International Journal of Information Technology and Web Engineering
Volume 17 • Issue 1

MHDNNL:
A Batch Task Optimization Scheduling Algorithm in Cloud Computing

Qirui Li, Guangdong University of Petrochemical Technology, China
Zhiping Peng, Jiangmen Polytechnic, China
Delong Cui, Guangdong University of Petrochemical Technology, China*
Jianpeng Lin, South China University of Technology, China
Jieguang He, Guangdong University of Petrochemical Technology, China

ABSTRACT

Task optimization scheduling is one of the key concerns of both cloud service providers (CSPs) and cloud users. The CSPs hope to reduce the energy consumption of executing tasks to save costs, while the users are more concerned about shorter task completion time. In cloud computing, multi-queue and multi-cluster (MQMC) is a common resource configuration mode, and batch is a common task commission mode. The task scheduling (TS) in these modes is a multi-objective optimization (MOO) problem, and it is difficult to get the optimal solution. Therefore, the authors proposed a MOO scheduling algorithm for this model based on multiple heterogeneous deep neural networks learning (MHDNNL). The proposed algorithm adopts a collaborative exploration mechanism to generate the samples and use the memory replay mechanism to train. Experimental results show that the proposed algorithm outperforms the benchmark algorithms in minimizing energy consumption and task latency.

KEYWORDS

Cloud Computing, Deep Learning, Deep Neural Network, Distributed, Heterogeneous, Resource Allocation, Server Cluster, Task Scheduling

INTRODUCTION

For more than 10 years since the concept was proposed, cloud computing has taken a huge leap forward and had drastic changes. Cloud computing is regarded as a revolution in the field of computer networks. Because of its emergence, social working methods and business models are also undergoing tremendous changes. Cloud computing platform has powerful computing and storage capacities, and can provide high-quality customized services. However, cloud computing is not a brand-new network technology, but a brand-new network application concept. In fact, cloud computing is the result of the mixed evolution of computer technologies such as distributed computing, utility computing, load balancing, parallel computing, network storage, and virtualization. The core concept of cloud computing is to provide fast and secure computing service and data storage on the network so that everyone can use the huge computing resources and data centers through Internet. Therefore, cloud computing is essentially a network that provides resources, and can be regarded as unlimited expansion.

DOI: 10.4018/IJITWE.310053 *Corresponding Author

Users can obtain resources from the "cloud" at any time, use them as needed, as long as they pay for it according to usage. CSPs build cloud computing resource pool in the form of data center and provide it to users in the form of virtual machine (VM). After a user rents VMs in the data center, she/he can submit her/his tasks to the VMs for processing via network. After the tasks are completed, the VMs return the results to the user via network too. Cloud users only need to install a simple client in local. During the service, the CSP and the user who want to use cloud computing resources sign a service level agreement (SLA) to agree on the quality of service (QoS). Under SLA condition, the CSP tries her/his best to provide better QoS and resources management, such as faster task response, less error probability, and lower data center energy consumption, etc., to attract more users and get better economic benefits. Task response time and other indicators that affect QoS are closely related to the cloud platform's TS strategy. So, the quality of TS strategy determines a cloud platform's service quality and business profit (Madni, Latiff, Coulibaly & Abdulhamid, 2017). However, the problem of cloud TS optimization has not yet been well solved, and it is currently one of urgent problems of cloud computing to be solved.

Due to the advancement and convenience of the cloud computing service model, more and more users abandon the software service model in which they manage physical servers themselves, begin to use the cloud computing service model and migrate their businesses to virtual servers in remote data center. The tasks generated by different businesses usually have different characteristics. For example, some are real-time, some are with a large amount of data, some can be processed offline, and some need to be processed online. Because of the diversity of user tasks and the huge number of users, cloud computing system has to deal with a large number of tasks and data, especially when batch tasks are submitted. Due to the large number, the tasks submitted by user must first enter task queues to wait for scheduling. In this case, if all tasks are submitted into a single queue, the tasks are prone to long queuing delay when the business bursts. Furthermore, to deploy tasks on a single VM is prone to overload the VM server, leading to slow task response. The service mode of single queue and single server will severely reduce the QoS of the cloud platform. Therefore, CSPs and users tend to use a service mode with MQMC, that is, different types of tasks are loaded into different queues for submission, and multiple VMs are clustered to execute the tasks. However, The TS in this MQMC mode will be more complicated. How to perform efficient TS and reasonable resource allocation under this mode has not been studied much.

