Load Balancing for Task Scheduling Based on Multi-Agent Reinforcement Learning in Cloud-Edge-End Collaborative Environments

Zhuo Li
School of Computer Science, National University of Defense Technology, Changsha, China

Jie Yu*
School of Computer Science, National University of Defense Technology, Changsha, China

Xiaodong Liu
School of Computer Science, National University of Defense Technology, Changsha, China

Long Peng
School of Computer Science, National University of Defense Technology, Changsha, China

ABSTRACT

With the increasing variety of computational scenarios and task types in cloud-edge-end collaborative networks, task scheduling in cloud-edge-end collaborative environments can better adapt to various task types and application scenarios, thereby enhancing the flexibility and adaptability of cloud-edge-end systems. This paper introduces a multi-agent reinforcement learning approach to conduct research on task load balancing scheduling in the context of cloud-edge-end collaboration, aiming to improve the efficiency of finding optimal task scheduling strategies in a distributed cloud-edge computing environment. In this paper, task scheduling is viewed as a competitive multi-agent system, where intelligent agents compete for a sufficient number of computing resources through the design of efficient task scheduling algorithms. This competition allows agents to reduce task completion latency and energy consumption while meeting task computational requirements. The paper employs Decentralized Partially Observable Markov Decision Process to model the reward maximization problem and designs a multi-agent reinforcement learning algorithm based on attention communication to solve it. Finally, experimental validation is conducted to evaluate the performance of the proposed task scheduling method.

CCS CONCEPTS

• **Scheduling methodologies;**; • **Reinforcement learning;**; • **Reinforcement learning approaches;**;

KEYWORDS

Collaborative task scheduling, Cloud-Edge-End collaborative, Multi-Agent reinforcement learning

*Corresponding author is Jie Yu (e-mail: yj@nudt.edu.cn).

ICMLSC 2024, January 26–28, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-1654-6/24/01
https://doi.org/10.1145/3647750.3647765

ACM Reference Format:
Zhuo Li, Jie Yu, Xiaodong Liu, and Long Peng. 2024. Load Balancing for Task Scheduling Based on Multi-Agent Reinforcement Learning in Cloud-Edge-End Collaborative Environments. In *2024 The 8th International Conference on Machine Learning and Soft Computing (ICMLSC 2024), January 26–28, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3647750.3647765

1 INTRODUCTION

In the face of a complex environment that encompasses multiple tasks and a diverse array of edge and cloud devices, the efficient scheduling for these tasks are imperative to achieve overall optimization [1]. The practice of collaborative computation resource scheduling at the end, edge and cloud is indeed an effective approach. However, exclusive reliance on distributed resource allocation proves insufficient, as it tends to overlook the unique characteristics of individual computing nodes. As each computing node exhibits distinct performance capabilities and varying workloads, rendering a one-size-fits-all allocation strategy inappropriate. Such an approach may result in certain nodes being burdened with excessive workloads, while others suffer from resource underutilization.

Traditional task scheduling algorithms typically rely on static rules or greedy strategies, and lack dynamic adjustments and adaptability, making them ill-suited for the complex task environments and network topologies at the end, edge and cloud [2]. Furthermore, these algorithms often struggle to meet real-time and efficiency requirements, especially when task complexity and scale continue to grow, leading to a significant decrease in their performance.

With the significant success of reinforcement learning techniques in the field of decision-making, both single-agent reinforcement learning-based task scheduling and distributed reinforcement learning-based task scheduling have become mainstream approaches. However, single-agent approaches may not be able to meet the demands of large-scale task scheduling [3]. On the other hand, distributed reinforcement learning-based task scheduling often lacks consideration for the collaborative relationships among individual agents [4].

2 RELATED WORKS AND MOTIVATION

2.1 Related Works

The current research on task scheduling optimization encompasses various optimization objectives based on real-world application

scenarios and requirements. The primary objectives include optimizing task scheduling for latency, energy consumption, and load balancing.

In the context of optimizing task scheduling for latency and energy consumption, Liu et al. introduced a latency-optimized task scheduling algorithm. This algorithm, through steps such as state partitioning, delay prediction, and task allocation, achieves latency-optimized task scheduling in edge computing systems [5]. Shen et al. proposed an adaptive task scheduling strategy that minimizes energy consumption while ensuring latency performance [6].

