



# Edge Collaborative Task Scheduling and Resource Allocation Based on Deep Reinforcement Learning

Tianjian Chen<sup>1</sup>, Zengwei Lyu<sup>1,3(✉)</sup>, Xiaohui Yuan<sup>2</sup>, Zhenchun Wei<sup>1,3(✉)</sup>,  
Lei Shi<sup>1,3</sup>, and Yuqi Fan<sup>1,3</sup>

<sup>1</sup> School of Computer Science and Information Engineering,  
Hefei University of Technology, Hefei 230009, China

<sup>2</sup> Department of Computer Science and Engineering University of North Texas  
Denton, Denton, TX 76203, USA

<sup>3</sup> Engineering Research Center of Safety Critical Industrial Measurement  
and Control Technology, Ministry of Education, Hefei 230009, China  
986397142@qq.com

**Abstract.** With the development of the sixth generation mobile network (6G), the arrival of the Internet of Everything (IoE) is accelerating. An edge computing network is an important network architecture to realize the IoE. Yet, allocating limited computing resources on the edge nodes is a significant challenge. This paper proposes a collaborative task scheduling framework for the computational resource allocation and task scheduling problems in edge computing. The framework focuses on bandwidth allocation to tasks and the designation of target servers. The problem is described as a Markov decision process (MDP). To minimize the task execution delay and user cost and improve the task success rate, we propose a Deep Reinforcement Learning (DRL) based method. In addition, we explore the problem of the hierarchical hash rate of servers in the network. The simulation results show that our proposed DRL-based task scheduling algorithm outperforms the baseline algorithms in terms of task success rate and system energy consumption. The hierarchical settings of the server's hash rate also show significant benefits in terms of improved task success rate and energy savings.

**Keywords:** Edge collaborative · Task scheduling · Deep reinforcement learning · Hierarchical server

## 1 Introduction

Field of metaverse and autonomous driving, massive amounts of data place incredibly high demands on hash rate, and the construction of a mobile edge computing (MEC) network platform is expected to cope with this challenge

---

Supported by the Natural Science Foundation of Anhui Province (2108085MF202).

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022

L. Wang et al. (Eds.): WASA 2022, LNCS 13473, pp. 598–606, 2022.

[https://doi.org/10.1007/978-3-031-19211-1\\_49](https://doi.org/10.1007/978-3-031-19211-1_49)

[1, 2]. The network of edge nodes allocates computing resources upon requests of offloading a task to the network. Scheduling methods have been developed, e.g., workflow-based dynamic scheduling algorithms [3, 4], self-adaptive learning particle swarm optimization (SLPSO) [5, 6]. Methods based on Deep Reinforcement Learning are developed for the task scheduling problem [7] in the MEC system.

In addition, setting optimization goals is essential in algorithm evaluation. Many scholars use task delay [8] and energy consumption [9, 10] as evaluation goals [11–14].

This paper proposes an optimal scheduling strategy suitable for delay constraints and user rent constraints to address this challenge. We propose an improved DRL-based task scheduling algorithm to solve multiple tasks' edge cooperative scheduling problem. Our main contributions are as follows:

- An edge collaborative task scheduling framework is proposed for the computational resource allocation and task scheduling problems in edge scenarios. The problem considers the bandwidth resources of the server, task characteristics, task queue state of the target server, and arithmetic power level.
- We propose a DRL-based algorithm to solve this scheduling problem, improving algorithm convergence and minimizing the weighted sum of task latency and cost by allocating appropriate bandwidth resources and target servers.
- Simulation results demonstrate that the proposed method outperforms the baseline algorithms. In addition, the hierarchical settings of arithmetic power for servers in the network effectively reduces the system energy consumption.

The rest of this article is organized as follows. Section 2 introduces the system model and describes the problem. Section 3 presents our proposed algorithm including algorithm design and process description. Section 4 presents our experimental results and discussion. Section 5 concludes this paper with a summary.

## 2 System Model and Problem Description

### 2.1 Model Overview

As shown in Fig. 1, our model consists of servers with hierarchical hash rate and multiple end-users. There are two waiting queues on each server, the waiting scheduling queue, and the waiting execution queue. The edge servers can be represented as  $\mathcal{M} = \{1, 2, \dots, M\}$ .  $\mathcal{N} = \{1, 2, \dots, N\}$  represents a task generated by the end-user.

Users send task requests to the closest server  $m'$  first, which we define as the local server. The local server receives the task  $A_{m'n} = \{C_{m'n}, L_{m'n}, Cost_{m'n}\}$ , where  $C_{m'n}$  denotes the task size and  $L_{m'n}, Cost_{m'n}$  denotes the task's latest response time and the maximum acceptable overhead for that task, respectively.