The scheduling of cloud tasks is an NP-complete problem that has never been completely solved (Nabi, Lbrahim & Jimenez, 2021). When submitting tasks to multiple clusters in batches, the scheduling possibilities are exponential. For example, if there are M user task queues of length N and there are K computing clusters for task processing, the possibility of TS will be K^{M*N}. When the problem scales up (that is, M and K become larger), the traditional exact and approximate methods such as exhaustive method will consume a huge amount of calculation and a long calculation time, and will not be able to cope with it. Heuristic algorithms have been tried, but they require specific conditions to obtain the optimal solution. And their adaptability is not good in the complex, changeable cloud environment. They can easily fall into a local optimal solution for MOO problem. In recent years, artificial intelligence (AI) technologies such as deep learning (DL) have made great progress and have been successfully applied in many fields such as unmanned driving. DL is hot in the current research and application of AI. DL has strong feature perception capability, and it can deal with large-scale state spaces, and provides a new solution for TS optimization problems in complex cloud computing environment.

DL is to learn the internal laws and the indication hierarchies of sample data. The information obtained through learning is of great help to the interpretation of data such as text, image and sound. Its ultimate goal is to enable machines to analyze and learn like humans, and to recognize data such as text, images, and sound. DL is a complex machine learning algorithm that has achieved results in sound and image recognition far surpassing previous related technologies. However, DL is a supervised learning method and requires a large number of samples during training and learning. However, in

the dynamic and changeable cloud computing environment, the training samples are not easy to obtain. Therefore, this paper expands the traditional TS framework that uses a single deep network as the scheduler, and upgrades the scheduler to make it combined with multiple heterogeneous deep neural networks (DNNs). The cloud task optimization scheduling framework based on heterogeneous distributed DNNs is formed. The main idea of the framework is that in each iteration, multiple DNNs perform paralleled and independent learning, and they will generate different scheduling strategies for the same task set. We save the optimal strategy among these strategies for the current task set. The saved samples can be used for the training of all DNNs in the framework. This mechanism is similar to memory replay of deep reinforcement learning (DRL), but is simpler. With this mechanism, the framework can automatically generate samples for training of each DNN, and improve the effect of training while solving the problem of training samples.

We propose a MHDNNL algorithm for cloud TS problem of MQMC mode to minimize energy consumption and task. MHDNNL can generate an optimal scheduling strategy to improve user service quality and the service provider's revenue.

The main innovations and contributions of this article are as follows:

- We propose a novel cloud optimization scheduling model to solve the TS problem in the MQMC cloud computing scenario. The complex dynamic scheduling problem is converted to a static problem whose near-optimal solution is obtained by minimizing the task latency and energy consumption.
- The MHDNNL algorithm we proposed takes multiple DNNs which are heterogeneous as joint schedulers to make scheduling decisions together. Moreover, historical prior knowledge and experience replay back mechanism are adopted to accelerate the training convergence speed and improve the optimization effect.
- The effectiveness and performance advantages of the proposed algorithm are verified through extensive comparative experiments.

The remainder of this paper is organized as follows: The Literature Survey is introduced in Section 2. The system framework, including the various model components, problem formulation, and MHDNNL algorithm, are introduced in Section 3. The simulation experiments and their results' analysis are provided in Section 4. The summary and the future prospects are provided in Section 5.

RELATED WORK

In cloud computing, the TS optimization problem has been studied by many researchers and institutions. To maximize TS performance and minimize unreasonable task allocation in clouds, (Zhang & Zhou, 2018) proposed a TS method based on two-stage strategy. At the first stage, they classified the task with a Bayes classifier and precreated VMs according to historical scheduling data to minimize the waiting time of creating VMs. At the second stage, they proposed a dynamic TS algorithm based on the dynamical tasks match with concrete VMs. Similarly, (Xiong, Huang, Wu, She & Jiang, 2019) proposed a Johnson's-Rule-Based genetic algorithm (JRGA) for two-stage-TS problem in data centers of cloud computing. The JRCA takes account the characteristics of multiprocessor scheduling, uses new crossover and mutation operations to make the algorithm converge more quickly, and applies the Johnson's rule to optimize the makepsan of tasks. Although the two-stage TS strategies can effectively improve the cloud's scheduling performance, their models use a single queue submission for user tasks, not multiple queues submission.

