IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 9, 2020, accepted October 20, 2020, date of publication October 26, 2020, date of current version November 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3033557

A Two-Stage Framework for the Multi-User Multi-Data Center Job Scheduling and Resource Allocation

JIANPENG LIN, DELONG CUI, ZHIPING PENG, QIRUI LI, AND JIEGUANG HE
Guangdong University of Petrochemical Technology, Maoming 525000, China

Corresponding authors: Delong Cui (delongcui@163.com) and Zhiping Peng (pengzp@foxmail.com)

This work was supported by the National Natural Science Foundation of China under Grant 61772145 and Grant 61672174.

ABSTRACT With the development of artificial intelligence and the Internet of things, the prospects of cloud computing applications have become broader, and the number of users and cloud data centers (CDCs) has exploded. It is a challenge to realize efficient job scheduling and resource allocation of multiple users and data centers. However, the traditional scheduling model based on heuristic algorithm has some limitations in the complex and changeable cloud environment. In addition, many existing single-agent models rarely consider the multi-objective global optimization of the system. Therefore, this paper proposes a two-stage job scheduling and resource allocation framework that adopts multiple intelligent schedulers to solve the cooperative scheduling problem between job scheduling and resource allocation. A heterogeneous distributed deep learning (HDDL) model is used in the job scheduling stage to schedule multiple jobs to multiple cloud data centers. The deep Q-network (DQN) model is a resource scheduler to deploy virtual machine to physical servers for execution. Extensive numerical results show that both HDDL-baesd job scheduler and DQN-based resource allocator outperform the benchmark algorithm in terms of optimizing energy consumption and job delay. Furthermore, the proposed framework not only can achieve a global near-optimum by achieving local optimization at each stage but also has good scalability and low computation delay.

INDEX TERMS Cloud computing, multi-user multi-data center, resource allocation, job scheduling.

I. INTRODUCTION

The rapid implementation of cloud computing has resulted in significant investment and fast growth. Continuous research of cloud computing has become a powerful engine for the development of artificial intelligence, further accelerating the development of applications such as big data and the Internet of things [2]. With the explosive growth of applications, users, and cloud service providers (CSPs), efficient job scheduling and resource allocation of multi-user multi-CSPs has become a major challenge [3]. An effective scheduling strategy must consider quality of service (QoS) and CSP benefits, and significant progress has been made.

An improved heuristic algorithm is the most common solution to the scheduling problems of cloud computing [5]. Alkayal *et al.* [6] combined multi-objective optimization (MOO) and particle swarm optimization (PSO) to optimize resource allocation, aiming to schedule jobs to virtual machines (VMs) with minimal waiting time and maximum system throughput. Hu *et al.* [7] devised a scientific workflow multi-objective scheduling algorithm for the reliability of workflow scheduling in a multi-cloud environment, with a goal to minimize the completion time and cost of workflow under reliability constraints. A number of hybrid algorithms [8]–[10] combine the excellent characteristics of multiple heuristic algorithms to achieve good performance.

Some scholars have adopted the decision-making ability and trial-error mechanism of reinforcement learning (RL) to explore optimal scheduling strategies. Cui *et al.* [11] and Peng *et al.* [12] combined RL and queue theory to solve task scheduling and resource allocation problems in complex cloud environments, cleverly transforming a scheduling problem to a sequence decision problem and adopting the RL agent to continuously interact with the cloud environment to

The associate editor coordinating the review of this manuscript and approving it for publication was Zhiwu Li.

explore optimal scheduling strategies. Considering the automatic scaling of applications in the cloud, Wei *et al.* [13] presented a reinforcement learning-based auto-scaling approach for SaaS providers in a dynamic cloud environment. However, RL has certain limitations and has convergence problems in the case of a high-dimensional state space.

Against this background, a number of researchers have adopted deep reinforcement learning (DRL) to achieve breakthroughs in automatic control [14] and game competition [15], [16] and better solve cloud computing scheduling problems. Bu and Wang [17] devised a DRL-based hierarchical system resource allocation and power management framework, which includes a global layer to allocate VMs to physical servers and a local layer for server power management. The framework achieves the best balance between server cluster latency and energy. For large-scale cluster resource scheduling in cloud computing, Ran *et al.* [18] presented a DRL-based elastic resource provision system that can automatically and dynamically allocate computing resources with the best resource management strategy according to changes in user workload demands. Bitsakos *et al.* [19] proposed a hierarchical hybrid online task scheduling framework based on DRL, which takes into account various optimization objectives. Peng *et al.* [20] developed a task scheduling and resource allocation framework based on a DQN algorithm, which can dynamically tradeoff multi-objective optimization relationships in accordance with dynamic requirements.

The above research can effectively solve specific scheduling problems, but it has several deficiencies. First, optimization is specific to job scheduling or resource allocation, rather than global. Second, optimization tends to focus on user service quality or CSP benefit and is not multi-objective. We devise a two-stage cloud job scheduling and resource allocation framework with multiple intelligent schedulers to complete scheduling tasks in different stages. The HDDL [21] model schedules multiple jobs to multiple data centers. At the resource allocation stage, DQN configures VMs for tasks and deploys them to physical servers for execution. The framework considers user service quality and CSP benefits to realize multi-objective optimization.

Our work makes the following three main contributions.

(1) To figure out the cooperative problem of job scheduling and resource allocation for a multi-user and multi-data center, a two-stage scheduling framework is proposed, with stages of job scheduling and resource allocation.
(2) Various intelligent schedulers which combine the excellent features of deep learning and reinforcement learning, are adopted to effectively solve large-scale cloud scheduling problems of different scheduling stages. In addition, different agents realized multi-objective optimization of cloud system scheduling based on cooperation mechanisms.
(3) The proposed framework is flexible, extensible and low latency, which can not only dynamically adjust system optimization goals according to the actual requirements, but also scale the size of data center.

This paper is organized as follows. Some of the related research work is organized in section 2. Section 3 introducess the key stages and components of the system framework and defines the job scheduling and resource allocation models. Section 4 discusses our extensive simulations to test the performance and advantages of the proposed framework. Section 5 draws our conclusions and makes a future prospect.

II. RELATED WORKS

Scheduling of cloud computing is NP-complete and is not fully solved. Pinedo and Hadavi [23] give the following definition: ''Scheduling is a decision-making process that is used on a regular basis in many manufacturing and services industries. It deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives''. In large-scale cloud systems, multiple resources (e.g., computing and network) and user requirements change dynamically. Therefore, to find the optimal solution for a dynamic scheduling problem is a difficult but forced challenge. It is feasible to solve the scheduling problem using a deep reinforcement model combining the excellent perception ability of DL with the decision-making ability of RL. DRL is an adaptive intelligent learning agent, which continuously interacts with the dynamic environment and explores the optimal scheduling strategy through trial-error and reward [24]. DRL has been used to solve cloud job scheduling and resource allocation problems. Some recent work has used an innovative architecture-hybrid hierarchical scheduling framework in cloud computing systems. Multiple intelligent schedulers achieve collaborative job scheduling and resource allocation to realize superior optimized performance.

A. RESEARCH ON JOB SCHEDULING

Workflow scheduling methods are still limited in aspects such as task dependency. Tong *et al.* [25] proposed an intelligent method called deep Q learning task scheduling (DQTS) to solve the problem of directed acyclic graph (DAG) task scheduling. Wang *et al.* [26] realized multi-objective workflow scheduling based on multi-DQN agent reinforcement learning. For the uncertainty of real-time workflow scheduling, Chen *et al.* [27] developed an unceRtainty-aware Online Scheduling Algorithm (ROSA), which skillfully integrates both the proactive and reactive strategies, to schedule dynamic and multiple workflflows with deadlines. With the growth of cloud computing applications, mobile edge computing (MEC) has seen broad application prospects [28]. In the area of MEC task offloading, Liu *et al.* [29] considered the dynamic changes of user equipment (UE) location and service requests and proposed a DRL-based task scheduling scheme. Large numbers of vehicles are used as mobile edge servers to provide computing services to nearby equipment, effectively solving the task scheduling problem in a changing environment.