### 2.2 Task Scheduling Model

After the user sends a task request to the local server  $m'$ , the scheduler in the local server  $m'$  allocates transmission bandwidth and gives it to the specified target server  $m$  for execution. The entire scheduling process can be divided into the following four stages.

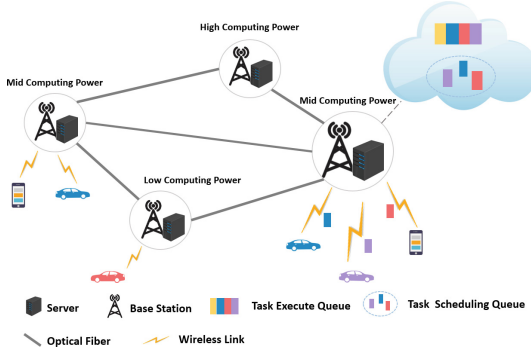


Fig. 1. The structure of our system model.

**Task Transmission:** After the local server  $m'$  receives the task request, it allocates appropriate bandwidth resources to it, and the end-user transmits the task to the local server  $m'$  through the wireless channel, we ignore the time delay, the uplink transmission rate can be defined as:

$$o_{m'n} = Q \log \left( 1 + \frac{p_{m'n} |h_{m'n}|^2}{\sigma^2} \right) \tag{1}$$

where  $p_{m'n}$  is transmission channel bandwidth between the task  $n$  and the local server  $m'$ ,  $h_{m'n}$  represents the channel gain, which is time-varying, and  $\sigma^2$  is the noise. The transfer time of the task is:

$$ST_{m'n} = \frac{C_n}{o_{m'n}} \tag{2}$$

**Task Scheduling:** The scheduler of the local server gives the optimal task scheduling policy according to the currently observed network status, such as the bandwidth of the local server and the available computing resources of each server in the network.

**Waiting for Execution:** We use  $Que_m$  to indicate the number of tasks in the waiting execution queue of the target server  $m$ . Therefore, the waiting execution time of the task is:

$$WT_{mn} = \begin{cases} 0, & Que_m = 0 \\ freeT_{mn} - AT_{mn}, & else \end{cases} \tag{3}$$

where  $freeT_{mn}$  represents the idle time of the server  $m$  after task  $n$  arrives.  $AT_{mn}$  is the task  $n$  arrival time to the target server. If it is in working condition, the task needs to wait. At this time, the idle time of the target server can be expressed as:

$$freeT_{mn} = \begin{cases} freeT_{mn'} + ET_{mn'}, & if\ freeT_{mn'} > AT_{n'} \\ AT_{n'} + ET_{mn'}, & else \end{cases} \tag{4}$$

where  $n'$  represents the previous task of the task  $n$ ,  $freeT_{mn'}$ ,  $ET_{mn'}$ ,  $AT_{n'}$  denote respectively the idle time of the target server after task  $n'$  arrive, the execution time of the task  $n'$ , and the arrival time after the task  $n'$  arrives at the target server.

**Task Execution:** The task size  $C_n$  and server compute speed  $f_m$  are known. So the estimated computing time of the task  $n$  on the server  $m$  can be expressed as

$$ET_{mn} = \frac{C_n}{f_m}. \quad (5)$$

In summary, after the completion of task  $n$ , the estimated completion time is the sum of the task  $n$ 's transmission time, waiting time for execution, and task execution time, which is expressed as follows

$$T_n = ST_{m'n} + WT_{mn} + ET_{mn}. \quad (6)$$

### 2.3 User Cost Model

The unit cycle price is  $u_m$ , and the higher the hash rate, the greater the  $u_m$ . Therefore, the total cost to be paid to the service after the execution of task  $n$  is

$$U_{mn} = \frac{C_n}{f_m} * u_m. \quad (7)$$

### 2.4 Problem Description

Our optimization goal is to minimize task execution delay and user cost overhead. The optimization problem can be described as follows

$$\begin{aligned} P1 : \min_p & \left\{ \alpha * \sum_{n=1}^N T_n + (1 - \alpha) * \sum_{n=1}^N U_{mn} \right\} \\ s.t. \quad C1 : & \alpha \in (0, 1) \\ C2 : & 0 < f_m \leq f_m^{max}, & \forall m \in \mathcal{M} \\ C3 : & 0 < p_{m'n} \leq p_{m'n}^{max}, & \forall m' \in \mathcal{M}, \forall n \in \mathcal{N} \\ C4 : & T_n \leq L_n, & \forall n \in \mathcal{N} \\ C5 : & U_{mn} \leq Cost_n, & \forall m \in \mathcal{M}, \forall n \in \mathcal{N} \end{aligned} \quad (8)$$

where  $\alpha$  represents the weight factor, which the scheduler can define according to different task requirements [15].  $C2$  indicates that the server's hash rate is limited, and  $C3$  indicates that the transmission power between each server and the task does not exceed the maximum value. Conditions  $C4$  and  $C5$  represent the task's constraints on time and cost.