Computation cost is an important issue of TS and needs to be considered in cloud computing. (Zuo, Shu, Dong, Zhu & Hara, 2017) proposed a resource-cost model which reflects the relationship between the cost of their budget and the cost of user resources in cloud computing. Based on this model and improved ant colony algorithm, they proposed an optimization scheduling method to

achieve MOO of system performance. (Al-Maytami, Fan, Hussain, Baker & Liatsis, 2019) proposed a TS algorithm with improved makespan based on prediction of tasks computation time (PTCT) algorithm for cloud computing. They tackled the problem of static scheduling of a single application in a widely distributed and heterogeneous environment. To predict the ideal algorithm for incoming/available data as and when needed, they performed a systematic analysis of techniques for resource utilization by means of Principal Components Analysis (PCA) in the cloud environment. The PTCT algorithm improves the performance of TS, while reducing computation cost. However, the system models of the above algorithms are different from ours.

In green infrastructure-as-a-service clouds (GICs), cost is still one of the most concerned issues for CSPs. The arrival of irregular tasks forces the private GIC cloud to migrate some tasks to the public clouds for processing, thus forming a hybrid cloud. However, the VM running prices are temporal differences in public clouds, so it difficult to schedule all tasks in a cost-effective way subject to the constraints of users' specified response time. In order to solve this problem, (Yuan, Bi & Zhou, 2019) gave the mathematical relations between rejected tasks and service rates of servers in private GIC, designed and implemented a novel meta-heuristic optimization approach by combining simulated annealing, particle swarm optimization (PSO), with genetic algorithm (GA). The approach investigates such temporal differences in hybrid GICs (HGICs). The approach is also multiple queues scheduling, and its goal is to maximize the profit and throughput with meeting the constraints of user tasks' response time, however, it doesn't optimize user tasks' response time.

The quality of TS performance directly affects the user gratifications and CSP benefits. TS need to perform MOO to meet the needs of both supply and demand of cloud service. To improve resource utilization and task response time, (Nabi, Ibrahim & Jimenez, 2021) proposed a dynamic and resource aware load balanced algorithm (DRALBA) for TS in cloud data center. The DRALBA maps a set of independent, non-preemptive, and computationally expensive tasks on the available resources in a load-balanced way, calculates the computation share for each VM based on a set of tasks and then select a VM (with maximum computation share for assigning larger size task) whose length is less than or equal to the computation share of the selected VM, and updates the VM's load and computation share at run time. Moreover, DRALBA cannot support multiple requested queues of user tasks.

For more goals of TS optimization, (Geng, Wu, Wang & Cai, 2020) established MOO TS model, whose optimization indicators include response time, total costs, resource utilization, and balancing load. They designed a hybrid angle strategy to solve this model. The hybrid angle is combined individual-to-individual angle with individual-to-reference point angle. They proposed one by one elimination strategy to remain individuals with better performance. Based on the hybrid angles, they designed and implemented a many-objective optimization algorithm (MaOEA-HA) for TS in cloud computing. The proposed algorithm achieves the best performance on the DTLZ test suite, comparing with NSGAIII, GrEA, KnEA, VaEA and Two-Arch2. (Ma, Wang, Gu, Meng & Wu, 2021) proposed a multi-objective knowledge-driven evolutionary algorithm (MGR-NSGA-III) for microservice deployment and startup strategy problem in different data centers. The MGR-NSGA-III takes completeness of service, total amount of storage resources, and total number of microservices as the constraints, considers the computation and storage resource utilization rate, load balancing rate, and actual microservice usage rate in resource service centers, and gains a better performance than the traditional microservice instance deployment and startup strategy does. However, the MaOEA-HA does not support MQMC scheduling in cloud computing.