Similarly, Huang *et al.* [30] presented a DQN-based MEC task offloading and resource allocation algorithm that efficiently offloads tasks of multiple mobile terminals to multiple edge servers to achieve reasonable resource allocation and minimize global costs. Huang *et al.* [30] and Liu [31] proposed actor-critic DRLS-based computation offloading for three-tier mobile computing networks. This model minimizes the weighted sum of task delays and energy consumption by optimizing task offloading strategies in dynamic network systems. Lu *et al.* [32] developed an improved IDRQN algorithm to solve the offloading problem of multi-service nodes and multi-mobile tasks in large-scale heterogeneous MEC clusters. Considering task scheduling for fog-based internet of things applications, a double deep Q learning (DDQL) scheduling algorithm [33] was adopted to minimize long-term service delays and computation costs under resource and deadline constraints.

B. RESEARCH ON RESOURCE ALLOCATION

Revenue is the primary goal of CSPs, so it is crucial to reduce the energy consumption of data centers while ensuring the quality of user services. Yadav *et al.* [34], [35] proposed several adaptive heuristic algorithms to reduce energy consumption while maintaining the required performance levels in CDCs. Moreover, to reduce the energy consumption of large-scale mobile cloud data centers, a novel adaptive heuristic host overload detection algorithm called MeReg was introduced, takeing energy consumption and SLA violations into full consideration [36]. Li *et al.* [37] applied the DRL model to cloud computing and residential smart grid systems to effectively solve cloud resource allocation and smart grid user task scheduling problems. Seyed *et al.* [38] devised a controller based on RL and implemented a distributed architecture that can rapidly expand system resources to meet demands, and shut down idle servers to save costs. Xiao *et al.* [39] developed a DRL-based system, SARA, to find the optimal cloud allocation strategy for workloads. It is particularly stable and fast for heterogeneous data-intensive workloads. To improve the QoS of edge computing, Carpio *et al.* [40] adopted a resource allocation algorithm based on DQN, which has strong adaptability. Wang *et al.* [41] paid more attention to efficient resource allocation in a dynamically changing MEC environment to meet the needs of mobile devices. They proposed an intelligent resource allocation scheme based on DRL (DRLRA) to adaptively allocate computing and network resources and reduce average service time.

C. RESEARCH ON COOPERATIVE SCHEDULING

Researchers have proposed a variety of hybrid hierarchical scheduling frameworks to address the issues of job scheduling and resource allocation, adopting multiple intelligent schedulers for the global multi-objective optimization of a system. A hierarchical resource allocation and power management framework based on DRL was devised, including a global layer to assign VMs to physical machines and a local layer for power management [42]. Cheng *et al.* [43] introduced a DRL-based resource provision (RP) and task scheduling (TS) system to minimize the energy cost of CSPs. The two-layer RP-TS processor automatically generates the best long-term decisions by learning from changing environments while considering the task offload of mobile devices with limited resources. Quan *et al.* [44] developed a two-layer RL (TLRL) algorithm. The DQN scheduler at the first layer assigns clusters to offload tasks, and the second layer specifies the physical machine for a task. Cheng *et al.* [45] presented a hierarchical hybrid online task scheduling framework based on DRL. The framework comprehensively considers various key factors of cloud platform and has high efficiency in terms of energy and cost.

III. SYSTEM FRAMEWORK

As shown in Figure 1, the system framework of this paper divides the cloud scheduling process into the two key stages of job scheduling and resource allocation.

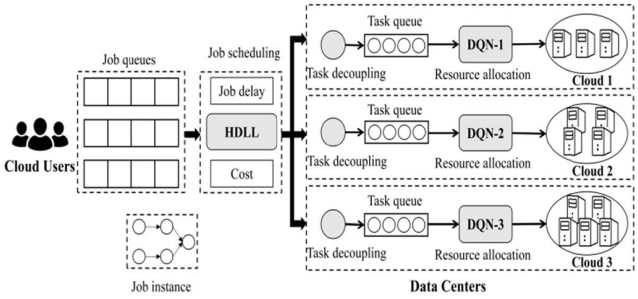

FIGURE 1. System framework.

A. JOB SCHEDULING

Multiple users submit jobs with different requirements to the cloud data center through the network and pay on demand for cloud computing services [5]. Jobs may be computation- or data-intensive, for example, while data centers differ in computing power, bandwidth, and energy consumption. The job scheduling stage selects appropriate data centers for jobs according to user requirements and data center attributes, so as to execute jobs with minimum delay and cost. Moreover, users and data centers have an intricate geographic relationship [3]. The system forms multiple job queues by gathering workloads from geographically close locations. The HDLL schedules jobs to multiple data centers according to the optimal scheduling strategy, taking into account both QoS and CSP benefits.

B. RESOURCE ALLOCATION

The cloud system decouples subtasks of jobs that have task dependencies and arranges them in the task queue in topological order to meet their dependencies [46]. The data center links multiple heterogeneous physical servers through high-speed bandwidth. Note that data placement factors and data transmission costs and time in the same data center are

not considered in this work. Various servers have different maximum load capacity and energy consumption. DQN, as a resource scheduler, deploys VMs to physical servers according to the state of a system, aiming to achieve the optimal server load and reduce the energy consumption of data centers while ensuring QoS.

IV. SYSTEM MODEL

A. JOB SCHEDULING MODEL AND ALGORITHM DESIGN

For the job scheduling model, we consider N cloud job queues, $\{Q_1, \ldots, Q_n\}$, each with a fixed number of jobs M, represented as $\{J_1, \ldots, J_m\}$, thus, the total number of jobs is $N \cdot M$. K data centers are denoted as $\{DC_1, \ldots, DC_k\}$. Job J_{nm} (job m in queue n) is associated with a parameter tuple $\{\alpha_{nm}, \beta_{nm}\}$, representing the required CPU cycles and size of data transmission, respectively. We set the linear correlation between the required CPU cycles and the size of data transmission [47], $\alpha_{nm} = q \cdot \beta_{nm}$, where q is the computation-to-data ratio. Each data center DC_k is associated with a parameter tuple, $\{C_k, P_k^{run}, BW_k, P_k^{comm}\}$, which represents computational power, running power, bandwidth, and communication power of DC_k, respectively. Note that the data center adopts an average resource allocation strategy to divide the computing power and bandwidth equally between its cloud jobs. The system state space s consists of all cloud jobs in the queue, represented as $\{\alpha_{11}, \beta_{11}, \ldots, \alpha_{nm}, \beta_{nm}\}$, and the output action decision d is $\{a_{11}, a_{12}, \ldots, a_{nm}\}$, where a_{nm} contains K binary numbers. anm is denoted as $\{b_1, b_2, \ldots, b_k | k \in K\}$, $b_k \in \{0, 1\}$. For example, $b_k = 1$ represents that J_m in Q_n is deployed to DC_k. The optimization goal of the job scheduling model is to reduce job delay and energy consumption.