When considering both latency and energy consumption in task scheduling, Yang et al. introduced the DEBTS algorithm. This algorithm employs a greedy strategy to divide task scheduling into multiple stages. In each stage, the algorithm prioritizes meeting task latency constraints while minimizing the overall energy consumption required for task completion [7]. Hosseini et al. presented a task scheduling algorithm based on the Fuzzy Analytic Hierarchy Process. By assigning different weight factors to CPU-intensive and I/O-intensive tasks, this algorithm effectively balances latency and energy consumption [8].

Load balancing optimization typically involves how to distribute tasks reasonably among multiple nodes to achieve load balancing optimization. Gupta et al. proposed a load balancing task scheduling algorithm based on the Ant Colony Optimization. This algorithm monitors the load of virtual machines and assigns tasks to virtual machines with lower load to achieve load balancing, thereby maximizing the reduction in overall energy consumption [9]. Ebadifard and Babamir introduced a dynamic task scheduling algorithm based on load balancing, which can dynamically adjust task allocation and resource allocation based on the system's load and task priorities, thus improving task scheduling efficiency and overall system performance [10]. Alnusairi and their group presented a load balancing task scheduling algorithm based on Binary Particle Swarm and Ant Colony Optimization. By dynamically adjusting the parameters in the Particle Swarm Optimization algorithm, the algorithm can effectively handle load balancing issues in cloud environments [11]. Panda and Jana proposed a load balancing task scheduling algorithm based on a probabilistic model. This algorithm randomly allocates task sets to different virtual machines and calculates the task load for each virtual machine using a probabilistic model to achieve load balancing [12].

Research in task scheduling requires the integration of load balancing optimization with the development of more intelligent and efficient algorithms to meet the demands of task scheduling in cloud-edge-end collaborative scenarios. Furthermore, machine learning and adaptive technologies will also find widespread applications in load balancing algorithms for task scheduling to achieve better performance and outcomes.

2.2 Load Balancing for Task Scheduling

With the increasing complexity of computational scenarios and the diversification of tasks in cloud-edge collaborative networks, task-oriented cloud-edge coordination scheduling can better adapt to various task types and application scenarios. This enhances the flexibility and adaptability of cloud-edge systems. Furthermore, dynamic task allocation according to the actual situation of each computational node is necessary to ensure load balancing within the task queue. This helps prevent situations of excessive load or underutilization. Therefore, in cloud-edge collaborative task scheduling, it's essential to introduce load balancing mechanisms to better distribute the workload across each node, avoiding situations of either excessive or insufficient load.

3 PROBLEM AND FORMULATION

This section introduces the collaborative scheduling model and optimization problems in the Cloud-Edge-End collaborative environment. Table 1 provides explanations for the model parameters.

3.1 Cloud-Edge-End Collaborative Scheduling Model

3.1.1 Cloud-Edge-End Collaborative Framework. In the distributed Cloud-Edge-End environment, task scheduling involves collaboration and competition among multiple edge servers. It also needs to consider system performance and efficiency. Traditional single-agent reinforcement learning algorithms struggle with these issues because they only optimize individual agents and overlook the interactions and influences among agents. A Multi-Agent Reinforcement Learning (MARL) framework is well-suited to address task scheduling problems in the Cloud-Edge-End distributed environment. To achieve this, this paper establishes a Cloud-Edge-End collaborative task scheduling framework based on MARL.

The MARL framework is deployed in the distributed computational environment of the cloud, edge, and end. Task scheduling servers on the cloud and edge devices act as multiple agents that collaborate. Each agent formulates task scheduling strategies by observing environmental states and communicating with other agents. They update their learning based on received reward signals to maximize task completion rates and quality, thereby enhancing user experience and system performance.

3.1.2 Load Balancing Scheduling Model. To evaluate the optimization performance of a task scheduling strategy, it is common to consider a combination of three aspects: task completion latency, energy consumption, and system utility. In this paper, system utility is defined as the state of load balancing. When there is an imbalance in the workload among the nodes, it reduces the resource utilization of the nodes. Therefore, the proposed load balancing scheduling model aims to maximize system utility by adjusting task allocation and resource scheduling to effectively balance the workloads across all nodes, which addresses the issue of workload imbalance, leading to improved resource utilization and system performance.