Some researches tried to use heuristic algorithm to solve TS in cloud computing. Aiming at the performance of PSO algorithm decreases with the increase of the number of tasks, (Saleh, Nashaat, Saber & Harb, 2018) proposed an improved PSO (IPSO) algorithm. The IPSO achieves the optimal scheduling for a large number of tasks by splitting them into batches dynamically. When creating a batch, the IPSO considers the resources utilization states and gets the optimal solution. (Duan, Chen, Min & Wu, 2017) proposed PreAntPolicy, a VM scheduling method based on improved ant colony optimization (ACO) algorithm and a prediction model which assists the scheduler to make rational

schedules by predicting load trends. (Srichandan, Kumar & Bibhudatta, 2018) designed a TS algorithm that combines the ideal characteristics of a GA with bacterial foraging (BF) algorithm. The algorithm can perform efficient scheduling while adhering to the service quality defined in the service level agreement (SLA). However, the optimal solution such as PSO, ACO and GA can only be obtained under the specific conditions required by traditional heuristic algorithms, so the algorithm has poor generality in a complex and variable cloud environment. Also, it easily falls into local optima and fails to obtain global optimal solution of MOO problems.

Some researchers began to use metaheuristic instead of earlier heuristic for TS in cloud environment. Based on non-dominated sorting genetic algorithm (NSGA)-II, (Alsadie, 2021) proposed a metaheuristic framework for dynamic VM allocation in data center. This framework was named MDVMA. The MDVMA is to optimize TS in cloud computing and its goals not only include minimizing energy consumption, makespan and cost, but also provide trade-off the CSPs according to their requirements. To improve the TS performance with resource constraints, (Chen et al. 2020) proposed a latest metaheuristics whale optimization algorithm (WOA) for cloud TS with a MOO model. To further improve the optimal solution search capability of the WOA-based approach, they proposed an improved WOA for cloud TS (IWC). Compared with current metaheuristic algorithms, the IWC achieve better convergence speech, better accuracy in searching for the optimal TS strategy, and higher resource utilization in the presence of both small and large-scale tasks. However, the MDVMA and IWC do not support MQMC scheduling in cloud computing either.

With the development of machine learning technologies such as DL and RL, some scholars used machine learning to do TS in cloud computing. In terms of TS using DL, (Rangra, Sehgal & Shukla, 2019) proposed a scheduling approach with multi-tasking convolution neural network (M-CNN) to solve the contradiction between running time and running cost of TS in cloud computing. The M-CNN can make a effective decision for TS in cloud computing. They performed experiments on tweets dataset and workflow dataset. The experiment results show that the M-CNN can great reduce the time and cost of scheduling, compared with PSO and PSO-GA. Forecasting resource usage of VM is importance when we use prediction method to deal with TS in cloud computing. But it is difficult to accurately predict the future workload because the resources requested in data center are dynamic. (Hasan Shuvo, Shahriar Maswood & Alharbi, 2020) proposed a novel method whose name is LSRU to improve the accuracy of prediction. LSRU is a hybrid-method, which is combined Gated Recurrent Unit (GRU) with Long Short Term Memory (LSTM). LSRU can not only perform short-time ahead workload prediction but also perform long-time ahead workload prediction, especially sudden workload prediction. However, because of the different scheduling ideas and system models, the above methods are not suitable for the MQMC TS in cloud computing.

RL is a machine learning algorithm with strong decision-making ability. It explores the optimal solution through a constant trial-and-error mechanism. RL is an effective method to solve multi-constrained MOO problems. Some researchers try to deal with the TS problems based on RL in cloud computing (Coello, Brambila, Gamboa, Ma & Gómez, 2019). (Peng, Cui, Zuo & Lin, 2015) solved TS problems by combining RL with queue theory in complex cloud environment and obtained the optimal scheduling strategy by the trial-and-error mechanism of RL. With the concepts of segmentation service level agreement (SSLA) and utilization unit time cost (UUTC), they viewed the TS and resource provisioning problem in data center as a Markov decision process. To solve it they proposed novel resource management scheme with RL and queuing theory. Furthermore, to solve the problem of accurate scaled cloud computing environment and the problem of efficient TS subject to resource constraints, (Peng et al., 2015) proposed a fine-grained cloud computing system model and optimization TS algorithm with RL and queuing theory. They divided the system model into TS submodel, task execute submodel and task transmission submodel, so as to accurately analyze the task processing of user requests. The algorithm they proposed not only improves the efficiency of TS, but also reveals the relationship between the arrival rate, server rate, number of VMs and the number of buffer size. Similarly, (Thein, Myo, Parvin & Gawanmeh, 2018) realized the energy-saving resource