The communication model includes communication delay and energy consumption for data transmission. The CPU cycles and bandwidth allocated to each job in the data center DC_k can be calculated as

$$R_k^{cpu} = \frac{C_k}{A_k} \tag{1}$$

$$R_k^{bw} = \frac{BW_k}{A_k}, \tag{2}$$

where A_k is the number of jobs in DC_k. Thus the job communication delay and communication energy consumption are, respectively,

$$D_{nm}^{comm} = \frac{\beta_{nm}}{R_k^{bw}} \tag{3}$$

$$E_{nm}^{comm} = P_k^{comm} \cdot D_{nm}^{comm}. \tag{4}$$

The computation model includes computation delay and computation energy consumption, which are calculated, respectively, as

$$D_{nm}^{comp} = \frac{\alpha_{nm}}{R_k^{cpu}} \tag{5}$$

$$E_{nm}^{comp} = P_k^{comp} \cdot D_{nm}^{comp}. \tag{6}$$

Considering the optimization goals of job delay and energy consumption, the cost function obtained by taking decision d in state space s is calculated as

$$Cost(s, d) = w1 \cdot \max_{n \in N, m \in M} (D_{nm}^{comm} + D_{nm}^{comp}) + w2 \cdot \sum_{n \in N} \sum_{m \in M} (E_{nm}^{comm} + E_{nm}^{comp}) \tag{7}$$

where $w1$, $w2$ are the reward weights of delay and energy consumption, respectively. In addition, the maximum delay (the sum of communication delay and computation delay) of all tasks in the queue is used to represent the delay cost, while the total energy consumption of all tasks in the queue is taken as the system energy cost. The sum of the two can accurately calculate the cost of the decision. The ultimate goal is to minimize the job delay and energy consumption, as

$$Min\ Cost(s, d). \tag{8}$$

The goal is to schedule jobs in multiple queues to specific data centers according to a scheduling strategy. HDDL is used as the scheduler. The model [21] uses multiple heterogeneous deep neural networks (DNNs) as the fitting function, which not only accelerates the convergence speed, but also explores the optimal solution of the scheduling problem more efficiently. In addition, the experience replay mechanism [16] of DRL is used in this model to improve the utilization of training samples. The DNNs have the same number of hidden layers but different numbers of hidden layer nodes. Figure 2 shows the HDDL model structure.

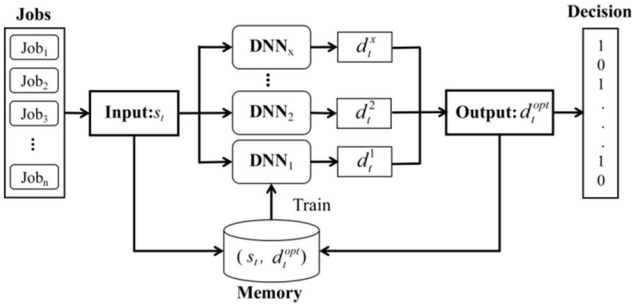

FIGURE 2. Architecture of HDDL model.

We describe the training steps of the HDDL model. The job state s_t of the system contains the job properties in all queues, which represented as $\{\alpha_{11}, \beta_{11}, \ldots, \alpha_{nm}, \beta_{nm}\}$. s_t is the input of x DNNs, which output x action decisions, f_θ^x: $s_t \rightarrow d_t^x$, where θ^x is the x-th DNN parameter. The cost of each action decision is calculated according to the cost function $Cost(s_t, d_t)$. The HDDL model takes the action decision with the maximum reward as the optimal decision d_t^{opt} of this episode. Memory is a key component in the model, which store s_t and optimal decision d_t^{opt} of each training episode of each DNN model as samples. Min-batch samples are randomly selected from the memory as a common training sample set for each DNN. The training goal is to minimize the cost. The cross-entropy loss function is defined as follows,

$$L(\theta^x) = -d_t^T \log f_{\theta^x}(s_t) - (1 - d_t)^T \log(1 - f_{\theta^x}(s_t)). \tag{9}$$

and updates θ^x with stochastic gradient descent (SDG). The pseudo-code of the job scheduling algorithm based on HDDL is shown as Algorithm 1.

ALGORITHM 1 PSEUDO - CODE OF JOB SCHEDULING ALGORITHM BASED ON HDDL

Algorithm 1 Pseudo-Code of Job Scheduling Algorithm Based on HDDL

1. Initialize all X DNNs with different random weights θ^x, $x \in X$.
2. Initialize memory M to capacity D.
3. **Input:** all job properties in job-ready queues.
4. **Output**: job scheduling decisions d^x.
5. **For** $t = 1, 2, \ldots, T$ **do**:
6. Input s_t to each DNN.
7. Generate $\boldsymbol{X}$ scheduling decisions, $\{d_t^x\} = f_{\theta_t^x}(s_t)$, from the DNNs.
8. Select the optimal decision

$$d_t^{opt} = \arg\min_{x \in X} Cost(s_t, d_t^x)$$

9. Store transition (s_t, d_t^{opt}) into $M(D)$.
10. **If** transition size reaches the threshold **do**:
11. Randomly sample mini-batch of transitions (s, d) from M (D) to train the DNNs.
12. Define loss function

$$L(\theta^x) = -d^T \log f_{\theta^x}(s) - (1-d)^T \log(1 - f_{\theta^x}(s)).$$

13. Update X DNN parameters using SDG.
14. **End If**
15. **End For**

B. RESOURCE ALLOCATION MODEL AND ALGORITHM DESIGN

In resource allocation, the cloud system configures a VM that meets the resource requirements for each task. Subsequently, as a scheduling unit, the VM is deployed to the server according to the scheduling strategy. We consider X physical servers in data center DC_k, expressed as $\{DC_k|S_1, S_2, \ldots, S_x\}$. The data center consists of multiple heterogeneous servers with different maximum load VMs, $\{VM_1^{max}, VM_2^{max}, \ldots, VM_x^{max}\}$. The number of VMs running on the server at time t is $\{VM_1^{run}(t), VM_2^{run}(t), \ldots, VM_x^{run}(t)\}$. Thus the load rate $U_x(t)$ of server S_x at time t can be calculated as

$$U_x(t) = \frac{VM_x^{run}(t)}{VM_x^{\max}} \cdot 100\%. \tag{10}$$

The energy consumption of server S_x at time t includes static energy consumption $P_{static}^x(t)$ and dynamic energy consumption $P_{dynamic}^x(t)$ of the server, both depending on the load rate of the server. $P_{static}^x(t)$ is constant when $U_x(t) > 0$, and otherwise is zero. A complicated relationship exists between the dynamic energy consumption and the load rate of the server. In work [48], the server has an optimal load rate U_{opt}^x. The dynamic energy consumption $P_{dynamic}^x(t)$ increases linearly with the load rate when $U^x(t) \leq U_{opt}^x$. However, the dynamic energy increases nonlinearly when $U^x(t) > U_{opt}^x$. Therefore, the dynamic energy $P_{dynamic}^x(t)$ can be calculated as

$$P_{dynamic}^x(t) = \begin{cases} U^x(t) \cdot \alpha_x, & (U^x(t) \leq U_{opt}^x) \\ U_{opt}^x \cdot \alpha_x + (U^x(t) - U_{opt}^x)^2 \cdot \beta_x, & (U^x(t) > U_{opt}^x), \end{cases} \tag{11}$$

where α_x is the linear growth rate, β_x is the nonlinear growth rate, and U_{opt}^x is the optimal load rate.

