performance and guide the actor's subsequent actions.

DDPG uses an Actor-Critic structure, where an Actor neural network is used to output an action and another Critic neural network is used to evaluate how good this action is (i.e., to estimate the value of the state-action). actor uses the structure of a goal-policy network and a current-policy network. critic uses the structure of a goal-value network and a current-value network. The current policy network outputs action a in environment state s, and when action a interacts with the environment, the environment generates the next state s' and the corresponding reward value r, so the action interacts with the environment to obtain the sample data $\langle s,a,r,s\rangle$, and then the data $\langle s,a,r,s\rangle$ in the experience playback pool is taken out for training, and the environment state s and action a in $\langle s,a,r,s'\rangle$ are put into the critic current value network, and $Q(s,a;\theta)$ is obtained, and the target strategy network is under the environment state s', output action a', and then the environment state s' and output action a' are input to the target value network, To get the target value of Q and

$y = r + \gamma Q(s^{'}, a^{'}; \theta^{'})$, if you want to get the current value of the network output value of $Q(s, a; \theta)$ and the target value of $y = r + \gamma Q(s^{'}, a^{'}; \theta^{'})$, you need supervised learning. The updated formula for the current value network:

$$L = \frac{1}{N} \sum_{i} (y_i - Q(s, a; \theta))^2 \tag{10}$$

Current Strategy Network Update Formula:

$$J(\theta) = -\frac{1}{N} \sum_{i} Q(s, a; \theta) \tag{11}$$

DDPG is mainly applied to solve continuous control problems, in which an intelligent body needs to choose a continuous action to maximize the cumulative reward. In order to improve the stability of the algorithm, DDPG uses soft updating, instead of exactly copying the parameters of the policy network to the target network at each iteration, the soft updating employs a small update rate that gradually adjusts the parameters of the target network.The performance of DDPG relies heavily on the selection of hyper-parameters, such as the learning rate, the batch size, and the update rate of the target network. Tuning these hyperparameters is a crucial step in the training process and usually requires experience and experimentation to find the right values. Overall, the DDPG algorithm enables good results in solving continuous control problems by skillfully combining the ideas of deep learning and reinforcement learning.

3 Experiments and analysis of results

The experiments use the Java-based CloudSim simulation tool in order to simulate the edge cloud environment and task generation, Python 3.10 to implement the algorithms in this paper, and a simulated task dataset for training and testing.

3.1 Experimental setup

In the edge computing environment, the number of MESs is set to be 4 and each MES has 5 heterogeneous and resource-constrained virtual machines. The number of cloud computing servers is 1 and each cloud computing server has 3 heterogeneous and resource rich virtual machines. In the generation process of task dataset, task arrival rate and quintuple are used to control the task generation. Task arrival rate is the ratio of the size of the task dataset to the time period (in hours) and indicates the number of tasks generated in each hour. By controlling the values of the variables within the task arrival rate and quintuple, a large number of heterogeneous tasks can be randomly generated to fully simulate the task generation characteristics of IoT devices in real-world environments.

3.2 Comparison of baseline algorithms

Compare the optimal resource allocation method based on Lagrange multiplier method and DDPG-based offloading decision making method (Lagrange-DDPG) in this paper with the following three baseline algorithms:

(1) Random strategy (Random). For a task, an offloading decision is made randomly.

(2) Edge-only execution (OE). All tasks are offloaded to the MES and not to the cloud