Assuming there are n tasks to be assigned to m computing nodes, where each task requires different computing resources, and each computing node has different processing capabilities, the completion time for all tasks is defined as:

$$T_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{m} t_{ij} \times x_{ij} \tag{1}$$

3.2 Scheduling Optimization Problem

Building upon the cloud-edge-end collaborative framework and utilizing the load balancing scheduling model, we introduce a joint optimization problem aimed at minimizing task completion latency and task scheduling energy consumption. The problem is defined

Table 1: Task Scheduling model parameters

Symbol	Definition
$n,\ m$	tasks, nodes
x_{ij}	task assigned to computing node
T_{ij}	task completion time
$E_{i,j}$	task completion energy consumption
σ	load standard deviation
L_j	node current load
C_j	node load capacity

as follows:

$$\begin{aligned} \min_{x} \quad & f(x) = w_1 T_{ij} + w_2 E_{i,j} + w_3 \sigma \\ & \text{s.t.} \sum_{j=1}^{m} x_{ij} = 1, \forall i \ \ (1) \\ & \sum_{i=1}^{J} x_{ij} L_j \le C_j, \forall j \ \ (2) \\ & x_{ij} \in 0, 1, \forall i, j \end{aligned} \tag{2}$$

where the load standard deviation σ is one of the optimization objectives, aiming to measure the balance of task allocation across various computing nodes. A smaller value indicates better load balancing. The constraint 1 represents that each task must be allocated to a computing node, and the constraint 2represents that the load on each computing node must not exceed its capacity.

The Dec-POMDP model can effectively describe the optimization problem of the task scheduling mentioned above. In this model, each agent acts as an independent decision-maker, allowing for the modeling of observations and actions for each agent while considering their interactions, which enables each agent to make more accurate decisions.

The scheduling optimization problem can be modeled using Dec-POMDP and represented as $\langle S, A, P, R \rangle$, where S is the state space, A is the action space, P is the state transition probability function $P(s^{'} \mid s, a)$, and R is the reward function $R(s, a, s^{'})$.

The task scheduling state refers to the node state in this paper, and each node's task execution state vector is denoted as s_i^'∈S, the task scheduling state can be represented as the Cartesian product of the state vectors of each node and can be expressed as:

$$s^{'} = \prod_{i=1}^{N} s_i^{'} \tag{3}$$

The task scheduling action a is an action taken by the agent based on the current observation and state to choose a task to perform. Therefore, the task scheduling action can be represented as $a \in 1, 2, \ldots, n$, indicating the task's number to be executed.

The reward function is employed to assess the quality of each scheduling decision, and it is defined as the negation of the optimization objective in joint optimization problems and formulated as follows:

$$R_i(s, a_i) = -\omega_1 T_{ij} - \omega_2 E_{i,j} - \omega_3 \sigma \tag{4}$$

where ω_1, ω_2, and ω_3 are weight factors for latency, energy consumption, and load balancing, respectively. By minimizing the reward function, the optimization objectives can be achieved.

According to the Bellman equation for Dec-POMDP, the optimization problem of Dec-POMDP can be transformed into an optimization problem of a value function. The Bellman equation can be expressed as:

$$V^*(s) = \max_{a_i \in A} R(s, a) + \gamma \sum_{s' \in S} P\left(s^{'} s, a\right) \sum_{i=1}^{n} \omega_i V_i^* \left(s^{'}\right) \tag{5}$$

where $V^*(s)$ represents the maximum cumulative reward achievable by following a task scheduling strategy in state s, which can be obtained by training the agents using an appropriate MARL algorithm.

4 MARL ALGORITHM DESIGN

To address the collaborative task scheduling optimization problem described above, this section builds upon the Multi-Actor-Attention-Critic (MAAC) algorithm and introduces an attention communication mechanism, resulting in the design of the Multi-Agent Attention-communication Actor-Critic (MA3C) algorithm.

