D2OP: A Fair Dual-Objective Weighted Scheduling Scheme in Internet of Everything

Zhao Tong, *Member, IEEE*, Bilan Liu, Jing Mei, Jiake Wang, Wenbin Li, and Keqin Li, *Fellow, IEEE*

***Abstract*—In times of the Internet of Everything (IoE), the power of the Internet is growing exponentially, followed by a surge in the number of network requests. The conflict between people's high requirements for the Quality of Experience (QoE) and limited computing resources are becoming increasingly prominent. Therefore, an appropriate offloading method is required to better ease this conflict. In this article, a highly efficient scheduling architecture of information processing under the big data flow of the IoE is proposed to enhance the scheduling performance. First, we construct a dual-channel processing model to describe the entire data flow and node devices. Second, we carefully consider the choice of the weighting method to better find a balance between dual objectives. Third, a dual-objective deep *Q*-network (DQN)-based offloading algorithm with principal component analysis weighting method (D2OP) is proposed to collaboratively minimize task response time and machine load in a more reasonable allocation. To verify the performance of the D2OP, a series of experiments are conducted from multiple angles. The experimental results demonstrate its better performance than the three comparison algorithms in reducing response time, load balance, and increasing task success ratio.**

***Index Terms*—Collaborative optimization, deep *Q*-network (DQN), Internet of Everything (IoE), principal component analysis (PCA), service-level agreement (SLA), task offloading.**

I. Introduction

WITH the research and development of fifth-generation (5G) technology, we have entered the Internet of Everything (IoE) era. Tens of thousands of devices are connected to the network, in addition to traditional devices such as mobile phones, computers, and printers, also including smart home devices, private vehicles, public facilities, and so on [1]. This has greatly increased the value of the Internet and enriched our lives, but it also means an explosive growth in the number of network requests. In the face of such a large number of requests, the local computing power is insufficient to cope. Therefore, we need to offload partial tasks to a place where many computing resources are gathered, which is task scheduling [2], [3].

Benefiting from the great progress in hardware technology, edge technology has emerged as a supplement, filling the insufficiency and relieving the pressure of cloud computing (CC). A better advantage of mobile-edge computing (MEC) is that it has a faster response to tasks, because of its nearby deployment characteristics. However, even in the face of relatively abundant computing resources, it is also crucial to develop an appropriate scheduling strategy [4]. While ensuring the quality of processing, it is also essential to find a better balance among conflicting optimization targets [5], [6]. Only by better coordinating the relationship between computing resources and tasks can satisfactory results be achieved [7].

Many solutions have been proposed for scheduling problems in the MEC environments. The optimization problem in large-scale task scenarios is an NP-hard problem, and solutions based on the optimization theory may take a long time to design strategies. Heuristic algorithms are also widely used to design a low-complexity strategy [8], but they often lack certain theoretical support, and the performance of the algorithm is difficult to guarantee. With the rise of artificial intelligence, many solutions based on deep learning have been applied to many fields, such as natural language processing, automatic driving, and medical applications [9], [10]. However, the inherent characteristics of deep learning require a large number of prior samples to be put into model training, which limits the scope of application. Furthermore, the trained model cannot correct the results in time according to the environment, which means that the entire model must be retrained for each local change. In the face of simultaneous optimization of multiple objectives, some studies are simply subjective weighting. However, there is an overlap between the information that each target depends on, which cannot ensure the full use of information. Therefore, an effective solution that comprehensively considers the scenario requirements of big data, the environmental adaptability of the model, and the weight tradeoff between objectives are proposed in this article. Taking the environment status as the system's input, an effective scheduling scheme can be obtained quickly after sampling training. This method has better adaptability.

Manuscript received 15 May 2022; revised 26 September 2022, 2 December 2022, and 23 December 2022; accepted 1 January 2023. Date of publication 4 January 2023; date of current version 9 May 2023. This work was supported in part by the Program of National Natural Science Foundation of China under Grant 62072174 and Grant 61502165; in part by the National Natural Science Foundation of Hunan Province, China, under Grant 2022JJ40278 and Grant 2020JJ5370; and in part by the Scientific Research Fund of Hunan Provincial Education Department, China, under Grant 22A0026. *(Corresponding author: Zhao Tong.)*

Zhao Tong, Bilan Liu, Jing Mei, and Jiake Wang are with the College of Information Science and Engineering, Hunan Normal University, Changsha 410012, China (e-mail: tongzhao@hunnu.edu.cn; liubilan_0@hunnu.edu.cn; jingmei1988@163.com; 202170293854@hunnu.edu.cn).

Wenbin Li is with the College of Information and Communication Engineering, Hunan Institute of Science and Technology, Yueyang 414006, China (e-mail: wenbin_lii@163.com).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA, and also with the College of Information Science and Engineering, Hunan University, Changsha 410082, Hunan, China (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/JIOT.2023.3234078

In this article, our target is to simultaneously optimize the response time and machine load. Among them, response time is based on the user's perspective. To assure users' Quality of Experience (QoE) [11], [12], a service-level agreement (SLA) constraint is considered, setting the deadline for task finish. If the finish time is later than the deadline, we treat this allocation as a failure. Another objective is to balance the load [13], [14] of all computing nodes as much as possible, based on the service provider's perspective under the premise of guaranteeing response time. Reaching the state of service load balance (SLB) could improve responding speed by decreasing extra waiting time and, thus, reduce network congestion. Nevertheless, it could also avoid the situation in which only a few machines keep working while some have been idle for a long time. This makes a waste of resources, and it is adverse to centralized management of resources. However, only paying attention to user needs will result in the idleness of low-performance machines and the nonstop operation of high-performance machines. In contrast, it will make the response too slow, and the user cannot bear it. Therefore, taking into account the needs and contradictions of the two is the problem studied in this article [15], [16].

Thus, a novel offloading algorithm is proposed to provide a solution for this problem, called the dual-objective deep *Q*-network (DQN)-based offloading algorithm with the principal component analysis (PCA) weighting method (D2OP). The main contributions in this article are as follows.

1) We construct a novel scheduling model in the IoE environment. This model contains a dual-channel design to reflect the actual scheduling situation.
2) We model the scheduling problem in the IoE environment as a Markov decision process (MDP) problem and propose a solution based on deep reinforcement learning (DRL).
3) We integrate the thoughts of parallel and distributed to propose the D2OP algorithm, based on the DRL method, the solution with response time and workload as optimization goals, and to design the feedback of the system.
4) We design a series of experiments to verify the performance of D2OP, and the simulation result shows a better exhibition compared to other machine learning (ML) algorithms.

The remainder of this article is organized as follows. Section II describes the current state of research in this field. Section III introduces the system's entire model and several submodels. Section IV analyzes the proposed D2OP algorithm in detail. Section V presents a series of performance verification experiments of the proposed algorithm. Finally, we summarize the whole article and mention possible future research directions.

II. Related Work

Presently, some scientists have also discovered the defect of the imbalance in the development of hardware and the massive data processing capacity and have performed many studies in this field [17], [18], [19], [20]. This section reviewed four aspects: 1) architecture; 2) objective; 3) weighting; and 4) method. Table I analyzes the comparison between some typical study and this work in four aspects.

A. System Architecture

Chen et al. [21] developed a new advanced method based on the whale optimization algorithm. The algorithm converges faster and with higher convergence accuracy. Abdel-Basset et al. [22] suggested a meta-heuristic algorithm in fog computing, with outstanding effects in many aspects. Mohanraj and Santhosh [23] proposed a novel approach in a multicloud environment with a conspicuous improvement in many facets. Liu et al. [24] studied the impact of different load-based degree methods on system performance, and proposed a scheduling model for cloud manufacturing. Simulation experiments showed that assigning higher priority to tasks with heavy loads could improve service utilization. Teng et al. [25] transformed the scheduling problem between multiple servers under MEC into a noncooperative game, and proposed the solutions of a distributed scheme and a centralized scheme, respectively, showing good performance in multiple indicators.

Most of the above-mentioned studies considered scenarios of task scheduling in multiple environments, and they all raised corresponding models. However, according to real application scenarios, we additionally take the dual-channel conception into account in this article, which considers the heterogeneity of resources.

B. Objective Optimization

Wang et al. [26] proposed a HetMEC structure, a heterogeneous multilayer structure for processing. The solution proposed on this basis had a good representation in reducing task delay. Zhang et al. [27] considered a combination scheme of scheduling and containerization. The proposed scheme enormously boosted the efficiency of program execution. Yu [28] developed a credible and sensitive algorithm in the cloud. The proposed algorithm vastly protected user data and improved user satisfaction. Lee et al. [29] proposed a mobility management scheme based on MEC, significantly reducing latency, so users could use the service normally even on the move frequently. Zhu et al. [30] considered the problem of vehicle computing unloading under resource constraints, and proposed a multiagent DRL scheduling scheme, which has a good optimization effect on task processing delay.

The above-mentioned approaches have made obvious progress in their target. However, most of them only considered a single goal. When compared with the complexity of reality, there is often more than one goal, and the goals generally contradict each other. Thus, in this article, we consider a scheduling situation with dual goals, response time, and SLB, to get closer to real-world scenarios.