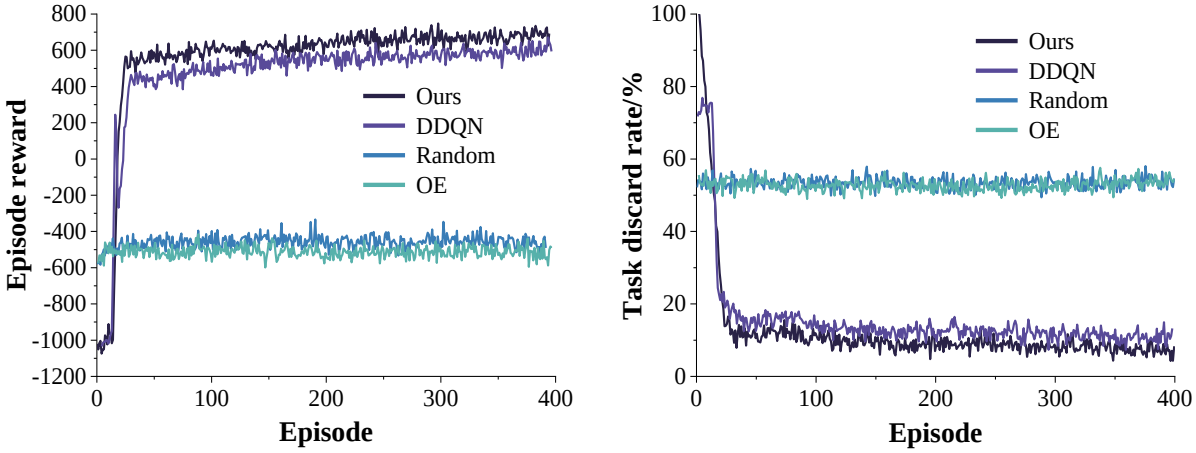
server.

(3) DoubleDQN (DDQN). This is a commonly used deep reinforcement learning algorithm that is adapted and thus adapted to the edge cloud environment of this paper.

3.2.1 Comparison of training

The evaluation metrics of the experiment use cumulative reward and task discard rate to measure the performance of the algorithm in task offloading and resource allocation. Obviously, a larger cumulative reward and a smaller task discard rate indicate a superior algorithm performance.

The experiments used a task count set with a time period of 6h and a total of 400 rounds to study the variation of cumulative rewards and task discard rate of this paper's algorithm and the baseline algorithm during the training process as shown in Fig. 1. In Figs. (a)(b), as training proceeds, the cumulative reward and task discard rate of this paper's Lagrange-DDPG algorithm and DDQN increase and decrease sharply in the first 30 rounds, respectively, and then they both converge slowly in the later rounds, but this paper's Lagrange-DDPG algorithm has more cumulative rewards, up to about 700, and a lower and faster convergence of task discard rate, up to 5 percent The cumulative rewards and task discard rates of Random and OE both fluctuate relatively smoothly within a certain range, but both are worse than the other two algorithms, which suggests that these two algorithms are not able to adapt to the highly stochastic and dynamic edge environments and heterogeneous tasks. Therefore, the Lagrange-DDPG algorithm in this paper outperforms the baseline algorithm in terms of convergence speed, cumulative rewards, and task discard rate during the training process. Also, the Lagrange-DDPG algorithm in this paper can complete the training faster in real deployment.



(a)Episode reward (b)Task discard rate

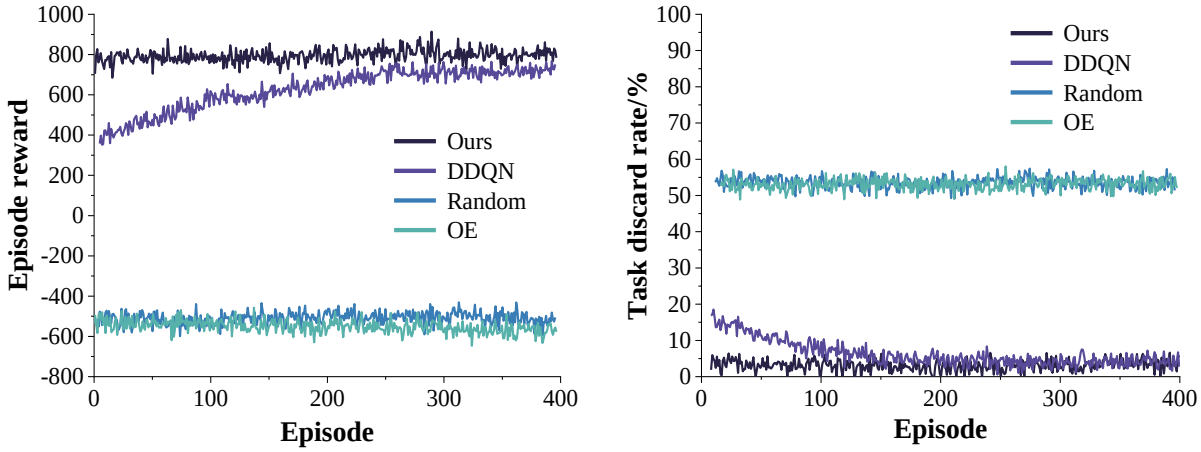
Figure 1: Training performance of our algorithm and baseline algorithms