The total energy consumption of the cluster at time t is

$$E_{total}(t) = \sum_{x=1}^{X} (P_{static}^x(t) + P_{dynamic}^x(t)). \tag{12}$$

The resource allocation model selects the optimal VM deployment scheme based on the load and power consumption of each server in the cluster. Therefore, the load rate of all servers can be collected as the state space of the system, $\{U_1, U_2, \ldots, U_x\}$, and the action space is $\{a|S_1, S_2, \ldots, S_x\}$. The optimization goal of the resource allocation model is to reduce task delay and energy consumption; thus, the reward function for task delay optimization can be specified as $R_{delay} = 1$ if the VM can be successfully deployed to the target server, and otherwise, $R_{delay} = -1$. The reward function for energy consumption optimization can be calculated as

$$R_{ec} = E_{total}(t) - E_{total}(t-1). \tag{13}$$

Thus the total reward function of the system is

$$R_{total} = w^d \cdot R_{delay} + w^{ec} \cdot R_{ec}, \tag{14}$$

where the role of weighting parameters w^d, w^{ec} is to avoid a large difference between R_{delay} and R_{ec}, so as to ensure the training effect of the model. Furthermore, the weight ratio of the above two parameters can be dynamically adjusted to meet the actual requirements of the system.

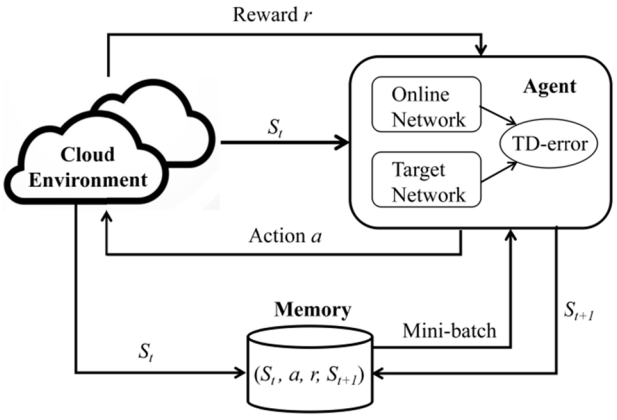

FIGURE 3. Architecture of DQN model.

As a classic model of deep reinforcement learning, DQN [16] is adopted as the scheduler in the resource allocation stage, with architecture as shown in Figure. 3.

Agent continuously interacts with the cloud environment, accumulating learning experience through reward and the experience replay mechanism to find the optimal scheduling strategy. The training steps for the DQN model are described. First, the system state s_t of the cloud environment at timestep t is used as the input of the neural network in Agent, which selects action a_t according to the strategy π, calculates the reward r_t, and returns the state s_{t+1} of the next time step. Subsequently, the model stores transitions (s_t, a_t, r_t, s_{t+1}) as training samples in Memory. Until the number of samples in Memory exceeds the threshold, mini-batch samples are randomly extracted as training set. The optimization goal of the model is to maximize the expected cumulative discount reward,

$$Q^*(s,a) = \max_{\pi} E\Big[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \mid s_t = s, a_t = a, \pi\Big]. \quad (15)$$

For each training episode, M transitions (s, a, r, s') are taken out of the memory, where the state s is the input of the online network to output the current value Q of action a. Similarly, the next state, s', is the input of the target network to obtain the maximum Q of all actions. Mean-square error (MSE) is adopted as the loss function:

$$L_i(\theta_i) = E_{(s,a,r,s')\sim D(M)}[(r + \gamma \max_{a'} Q(s', a'; \theta_i^{\sim}) - Q(s,a;\theta_i))^2] \quad (16)$$

where $\gamma \in [0,1]$ is the discount factor. θ_i and $\theta_i^{\sim}$ are the parameters of the online and target network, respectively, at the i-th iteration. Stochastic gradient descent (SDG) method is adopted to update θ_i. Every C iterations, the parameters of the online network are copied to the target network to update its parameters. The pseudo-code of the resource allocation algorithm based on DQN is shown as Algorithm 2.

For convenience, the notations used above are listed in Table 5 at the end of this paper.

V. SIMULATION EXPERIMENT AND RESULT ANALYSIS

We performed simulation experiments to test the performance of the proposed scheduling framework. First, the optimization effects of the job scheduler and resource allocator were compared and analyzed. Then, we selected multiple scheduler combinations as benchmarks and compared the global scheduling effects with the HDDL and DQN scheduler combinations adopted in the proposed framework. Finally, the average CPU computation time of each algorithm to make a scheduling decision is compared.

A. EXPERIMENTAL STEPS AND PARAMETER SETTING

In the job scheduling stage, we examine the effectiveness and convergence of the algorithm by observing the change in the cost ratio, which is equal to the ratio of the costs between the optimal decision and the HDDL. The closer the ratio is

ALGORITHM 2 THE PSEUDO CODE OF THE SCHEDULING ALGORITHM to 1,

Algorithm 2 Pseudo-Code of Resource Allocation Algorithm Based on DQN

1. Initialize memory D to capacity M
2. Initialize online network and target network with random weights θ, $\theta^{\sim}$
3. Initial original state space s with load rate of each server
4. **For** each episode **do**:
5. **For** each task in task-queue **do**:
6. Select $a_t = \text{argmax}\, Q(s_t, a)$ with probability $\boldsymbol{\varepsilon}$ or select random action a_t with probability 1- ε.
7. Execute action a_t, calculate the total reward
$$R_{total} = w^d \cdot R_{delay} + w^{ec} \cdot R_{ec}$$
8. Observe new state s_{t+1}, store transition (s_t, a_t, r, s_{t+1}) in memory $D(M)$
9. **End For**
10. **If** transition size reaches the threshold **do**:
11. Sample random mini-batch of transition (s, a, r, s') from $D(M)$
12. Define loss function
$$L(\theta) = E_{(s,a,r,s')\sim D(M)}[(r + \gamma \max_{a'} Q(s', a'; \theta^{\sim}) - Q(s,a;\theta))^2]$$
13. Update the online network θ using SDG:
$$\nabla_\theta L(\theta) = E_{(s,a,r,s')\sim D(M)}[(r + \gamma \max_{a'} Q(s', a'; \theta^{\sim}) - Q(s,a;\theta))\nabla_\theta Q(s,a;\theta)]$$
14. **End If**
15. Every $\boldsymbol{C}$ episodes, update target network $\theta^{\sim} = \theta$
16. **End For**

the better the performance of HDDL. Then we compare the optimization effects of the proposed and benchmark algorithms in task delay and energy consumption. Finally, the proportion of the two optimization goals in the reward function is adjusted to test whether the HDDL algorithm can flexibly adjust the system optimization goals. The benchmarks include the random, round-robin (RR), Greedy, and multi-objective particle swarm optimization (MoPSO) algorithm [6]. The Greedy algorithm enumerates $K^{M \bullet N}$ scheduling schemes, calculates cost for each, and selects the scheme with the smallest cost as optimal. However, it is costly and uses much calculation time because the number of scheduling schemes increases exponentially with the number of data centers, queues, and jobs. In addition, the MoPSO algorithm based on swarm intelligence has been widely used in the field of engineering optimization due to its unique searching mechanism and excellent convergence performance. Note that 50 particle swarms were initialized in the MoPSO simulation.

In the simulation, we considered three job queues, four jobs per queue, and three data centers, for a total of $3^{3\cdot4}$

scheduling schemes. The data transmission amount of the job was distributed between [100, 500] MB and the number of subtasks contained in each job ranges from [10, 20]. Note that the quantitative relationship between the amount of data and number of CPU cycles is $\alpha_{nm} = q \cdot \beta_{nm}$, where $q = 330$ cycles/byte. The respective key parameters of the three data centers were set as follows. The calculation cycle number was $1.5 \cdot 10^{15}$ cycle/s, $2.5 \cdot 10^{15}$ cycle/s, and $3.5 \cdot 10^{15}$ cycle/s; the running power was $1.0 \cdot 10^5$ W, $2.5 \cdot 10^5$ W, and $4.0 \cdot 10^5$ W; the bandwidth was 250 Mbps, 550 Mbps, and 750 Mbps; the communication power was 0.2 W, 0.6 W, and 0.75 W; and the number of maximum load virtual machines was 410, 680, and 990. Using the above parameter settings, 500 sets of job sets were generated as datasets, each containing 12 jobs and corresponding minimum cost. Datasets were divided into training and test sets with the ratio 8: 2.