## 3 Proposed Method

We transform Problem P1 into an MDP problem and design a Deep Deterministic Policy Gradient-based online scheduling and allocation (SA-DDPG) algorithm to solve it. The structure of our proposed network is shown in Fig. 2.

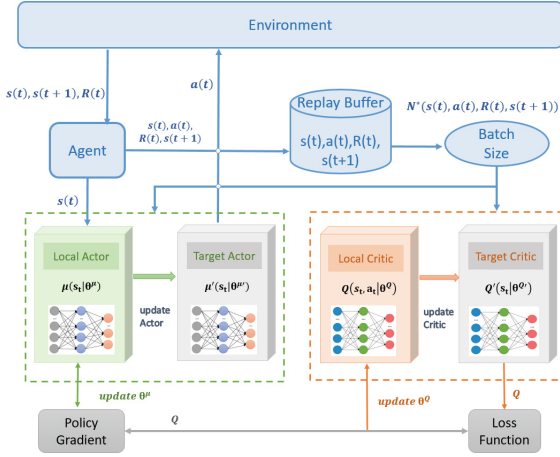


Fig. 2. SA-DDPG algorithm structure.

### 3.1 MDP-Based Task Scheduling Problems

**State Space:** The system state  $s(t)$  consists of two components: the channel gain  $h_{m'n}(t)$  between the user and the local server and the server state information  $s_m(t)$  in the network. Thus, the state space at moment  $t$  can be expressed as :

$$s(t) = \{h_{m'n}(t), s_1(t), s_2(t), \dots, s_m(t)\} \tag{9}$$

**Action Space:** We define the action space as:

$$a(t) = \{p_{m'n}(t), a_{1n}(t), a_{2n}(t), \dots, a_{mn}(t)\} \tag{10}$$

Here  $a_{mn} \in \{0, 1\}$ , and  $a_{mn} = 1$  represents offloading task  $n$  to server  $m$ ,  $a_{1n} + a_{2n} + \dots + a_{mn} = 1$ .

**Reward Function:** Our optimization problem  $P1$  is to minimize the delay and cost of task completion, so we set the reward to:

$$R = \frac{L_{m'n} - T_n}{L_{m'n}} + \frac{Cost_{m'n} - U_{mn}}{Cost_{m'n}} \tag{11}$$

### 3.2 SA-DDPG Algorithm Framework

We describe the network structure of the algorithm in Fig. 2. The weight parameters  $\theta^\mu$  and  $\theta^Q$  of the Actor network and the Critic Network are randomly initialized at the beginning of the algorithm. We use  $Q'$  and  $\mu'$  to improve learning stability in the target network.

**Algorithm 1** Deep Deterministic Policy Gradient-based Online Scheduling and Allocation algorithm

---

**Input:** Task  $A_{m'n} = \{C_{m'n}, L_{m'n}, Cost_{m'n}\}$ , channel gain  $h_{m'n}$   
**Output:** Optimal scheduling policy  $a^*(t)$   
Initialize critic network  $Q(s, a|\theta^Q)$  and actor network  $\mu(s|\theta^\mu)$  with weights  $\theta^Q$  and  $\theta^\mu$ ;  
Initialize target network  $Q'$  and  $\mu'$  with weights  $\theta^{Q'} \leftarrow \theta^Q$  and  $\theta^{\mu'} \leftarrow \theta^\mu$   
Initialize the empty replay buffer  $\mathcal{B}$ ;  
**for**  $t = 1, 2, \dots, T$  **do**  
  Takes system state  $s(t)$  as an input to the actor network and obtains action  $a(t) = \mu(s(t)|\theta^\mu) + \delta_t$   
  Execute action  $a(t)$ , obtain the reward  $R(t)$  and next state  $s(t+1)$   
  Store transition tuple  $s(t), a(t), R(t), s(t+1)$  into  $\mathcal{B}$   
  **if** learning time reaches **then**  
    Agent collects  $K$  samples from  $\mathcal{B}$   
    Update critic by minimizing the loss function in Equation (13)  
    Update actor policy by the deterministic policy gradient in Equation (14)  
  **end if**  
  regularly update the target networks:  

$$\theta^{Q'} \leftarrow \tau\theta^{Q'} + (1 - \tau)\theta^Q$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu'} + (1 - \tau)\theta^\mu$$
  