4.1 Multi-Actor-Attention-Critic

The MAAC algorithm incorporates an attention mechanism in the Actor network of each agent, enabling them to focus on the states and actions of other agents. Additionally, in the MAAC algorithm, all agents share the same parameters for the Critic network, referred to as a centralized Critic. This mechanism allows for the more reasonable integration of Critic information using centralized data and helps the MAAC algorithm to better handle competition and cooperation relationships in multi-agent environments. This approach becomes increasingly advantageous as the number of agents rises.

Specifically, the goal of the MAAC algorithm is to maximize the expected cumulative return for each agent. This can be defined as follows:

$$J_i(\theta_i) = \mathbb{E}_{\tau} \sim p\theta_i \left[\sum_{t=0}^{T} \gamma^t r_{i,t} \right] \tag{6}$$

where τ represents a sequence of actions and states executed by an agent, $r_{i,t}$ represents the instantaneous reward and T represents the maximum time steps for agent execution.

MARL algorithms based on a centralized Critic, exemplified by MAAC, primarily focus on studying how to learn independently executable policies through centralized training. However, these approaches often overlook the importance of communication, resulting in the need to transmit a substantial amount of information.

4.2 Multi-Agent Attention-communication Actor-Critic

Despite communication being considered a facilitator of collaboration among multiple agents, much of the prior work has assumed that information is shared among all agents. However, when the number of agents is very large, individual agents may struggle to distinguish valuable information from the vast pool of globally shared data that can truly enhance collaborative decision-making. In such cases, communication may become ineffective, or even hinder collaboration among agents. Moreover, the transmission of an extensive volume of information requires significant bandwidth, resulting in substantial latency and computational complexity, along with associated costs that can be deemed unacceptable. This paper introduces the MA3C algorithm, a Multi-Agent Attention-communication Actor-Critic algorithm. It employs a communication model based on an attention mechanism that determines when communication is necessary and how to integrate shared information effectively.

MA3C utilizes an Actor-Critic algorithm with a baseline to update the parameters of the policy network. At each time step t, agent i observes the environmental state s_t, selects an action $a_{i,t}$, and receives a reward $r_{i,t}$. The parameters θ_i of the policy network are updated using an Actor-Critic algorithm with a baseline:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s_t, a_{i,t} \sim \pi\theta_i} \left[\nabla_{\theta_i} \log \pi_{\theta_i} (a_{i,t} s_t) \left(r_{i,t} - b(s_t) \right) \right] \quad (7)$$

where $\pi_{\theta_i}(a_{i,t} s_t)$ represents the output of the policy network, denoting the probability of taking action $a_{i,t}$ in state s_t. The term $b(s_t)$ refers to the baseline for state s_t, which is employed to reduce variance. The baseline is learned by a Critic network that utilizes an attention mechanism to focus on information from other agents.

ALGORITHM 1: Multi-Agent Attention-communication Actor-Critic (MA3C) Algorithm

Initialize critic network $Q(s, a_1, \ldots, a_n)$, actor networks $\mu_1(s), \ldots, \mu_n(s)$,
Initialize target networks $Q'(s, a_1, \ldots, a_n)$, $\mu'_1(s), \ldots, \mu'_n(s)$ with weights $\theta_Q, \theta_{\mu_1}, \ldots, \theta_{\mu_n}$, replay buffer R
Initialize attention network $A_1(s, a_1), \ldots, A_n(s, a_n)$, communication network for each agent pair $C_{ij}(s, a_i, a_j)$
for episode = 1, M do
  Initialize a random process N for action exploration
  Receive initial observation state s_1
    for $t = 1, T$ do
      for each agent i:
    Sample an action $a_i = \mu_i(s) + N$
    Execute action $a = [a_1, \ldots, a_n]$ and observe reward r, next state s'
    Store transition (s, a, r, s') in R
  for each agent pair (i, j):
    Sample a batch of transitions (s, a, r, s') from R
    Compute attention weights $a_{ij}^c = C_{ij}(s, a_i, a_j)$
    Compute the weighted sum of the gradients for communication network
    Compute attention weights for each agent: $a_i^c = \sum_{j \neq i} a_{ij}^c$
    Compute the message received by each agent: $m_i = \sum_{j \neq i} a_{ij}^c C_{ij}(s, a_i, a_j)$
  for each agent i:
    Compute attention weights $a_i^a = A_i(s, a_i, m_i)$
    Compute the weighted sum of the gradients for actor network i, the attention-based value function, the gradients for the attention network, the gradients for the communication network;
    Update actor networks using the sampled policy gradient: $\theta_{\mu_i} = \theta_{\mu_i} + \alpha \nabla_{\theta_{\mu_i}} J$
    Update attention network using the sampled policy gradient: $\theta_{A_i} = \theta_{A_i} + \beta \nabla_{\theta_{A_i}} J$
    Update the target networks by polyak averaging:
$$\theta_{Q_1'} = \tau\theta_{Q_1} + (1-\tau)\theta_{Q_1'}, \theta_{Q_2'} = \tau\theta_{Q_2} + (1-\tau)\theta_{Q_2'}, \theta_{\mu_i'} = \tau\theta_{\mu_i}$$
  end for
  Update state $s = s'$
  end for
end for