allocation of cloud data center based on RL, and achieved the purpose of high energy efficiency and SLA guarantee. (Wei et al. 2019) proposed an adaptive cloud computing resource lease plan generating approach based on RL, which can help Software-as-a-Service (SaaS) CSP to make effective facility adjustment strategies dynamically. Although cloud TS algorithms based on RL can obtain the optimal strategy through constant trial-and-error mechanism, when the scale of state space becomes large, the RL may not converge or converge slowly. DNN has strong perception capabilities, and can deal with large-scale state spaces effectively and overcome the shortcomings of RL.

Recent breakthroughs have occurred in deep reinforcement learning (DRL) in many fields, such as natural language processing, games (Volodymyr et al., 2019), robotic control, and so on. Some researchers have adopted a DRL model to solve TS problems in cloud computing. (Bitsakos, Konstantinou, & Koziris, 2018) proposed a flexible TS system based on DRL for resource scheduling in large-scale clusters. The system can automatically allocate computing resources according to users' fluctuating workload demands and follow an optimal resource management strategy. (Huang, Feng, Zhang, Qian & Wu, 2019) combined the reinforcement learning training method with distributed deep learning model to solve the task offloading problem of mobile edge computing to reduce energy consumption and guarantee quality of service (QoS). To schedule large-scale workloads atomically in cloud data center, (Rjoub, Bentahar, Wahab & Bataineh, 2020) developed four DRL-based scheduling algorithms and achieved good scheduling results. Most of the above studies based on DRL can solve various cloud computing scheduling problems and achieve good performance, and are very suitable for dynamic TS. However, they are too complex and requires a lot of training, and are not suitable for static TS with batch submission.

Based on the above studies, we propose a cloud TS optimization scheduling algorithm based on MHDNNL for MOO problem. MHDNNL proposed a collaborative exploration mechanism to generate the training samples, and drew on the memory playback mechanism of DRL to train. MHDNNL takes minimizing task latency and energy consumption as the system optimization goal, and generate an optimal scheduling strategy to improve QoS of users and revenue of CSPs. The MHDNNL algorithm we proposed can effectively solve the TS problem of MQMC mode in cloud computing.

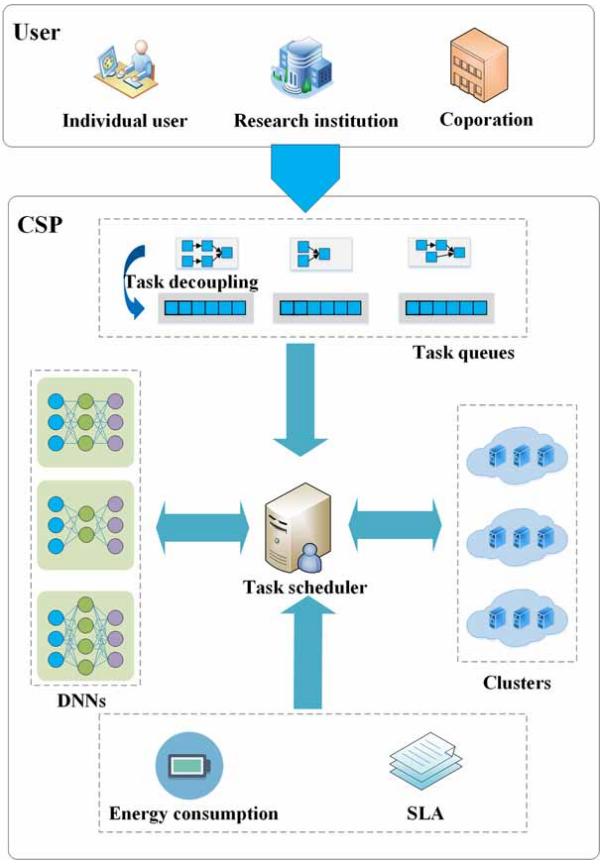
SYSTEM FRAMEWORK AND PROBLEM DESCRIPTION

Main System Framework

Since cloud users get personalized cloud services from CSP by paying on demand, they then submit their tasks to cloud service system through network, and CSP provides virtual resources that meet user needs to deploy tasks. The system framework is shown in Figure 1.