**end for**

---

**Algorithm Training:** The Agent obtains states  $s(t)$  from the environment and selects the current best action  $a^*(t)$ . After taking action  $a^*(t)$ , the Agent receives a reward  $R_t$  and subsequently observes the next state  $s(t+1)$ . The transition buffer is  $\{s(t), a(t), R(t), s(t+1)\}$ , which can be stored into the experience replay memory  $\mathcal{B}$ . In addition, the loss function in Fig. 2 is:

$$L = \mathbb{E} \left[ \left( R(t) + \gamma Q' \left( s(t+1), a(t+1) | \theta^{Q'} \right) - Q \left( s(t), a(t) | \theta^Q \right) \right)^2 \right] \quad (12)$$

where  $\gamma$  is the attenuation coefficient. And policy gradient can be expressed as:

$$\nabla_{\theta} \mathcal{T} = E \left[ \nabla_a Q \left( s, a | \theta^Q \right) \Big|_{s=s(t), a=\mu(s(t))} \nabla_{\theta^\mu} \mu \left( s | \theta^\mu \right) \Big|_{s(t)} \right] \quad (13)$$

We formalize the SA-DDPG algorithm process in Algorithm 1.

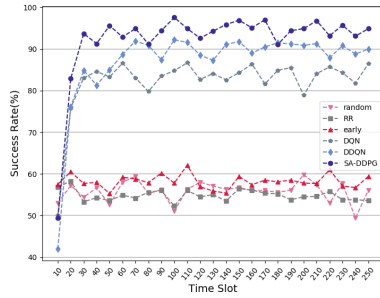
## 4 Experimental Analysis

### 4.1 Settings

We compare the task success rate and energy consumption with the random scheduling algorithm, the round-robin [16] scheduling algorithm, the earliest scheduling algorithm, and the DRL algorithms of DQN [17] and DDQN. The detailed simulation parameters are shown in Table 1.

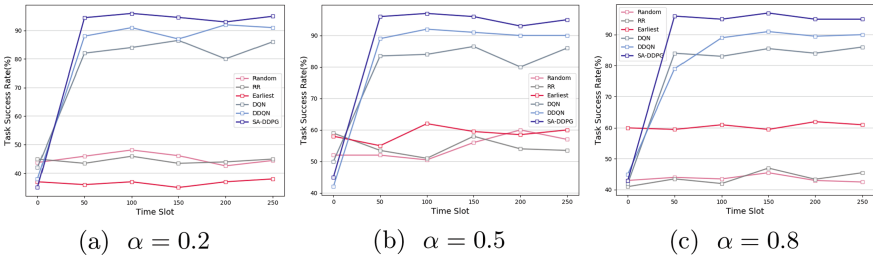
**Table 1.** Simulation Parameters

Parameters	Description	Value
$f_m$	Server hash rate	[12.5 MIPS/s,15 MIPS/s,17.5 MIPS/s]
$C_n$	Size of task n	[500 KB, 1200 KB]
$M$	Number of servers	15
$Mcapacity$	Computing capacity of the server	1000 MIPS
$p_m^{max}$	Maximum transmission power	0.2 M
$\omega$	channel bandwidth	0.5 MHz
$\alpha_{actor}$	Actor network learning rate	0.01
$\alpha_{critic}$	Critic network learning rate	0.02
$\gamma$	The discount factor	0.9



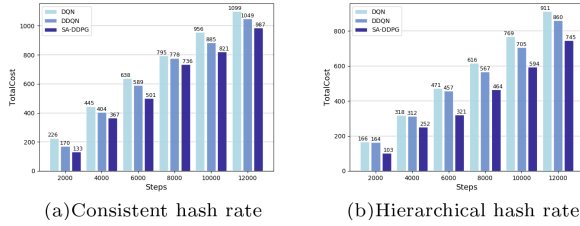
**Fig. 3.** The task success rate.

As shown in Fig. 3, the task success rate of the DRL-based family of algorithms is 30%-40% higher than that of the conventional algorithms because the DRL-based algorithms can make learning based on historical experience and continuously train to optimize the decisions.



**Fig. 4.** Impact of different weighting factors  $\alpha$  on task success rate.

We set  $\alpha$  to 0.2, 0.5, and 0.8 for cost-sensitive tasks, balanced tasks, and delay-sensitive tasks, respectively. As shown in Fig. 4, the task success rate of our proposed algorithm consistently outperforms the other baseline algorithms in the three task type arrival scenarios, converging to about 93%.