5 PERFORMANCE EVALUATION

5.1 Experimental Setup

To evaluate the performance of the proposed MA3C algorithm, an open-source multi-agent framework called MPE, developed by OpenAI, was selected to set up the testing environment [13]. MPE provides a foundational environment and testing platform for the development of new multi-agent reinforcement learning algorithms. As a comparison, the MADDPG algorithm [14], MAAC algorithm [13], and a communication-based multi-agent reinforcement learning algorithm, ATOC [15], were chosen.

To ensure fairness, the same core training parameters were used in specific training instances. In order to assess the effectiveness of the collaborative task scheduling approach proposed in this paper, the collaborative task scheduling method based on MA3C, as presented in this paper, was compared in terms of performance with several typical task scheduling approaches.

DRL: A task scheduling method based on distributed reinforcement learning algorithms, where multiple agents run in a distributed system, with each agent representing a computational resource node or processor. They collaborate through reinforcement learning to jointly learn how to optimally schedule tasks and allocate resources [16].

SRL: An adaptive task scheduling method based on single-agent reinforcement learning, treating the entire system as a reinforcement learning environment. It involves centralized task scheduling and resource allocation by a single agent for tasks across the entire system [17].

5.2 Experimental Results and Analysis

The experiments initially compared the convergence of different Multi-Agent Reinforcement Learning (MARL) algorithms, highlighting their convergence performance using average rewards. Figure 1 illustrates the average reward values obtained at each iteration during the training phase for four MARL algorithms. In total, the algorithms were trained for 5000 episodes, with each episode consisting of 25 steps.

On the one hand, compared to MADDPG and MAAC algorithms based on a centralized Critic network, the proposed MA3C and ATOC algorithms, which leverage communication among collaborating multi-agent systems in partially observable scenarios, achieved higher average reward values during the later stages of training, maximizing the utility function of the multi-agent system. On the other hand, compared to ATOC, which also relies on an attention mechanism, the experimental results demonstrated that the proposed MA3C algorithm converges more rapidly, to some extent, showcasing the convergence performance of the MA3C algorithm presented in this paper.

Task completion latency can be used to measure the performance of task scheduling systems. In the experiments of this paper, five different sets of task quantities were used. To make the observed average task completion latency more convincing, the numerical values of the average task completion latency were sampled ten times for three task scheduling schemes. The comparative results of these three schemes are presented in the form of box plots in Figure 2. It can be observed that as the number of tasks increases, the average task completion latency for each algorithm also increases.