C. Weighting Method

Xu et al. [31] designed a new architecture in the cloud. The architecture considered the situation of fault task recovery. On this basis, they proposed a dynamic scheduling algorithm under this architecture. Tong et al. [32] designed a deep *Q*-learning task scheduling (DQTS) method to deal with the DAG task to

TABLE I
TYPICAL STUDY ABOUT TASK SCHEDULING STRATEGIES

Author	Architecture	Objectives	Methods	Weighed approach
Chen *et al.* [21]	CC	Resource utilization	Heuristic algorithm	Subjective, fixed
Mohamed *et al.* [22]	FC	Response time, energy, cost, flow time, emission rate	Heuristic algorithm	Subjective, fixed
Mohanraj *et al.* [23]	Single cloud, multi-loud	Response time, resource utilization, delay, throughput, waiting time, efficiency	Swarm optimization	None
Liu *et al.* [24]	Cloud manufacturing	Response time, cost, quality	Workload-based heuristic algorithm	None
Teng *et al.* [25]	MEC	Task completion rate, system computation amount, profit, cost, process time, execution time, wait time	Non-cooperative game	None
Wang *et al.* [26]	MEC	Response time	Latency minimization algorithm	None
Zhang *et al.* [27]	MEC	CPU efficiency	Containerization algorithm	None
Yu *et al.* [28]	CC	Throughput, service	Fuzzy clustering algorithm	None
Zhu *et al.* [30]	MEC	Processing delay	DRL	None
Xu *et al.* [31]	CC	Response time, SLB	Genetic algorithm	None
Chen *et al.* [33]	CC	Response time, dismissing rate	DRL	None
Wang *et al.* [35]	MEC	Transmission delay	Greedy algorithm	None
Wang *et al.* [36]	Industrial Internet of Things	Power	RL	None
Min *et al.* [37]	MEC	Energy	Mathematics	None
This work	MEC with dual channel	Response time, SLB	DRL	Objective, dynamic

optimize makespan and SLB. The experimental results show a great performance of the proposed algorithm compared with a few algorithms in WorkflowSim. Chen et al. [33] proposed an adaptive allocation method, aiming at enhancing energy efficiency, reducing latency and dismissing rate, and it was proved through experiments that good results had been obtained. Zhou et al. [34] focused on saving energy and improving security quality under the constraint of time, applying a mixed-integer linear programming approach.

Most of the above-mentioned studies considered the multi-goals in the scheduling process. However, they did not consider joint optimization between goals or did not combine multiple goals in an appropriate and objective weighted manner through verification. In this article, we seriously consider the combination between the goals and the way the weights are given and, thus, select the best method through a comparison of a variety of objective weighting methods.

D. Optimization Methods

Wang et al. [35] considered the problem of the long delay of the cache server while ensuring the security of the data cache in the MEC environment. Based on the greedy algorithm, a secure decentralized data caching strategy was proposed, which could effectively improve the cache hit rate and reduce latency. Wang et al. [36] proposed a distributed Q-learning-assisted power allocation algorithm for two-layer heterogeneous networks, and studied fixed or variable learning rates and different kinds of multiagent cooperation modes. The simulation results verified the superiority of the proposed algorithm. Sheng et al. [37] proposed a solution that considered the coupling relationship between tasks and the allocation of computing resources. The simulation results showed an obvious improvement in saving energy of 73.8% compared with traditional schemes.

Most of the above-mentioned studies considered solving scheduling problems in different scenarios through heuristic methods or optimization theories. However, for complex large-scale scenarios, the time cost of solving the problem cannot be ignored. In this article, we consider methods incorporating neural networks to enhance the adaptability of the algorithm to large-scale scenarios. At the same time, the algorithm solution time is not significantly improved.

III. OVERVIEW OF SYSTEM ARCHITECTURE

In this section, we will describe the particulars of this research problem. After that, an overall architecture is given for this specific problem, and we describe several submodels around the problem. A summary of the main symbols involved is as shown in Table II.

A. System Model

To more clearly describe the process from task submission to finish, an offloading architecture in the IoE environment (OIOE) is proposed in this article, as shown in Fig. 1.

TABLE II
D2OP Parameter Settings

Symbol	Definition
I_{vio}	the indicator to indicate the default status of the task
lei_j^{fiber}	the available time of channel j
L_j^{last}, L_j^{cur}	the load status of machine j at the previous and the current
O_i^{arr}, O_i^{ddl}	the arrival time at the cloud and the default time of task i
O_i^{est}, O_i^{eft}	the earliest start processing time and the earliest finish time of task i
O_j^{idle}, O_{sta}	the available time of machine j and the computing nodes boot time
S_i	the task length of task i
A_j^{mach}	the processing rate of machine j
A_{slo}^{mach}	the processing rate of the slowest machine
A_{wire}	the wireless communication rate
T_i^{back}, T_{endu}	the result return time of task i and the user maximum endurance time
T_i^{res}, T_i^{trans}	the total response time and the total communication time of task i
T_{wait}	the time to wait for channel to be available

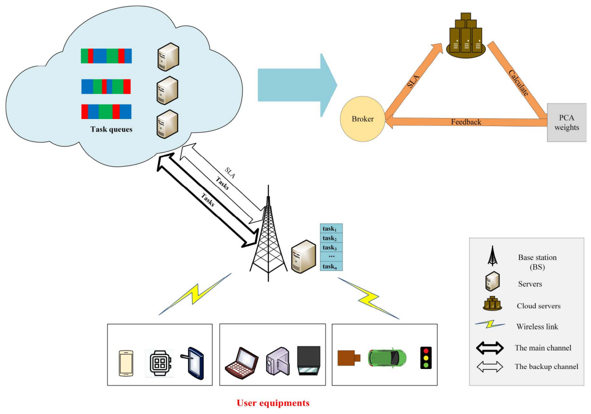


Fig. 1. OIOE architecture of the system.

While a task is generated from the client side, it is transmitted to the base station (BS) through wireless transmission. The BS collects all tasks from the user and sends them to the MEC through two fiber channels. This proposal, which can provide a much better performance, builds multiple paths via different interfaces and channels from the source to the destination and builds multiple backup paths. Due to the limitation of transmission resources, there is a process of waiting in line. In the MEC, a task is executed on a specified machine depending on a certain strategy. After executing, feedback based on two objectives is returned to the broker, and the result is returned to the client.

B. Submodels

1) Transmission Model: The transmission process is divided into two stages. In the first stage, the BS collects tasks sent by users, applying the frequency-division multiple access (FDMA) protocol. In the second stage, the BS sends those tasks to the MEC, applying the time-division multiple access (TDMA) protocol. It is a wireless transmission from the user to the BS with a wireless transmission rate, defined as

$$A_{\text{wire}} = B\log_2\left(1+\frac{P_m y_{i,j}}{NB+h_i}\right) \tag{1}$$

$$y_{i,j} = \varphi_i \times g_0 \times \left(d_0/\text{dist}_{i,j}\right)^{\theta} \tag{2}$$

where B is the communication bandwidth. P_m is the emission power of the client. N is the noise power. yi, j is the channel gain, and it can be obtained through (2). h_i is the interference. d_0 is the reference distance. dist_j is the distance between BS and client j. g_0 is the path-loss constant. φ_i is the fading channel power gain. θ is the path-loss exponent.

The process of transmitting data from the BS to the MEC is transmitted through wired transmission and optical fiber. To be in line with reality, a dual-channel is adopted in this model, with a different configuration. Thus, considering these two transmission stages, the total transmission time of task i is defined as

$$T_i^{\text{trans}} = \frac{S_i}{A_{\text{wire}}} + \frac{S_i}{A_j^{\text{fiber}}} + T_{\text{wait}}. \tag{3}$$

S_i is the calculation size of task i. A_j^{fiber} is the communication rate of channel j, and the value of j is 0 or 1, which symbolizes two different rate channels. Communication resources are not endless. Therefore, when communication is tight, tasks need to be queued to wait for the release of communication resources. T_{wait} is defined as

$$T_{\text{wait}} = \max\left(0,\ \text{lei}_j^{\text{fiber}} - O_{\text{cur}}\right) \text{ if choose fiber}_j \tag{4}$$

where $\text{lei}_j^{\text{fiber}}$ is used to record the available time of channel j, and O_{cur} is the current time. If $\text{lei}_j^{\text{fiber}}$ is earlier than O_{cur}, it means the task could be transmitted directly without waiting. Otherwise, the task should wait for transmitting until the channel is free. The main service channel is preferred under the same conditions.

2) Response Time Model: One of the objectives in this model is T^{res}, which relates to user satisfaction toward provides. T^{res} is the time from task submission to the result returning to the user, containing data transmission time, waiting time in line, and calculation time on computing nodes. The time when the task arrives at the MEC after the transmission is defined as

$$O_i^{\text{arr}} = O_i^{\text{sub}} + T_i^{\text{trans}} \tag{5}$$

where O_i^{sub} is the submission time of user equipment (UE) i.