In the resource allocation stage, we selected one data center as the research object to verify the performance of the DQN scheduler. We observed the total reward of the DQN algorithm during training under a specific cluster load to verify the convergence and effectiveness of the scheduling algorithm. Then the scheduling effect of DQN scheduler under different cluster loads was tested. Finally, we explored the scheduling strategy of DQN scheduler by analyzing the load distribution of various servers in the cluster. The benchmarks at this stage include the Random, Round-Robin (RR), Minimum Load First (MLF), and Fast and Energy-aware Resource Provisioning and Task Scheduling (FERPTS) algorithm [46]. The advantage of the RR algorithm is its simplicity, with the scheduler taking turns assigning tasks to the server at each time step. With MLF, the scheduler chooses the server with the lowest load rate as the target server, which can achieve load balancing among the servers. With FERPTS, one contemporary iterative algorithm which divide the provisioning and scheduling to multiple steps, and can effectively reduce the complexity and minimize the run time while achieving a reasonable energy cost.

We considered that the data center has eight servers, consisting of four types of servers, two for each type. The respective parameters were set as follows for the four servers. The numbers of maximum load VMs were 40, 50, 55, and 60; the optimal loads were 0.6, 0.65, 0.7, and 0.75; the linear growth rates were 0.5, 0.65, 0.75, and 0.85; and the nonlinear growth rates were 14, 13, 11, and 9. Note that a large number of simulation results show that better optimization effect can be achieved by setting parameters w^d and w^{ec} to 1.0 and 20.0, respectively. To more clearly represent the dynamic energy consumption relationship of various servers, the dynamic energy consumption curves are shown in Figure 4.

In the global scheduling simulation experiment, we selected seven scheduler combinations as benchmarks to test the global optimization performance of our proposed framework. We conducted 50 global scheduling experiments to compare the optimization effects of different scheduler combinations on QoS as well as energy consumption. Job delay and delay rate were adopted as indicators of QoS.

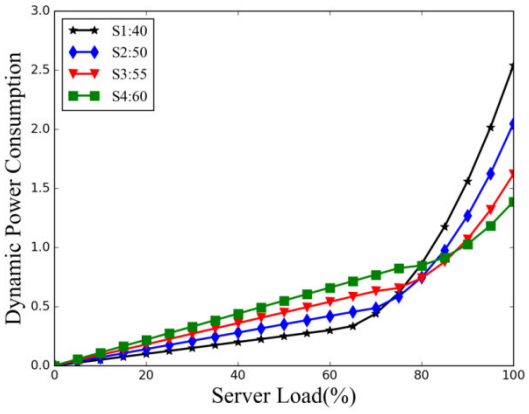

FIGURE 4. Relationship between server load and dynamic energy consumption.

TABLE 1. HDDL model parameters.

Parameter	Value	Parameters	Value
Episodes	500	Learning Rate	0.01
Memory Size	1024	Hidden Layers	4
Mini-batch	128	Network Number	5
Training Interval	10	Optimizer	RMSProp

TABLE 2. DQN model parameters.

Parameters	Value	Parameters	Value
Episodes	600	Frequency of Replacement	300
Learning Rate	0.01	Initial Value of ε	0.4
Discount Factor	0.9	Maximum Value of ε	0.9
Memory Size	2048	ε-greedy Increment	0.002
Mini-batch	64	Hidden layers	3

Simulation experiments were conducted on an experimental platform with the Python-3 and Tensorflow-1.2. The key parameters of the HDDL and DQN models adopted in the experiments are shown in Tables 1 and 2, respectively.

B. EXPERIMENTAL RESULTS AND ANALYSIS

1) EXPERIMENTAL RESULTS OF JOB SCHEDULING STAGE

Figure 5 shows the changes in cost ratio of each algorithm during training, from which it is clear that the cost ratio of HDDL gradually increases with the amount of training, and the final convergence value is slightly better than that of MoPSO. It is worth noting that the convergence value of the HDDL curve is close to 1, which means that the closer the ratio is to 1, the better the performance of scheduling model.

Figure 6 shows the job delay and energy consumption of each algorithm running 10 sets of jobs when $\lambda^d = 0.5$ and $\lambda^e = 0.5$. It is seen that the delay and energy consumption of HDDL are closer to those of the greedy algorithm among the benchmark algorithms.

Figure 7 shows the variation trend of the delay and energy consumption rewards obtained by the HDDL algorithm running 10 sets of jobs under different weights of *w1* and *w2* in formula (7). It can be seen that the delay reward decreases with the increase of *w1*, indicating that the optimization goal of the scheduler is more inclined to reduce job delay. As the

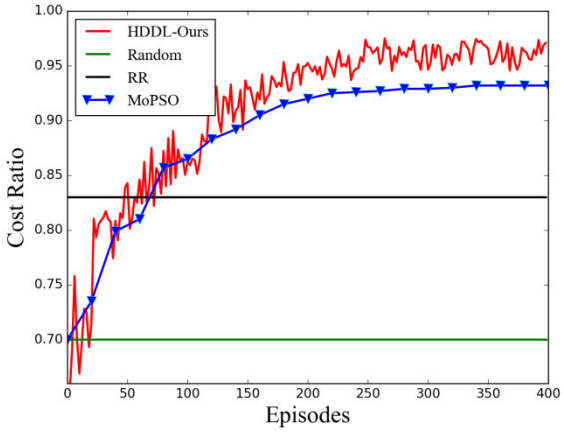

FIGURE 5. Cost ratio.

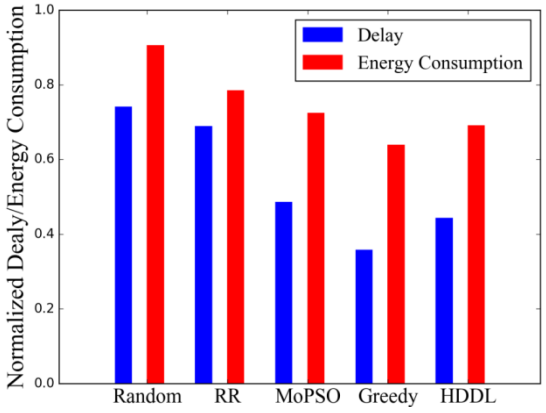

FIGURE 6. Job delay and energy consumption of each algorithm.

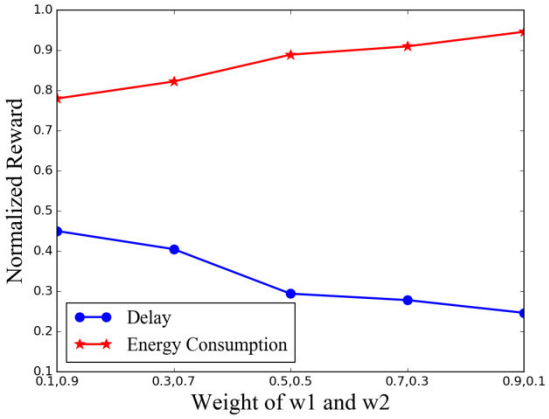

FIGURE 7. Trend of delay reward and energy consumption reward.

energy consumption weight *w2* decreases, the energy consumption gradually increases. Therefore, it can be inferred that HDLL can dynamically adjust the optimization goal according to different weight values to meet scheduling requirements with good flexibility.