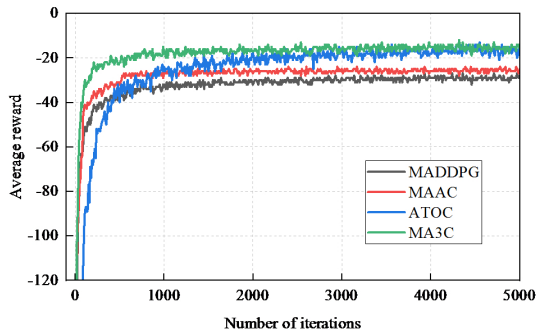

Figure 1: Convergence performance of MARL algorithms

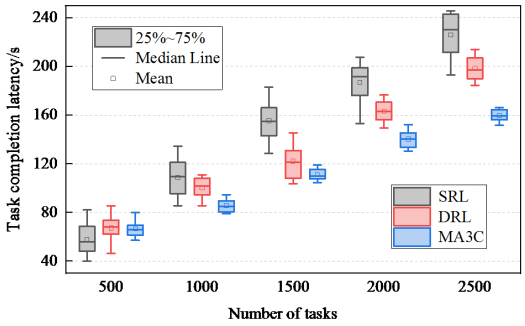

Figure 2: Task completion latency under different task quantities

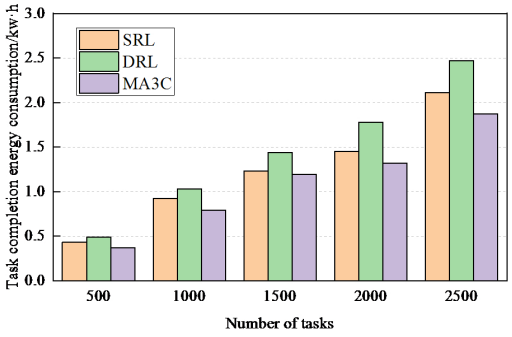

Figure 3: Task completion energy consumption under different task quantities

The proposed MA3C algorithm has the lowest average task completion latency compared to SRL and DRL algorithms. This is because in DRL and MA3C algorithms, multi-agent systems can operate in a distributed environment, with each agent obtaining different information from the environment and taking independent actions, allowing them to better adapt to various environments and tasks.

Furthermore, in comparison to the DRL algorithm, the proposed MA3C algorithm reduces the average task latency by a maximum of 19.5%. This to some extent validates that in the proposed MA3C algorithm, the multi-agent system can better explore unknown environments and tasks through collaboration among agents, thereby improving the overall system performance.

Energy consumption in task scheduling needs to take into account the energy consumption of computational nodes, data transfer strategies, and task scheduling strategies. In the experiments of this paper, different task quantities were used to observe the

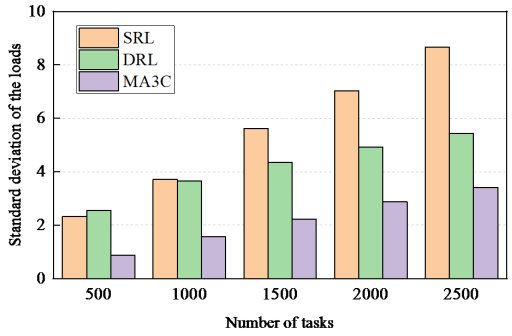

Figure 4: Standard deviation of the loads for each node at the cloud, edge and end

task completion energy consumption of three task scheduling algorithms. The experimental results are shown in Figure 3.

As the number of tasks increases, we observed that the task completion energy consumption of the DRL algorithm, based on distributed reinforcement learning, is the highest, followed by the SRL algorithm, while the lowest energy consumption is achieved by the MA3C algorithm proposed in this paper. The higher energy consumption of the SRL algorithm is mainly due to its use of a single-agent reinforcement learning-based task scheduling approach, which introduces significantly higher energy consumption due to its centralized task scheduling strategy compared to the other two algorithms. The higher latency of the DRL algorithm is attributed to the fact that distributed reinforcement learning typically learns good policies faster than single-agent reinforcement learning but also involves communication and collaboration between multiple agents, leading to increased data transfer energy consumption and consequently higher task energy consumption.

Compared to the SRL and DRL algorithms, the MA3C algorithm reduces task completion energy consumption by up to 14.1% and 25.9%, respectively. This reduction is mainly due to the collaborative task scheduling in the MA3C algorithm, which reduces both the static power consumption within computational nodes and the dynamic power consumption during task execution.

The standard deviation of computational node load is used to represent load balancing. It is calculated by first determining the average load of all computational nodes. Then, the absolute difference between each node's load and the average load is calculated. The absolute differences are summed, and the sum of all absolute differences is divided by the number of computational nodes. This process yields the standard deviation of load for different task quantities in the cloud-edge environment, as shown in Figure 4.