3.2.2 Comparison of test situations

Fig. 2 shows the changes in the cumulative rewards and task discard rates of this paper's method and the baseline algorithm during the testing process in a task dataset with a time period of 6h and a total of 400 rounds to explore the changes in the cumulative rewards and task discard rates of this paper's method and the baseline algorithm. In Fig. (a), the cumulative rewards of Random and OE are much smaller than those of the Lagrange-DDPG algorithm and DDQN, and the cumulative rewards of DDQN show a slow increase in the first 280 rounds before stabilizing in the last 120 rounds, while the cumulative rewards of the

Lagrange-DDPG algorithm in this paper are always stable and larger than the cumulative rewards of DDQN's cumulative reward, which reaches about 790. In Fig. (b), the task discard rate of Random and OE is much larger than that of Lagrange-DDPG algorithm and DDQN, in the range of 50% to 55%, while the task discard rate of DDQN shows a decreasing trend until convergence, but the task discard rate of Lagrange-DDPG algorithm is always at the lowest level of about 3%.

Therefore, during the testing process, both the cumulative reward and task discard rate of this paper's method outperform the baseline algorithm, which indicates that this paper's method can realize efficient task offloading decision and resource allocation for real-time control of mobile edge computing environments when deployed in mobile edge computing environments.



(a)Episode reward (b)Task discard rate

Figure 2: Test performance of our algorithm and baseline algorithms

3.3 Analysis of system energy consumption

3.3.1 Number of different equipment

This section analyzes the impact of varying the number of virtual machines owned by each MES in the edge computing system on the total system energy consumption. Six experimental scenarios are considered, in which the number of VMs in the six sets of experiments are set to 5, 6, 7, 8, 9, and 10, respectively.The comparison of the algorithmic system energy consumption for different numbers of VMs is shown in Fig. 3. As the number of user terminals increases, the total system energy consumption rises, which is mainly due to the fact that when the number of VMs increases, the number of VMs that need to be serviced in each MEC coverage increases, and the tasks that need to be processed also increase dramatically. It can be seen that the Lagrange-DDPG algorithm of this paper achieves the minimum energy consumption, and the average energy consumption of the six groups of experiments is 1271 J, which is reduced by 88.67-203.5 J on average compared with the comparison methods, and the Lagrange-DDPG algorithm has been maintaining a significant advantage over the DDQN, Random and OE algorithms. This indicates that the proposed Lagrange-DDPG algorithm in this paper has good adaptability in terms of the variation of the number of user terminals in the edge computing system and demonstrates good computational energy saving.

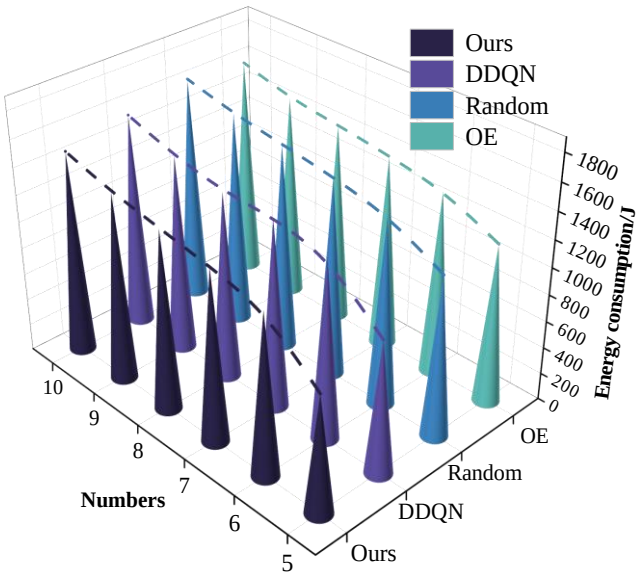


Figure 3: The energy consumption of different virtual machine number algorithm system

3.3.2 Different task sizes

Setting the task size to 2, 3, 4, 5, 6, and 7 Mb, Fig. 4 verifies the effect of task size variation on system energy consumption in edge computing systems. As the task size increases, the system energy consumption also increases. Throughout the task size increase, the Lagrange-DDPG algorithm of this paper has remained flat and achieved the minimum computational energy consumption, and the average energy consumption of this paper's algorithm for different task size experiments is 704 J. Its average energy consumption is reduced by 12.80%, 25.75%, and 19.77% compared to DDQN, Random, and OE algorithms. This shows that the Lagrange-DDPG algorithm more significantly reduces the computational energy consumption for task offloading in edge computing systems as the task size increases. This is because the Lagrange multiplier method and the DDPG algorithm can find the optimal task offloading and resource allocation strategies more comprehensively and quickly when the task size changes.