2) EXPERIMENTAL RESULTS OF RESOURCE ALLOCATION STAGE

Figure 8 shows the total return of each algorithm for completing task scheduling under 90% load of the cluster. From a macro-perspective, as the number of episodes increases, the reward curve of DQN gradually increases and eventually converges. From a micro-perspective, during the first 300 training episodes, the DQN reward curve gradually increases, exceeding the random, RR, and approaching the MLF reward curve. After 300 episodes of training, the reward of DQN begins to surpass the MLF curve and gradually converge. In addition, the convergence trend of DQN is similar to that of FERPTS [42] during the whole iteration, but get more reward.

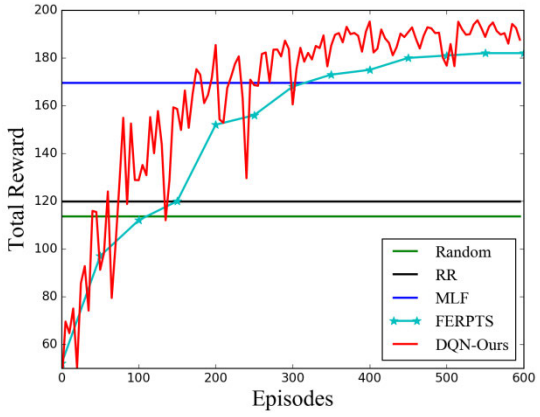

FIGURE 8. Total reward under 90% workload.

Figure 9 shows the changes in the total reward obtained by various algorithms for scheduling tasks under different cluster loads. There are three notable trends. First, when the cluster load is less than 65%, the reward curve of each algorithm shows an upward trend. The rewards of DQN, MLF and FERPTS are similar and are better than those of other benchmark algorithms. Second, the gap in reward between them is small at less than 75% of the cluster load. Finally, when the load exceeds 75%, three of them rewards begin to decline, DQN declines slower than MLF and FERPTS, and the total reward is better than those of other benchmark algorithms.

The analysis is as follows. The MLF scheduling strategy is more inclined to ensure the load balance of each server. It is a better strategy for low loads, but under high loads, the dynamic energy consumption of different servers increases at different rates, resulting in poor scheduling. Moreover, FERPTS pays more attention to the reduce run time of task scheduling, which leads to the increase of cluster energy consumption under high load. The DQN algorithm learns the scheduling strategy from historical experience, considering the relationship between dynamic energy consumption and load under high loads. Thus the scheduler can dynamically adjust the scheduling strategy according to the load state of the system for further optimization.

Figure 10 shows the load distribution of various servers under different cluster loads. As shown in the box-plot, when the cluster is under a low load, the load rate of weak servers is larger, and the load rate of strong servers is smaller. As the

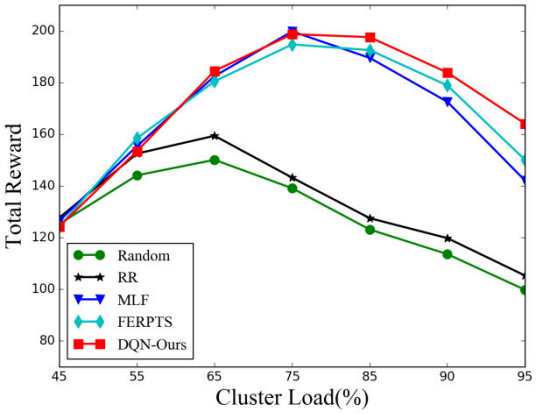

FIGURE 9. Total reward under different cluster loads.

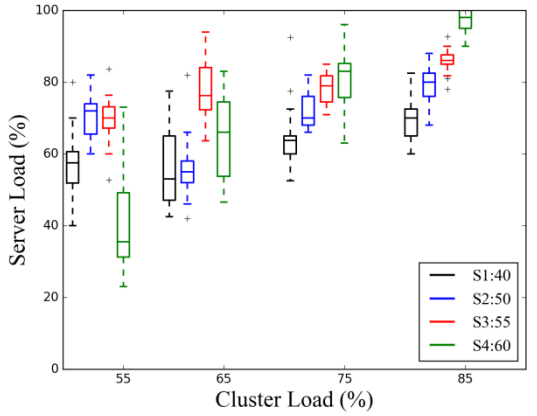

FIGURE 10. Load distribution of various servers under different cluster loads.

TABLE 3. Optimization results of each scheduler combination.

No.	Job Scheduler	Resource Scheduler	QoS		Energy Consumption
			Job Delay	Delay Rate	
1	Random	DQN	68.9	14.2%	17.75
2	RR	DQN	66.5	12.4%	15.86
3	MoPSO	DQN	35.50	1.2%	11.65
4	**HDDL**	**DQN**	**33.21**	**0.37%**	**10.48**
5	HDDL	MLF	32.01	0	11.12
6	HDDL	FERPTS	31.43	0	11.61
7	HDDL	RR	54.04	1.4%	14.86
8	HDDL	Random	56.65	2.3%	15.63

cluster load increases, the load rate of strong servers increases significantly, approaching full load.

The analysis is as follows. When the cluster load is below 65%, the linear increase in energy consumption of the weak server is smaller than that of the strong server, so the scheduler will get more reward for deploying more tasks to the weak server. With the increase of cluster load, the nonlinear growth rate in energy consumption of the strong server is less affected by the load, so its energy consumption growth rate is less than that of the weak server. Therefore, when the cluster is under high load, the scheduler tends to deploy more tasks to the server with strong load capacity to reduce the total energy consumption of the cluster.

TABLE 4. Average CPU computation time of different algorithms.

Job Scheduler		Resource Scheduler	
Algorithms	Time (s)	Algorithms	Time (s)
Random	$1.92 \cdot 10^{-5}$	Random	$9.10 \cdot 10^{-6}$
RR	$2.08 \cdot 10^{-5}$	RR	$1.59 \cdot 10^{-6}$
Greedy	10.8	FERPTS	$2.78 \cdot 10^{-2}$
MoPSO	$3.7 \cdot 10^{-2}$	MLF	$1.87 \cdot 10^{-5}$
HDDL	$\mathbf{1.8 \cdot 10^{-3}}$	**DQN**	$\mathbf{5.69} \cdot 10^{-4}$

TABLE 5. Notation used in this paper.