As the number of tasks increases, task scheduling algorithms distribute tasks to various computational nodes, resulting in changes in the load of each computational node. It can be observed that the proposed MA3C algorithm has the lowest load standard deviation, indicating that the MA3C algorithm's collaborative task scheduling functionality effectively selects appropriate computational nodes to execute tasks, thereby achieving load balancing and optimizing the utilization of computational resources.

6 CONCLUSION

This paper conducts research on task load balancing scheduling based on multi-agent reinforcement learning in a cloud-edge collaborative environment. It establishes a task load balancing scheduling model within the cloud-edge collaborative scheduling framework. It uses Dec-POMDP to model the multi-objective optimization problem, considering task completion latency, energy consumption, and load balancing. An actor-critic algorithm, MA3C, based on attention communication, is proposed for solving the problem. Experimental results indicate that the proposed task scheduling approach achieves better load balancing while maintaining favorable task completion latency and task completion energy consumption performance.

REFERENCES

[1] Saleem U, Liu Y, Jangsher S, *et al.* Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing[J]. IEEE Transactions on Wireless Communications, 2020, 20(1): 360-374.
[2] Arunarani A R, Manjula D, Sugumaran V. Task scheduling techniques in cloud computing: A literature survey[J]. Future Generation Computer Systems, 2019, 91: 407-415.
[3] Elgendy I A, Zhang W Z, He H, *et al.* Joint computation offloading and task caching for multi-user and multi-task MEC systems: reinforcement learning-based algorithms[J]. Wireless Networks, 2021, 27(3): 2023-2038.
[4] Wang H, Liu Z, Shen H. Machine learning feature based job scheduling for distributed machine learning clusters[J]. IEEE/ACM Transactions on Networking, 2023, 31(1): 58-73.
[5] Liu J, Mao Y, Zhang J, *et al.* Delay-optimal computation task scheduling for mobile-edge computing systems[C]//2016 IEEE International Symposium on Information Theory (ISIT). IEEE, 2016: 1451-1455.
[6] Shen Y, Bao Z, Qin X, *et al.* Adaptive task scheduling strategy in cloud: when energy consumption meets performance guarantee[J]. World Wide Web, 2017, 20: 155-173.
[7] Yang Y, Zhao S, Zhang W, *et al.* DEBTS: Delay energy balanced task scheduling in homogeneous fog networks[J]. IEEE Internet of Things Journal, 2018, 5(3): 2094-2106.
[8] Hosseini E, Nickray M, Ghanbari S. Optimized task scheduling for cost-latency trade-off in mobile fog computing using fuzzy analytical hierarchy process[J]. Computer Networks, 2022, 206: 108752.
[9] Gupta A, Garg R. Load balancing based task scheduling with ACO in cloud computing[C]//2017 International Conference on Computer and Applications (ICCA). IEEE, 2017: 174-179.
[10] Ebadifard F, Babamir S M. A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment[J]. Concurrency and Computation: Practice and Experience, 2018, 30(12): e4368.
[11] Alnusairi T S, Shahin A A, Daadaa Y. Binary PSOGSA for load balancing task scheduling in cloud environment[J]. arXiv preprint arXiv:1806.00329, 2018.
[12] Panda S K, Jana P K. Load balanced task scheduling for cloud computing: A probabilistic approach[J]. Knowledge and Information Systems, 2019, 61(3): 1607-1631.

[13] Lowe R, Wu Y I, Tamar A, *et al.* Multi-agent actor-critic for mixed cooperative-competitive environments[J]. Advances in Neural Information Processing Systems, 2017, 30.
[14] Iqbal S, Sha F. Actor-attention-critic for multi-agent reinforcement learning[C]//2019 International Conference on Machine Learning. PMLR, 2019: 2961-2970.
[15] Jiang J, Lu Z. Learning attentional communication for multi-agent cooperation[J]. Advances in Neural Information Processing Systems, 2018, 31.
[16] Wang H, Liu Z, Shen H. Machine learning feature based job scheduling for distributed machine learning clusters[J]. IEEE/ACM Transactions on Networking, 2023, 31(1): 58-73.
[17] Pandit M K, Mir R N, Chishti M A. Adaptive task scheduling in IoT using reinforcement learning[J]. International Journal of Intelligent Computing and Cybernetics, 2020, 13(3): 261-282.