There is diversity in user workload (Gao, Wang, Gupta & Pedram, 2013), which contains multiple tasks with dependencies and data transmission. The task decoupler of this framework should ensure the order and dependencies when the tasks are scheduled. The data center is composed of a large number of physical machine servers, and the scale is very large. In data center, the neighboring servers can be clustered into computing clusters according to user needs (Li, Ji, Wang, Nazarian & Wang, 2017). In clusters or between clusters, because communication is usually carried out through high-speed optical fibers, the transmission speed is very fast. Therefore, this part of transmission latency and energy consumption can be ignored. However, the communications between users and clusters are usually long-distance transmission, and there are no dedicated high-speed network channels. At a certain scheduling timestep, the bandwidth between each user and different cluster is different, and the computing powers of different clusters allocated to each user are also different. Therefore, the communicational latency and energy consumption of this part are important factors to be considered in the optimization problem. In addition, because computing ability and computing power of each cluster are different, they are also key factors that affect the efficiency of TS.

Figure 1. System framework



Problem Formulation

The two key factors of the problem we are focusing on are task latency and energy consumption. They are related to the communication between task queues and clusters and the execution on the clusters. Below, we will clarify the definitions of the communicational sub-model and computational sub-model used in this paper.

The task of cloud system is to schedule atomic tasks in multiple queues and multiple clusters. We assume that the number of waiting task queues in the system is N, represented as $\{Q_1, Q_2, \ldots, Q_N\}$, and each queue contains M tasks, denoted as $\{T_1, T_2, \ldots, T_M\}$, the total number of tasks is $M * N$, and there are K clusters, denoted as $\{Clu_1, Clu_2, \ldots, Clu_K\}$. Task T_{nm} represents the m-th task in the n-th queue, and the attribute of task T_{nm} is represented by a binary group $(r_{nm}^{cpu}, r_{nm}^{data})$, where r_{nm}^{cpu} indicates the number of required CPU cycles, and r_{nm}^{data} indicates amount of data of the T_{nm}. In addition, we assume that the required number of CPU cycles for each task is linearly related to the amount of task data (Miettinen & Nurminen, 2010): $r_{nm}^{cpu} = \mu * r_{nm}^{data}$, where μ represents the computation-to-data ratio. The attributes of cluster Clu_k are represented by a triplet, $(CP_k, P_k^{comm}, P_k^{comp})$, where CP_k represents the computing power of the Clu_k, that is, the number of cycles of the CPU. P_k^{comm} represents the communication power consumption of

the Clu_k and P_k^{comp} represents the computing power consumption of the Clu_k. The action is $a_{nmk} \in \{0,1\}$, $1 \le n \le N$, $1 \le m \le M$, $1 \le k \le K$. If $a_{nmk} = 1$, then the T_{nm} is scheduled to the Clu_k. In addition, the bandwidth between multiple queues to multiple clusters is represented by $\{BW_{11}, BW_{12}, \ldots, BW_{nk}\}$, and BW_{nk} represents the bandwidth from the Q_n to the Clu_k.

1. Communicational sub-model

The communicational sub-model includes the time required for the transmission of the task data and the energy consumption. When multiple tasks in the same queue are simultaneously scheduled to the same cluster, the bandwidth is distributed equally to each task, so the bandwidth occupied by T_{nm} is

$$R_{nm}^{bw} = \frac{BW_{nk}}{\mathrm{A}_{nk}}, \tag{1}$$

where BW_{nk} is the bandwidth from the Q_n to Clu_k, and A_{nk} is the number of tasks scheduled from the Q_n to Clu_k. The communicational latency is the time it takes to upload the task data to the server,

$$TD_{nm}^{comm} = \frac{\alpha_{nm}}{R_{nm}^{bw}}, \tag{2}$$

where α_{nm} is the amount of data of T_{nm}.