Notation	Description	Notation	Description
α_{nm}	CPU cycle requirement of job m of queue n	β_{nm}	Data size of job m of queue n
C_k	Computational power of data center DC_k	P_k^{run}	Running power of data center DC_k
BW_k	Bandwidth of data center DC_k	P_k^{comm}	Communication power of data center DC_k
a_{nm}=(0,0,1)	Deploy job m in queue n to the 3rd data center	R_k^{cpu}	CPU cycles allocated to each job in data center DC_k
R_k^{bw}	Bandwidth allocated to each job in data center DC_k	A_k	Number of jobs in data center DC_k
D_{nm}^{comm}	Communication delay of jobs scheduled to DC_k	E_{nm}^{comm}	Communication energy consumption of jobs scheduled to DC_k
D_{nm}^{comp}	Computational delay of jobs scheduled to DC_k	E_{nm}^{comp}	Computational energy consumption of jobs scheduled to DC_k
$VM_x^{\max}$	Maximum number of loaded virtual machines on server S_x	$VM_x^{run}(t)$	Number of virtual machines running on server S_x at time t
$U_x(t)$	Load rate of server S_x at time t	U_{opt}^x	Optimal load rate for server
$P_{static}^x(t)$	Static energy consumption of server S_x at time t	$P_{dynamic}^x(t)$	Dynamic energy consumption of server S_x at time t

3) EXPERIMENTAL RESULTS OF GLOBAL OPTIMIZATION

Table 3 shows the results of global optimization using different scheduler combinations. The purpose of experiments 1-4 was to verify the impact of different job schedulers on global optimization. Different combinations adopt various job schedulers, but they uniformly use DQN as a resource scheduler. These experiments show that HDDL and MoPSO adopt more efficient scheduling strategies according to the resource configuration of each data center, to effectively

reduce job delay and energy consumption. In addition, HDDL performs better than MoPSO. To examine the impact of different resource schedulers on global optimization, experiments 4-8 adopt HDDL as the job scheduler and use different resource schedulers. Results show that FERPTS and MLF are superior to DQN at reducing job delay. However, DQN reduced energy consumption by 5.7%, 9.7% compared to MLF and FERPTS respectively, while with an acceptable job delay rate increase.

In Table 4, we compared the average CPU computation time taken by different algorithms to make a scheduling decision. The results clearly show that the computation time of both HDDL and DQN is less than the classic heuristic algorithm MoPSO and iterative algorithm FERPTS, respectively. Moreover, the computation time of both is close to milliseconds, which is applicable for real-time application.

VI. CONCLUSION AND FUTURE WORK

For the cooperative scheduling problem of job scheduling and resource allocation for multi-user multi-cloud data centers, this paper proposes a novel two-stage scheduling framework with multiple intelligent schedulers. The proposed framework decomposes the complex cloud scheduling problem into two sub-scheduling problems, and uses different DRL-based intelligent models to act as the scheduler according to the optimization goals of different stages. Numerical simulation experiments show that both HDDL-baesd job scheduler and DQN-based resource allocator can achieve better performance and computation delay than baseline. Specifically, the proposed framework can achieve a global near-optimum by achieving local optimization at each stage. Besides, each intelligent scheduler in the cloud system is deployed independently, so it has good scalability. Both of them have low latency, which can be applied to real-time cloud platforms.

Our future research will focus on improving the cooperative ability of multiple learning models to achieve the local optimization of each sub-scheduling task, so as to obtain near-global optimization. Effective load forecasting will allow data centers to adjust the resource allocation in advance, such as by turning on more servers to respond to burst requests, or turning off idle servers to reduce the cost of cloud systems.

APPENDIX

HDDL: Heterogeneous distributed deep learning
DL: Deep learning
DQN: Deep Q-network
MOO: Multi-objective optimization
CDCs: Cloud data centers
GA: Genetic algorithm
DRL: Deep reinforcement learning
PSO: Particle swarm optimization
QoS: Quality of service
SLA: Server-level agreement
DAG: Directed acyclic graph
CSP: Cloud service provider
MEC: Mobile edge computing
MoPSO: Multi-objective particle swarm optimization
VM: Virtual machine
RL: Reinforcement learning
DNN: Deep neural networks
SDG: Stochastic gradient descent
RR: Round-robin
MLF: Minimum load first

REFERENCES

[1] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "Recent advancements in resource allocation techniques for cloud computing environment: A systematic review," *Cluster Comput.*, vol. 20, no. 3, pp. 2489–2533, Sep. 2017.

[2] R. Sturm, C. Pollard, and J. Craig, "The NIST definition of cloud computing," in *Proc. Appl. Perform. Manage. (APM) Digit. Enterprise*, 2017, pp. 267–269.

[3] L. F. Bittencourt, A. Goldman, E. R. M. Madeira, N. L. S. da Fonseca, and R. Sakellariou, "Scheduling in distributed systems: A cloud computing perspective," *Comput. Sci. Rev.*, vol. 30, pp. 31–54, Nov. 2018.

[4] N. Patil and D. Aeloor, "A review–different scheduling algorithms in cloud computing environment," in *Proc. 11th Int. Conf. Intell. Syst. Control (ISCO)*, Coimbatore, India, Jan. 2017, pp. 182–185.

[5] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Gener. Comput. Syst.*, vol. 91, pp. 407–415, Feb. 2019.

[6] E. S. Alkayal, N. R. Jennings, and M. F. Abulkhair, "Efficient task scheduling multi-objective particle swarm optimization in cloud computing," in *Proc. IEEE 41st Conf. Local Comput. Netw. Workshops (LCN Workshops)*, Nov. 2016, pp. 17–24.

[7] H. Hu, Z. Li, H. Hu, J. Chen, J. Ge, C. Li, and V. Chang, "Multi-objective scheduling for scientific workflow in multicloud environment," *J. Netw. Comput. Appl.*, vol. 114, pp. 108–122, Jul. 2018.

[8] M. Agarwal and G. M. S. Srivastava, "Genetic algorithm-enabled particle swarm optimization (PSOGA)-based task scheduling in cloud computing environment," *Int. J. Inf. Technol. Decis. Making*, vol. 17, no. 4, pp. 1237–1267, Jul. 2018.

[9] H. Ben Alla, S. Ben Alla, A. Touhafi, and A. Ezzati, "A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment," *Cluster Comput.*, vol. 21, no. 4, pp. 1797–1820, Dec. 2018.

[10] C.-L. Huang, Y.-Z. Jiang, Y. Yin, W.-C. Yeh, V. Y. Y. Chung, and C.-M. Lai, "Multi objective scheduling in cloud computing using MOSSO," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Rio de Janeiro, Brazil, Jul. 2018, pp. 1–8.

[11] D. Cui, W. Ke, Z. Peng, and J. Zuo, "Multiple DAGs workflow scheduling algorithm based on reinforcement learning in cloud computing," in *Proc. Comput. Intell. Intell. Syst. (ISICA)*, vol. 575, 2016, pp. 305–311.

[12] Z. Peng, D. Cui, Y. Ma, J. Xiong, B. Xu, and W. Lin, "A reinforcement learning-based mixed job scheduler scheme for cloud computing under SLA constraint," in *Proc. IEEE 3rd Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)*, Beijing, China, Jun. 2016, pp. 142–147.

[13] Y. Wei, D. Kudenko, S. Liu, L. Pan, L. Wu, and X. Meng, "A reinforcement learning based auto-scaling approach for SaaS providers in dynamic cloud environment," *Math. Problems Eng.*, vol. 2019, pp. 1–11, Feb. 2019.

[14] S. Phaniteja, P. Dewangan, P. Guhan, A. Sarkar, and K. M. Krishna, "A deep reinforcement learning approach for dynamically stable inverse kinematics of humanoid robots," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Macau, China, Dec. 2017, pp. 1818–1823.

[15] V. Mnih, K. Kavukcuoglu, and D. Silver "Playing Atari with deep reinforcement learning," in *Proc. Workshops 26th Neural Inf. Process. Syst. Comput. Sci.*, 2015, pp. 201–220.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[17] F. Bu and X. Wang, "A smart agriculture IoT system based on deep reinforcement learning," *Future Gener. Comput. Syst.*, vol. 99, pp. 500–507, Oct. 2019.

[18] L. Ran, X. Shi, and M. Shang, "SLAs-aware online task scheduling based on deep reinforcement learning method in cloud environment," in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Commun.*, Zhangjiajie, China, Aug. 2019, pp. 1518–1525.

[19] C. Bitsakos, I. Konstantinou, and N. Koziris, "DERP: A deep reinforcement learning cloud system for elastic resource provisioning," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2018, pp. 21–29.