**Fig. 5.** Total energy consumption in different hash rate environments.

Furthermore, we explore the impact of the hierarchical hash rate of the servers on system energy consumption in Fig. 5. And after the server hash rate is hierarchical, the total energy consumption of all algorithms is reduced by about 25% compared with the ungraded case.

## 5 Conclusion

In this paper, we study the task scheduling problem in edge scenarios. To solve the problem of allocating bandwidth resources to servers and scheduling tasks among servers, we propose a DRL-based algorithm to reduce the total task latency and user overhead to maximize the success rate of tasks. In addition, we verified that our algorithm is highly adaptable and capable of handling any type of task. In future work, we can further explore mobility under multi-edge networks.

## References

1. Kye, B., Han, N., Kim, E., Park, Y., Jo, S.: Educational applications of metaverse: possibilities and limitations. *J. Educ. Eval. Health Prof.* **18**, 32 (2021)
2. Abbas, M., Siddiqi, M.H., Khan, K., Zahra, K., Naqvi, A.U.: Haematological evaluation of sodium fluoride toxicity in oryctolagus cuniculus. *Toxicol. Rep.* **4**, 450–454 (2017)
3. Cai, Z., Li, X., Ruiz, R., Li, Q.: A delay-based dynamic scheduling algorithm for bag-of-task workflows with stochastic task execution times in clouds. *Futur. Gener. Comput. Syst.* **71**, 57–72 (2017)
4. Jiang, H., E, H., Song, M.: Dynamic scheduling of workflow for makespan and robustness improvement in the iaas cloud. *IEICE Trans. Inf. Syst.* **E100.D(4)**, 813–821 (2017)
5. Zuo, X., Zhang, G., Tan, W.: Self-adaptive learning pso-based deadline constrained task scheduling for hybrid iaas cloud. *IEEE Trans. Autom. Sci. Eng.* **11(2)**, 564–573 (2014)
6. Fu, Z., Tang, Z., Yang, L., Liu, C.: An optimal locality-aware task scheduling algorithm based on bipartite graph modelling for spark applications. *IEEE Trans. Parallel Distrib. Syst.* **31(10)**, 2406–2420 (2020)



7. Yan, J., Bi, S., Zhang, Y.J.A.: Offloading and resource allocation with general task graph in mobile edge computing: a deep reinforcement learning approach. *IEEE Trans. Wireless Commun.* **19**(8), 5404–5419 (2020)
8. Du, J., Yu, F.R., Chu, X., Feng, J., Lu, G.: Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization. *IEEE Trans. Veh. Technol.* **68**(2), 1079–1092 (2019)
9. Zhang, J., Hu, X., Ning, Z., Ngai, E.C.H., Zhou, L., Wei, J., Cheng, J., Hu, B.: Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks. *IEEE Internet Things J.* **5**(4), 2633–2645 (2018)
10. Hong, Z., Huang, H., Guo, S., Chen, W., Zheng, Z.: Qos-aware cooperative computation offloading for robot swarms in cloud robotics. *IEEE Trans. Veh. Technol.* **68**(4), 4027–4041 (2019)
11. Wang, Y., Sheng, M., Wang, X., Wang, L., Li, J.: Mobile-edge computing: partial computation offloading using dynamic voltage scaling. *IEEE Trans. Commun.*, 1 (2016)
12. Shah-Mansouri, H., Wong, V.W.S., Schober, R.: Joint optimal pricing and task scheduling in mobile cloud computing systems. *IEEE Trans. Wireless Commun.* **16**(8), 5218–5232 (2017)
13. Li, J., Qiu, M., Ming, Z., Quan, G., Qin, X., Gu, Z.: Online optimization for scheduling preemptable tasks on iaas cloud systems. *J. Parallel Distributed Comput.* **72**(5), 666–677 (2012)
14. Lu, H., He, X., Du, M., Ruan, X., Sun, Y., Wang, K.: Edge qoe: Computation offloading with deep reinforcement learning for internet of things. *IEEE Internet Things J.* **7**(10), 9255–9265 (2020)
15. Chun, B.G., Maniatis, P.: Augmented smartphone applications through clone cloud execution. In: *Proceedings of the 12th Conference on Hot Topics in Operating Systems, HotOS 2009*, p. 8. USENIX Association, USA (2009)
16. Devi, K., Paulraj, D., and B.M.: Deep learning based security model for cloud based task scheduling. *KSII Trans. Internet Inf. Syst.* **14**(9), 3663–3679 (2020)
17. Van Le, D., Tham, C.K.: A deep reinforcement learning based offload scheme in ad-hoc mobile clouds. In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 760–765 (2018)