However, even if the task arrives at the MEC, it may not be able to start execution immediately. Because of limited computing resources, the current state of the machine is also considered. The task can only be executed if the machine is idle. The available time of the machine j is defined as

$$O_j^{\text{idle}} = \begin{cases} O_{\text{sta}}, & \text{the first task in this } VM \\ O_j^{\text{last}}, & \text{otherwise} \end{cases} \tag{6}$$

where the O_{sta} is the beginning time of machine operation, and that of all machines are the same. The O_j^{last} is the finish time of the last task executed on machine j.

Considering two aspects, the task arrival time and machine state, the earliest start time of task i on given machine j is defined as

$$O_i^{\text{est}} = \begin{cases} O_i^{\text{arr}}, & O_i^{\text{arr}} \geq O_j^{\text{idle}} \\ O_j^{\text{idle}}, & \text{otherwise.} \end{cases} \tag{7}$$

Thus, the earliest finish time of a task is the time to return the calculation result to the client, which can be defined as

$$O_i^{\text{eft}} = O_i^{\text{est}} + \frac{S_i}{A_j^{\text{mach}}} + T_i^{\text{back}} \tag{8}$$

where A_j^{mach} is the calculation rate of machine j. T_i^{back} is the time to return the result to the user. Because the result size is generally small compared with the original, it is neglected in this article.

To provide a better experience to users, an SLA constraint is considered. All tasks that violate this constraint are observed as failures. Therefore, a variable that limits task completion deadline is adopted in this constraint, defined as

$$O_i^{\text{ddl}} = O_i^{\text{arr}} + \frac{S_i}{A_{\text{slo}}^{\text{mach}}} + T_{\text{endu}} \tag{9}$$

where the $A_{\text{slo}}^{\text{mach}}$ is the rate of the worst performing machine of all machines. T_{endu} is a value of the longest waiting time that the user could tolerate. We judge whether the task is successful or not through the index I_{vio}, defined as

$$I_{\text{vio}} = \begin{cases} \text{True}, & \text{if } O_i^{\text{eft}} > O_i^{\text{ddl}} \\ \text{False}, & \text{otherwise.} \end{cases} \tag{10}$$

Therefore, based on the above, the response time of task i is defined as

$$T_i^{\text{res}} = O_i^{\text{eft}} - O_i^{\text{sub}}. \tag{11}$$

3) Load Balancing Model: Another goal of this article is to balance machine load as much as possible in a heterogeneous environment. For users, fast response and high service quality are what they care about most. However, for service providers, an ideal state is that the load on each machine is not much different, which means full use of resources. So trying to achieve the state of SLB is meaningful. Load is an index revealing the overall working state of every machine, and the load of machine j is defined as

$$L_j^{\text{cur}} = L_j^{\text{last}} + \frac{S_i}{A_j^{\text{mach}}} \tag{12}$$

where L_j^{last} is the load of machine j at the previous moment. Therefore, the total load of a machine can be obtained as

$$L_j^{\text{tot}} = \sum_{i=1}^{n} \frac{S_i}{A_j^{\text{mach}}} \tag{13}$$

where n is the number of tasks that are executed on this computing node.

4) Feedback Designing Model: Every time a task is executed, feedback is given by the environment after execution. The feedback expresses the level of intelligence of the current decision. The larger the feedback value is, the more favorable we consider the current decision to be. In this way, the system could guide the broker to make strategies. The feedback R is a weighted sum of our two goals, defined as

$$R = w_1 \times \text{norRes}_i + w_2 \times \text{norLoad}_i \tag{14}$$

where w_1 and w_2 are the weights of the task delay and load, respectively. norRes_i and norLoad_i are defined as

$$\text{norRes}_i = \frac{T_i^{\text{res}} - T_{\text{min}}^{\text{res}}}{T_{\text{max}}^{\text{res}} - T_{\text{min}}^{\text{res}}} \tag{15}$$

$$\text{norLoad}_i = \frac{L_i - L_{\text{min}}}{L_{\text{max}} - L_{\text{min}}}. \tag{16}$$

Equations (15) and (16) perform standardized operations for two goals. T^{res} and L^{tot} are two indicators with different dimensions. Thus, to avoid the result being affected by the difference in their value ranges, these two values need to be adjusted to a uniform magnitude.

C. Problem Description

In the IoE environment, terminal devices, such as mobile phones, laptops, printers, traffic lights, vehicles with autonomous driving technology, etc., can all be regarded as users. Those devices that generate millions of requests do not have enough ability to process them. Thus, these requests would be transferred to the MEC for processing, where a large number of computing resources are gathered. However, no matter how enormous the resources there are, a reasonable resource allocation is still needed to make full use of them and, thus, satisfy users' high demands. At a certain timestamp t, a series of requests $J = (\text{task}_1, \text{task}_2, \ldots, \text{task}_n)$ will be emitted as an input. Each task contains specific attributes, which can be described as $task_i = (ueId, taskId, O_i^{sub}, S_i, reqMem_i, reqCpu_i, upSize_i)$. A series of offloading policies are output to simultaneously optimize task response time and SLB.

A simple case to interpret the importance of scheduling policy is given below. Suppose that there are only two computing nodes for processing, and multiple tasks arrive at different time slots. Each task is executed in the form of an exclusive task node during processing. We define each node computing resource as 10 units, and that of required resources of T_i, T_{i+1}, T_{i+2}, and T_{i+3} as 20, 30, 10, and 12 units, which means their processing durations are 2, 3, 1, and 1.2 s, respectively. As shown in Fig. 2(a), when tasks are not crowded, the computing resource is sufficient to process. In this case, each task can be handled directly when it arrives, so which is no need to wait. However, as the Internet develops by leaps and bounds, task jams have become the norm. During peak periods, an outstanding scheduling strategy is of paramount significance. Based on the above, a new task arrives at 6 s. There are two possible actions to deal with this task, yet they lead to completely different results, as shown in Fig. 2(b) and (c), respectively. The policy in Fig. 2(b) only needs 1.5 s, and the whole schedule

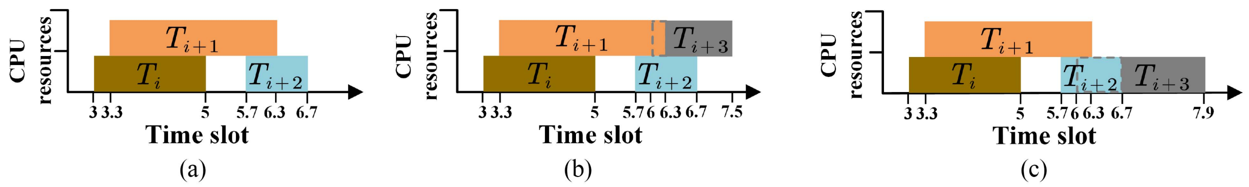


Fig. 2. Impact of two different scheduling strategies on performance. (a) Current workflow schedule. (b) Scheduling strategy. (c) Another scheduling strategy.

is finished at 7.5 s. However, the policy in Fig. 2(c) needs to wait for 0.7 s, and finish at 7.9 s. Compared with the former, T^{res} of T_{i+3} increases by 26.67%. Therefore, on the premise of identical circumstances, varying scheduling policies could also produce a significant impact on performance. To shorten the response time, enhance users' experience, and reach SLB as much as possible, a feasible solution is considered to optimize them simultaneously in this article. The objective function can be expressed as

$$\underset{a(t)}{\text{Min}}\ T_i^{\text{res}},\ L^{\text{tot}} \quad \forall i \in \{1, 2, \ldots, n\} \tag{17}$$

$$\text{s.t.}\ \ h \le \delta \tag{18}$$

$$\sum_{i=1}^{n} \text{mem}_i \le M \tag{19}$$

$$\sum_{i=1}^{n} \text{cpu}_i \le C. \tag{20}$$

n is the number of tasks. h is the number of parallel tasks, and δ is the number of parallel tasks that can be accepted. mem_i and M are the memory required to perform the task and the memory capacity of the hardware, respectively. cpu_i is the required CPU capacity, and C is the CPU capacity of the remote server.

D. Problem Complexity Analysis

The necessity of an appropriate scheduling scheme has been elaborated on above. This section demonstrates that the research problem is an NP-hard problem.

The problem studied in this article is to obtain an appropriate scheduling policy, making full use of computing nodes in a heterogeneous IoE environment. The task can be expressed as $t_i = (cr_j, mr_i)$, representing the requirement for CPU and memory resources. The computing node could be expressed as $cn_j = (c_j, m_j)$, representing the CPU and memory resources inherent in itself. The prerequisite for the successful execution of the task is that the assigned node must meet the requirements $c_j \ge cr_i$ and $m_j \ge mr_i$.

The knapsack problem refers to putting a bunch of objects O_i into several containers T_j through different combinations so that the remaining space of the container is as small as possible. The collection of objects $O = \{O_i\}$ is used as input, and the placement strategy of objects and containers is used as output. Each object could be expressed as $O_i = (ar_i, hr_i)$, representing the requirement for the cross-sectional area and height of O_i. The container can be expressed as $T_j = (a_j, h_j)$, representing the cross-sectional area and height inherent in itself. The object O_i can be placed in container T_j and must satisfy $a_j \ge ar_i$ and $h_j \ge hr_i$.