The communicational energy consumption is the energy consumed during the transmission task,

$$EC_{nm}^{comm} = P_k^{comm} \cdot TD_{nm}^{comm}. \tag{3}$$

So, the communicational energy consumption for all tasks in the Q_n is,

$$EC_n^{comm} = \sum_{m \in M} EC_{nm}^{comm}. \tag{4}$$

2. Computational sub-model

The computational sub-model contains the computational latency of the task and the computational energy consumption. The cluster computing power will be distributed equally to the tasks scheduled to the cluster, i.e., each task gets the following CPU cycles:

$$R_{nm}^{cpu} = \frac{CP_k}{\sum_{n \in N} \sum_{m \in M} a_{nmk}}. \tag{5}$$

The computational latency is the time consumed by the calculation of the task,

$$TD_{nm}^{comp} = \frac{r_{nm}^{cpu}}{R_{nm}^{cpu}}. \tag{6}$$

The computational energy consumption is the energy consumed in the calculation process of the task,

$$EC_{nm}^{comp} = P_{k}^{comp} \cdot TD_{nm}^{comp}. \tag{7}$$

The computational energy consumption for all tasks in the Q_n is,

$$EC_{n}^{comp} = \sum_{m \in M} EC_{nm}^{comp}. \tag{8}$$

3. Total cost

Since the factors considered in this paper are task latency and energy consumption, the total cost of scheduling decision *d* under system state *s* can be calculated as,

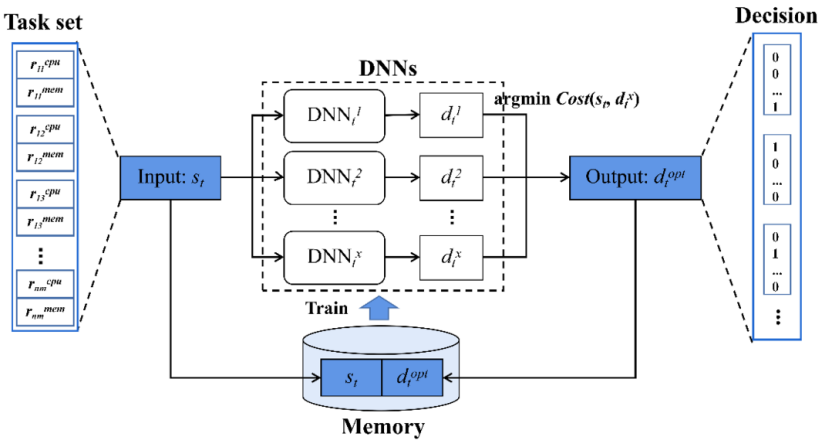
$$\mathrm{Cos}\, t(s,d) = \xi^{delay} \cdot \max_{n \in N, m \in M} (TD_{nm}^{comm} + TD_{nm}^{comp}) + \xi^{ec} \cdot \sum_{n \in N} (EC_{n}^{comm} + EC_{n}^{comp}), \tag{9}$$

where $\xi^{delay}, \xi^{ec} \in [0, \ 1]$ respectively indicate the fraction of optimization for task latency and energy consumption. The ultimate goal of the system is to obtain an optimal scheduling strategy to minimize task latency and energy consumption, i.e., to minimize the cost.

MHDNNL Model

The MHDNNL model as shown in Figure 2 uses multiple heterogeneous DNNs as joint schedulers to make scheduling decisions together.

Figure 2. MHDNNL model



The model has two key techniques. The first is to evaluate the scheduling decisions of multiple DNN outputs based on the prior knowledge of the system, so as to obtain the optimal decision that meets the optimization objectives. The second is the memory replay mechanism based on DRL [20]. After each scheduling, we combine the optimal scheduling decision generated by DNNs with the task set as a training sample and save it into the Memory. This approach can increase the number and diversity of samples. In addition, we select a mini-batch of samples from Memory randomly to train the network model and guide multiple agents to explore the optimal scheduling strategy. This model can not only enhance the agent's ability to explore the optimal strategy, but also improve the utilization of the training samples.