[20] Z. Peng, J. Lin, D. Cui, Q. Li, and J. He, "A multi-objective trade-off framework for cloud resource scheduling based on the deep Q-network algorithm," *Cluster Comput.*, Jan. 2020, doi: 10.1007/s10586-019-03042-9.

[21] L. Huang, X. Feng, L. Zhang, L. Qian, and Y. Wu, "Multi-server multi-user multi-task computation offloading for mobile edge computing networks," *Sensors*, vol. 19, no. 6, p. 1446, Mar. 2019.

[22] L. F. Bittencourt, A. Goldman, E. R. M. Madeira, N. L. S. da Fonseca, and R. Sakellariou, "Scheduling in distributed systems: A cloud computing perspective," *Comput. Sci. Rev.*, vol. 30, pp. 31–54, Nov. 2018.

[23] M. Pinedo and K. Hadavi, "Scheduling: Theory, algorithms, and systems," *AIIE Trans.*, vol. 28, no. 8, pp. 695–697, 2016.

[24] Q. Liu, J. W. Zhan, and Z. Zhang, "A survey on deep reinforcement learning," *Chin. J. Comput.*, vol. 1, pp. 1–17, Jul. 2018.

[25] 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.

[26] Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo, and H. Xie, "Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 39974–39982, 2019.

[27] H. Chen, X. Zhu, G. Liu, and W. Pedrycz, "Uncertainty-aware online scheduling for real-time workflows in cloud service environment," *IEEE Trans. Services Comput.*, early access, Aug. 21, 2018, doi: 10.1109/TSC.2018.2866421.

[28] S. S. Gill, S. Tuli, M. Xu, I. Singh, K. V. Singh, D. Lindsay, and S. Tuli, "Transformative effects of IoT, blockchain and artificial intelligence on cloud computing: Evolution, vision, trends and open challenges," *Internet Things*, vol. 8, pp. 100–118, Dec. 2019.

[29] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158–11168, Nov. 2019.

[30] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digit. Commun. Netw.*, vol. 5, no. 1, pp. 10–17, Feb. 2019.

[31] Y. Liu, Q. Cui, J. Zhang, Y. Chen, and Y. Hou, "An actor-critic deep reinforcement learning based computation offloading for three-tier mobile computing networks," in *Proc. 11th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Xi'an, China, Oct. 2019, pp. 1–6.

[32] H. Lu, C. Gu, F. Luo, W. Ding, and X. Liu, "Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning," *Future Gener. Comput. Syst.*, vol. 102, pp. 847–861, Jan. 2020.

[33] P. Gazori, D. Rahbari, and M. Nickray, "Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach," *Future Gener. Comput. Syst.*, vol. 110, pp. 1098–1115, Sep. 2020.

[34] R. Yadav, W. Zhang, K. Li, C. Liu, M. Shafiq, and N. K. Karn, "An adaptive heuristic for managing energy consumption and overloaded hosts in a cloud data center," *Wireless Netw.*, vol. 26, no. 3, pp. 1905–1919, Apr. 2020.

[35] R. Yadav, W. Zhang, O. Kaiwartya, P. R. Singh, I. A. Elgendy, and Y.-C. Tian, "Adaptive energy-aware algorithms for minimizing energy consumption and SLA violation in cloud computing," *IEEE Access*, vol. 6, pp. 55923–55936, 2018.

[36] R. Yadav and W. Zhang, "MeReg: Managing energy-SLA tradeoff for green mobile cloud computing," *Wireless Commun. Mobile Comput.*, vol. 2017, pp. 1–11, Dec. 2017.

[37] H. Li, R. Cai, N. Liu, X. Lin, and Y. Wang, "Deep reinforcement learning: Algorithm, applications, and ultra-low-power implementation," *Nano Commun. Netw.*, vol. 16, pp. 81–90, Jun. 2018.

[38] S. M. R. Nouri, H. Li, S. Venugopal, W. Guo, M. He, and W. Tian, "Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications," *Future Gener. Comput. Syst.*, vol. 94, pp. 765–780, May 2019.

[39] A. Xiao, Z. Lu, J. Li, J. Wu, and P. C. K. Hung, "SARA: Stably and quickly find optimal cloud configurations for heterogeneous big data workloads," *Appl. Soft Comput.*, vol. 85, Dec. 2019, Art. no. 105759.

[40] F. Carpio, A. Jukan, R. Sosa, and A. J. Ferrer, "Engineering a QoS provider mechanism for edge computing with deep reinforcement learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Waikoloa, HI, USA, Dec. 2019, pp. 1–6.

[41] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Emerg. Topics Comput.*, early access, Mar. 4, 2019, doi: 10.1109/TETC.2019.2902661.

[42] N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, Q. Qiu, J. Tang, and Y. Wang, "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Atlanta, GA, USA, Jun. 2017, pp. 372–382.

[43] M. Cheng, J. Li, and S. Nazarian, "DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *Proc. 23rd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jeju, South Korea, Jan. 2018, pp. 129–134.

[44] L. Quan, Z. Wang, and F. Ren, "A novel two-layered reinforcement learning for task offloading with tradeoff between physical machine utilization rate and delay," *Future Internet*, vol. 10, no. 7, pp. 10–60, 2018.

[45] M. Cheng, J. Li, P. Bogdan, and S. Nazarian, "H_2O-cloud: A resource and quality of service-aware task scheduling framework for warehouse-scale data centers," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2925–2937, Oct. 2020.

[46] H. Li, J. Li, W. Yao, S. Nazarian, X. Lin, and Y. Wang, "Fast and energy-aware resource provisioning and task scheduling for cloud systems," in *Proc. 18th Int. Symp. Qual. Electron. Design (ISQED)*, Santa Clara, CA, Mar. 2017, pp. 174–179.

[47] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. 2nd USENIX Conf. HotCloud*, Boston, MA, USA, 2010, p. 4.

[48] M. Pedram, "Energy-efficient datacenters," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 10, pp. 1465–1484, Oct. 2012.

JIANPENG LIN received the B.S. degree in computer science and technology from Foshan University, in 2016, and the M.S. degree in computer science from the Guangdong University of Technology (GDUT), China. His research interests include cloud computing, deep reinforcement learning, and multi-agent systems.

DELONG CUI received the M.S. degree in communication and information systems from Southwest Jiaotong University, in 2008. He is currently a Professor with the Guangdong University of Petrochemical Technology. He has published more than 20 articles in refereed journals and conference proceedings. His research interests include cloud computing and reinforcement learning.

ZHIPING PENG received the B.S. degree from the China University of Petroleum, Huadong, in 1996, the M.S. degree from the Huazhong University of Science and Technology, in 2001, and the Ph.D. degree in computer applications from the South China University of Technology, in 2007. He is currently a Professor with the Guangdong University of Petrochemical Technology. He has published more than 50 articles in refereed journals and conference proceedings. His research interests include cloud computing, machine learning, and multi-agent systems.

QIRUI LI received the M.S. degree in computer science and technology from the South China University of Technology, in 2008. He is currently pursuing the Ph.D. degree with Guangzhou University. He is currently a Lecturer with the Guangdong University of Petrochemical Technology. He has published more than ten articles in refereed journals and conference proceedings. His research interests include artificial intelligence, cloud computing, image processing, and pattern recognition.

JIEGUANG HE received the Ph.D. degree in mechanical engineering from the Guangdong University of Technology, in 2015. He is currently a Lecturer with the Guangdong University of Petrochemical Technology. He has published more than 20 articles in relevant journals and conferences. His research interests include cloud computing, intelligent optimization algorithms, and project scheduling.