Task t_i can be viewed as object O_i, and computing node cn_j can be viewed as container T_j. Thus, the problem of finding a suboptimal scheduling policy can be transformed to pack these objects into distinct containers. The target of making full use of computing resources can be viewed as minimizing the remaining space in the container, that is, using as few containers as possible under the premise that all objects are packed into the container. Assuming the computer hardware configurations are the same, then the containers are all the same size and, thus, the above problem can be reduced to the knapsack problem. The knapsack problem is a known NP-hard problem. Thus far, the original problem is an NP-hard problem.

IV. Proposed D2OP Algorithm

An approach that combines DQN and PCA is proposed to provide a solution for the scheduling problem in a heterogeneous environment. In this section, first, we introduce the underlying theory of the algorithm. Second, we introduce the framework basis of the algorithm. Finally, the whole workflow of the proposed algorithm is provided.

A. Machine Learning

The ML algorithm is divided into three categories: 1) supervised learning (SL); 2) unsupervised learning (USL); and 3) reinforcement learning (RL). SL learns from a large number of labeled datasets. USL mines effective features in unlabeled datasets, and is mainly used for data preprocessing such as denoising and feature extraction. RL is a way of learning while interacting with the environment.

Q-learning (QL) is a classical RL approach [38], [39]. The core idea of QL is to construct a 2-D table and record all possibilities in an exhaustive way. The *Q*-table is combined with $m \times n$, where m is the number of possible states, and n is the number of optional actions. The *Q*-value is the expected future reward that can be obtained when each action is taken in each state. The *Q*-value transition equation of state s for choosing action a is defined as

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\left(R + \gamma \max_{a'} Q_t(s',a') - Q_t(s,a)\right) \tag{21}$$

$$Q_t^{\text{targ}}(s,a) = R + \gamma \max_{a'} Q_t(s',a'). \tag{22}$$

R is the reward given from the environment to the broker. $R + \gamma \max_{a'} Q_t(s',a')$ is the actual *Q*-value as defined in (22), and $Q_t(s,a)$ is the evaluation value. α is the learning rate, which represents the learning step length of the network. γ is the discount factor indicating the importance of future reward.

However, a significant problem is that such an exhaustive approach can lead to memory overflow when the task volume is large enough and the number of possible states is excessively large.

To meet the needs of large-scale data, DQN introduces DL based on QL to replace the function of Q-table and directly obtains Q-value. This method avoids the shortcomings of the two. Only a single DL or RL has its deficiency.

1) DL belongs to SL, which means it needs a large number of labeled samples and is trained by judging the results with the labels. There are great limitations in applicable scenarios.
2) RL is a self-learning method, that can construct labels of the data through reward values. However, this method is not suitable for large-scale scenarios, owing to the termination of storage.
3) Tasks are regarded as independent of each other in DL, however, they are considered correlative in RL.

Even though combined with DL, DQN still retains the nature of general RL, learning adaptively through interacting with the environment. There are two networks in DQN, with consistent structure but inconsistent parameters. One is the current network with the newest parameters, used to get the estimated value. Another is the target network with older parameters, used to predict the real. In the beginning, there are no samples to learn from, so there is a process of accumulating samples. The environment enters the current state s as the input of the current network *EN*. *EN* would output the Q-value corresponding to each action, marked as $Q(s, a)$. The broker will adopt the action a corresponding to the largest Q-value, and the environment changes from s to s'. Afterward, a certain amount of samples is extracted from the pool D randomly. In each sample, s enters *EN*, and s' is distributed to the target network *TN* as input, and the evaluation value Q_{eval} and the actual value Q_{targ} are gotten. The parameters of *EN* update through the deviation between the two. For every certain number of iterations, parameters of *TN* would be updated by copying the current network's. The above two stages will alternate continuously until reaching preset training times. The network cannot be thought of as reaching convergence unless the discrepancy between the target and evaluation becomes quite small.

Combining the strong point of the two and making improvements on this basis, DQN has three notable characteristics.

1) Using the value function approximation method to replace the Q-table not only avoids the problem of dimension explosion but also reduces the cost of searching for the Q-table.
2) To break the correlation between samples in a continuous period, an experience replay approach is introduced. Every once in a while, the network will randomly distill a certain number of samples from the pool.
3) The double Q-network mechanism avoids the interference caused by the correlation between Q_{targ} and Q_{eval} and improves the stability of the network.

The pseudo code for DQN can be described in Algorithm 1. The time complexity is $O(PT)$, where P are the number of episodes, and T is the number of timestamps.

Algorithm 1: DQN Algorithm

```
Input: Mobile user equipment
Output: Offloading strategy
Initial the environment and parameters;
Initialize the experience replay pool $D$ and the two
  neural networks ;
for preset number of iterations $P$ do
    Obtain the initial state $s_1$ from environment;
    for $t = 1, T$ do
        Get action $a$ through $\epsilon$-greedy approach;
        Put $(s, a)$ into the evaluation network EN, getting
          $Q(s, a)$ and $R$;
        Transfer environment from state $s$ to $s'$;
        Add $(s, a, R, s')$ into $D$;
        Choose a sample in $D$ randomly, and put it into
          EN and the target network TN;
        Get $Q_{eval}$ and $Q_{targ}$ by Eq. (22);
        Calculate the divergence between $Q_{eval}$ and $Q_{targ}$
          through loss function;
        Update the parameters of EN;
        Copy the parameters of EN to TN every $k$ steps;
    end
end
```

B. MDP

A change in one state is potentially linked to an infinite number of previous states of the environment. It is difficult to model the problem due to its complexity. Therefore, it is necessary to simplify the problem model and assume that the state transition has Markov properties. The Markov property means that the current state is only related to the previous state, but has nothing to do with any other state, defined as

$$P_{ss'}^{a} = E\big(S_{t+1} = s' | S_t = s, A_t = a\big). \tag{23}$$

MDP is a process with Markov properties.

MDP is an RL frame, as well as the cornerstone of all theories of RL. The Markov reward process (MRP) is an MDP with a reward, which can be represented by a five-tuple, $\langle S, A, P, R, \gamma\rangle$. S is the collection of all possible states, represented through the state of every computing node in this article. A is the collection of all actions, that is, which node to be chosen to work. P is the probability of choosing action a under state s. R is the feedback given by the environment to the current decision. The sum of all future rewards starting from time t R_t is defined as

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \Sigma_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{24}$$

where γ is the discount factor, which reflects the extent of emphasis on future reward.

To represent the long-term value of a certain action, the behavior value function of the policy π is defined as

$$\begin{aligned} V^{\pi}(s) &= E^{\pi}[G_t | S_t = s, A_t = a] \\ &= E^{\pi}\Big[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s, A_t = a\Big] \\ &= E^{\pi}\big[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \cdots) | S_t = s, A_t = a\big] \end{aligned}$$

$$= E^{\pi}\left[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a\right]$$
$$= E^{\pi}\left[R_{t+1} + \gamma V^{\pi}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a\right]. \quad (25)$$

In this way, the optimal policy π^* is found by finding the largest action-value function, defined as

$$Q^*(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} Q^*(s', a'). \quad (26)$$

The solution to the problem is obtained by solving the equation using the RL method. Therefore, the model complies with the MDP.

C. Principal Component Analysis

According to (14), a method is needed for weighting two goals. There are two types of multivariate statistical weighting methods: 1) subjective weighting and 2) objective weighting. However, to ensure the fairness and reasonableness of the reward, an objective weighting method, PCA, is adopted. PCA [40], [41] is a data analysis method that can extract the main feature components of data. In real applications, a problem is always accompanied by lots of related variable factors, and these variables all reflect different degrees of information. Our mission is to give these two goals an equitable weight to better guide the broker's decision making.

Treat the whole sample data as a 2-D matrix. Each row is a sample record, and each column is a collection of the values to an indicator of all samples. The idea of the PCA algorithm is as follows.

1) Eliminate the dimension of the data because the two objectives are different physical quantities, so their units and value ranges are inconsistent. The data standardization method uses min-max normalization, defined as

$$x_{i,j} = \frac{x_{i,j} - \min(x_{1,j}, \ldots, x_{p,j})}{\max(x_{1,j}, \ldots, x_{p,j}) - \min(x_{1,j}, \ldots, x_{p,j})} \quad (27)$$
$$\text{Ind}_j = \{x_{i,j}\}, i \in [0, n] \quad (28)$$

where $x_{i,j}$ represents the value of the ith sample corresponding to the jth index. Ind_j is the collection of the value of all samples on indicator j.

2) Construct a covariance matrix C, which is used to express the correlation between indicators, as

$$C = \begin{bmatrix} \text{cov}(\text{Ind}_1, \text{Ind}_1) & \cdots & \text{cov}(\text{Ind}_1, \text{Ind}_j) \\ \text{cov}(\text{Ind}_2, \text{Ind}_1) & \cdots & \text{cov}(\text{Ind}_j, \text{Ind}_2) \\ \vdots & \ddots & \vdots \\ \text{cov}(\text{Ind}_j, \text{Ind}_1) & \cdots & \text{cov}(\text{Ind}_j, \text{Ind}_j) \end{bmatrix}. \quad (29)$$

The nonoverlapping information between the indicators appears as their covariance is 0.