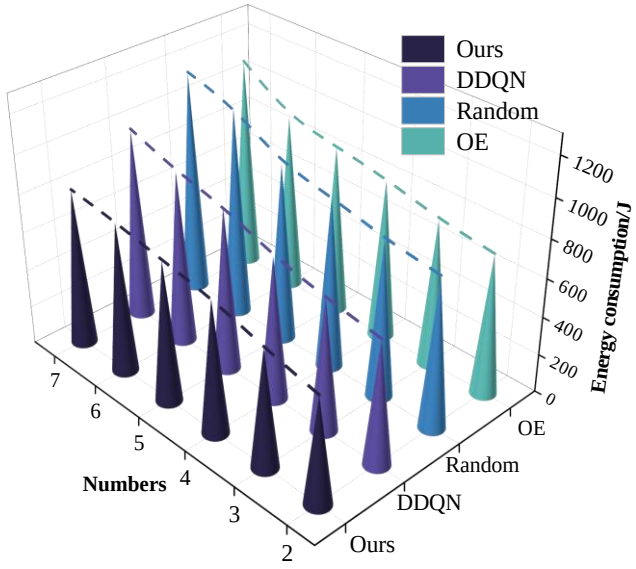


Figure 4: The impact of different task size changes on system energy consumption

3.4 Overall performance evaluation

In order to further validate the real-time control performance of the deep reinforcement learning-based task offloading and resource allocation algorithms proposed in this paper, in addition to the above three baseline algorithms, this paper compares their performance with three other classical algorithms, including the greedy algorithm (Greedy), the heuristic algorithm (Heuristic), and the algorithm based on queuing theory (QT), in terms of throughput, latency, and energy consumption Three aspects were tested. In the same simulation scenario, 10 experiments were run for each of the seven algorithms, each lasting 1000 time steps, and the average values of the key performance indicators were recorded. Table 1 shows the performance comparison of different algorithms in terms of system throughput, average task latency and edge server energy consumption. The algorithm proposed in this paper achieves optimal performance in all three metrics, and compared with other algorithms, the system throughput is improved by 9.98% to 28.61%, the average task latency is reduced by 8.05% to 22.48%, and the energy consumption of the edge server is reduced by 4.45% to 16.05%. These results show that the deep reinforcement learning algorithm is able to better adapt to the dynamic changes of the MEC environment and learn better task offloading and resource scheduling strategies, thus improving the overall efficiency and performance of the system while ensuring user experience and real-time control.

Table 1: Performance comparison of different algorithms

Algorithms	Throughput/GFLOPS/s	Average task delay/ms	Energy consumption of MES/W
DDQN	243.24	103.27	275.81
Random	211.35	122.50	306.92
OE	235.01	121.37	303.66
Greedy	228.78	119.91	313.92
Heuristic	247.14	105.14	289.45
QT	237.37	110.97	297.16
Ours	**271.81**	**94.96**	**263.54**

4 Conclusion

With the continuous evolution of mobile communication networks, the efficient resource scheduling required by access devices in mobile edge networks poses a great challenge to the network. In this study, we address the problem of real-time control of resource allocation in edge computing, construct a system model, combine Lagrange multiplier method and DDPG algorithm, and propose offloading decision and resource allocation method based on deep reinforcement learning. Its training and testing situations are analyzed through experiments, and its real-time control performance for edge computing systems is explored.