3) Calculate the eigenvalue $\text{value}_{\text{eig}}$ and eigenvector $\text{vector}_{\text{eig}}$ of C by setting $\det(C) = 0$. $\text{value}_{\text{eig}}$ represents the amount of information of corresponding $\text{vector}_{\text{eig}}$. Actually, when $i = j$, the value of $\text{cov}(\text{Ind}_i, \text{Ind}_j)$ equals to that of variance.

4) Sort the vectors from high to low variance and select k indicators that you need. In this way, the effect of dimensionality reduction can be achieved.

Algorithm 2: PCA Algorithm

```
Input: Original samples data
Output: Compressed data
for Each indicator do
    Standardize all samples according to Eq. (27);
end
Construct the covariance matrix $C$ through Eq. (29);
Obtain the $value_{eig}$ and $vector_{eig}$ by setting $\det(C) = 0$;
for Each pair of $value_{eig}$, $vector_{eig}$ do
    Sort $value_{eig}$, $vector_{eig}$ in descending order according
      to $value_{eig}$;
end
Select the top $k$ $vector_{eig}$ as the mapping relationship;
Multiply the original sample matrix with the projection
  matrix;
```

5) Map sample points to selected feature vectors, so that the sample point could be expressed linearly through new vectors. In this way, the process of converting from the original indicator to the new indicator is completed.

In this article, we aim to obtain a suitable weight for two objectives. The variance represents the information, so we think the bigger the value of the variance is, the more the information includes. Therefore, process the data in Ind_j by the way of (30), and the weight of indicators could be obtained, satisfying (31)

$$w_j = \frac{\text{value}_j^{\text{eigen}}}{\sum_{i=1}^{J} \text{value}_i^{\text{eigen}}} \quad (30)$$
$$\sum_{j=1}^{J} w_j = 1. \quad (31)$$

The general flow of PCA could be described in Algorithm 2. The computational complexity of the covariance matrix is $O(m^2 n)$, and that of eigenvalue decomposition is $O(m^3)$. Thus, the complexity of the whole workflow is $O(m^3 + m^2 n)$, where m is the number of indicators, and n is the number of samples.

Many deep learning algorithms have several parts in common: 1) determine the action of this episode according to a certain strategy; 2) execute actions to obtain immediate feedback R and the next state of the environment; and 3) update network model parameters. The PCA method is applied to dynamically assign weights to objectives to calculate network feedback. Therefore, PCA has a certain degree of universality for deep learning algorithms.

D. D2OP Algorithm

In this article, a novel algorithm D2OP is developed to give a strategy on an offloading scheme under the OIOE model, devoted to reducing T^{res} and L^{tot} through a better allocation strategy. The introduction of DQN solves the problem of Q-table storage explosion, on the premise of retaining the RL adaptive learning. PCA stands out among multiple objective weight distribution comparison methods to ensure the weight

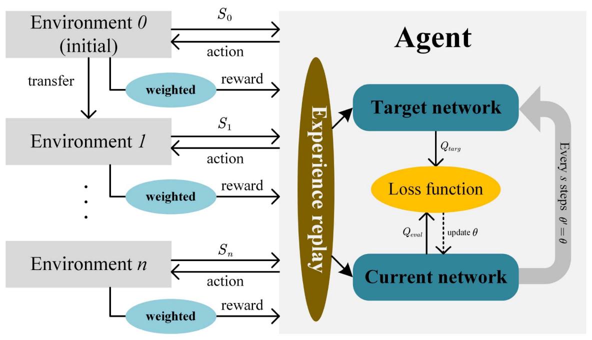


Fig. 3. Decision flow of the D2OP.

of targets. The dual-channel design and its selection strategy are based on the actual scenarios.

A schematic diagram of the D2OP decision-making process is shown in Fig. 3. The broker obtains a state message from the environment, and the current network would output an action that makes the Q-value maximum. The environment transfers to the next state after implementing the given action in the current state. For the change of state in a round, the broker will evaluate the quality of this transition on given objectives. The respective values of load and response time would be weighted through the PCA method and, thus, a global reward is returned to the broker to update the two networks. This transition will be put into the experience pool. Then, the deviation between the output Q-values of the two networks, called loss, is used to update the parameters of the current network. At every certain step, the parameters would be copied from the current network to the target one. Repeat the above steps until the loss is small enough and we think the network has already trained better.

1) Communication: On the facet of communication, a dual channel is designed during the transmission between BS and cloud to reflect the realistic scenario. This process is a wired transmission. Generally, the channel speed is not exactly the same in reality, so two channels with inconsistent communication speeds are applied in this architecture.

In actual applications, high-quality resources are generally called first under the same conditions. Moreover, trying to reduce task delay is one of the targets, which includes transmission time, according to (3), (5), and (11). Based on this point, a similar design idea is adopted to ensure the quality of service. When communication resources are abundant, the main service channel is in priority. Otherwise, the status of both two channels will be considered comprehensively. While communication is tight if the main channel will end the busy state before the other, choose the main channel, else, the backup channel. The pseudo code of this process could be expressed in Algorithm 3, and the time complexity is $O(n)$.

2) Workflow: The entire algorithm flow is as follows. Tasks are generated on the UE side, and there are many types of terminal devices, not just limited to smartphones. Next is a data transmission process. Users send their task requests to the BS side through wireless connections. The role of the BS side is to collect tasks sent by users and send them to the cloud through wired transmission. Here, is a process of queuing due to limited resources. After the data arrives in the cloud, the

Algorithm 3: Channel Selection Method

```
Input: Channel status, tasks
Output: Channel selection strategy
for Every time a task arrives do
    Calculate the time for the task to reach BS;
    Assess the current status of the two channels by
      Eq. (4);
    if Communication resources are in idle then
        Choose the faster channel $fiber_1$;
    else
        if $fiber_1$ ends current task sequence earlier than
          $fiber_0$ then
            Choose $fiber_1$;
        else
            Choose $fiber_0$;
    Calculate $T_i^{trans}$ by Eq. (3);
    Update selected channel available time;
end
```

Algorithm 4: D2OP Algorithm

```
Input: UEs, system parameters
Output: Offloading strategies
for Each task generated by UE do
    Calculate $T_i^{trans}$ through Alg. 3;
    Task $i$ arrives the cloud side;
    Pair the task $i$ with the machine $j$ selected by Alg. 1;
    if Machine $j$ is in busy then
        Wait until machine $j$ finishes the previous tasks;
    Put task $i$ on machine $j$ for executing;
    Observe the environment and return a reward to the
      broker, calculated by Alg. 2;
    Update network parameters;
end
```

broker makes decisions based on the overall situation of the system, choosing the allocated machine. The result of task execution will be back to the client. For each allocation, the impact of this decision on the two goals would be observed, respectively, by the environment. And these two values would be weighted through the PCA method as a total reward to guide the broker's decision making. The entire process could be described as Algorithm 4, and the time complexity is $O(m^2 n(m+n))$.

V. Simulation Experiments and Discussion

In this section, an introduction of the total simulation environment is given first, including some hardware configuration information and related parameters. Then, to prove the performance of the proposed D2OP algorithm, a series of experiments are listed and compared with several comparison algorithms. For every experimental result figure, its description and analysis are given below. The comparison algorithms consist of QL, Sarsa, and WRR. The first two belong to RL, which adaptively learns from the external environment. To ensure the fairness of the comparison results, the weight values of these

TABLE III
D2OP PARAMETER SETTINGS

Symbol	Definition	Value
B	the bandwidth between UE and BS	25MHz [42]
N	the noise power	-174dBm/Hz [42]
P_m	the transition power of user equipment	200mW [42]
dist	the distance from UE to BS	randint(50, 200)
d_0	the reference distance	1m [42]
g_0	the path loss constant	-40dB [42]
A_0^{fiber}	the speed of the backup channel	150Mbps
A_1^{fiber}	the speed of the main channel	200Mbps
γ	the discount factor	0.9 [43]
α	the learning rate	1e-5 [43]
θ	the path loss exponent	4 [42]
φ_i	the fading channel power gain	exp(10) [42]
/	the weighting method	PCA

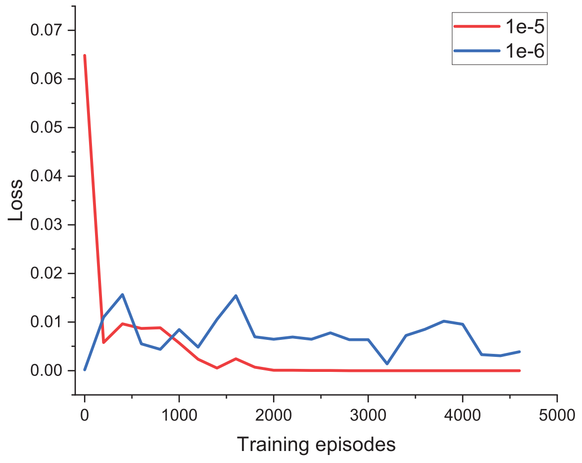


Fig. 4. Convergence for different learning rates.

two algorithms are obtained using consistent weight methods. WRR is a classical algorithm that allocates tasks to computing nodes with different weights.

A. Experimental Environment Setup

CloudSim is used as our simulation platform, using Python 3.6.4 as the programming language. The entire project runs under the operating system Windows 10 64 bit. The processor, CPU frequency, and running memory of the device are i5-8400 with six cores, 2.80 GHz, and 8 GB, respectively. To ensure the result's rationality, the overall parameters of D2OP are set uniformly, as listed in Table III.

B. Learning Rate Selection

Learning rate α is a critical parameter that could play an important role in the whole network's performance. Therefore, we need to find a suitable α value to make the network converges better with a smaller difference between the estimated value and the real value.

The network convergence effect of varying α is shown in Fig. 4. From the experimental result graph above, when α is

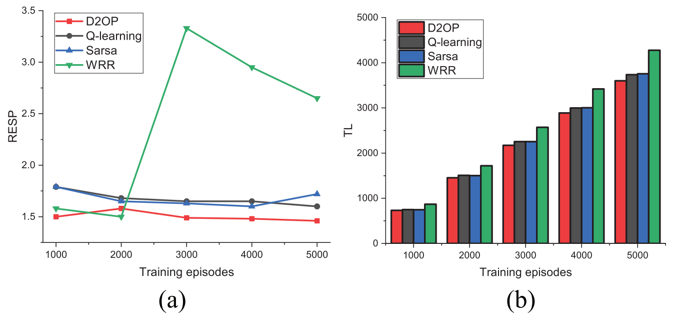


Fig. 5. Optimized performance for single-objective scenarios. (a) Optimization effect on RESP. (b) Optimization effect on TL.

set to 1e-6, it is obvious that the network does not tend to converge. The blue line in the figure oscillates repeatedly as the training episodes progress, while the red one achieves convergence at nearly 2000 episodes. Therefore, 1e-5 is chosen as the value of α. The reason is that if α is too small, the network can not reach converge in less time, and it may fall into a local optimum and miss the global optimum.

C. Results for Single Objective

In this section, the response time and SLB are measured among different methods separately and, thus, two metrics are advanced to judge the performance. RESP is defined as

$$\text{RESP} = \frac{1}{n}\sum_{i=1}^{n}\left(O_i^{\text{eft}} - O_i^{\text{sub}}\right) \tag{32}$$

judging for task average delay. From the time user submits the request until the result is fed back to the user, TL is the metric to observe the total load for all machines, defined as

$$\text{TL} = \sum_{i=1}^{m} L_i^{\text{tot}}. \tag{33}$$

We move closer to SLB by minimizing the total load for the reason that the settings of computing nodes are proximate and, thus, in these circumstances, we deem that the smaller the TL, the more balanced the system.

We take the training level as the horizontal axis and observe the changes in the two indicators. Fig. 5(a) shows the average response time among these four algorithms. The delay of WRR is the highest, and the best performer is the D2OP algorithm. The performance of QL and Sarsa is not much different. The trend of WRR is unstable, that of QL and Sarsa is maintained at a certain range, while that of D2OP is in reduction, which means that the performance of the proposed algorithm becomes better as training progresses.

From Fig. 5(b), we can see that as training advances, the load levels of all four algorithms have risen. The optimization effect of D2OP is always the best, followed by QL and Sarsa, and the worst is WRR.

Through this group of single-target control experiments, the optimization effect of the proposed D2OP algorithm is proved when there is only a single objective.

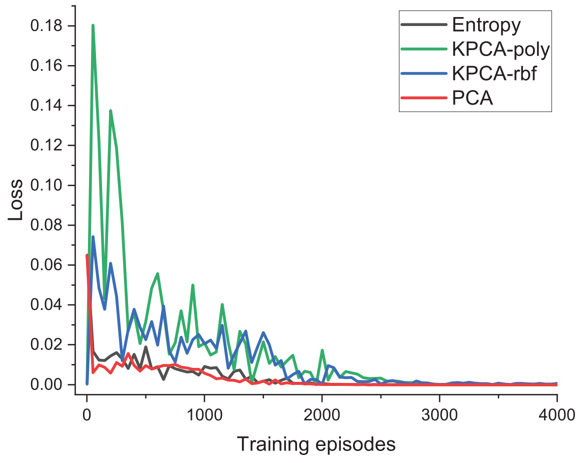


Fig. 6. Convergence for different weighting methods.

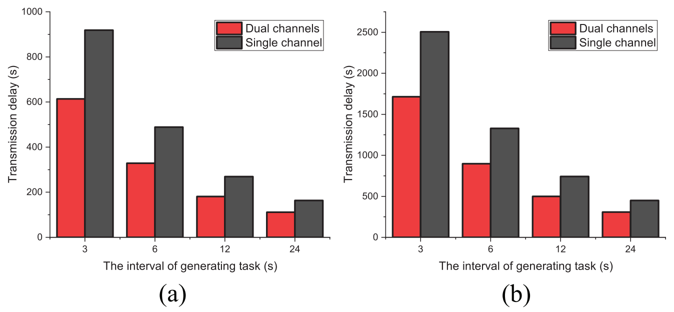


Fig. 7. Transmission delay of varying channel. (a) Small tasks. (b) Large tasks.

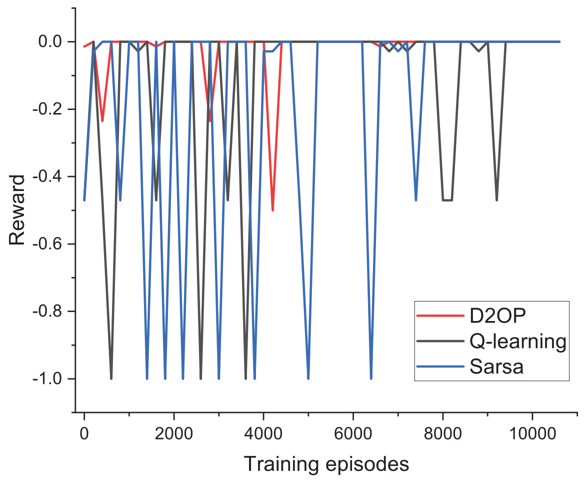


Fig. 8. Reward with different algorithms.

D. Results for Multi Objectives

1) Weighting Method Selection: In this article, a novel D2OP algorithm is proposed to minimize T^{res}, and reach SLB as possible. This is a two-objective optimization problem, so properly weighting these two goals is necessary. Those comparison weighting algorithms are PCA, Kernel PCA (KPCA), and Entropy, among which KPCA tests two kinds of kernel functions, the Gaussian kernel, and the polynomial kernel. PCA is a weighting approach that extracts information in a certain dimension to reach the purpose of dimensionality reduction. KPCA is a way of mapping data from low-dimensional to high-dimensional through kernel function transformation. Then extract information in higher dimensional space by using PCA, and finally map it back to lower dimensional space. Different kernel functions mean different mapping strategies. Entropy is a way of ensuring weight by the level of data confusion.

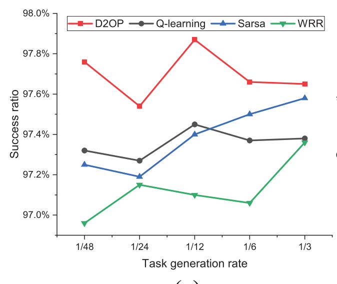
As shown in Fig. 6, all four weighting methods could converge as training progresses. The method with the best performance is PCA, so we apply it to this project. At the beginning of the training process, the PCA method starts from a lower loss value, with a faster convergence speed. The stability after convergence is also good, followed by the Entropy method. Last is KPCA with the Gaussian and polynomial kernel, with the slowest convergence speed.

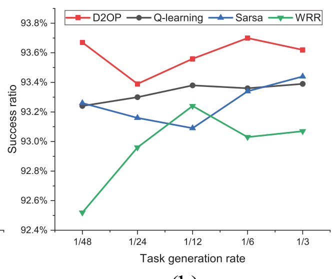
2) Channel Transmission Delay: Data transmission includes the process from UE to BS and from BS to cloud, so transmission delay equals the sum of wireless and wired transmission. To be in line with reality, the model proposed in Section III-A designs a dual-channel, one with a faster transmission rate, and one with a lower one. This design has the consistency of the backup scenario in many fields of life. This experiment is set to prove performance improvement at this point, in which the single channel represents the faster channel. The result is shown in Fig. 7.

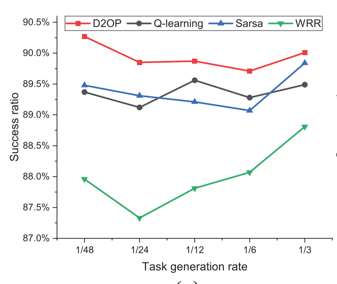
The interval of generating task is set as an independent variable, and the total transmission delay as a dependent variable. The bigger the interval of generating tasks, the smaller the number of generating tasks is and, thus, the smaller the total transmission delay will be. Fig. 7 shows the changes in the transmission delay of different task sizes and different channel designs under the change of the number of tasks. The S_i range of the small task is (100, 1000) kB, and that of the large task is (1000, 2000) kB. Whether it is a large task or a small task, the transmission delay will be prolonged as the number of tasks increases. However, compared with the single-channel, the dual-channel model greatly reduces the delay, and the more tasks, the greater the reduction. This also proves the advantages of the OIOP model and its stability.