During training and testing, the Lagrange-DDPG method proposed in this paper outperforms the baseline algorithm in terms of cumulative rewards and task discarding rate, with training results of around 700 and 5%, and testing results of around 790 and 3%, respectively. The energy consumption of this paper's method maintains a good advantage under different end devices and different task sizes, and the average value of energy consumption is reduced by 12.80% to 25.75% under different task sizes. In addition, in terms of overall performance, it improves the throughput by 9.98%~28.61% over other methods, and reduces the average task latency and edge server energy consumption by 8.05%~22.48% and 4.45%~16.05%, respectively. The above results demonstrate that the method in this paper

improves the overall efficiency and performance of the system and can realize real-time control of task offloading and resource allocation in edge computing.

This paper investigates the resource allocation method for mobile edge networks, which provides a meaningful reference for real-time control of task offloading and resource allocation in mobile edge networks. There are some shortcomings in this paper, which need to be further improved and optimized. At present, the hyperparameters of the DDPG algorithm rely on manual setting, and the subsequent need to study the acceleration mechanism of the algorithm and the way of parameter adaptive setting, and to explore the feasibility of using the algorithm in the actual system.

References

[1] Yang, T., Yi, X., Lu, S., Johansson, K. H., & Chai, T. (2021). Intelligent manufacturing for the process industry driven by industrial artificial intelligence. Engineering, 7(9), 1224-1230.

[2] Peres, R. S., Jia, X., Lee, J., Sun, K., Colombo, A. W., & Barata, J. (2020). Industrial artificial intelligence in industry 4.0-systematic review, challenges and outlook. IEEE access, 8, 220121-220139.

[3] Cioffi, R., Travaglioni, M., Piscitelli, G., Petrillo, A., & De Felice, F. (2020). Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions. Sustainability, 12(2), 492.

[4] Gao, Z., Wanyama, T., Singh, I., Gadhrri, A., & Schmidt, R. (2020). From industry 4.0 to robotics 4.0-a conceptual framework for collaborative and intelligent robotic systems. Procedia manufacturing, 46, 591-599.

[5] Bilancia, P., Schmidt, J., Raffaeli, R., Peruzzini, M., & Pellicciari, M. (2023). An overview of industrial robots control and programming approaches. Applied Sciences, 13(4), 2582.

[6] Chen, X., & Guhl, J. (2018). Industrial robot control with object recognition based on deep learning. Procedia CIRP, 76, 149-154.

[7] Yin, S., Rodriguez-Andina, J. J., & Jiang, Y. (2019). Real-time monitoring and control of industrial cyberphysical systems: With integrated plant-wide monitoring and control framework. IEEE Industrial Electronics Magazine, 13(4), 38-47.

[8] Arents, J., & Greitans, M. (2022). Smart industrial robot control trends, challenges and opportunities within manufacturing. Applied Sciences, 12(2), 937.

[9] Suri, S., Jain, A., Verma, N., & Prasertpoj, N. (2018, April). SCARA industrial automation robot. In 2018 international conference on power energy, environment and intelligent control (PEEIC) (pp. 173-177). IEEE.

[10] Xu, J., Wan, W., Pan, L., Sun, W., & Liu, Y. (2024, February). The fusion of deep reinforcement learning and edge computing for real-time monitoring and control optimization in iot environments. In 2024 3rd International Conference on Energy and Power Engineering, Control Engineering (EPECE) (pp. 193-196). IEEE.

[11] Xu, W., Cui, J., Li, L., Yao, B., Tian, S., & Zhou, Z. (2021). Digital twin-based industrial cloud robotics: Framework, control approach and implementation. Journal of Manufacturing Systems, 58, 196-209.

[12] Liu, X., Gan, H., Luo, Y., Chen, Y., & Gao, L. (2023). Digital-twin-based real-time optimization for a fractional order controller for industrial robots. Fractal and Fractional, 7(2), 167.

[13] Righettini, P., Strada, R., & Cortinovis, F. (2023). Neural network mapping of industrial robots' task times for real-time process optimization. Robotics, 12(5), 143.

[14] Lu, Y., Shen, Z., Hu, H., Zhuang, C., & Ding, H. (2024). A unified framework of in-situ calibration and synchronous identification for industrial robots using composite sensing. IEEE Transactions on Automation Science and Engineering.

[15] Chuang, W. L., Yeh, M. H., & Yeh, Y. L. (2021, June). Develop real-time robot control architecture using robot operating system and ethercat. In Actuators (Vol. 10, No. 7, p. 141). MDPI.