3) Reward: Different approaches make different rewards. Three algorithms, D2OP, QL, and Sarsa are all RL algorithms, learning from the environment, and will return feedback toward every action. If the reward is higher, usually we regard the action as more beneficial.

According to (14), the reward is a weighted sum of normalized T_i^{res} and normalized L_i^{tot}, whose range is from -1 to 0. Only when the task violates will the reward value reaches -1. In Fig. 8, in the beginning, the reward is lower because of inadequate training. As the training progresses, the feedback of D2OP is getting better, and the curve is gradually stable only with less fluctuation. However, the feedback of QL and Sarsa is still continuously shocking with lower rewards. This means that the strategies made by the D2OP algorithm among the three are more likely to perform well in the future, and the performance is more stable than the other two. The result of D2OP is more superior to QL and Sarsa, which is due to


(a)


(b)


(c)

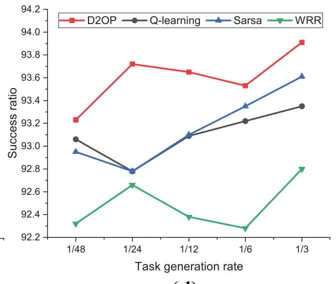

(d)

Fig. 9. Success ratio among different algorithms with changing task sizes. (a) Task size between 100 K and 1 M. (b) Task size between 1 M and 2 M. (c) Task size between 2 M and 3 M. (d) Task size between 100 K and 3 M.

the discrepancy in processing mechanism and, thus, we choose the D2OP to obtain a scheduling strategy.

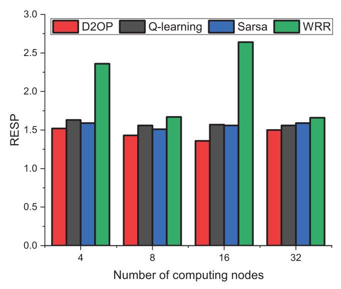
4) Success Ratio: The SLA constraint is set for ensuring users' QoE (response time and successful processing rate). Equation (8) defines the latest finished time, containing task transmission delay, time to wait for the computing node, and calculation time on the selected machine. If the response time exceeds this deadline, the current task will be regarded as a failure and will be given up. The metric to measure the situation of task execution is defined as

$$\text{success ratio} = \frac{\text{the amount of finished tasks}}{\text{the amount of all tasks}}. \quad (34)$$

The fewer abandoned tasks, the higher the ratio. We regard it to perform better.

Fig. 9 shows the task execution success ratio of four algorithms at different levels of task size. The horizontal axis represents the rate of task generation, which is proportional to the number of tasks. As the task generation rate increases, the results of WRR are the worst, whose curve is basically at the bottom, in repeated shocks. This is for the reason that the WRR is a classical weighted polling algorithm, having higher uncertainty, while the other three algorithms are RL algorithms. Relative to WRR, those three can remain stable within a certain range. As the task size increases, the success rate of all four algorithms has dropped, which is subject to the limited computing power of nodes. The trends of QL are keeping pace with Sarsa, and they are both below that of D2OP, for the reason that the frameworks of the two are roughly the same, only slightly different when updating the status. Under all circumstances, the D2OP performs the highest processing capacity.

5) Expanding Computing Nodes: In this article, we have proposed a D2OP algorithm to jointly optimize task response time and machine load. And this part is set to verify the optimized performance of the proposed method. Our objective is to make these two values of RESP and TL as low as possible at the same time. This experiment is conducted to observe D2OP's performance compared with QL, Sarsa, and WRR.


(a)

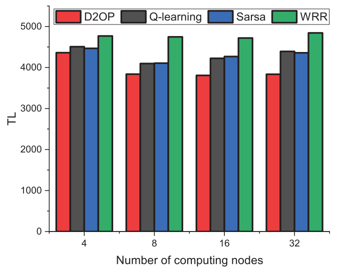

(b)

Fig. 10. Dual-objective performance verification under task node expansion. (a) Optimization effect on RESP. (b) Optimization effect on TL.

Fig. 10 shows the performance comparison of four algorithms on task delay and machine load, respectively. In Fig. 10(a), the value of RESP is reducing while the amounts of computing nodes increase from 4 to 8. However, if it continues to increase the number of computing nodes, RESP is slightly increasing. This is because the computing capability is not sufficient to handle so many tasks, so adding nodes save time waiting for nodes to be idle. While at this time, the nodes' capacity has reached saturation yet, even though continue to add nodes, the calculation time cannot be reduced. For the reason the system still needs to optimize the machine load simultaneously in a heterogeneous environment, partial tasks should be allocated to newly added slower nodes, leading to a growth in RESP. Yet, in all cases, D2OP has stronger performance than the other. In Fig. 10(b), the TL value of D2OP is also the slowest. This means in the case of optimizing RESP and TL at the same time, D2OP has the best optimization effect on both. The proposed algorithm has better adaptability in the case of computing node expansion.

In conclusion, in the node expansion experiment, D2OP performs better ability on jointly optimizing both task delay and balancing machine load than other comparison algorithms, with great robustness.

6) Expanding Task Density: This section is set to test the performance of the algorithm when the number of tasks increases as the user scale increases. The evaluation indicators are two optimization objectives, RESP and TL.

In Fig. 11, it can be seen that with the expansion of the task scale, the RESP and TL of the four algorithms are extended. This is the reason that although the scale of tasks is continuously expanding, the computing resources are limited. From the trends in the figure, the proposed D2OP algorithm has the best optimization performance for both RESP and TL, followed by QL and Sarsa. The worst performer is WRR. This is because WRR can not adjust the strategy according to the environment in time, which lacks adaptability. However, D2OP can better adapt to large-scale task scenarios due to the application of neural networks.

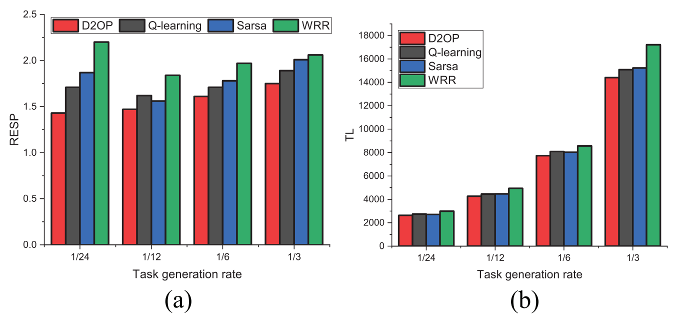


Fig. 11. Dual-objective performance verification under task density expansion. (a) Optimization effect on RESP. (b) Optimization effect on TL.

VI. Conclusion

In this article, we designed a scheduling model in the IoE environment. The scheduling problem in such a scenario was modeled as MDP, and a corresponding solution based on the DRL and PCA method was proposed. When jointly optimizing task response time and SLB, we carefully considered the measurement of the proportion between the two goals. Through comparison experiments with multiple objective weighting algorithms, we chose the most suitable weighting algorithm. For the proposed D2OP algorithm, we first elaborated on its theoretical basis in detail. Second, an exhaustive process and its pseudo code were spread out. Third, using multiple ML methods as comparison methods, multiple experiments are used to verify the performance of the proposed algorithm, and it shows a better performance in promoting users' QoE and SLB, with good robustness.

With the proliferation of the Internet, people have triggered increasing attention to privacy protection. Hence, in future work, we will focus on the data security issues in the scheduling process, making the model more perfect and in line with reality.

Acknowledgment

The authors thank the editor and reviewers for their insightful comments and valuable suggestions.

References

[1] W. Ji, B. Liang, Y. Wang, R. Qiu, and Z. Yang, "Crowd V-IoE: Visual Internet of everything architecture in AI-driven fog computing," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 51–57, Apr. 2020.

[2] W. Chen, Y. Yaguchi, K. Naruse, Y. Watanobe, and K. Nakamura, "QoS-aware robotic streaming workflow allocation in cloud robotics systems," *IEEE Trans. Services Comput.*, vol. 14, no. 2, pp. 544–558, Mar./Apr. 2021.

[3] R.-G. Stan, L. Băjenaru, C. Negru, and F. Pop, "Evaluation of task scheduling algorithms in heterogeneous computing environments," *Sensors*, vol. 21, no. 17, p. 5906, 2021.

[4] Z. Fu, Z. Tang, L. Yang, and C. Liu, "An optimal locality-aware task scheduling algorithm based on bipartite graph modelling for spark applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 10, pp. 2406–2420, Oct. 2020.

[5] J. Xu, Z. Zhang, Z. Hu, L. Du, and X. Cai, "A many-objective optimized task allocation scheduling model in cloud computing," *Appl. Intell.*, vol. 51, no. 6, pp. 3293–3310, 2021.

[6] M. S. A. Khan and R. Santhosh, "Task scheduling in cloud computing using hybrid optimization algorithm," *Soft Comput.*, vol. 26, pp. 13069–13079, Nov. 2021.

[7] Z. Tong, X. Deng, F. Ye, S. Basodi, X. Xiao, and Y. Pan, "Adaptive computation offloading and resource allocation strategy in a mobile edge computing environment," *Inf. Sci.*, vol. 537, pp. 116–131, Oct. 2020.

[8] X. Zhang, D. Zhang, W. Zheng, and J. Chen, "An enhanced priority-based scheduling heuristic for DAG applications with temporal unpredictability in task execution and data transmission," *Future Gener. Comput. Syst.*, vol. 100, pp. 428–439, Nov. 2019.

[9] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Comput. Sci. Rev.*, vol. 40, May 2021, Art. no. 100379.

[10] D. Wu, J. Yan, H. Wang, and R. Wang, "User-centric edge sharing mechanism in software-defined ultra-dense networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 7, pp. 1531–1541, Jul. 2020.

[11] X. He, H. Lu, H. Huang, Y. Mao, K. Wang, and S. Guo, "QoE-based cooperative task offloading with deep reinforcement learning in mobile edge networks," *IEEE Wireless Commun.*, vol. 27, no. 3, pp. 111–117, Jun. 2020.

[12] F. Murtaza, A. Akhunzada, S. Ul Islam, J. Boudjadar, and R. Buyya, "QoS-aware service provisioning in fog computing," *J. Netw. Comput. Appl.*, vol. 165, Sep. 2020, Art. no. 102674.

[13] W. Cai, J. Zhu, W. Bai, W. Lin, N. Zhou, and K. Li, "A cost saving and load balancing task scheduling model for computational biology in heterogeneous cloud datacenters," *J. Supercomput.*, vol. 76, pp. 6113–6139, May 2020.

[14] X. Wei, "Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing," *J. Ambient Intell. Humanized Comput.*, vol. 750, pp. 1–12, Oct. 2020.

[15] Y. Wang and X. Zuo, "An effective cloud Workflow scheduling approach combining PSO and idle time slot-aware rules," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 5, pp. 1079–1094, May 2021.

[16] C.-Q. Dai, C. Li, S. Fu, J. Zhao, and Q. Chen, "Dynamic scheduling for emergency tasks in space data relay network," *IEEE Trans. Veh. Technol.*, vol. 70, no. 1, pp. 795–807, Jan. 2021.

[17] M. Satyanarayanan, T. Eiszler, J. Harkes, H. Turki, and Z. Feng, "Edge computing for legacy applications," *IEEE Pervasive Comput.*, vol. 19, no. 4, pp. 19–28, Oct.–Dec. 2020.

[18] Z. Fang, J. Wang, Y. Ren, Z. Han, H. V. Poor, and L. Hanzo, "Age of information in energy harvesting aided massive multiple access networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 5, pp. 1441–1456, May 2022.

[19] J. Guo, G. Huang, Q. Li, N. N. Xiong, S. Zhang, and T. Wang, "STMTO: A smart and trust multi-UAV task offloading system," *Inf. Sci.*, vol. 573, pp. 519–540, Sep. 2021.

[20] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[21] X. Chen et al., "A WOA-based optimization approach for task scheduling in cloud computing systems," *IEEE Syst. J.*, vol. 14, no. 3, pp. 3117–3128, Sep. 2020.

[22] M. Abdel-Basset, D. El-Shahat, M. Elhoseny, and H. Song, "Energy-aware metaheuristic algorithm for Industrial-Internet-of-Things task scheduling problems in fog computing applications," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12638–12649, Aug. 2021.

[23] T. Mohanraj and R. Santhosh, "Multi-swarm optimization model for multi-cloud scheduling for enhanced quality of services," *Soft Comput.*, vol. 26, pp. 12985–12995, Oct. 2021.

[24] Y. Liu, X. Xu, L. Zhang, L. Wang, and R. Y. Zhong, "Workload-based multi-task scheduling in cloud manufacturing," *Robot. Comput. Integr. Manuf.*, vol. 45, pp. 3–20, Jun. 2017.

[25] H. Teng, Z. Li, K. Cao, S. Long, S. Guo, and A. Liu, "Game theoretical task offloading for profit maximization in mobile edge computing," *IEEE Trans. Mobile Comput.*, early access, May 6, 2022, doi: 10.1109/TMC.2022.3175218.

[26] P. Wang, Z. Zheng, B. Di, and L. Song, "HetMEC: Latency-optimal task assignment and resource allocation for heterogeneous multi-layer mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4942–4956, Oct. 2019.

[27] J. Zhang, X. Zhou, T. Ge, X. Wang, and T. Hwang, "Joint task scheduling and Containerizing for efficient edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 8, pp. 2086–2100, Aug. 2021.

[28] J. Yu, "Qualitative simulation algorithm for resource scheduling in enterprise management cloud mode," *Complexity*, vol. 2021, no. 5, pp. 1–12, 2021.

[29] J. Lee, D. Kim, and J. Lee, "ZONE-based multi-access edge computing scheme for user device mobility management," *Appl. Sci.*, vol. 9, no. 11, p. 2308, 2019.

[30] X. Zhu, Y. Luo, A. Liu, M. Z. A. Bhuiyan, and S. Zhang, "Multiagent deep reinforcement learning for vehicular computation offloading in IoT," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9763–9773, Jun. 2021.

[31] X. Xu, R. Mo, F. Dai, W. Lin, S. Wan, and W. Dou, "Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 6172–6181, Sep. 2020.

[32] Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, "A scheduling scheme in the cloud computing environment using deep *Q*-learning," *Inf. Sci.*, vol. 512, pp. 1170–1191, Feb. 2020.

[33] Z. Chen, J. Hu, G. Min, C. Luo, and T. El-Ghazawi, "Adaptive and efficient resource allocation in cloud datacenters using actor-critic deep reinforcement learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1911–1923, Aug. 2022.

[34] J. Zhou et al., "Security-critical energy-aware task scheduling for heterogeneous real-time MPSoCs in IoT," *IEEE Trans. Services Comput.*, vol. 13, no. 4, pp. 745–758, Jul./Aug. 2020.

[35] G. Wang, C. Li, Y. Huang, X. Wang, and Y. Luo, "Smart contract-based caching and data transaction optimization in mobile edge computing," *Knowl.-Based Syst.*, vol. 252, Sep. 2022, Art. no. 109344.

[36] J. Wang, C. Jiang, K. Zhang, X. Hou, Y. Ren, and Y. Qian, "Distributed *Q*-learning aided heterogeneous network association for energy-efficient IIoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2756–2764, Apr. 2020.

[37] M. Sheng, Y. Wang, X. Wang, and J. Li, "Energy-efficient multiuser partial computation offloading with collaboration of terminals, radio access network, and edge server," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1524–1537, Mar. 2020.

[38] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[39] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[40] R. Peakall and P. E. Smouse, "GENALEX 6: Genetic analysis in excel. population genetic software for teaching and research," *Mol. Ecol. Notes*, vol. 6, no. 1, pp. 288–295, 2006.

[41] B. L. Sun et al., "Role of secreted extracellular nicotinamide phosphoribosyltransferase (eNAMPT) in prostate cancer progression: Novel biomarker and therapeutic target," *EBioMedicine*, vol. 61, Nov. 2020, Art. no. 103059.

[42] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.

[43] H. Yuan, G. Tang, X. Li, D. Guo, L. Luo, and X. Luo, "Online dispatching and fair scheduling of edge computing tasks: A learning-based approach," *IEEE Internet Things J.*, vol. 8, no. 19, pp. 14985–14998, Oct. 2021.

Zhao Tong (Member, IEEE) received the Ph.D. degree in computer science from Hunan University, Changsha, China, in 2014.

He was a Visiting Scholar with Georgia State University, Atlanta, GA, USA, from 2017 to 2018. He is currently an Associate Professor with the College of Information Sciences and Engineering, Hunan Normal University, Changsha. He has published more than 20 research papers in international conferences and journals, such as the IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *Information Sciences*, and *Journal of Parallel and Distributed Computing*. His research interests include modeling and scheduling for parallel and distributed computing systems.

Dr. Tong is a member of CCF.

Bilan Liu is currently pursuing the master's degree with the College of Information Sciences and Engineering, Hunan Normal University, Changsha, China.

Her research interests include cloudedge collaborative computing, deep learning algorithms, and combinatorial optimization.

Jing Mei received the Ph.D. degree in computer science from Hunan University, Changsha, China, in 2015.

She is currently an Assistant Professor with the College of Information Sciences and Engineering, Hunan Normal University, Changsha. She has published more than 12 research articles in international conferences and journals, such as the IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON SERVICE COMPUTING, *Cluster Computing*, *Journal of Grid Computing*, and *Journal of Supercomputing*. Her research interests include parallel and distributed computing and cloud computing.

Jiake Wang is currently pursuing the master's degree with the College of Information Science and Engineering, Hunan Normal University, Changsha, China.

Her research interests include mobile-edge computing, deep learning algorithms, and combinatorial optimization.

Wenbin Li, photograph and biography not available at the time of publication.

Keqin Li, photograph and biography not available at the time of publication.