In training, the state space s_t is composed of multiple task properties in multiple queues, expressed as $\{r_{11}^{cpu}, r_{11}^{data}, r_{12}^{cpu}, r_{12}^{data}, \ldots, r_{nm}^{cpu}, r_{nm}^{data}\}$, as the X DNNs' input. Each DNN outputs different decision, expressed as $(d_t^1, d_t^2, \ldots, d_t^x)$. With time step t and taking s_t as an input, each DNN's output d_t^x is expressed as $f_{\theta_t^x} : s_t \rightarrow d_t^x$, where $f_{\theta_t^x}$ and θ_t^x represents the function and the parameters of of the x-th DNN, respectively. Decision d_t^x consists of a binary sequence, expressed as $d_t^x = \{a_{111}, a_{121}, a_{nmk}\}$. Once the scheduling decision of x-th DNN is determined, the cost of the decision is calculated with Equation (9). Among the decisions generated by each DNN, the decision with the minimum cost is selected as the optimal action decision for the current task set, expressed as d_t^{opt},

$$d_t^{opt} = \arg\min_{x \in X} \mathrm{Cos}\, t(s_t, d_t^x), \tag{10}$$

where (s_t, d_t^x) is stored in the Memory as the training sample. When the quantity of Memory's samples reaches the preset threshold, we select a mini-batch number of samples from Memory randomly for each DNN to train. By minimizing the cross-entropy loss, we optimized θ_t^x with the gradient descent method,

$$L(\theta_t^x) = -d_t^T \log f_{\theta^x}(s_t) - (1 - d_t)^T \log(1 - f_{\theta^x}(s_t)). \tag{11}$$

The optimization scheduling algorithm based on MHDNNL is shown as Algorithm 1.

Algorithm 1 pseudo-code of TS algorithm based on MHDNNL

1. Initialize all X DNNs with different random wights θ^x, $x\hat{I}X$.
2. Initialize replay memory D to capacity M.
3. **Input:** all task requirements in task ready queues.
4. **Output**: task scheduling decisions d^x.
5. **For** $t = 1, 2, .., T$ **do**
6. Input the same s_t to each DNN.
7. Generate X scheduling decisions from the DNNs $\{d_t^x\} = f_{\theta_t^x}(s_t)$.
8. Select the optimal decision $d_t^{opt} = \arg\min_{x \in X} Q(s_t, d_t^x)$.
9. Store (s_t, d_t^{opt}) into replay memory D.
10. **If** the number of samples exceeds the threshold:
11. Update DNNs' parameters with the stochastic gradient descent method
12. **End If**
13. **End For**

EXPERIMENT RESULTS

Experimental Design and Parameter Description

We designed a two-part simulation experiment to verify the validity and performance of the proposed MHDNNL algorithm. The first part is to verify the convergence of the MHDNNL algorithm with different numbers of queue and clusters. The second part is to verify the optimization results of MHDNNL algorithm and compare them with the benchmark algorithms, such as the Random, Round-Robin (RR), and multi-objective particle swarm optimization (MoPSO). In the experiments, ξ^{delay} and ξ^{ec} of equation (9) were set to 0.9 and 0.1, respectively. The number of queues was set from 3 to 12, the number of tasks in the queue was 4, the data of the task was set from 100MB to 500MB, and the computation-to-data ratio of different types of tasks were 330 cycles/byte, 1300 cycles/byte, 1900 cycles/byte, 2100 cycles/byte, respectively. The number of clusters was set from 3 to 12, the computing power of the cluster was set from 1.5_*10^{15} cycles/s to 10.0_*10^{15} cycles/s randomly, the computing power of the cluster was set from 1.0_*10^5 W to 10.5_*10^5 W according the computing power, the bandwidth between the queue and the cluster was set from 250MB to 900Mbps, and the communication power was set from 0.2W to 0.8W.

We took 1000 sets of tasks as the training dataset, and 100 sets of tasks as the test dataset. We designed 8 heterogeneous DNN as decision generators, each with 1 input layer, 3 hidden layers, and 1 output layer. The numbers of neurons in the first hidden layer, second hidden layer and the third hidden layer were randomly set from 100 to 200, from 30 to 50, and to 10, respectively.

The simulation experiment platform was developed base on Python3.6.2 and TensorFlow1.2.1. We carried out the experiment on a Windows 10 operating system, with an Intel core i7-8550U dual-core CPU at 1.80 GHz and 16 GB memory.

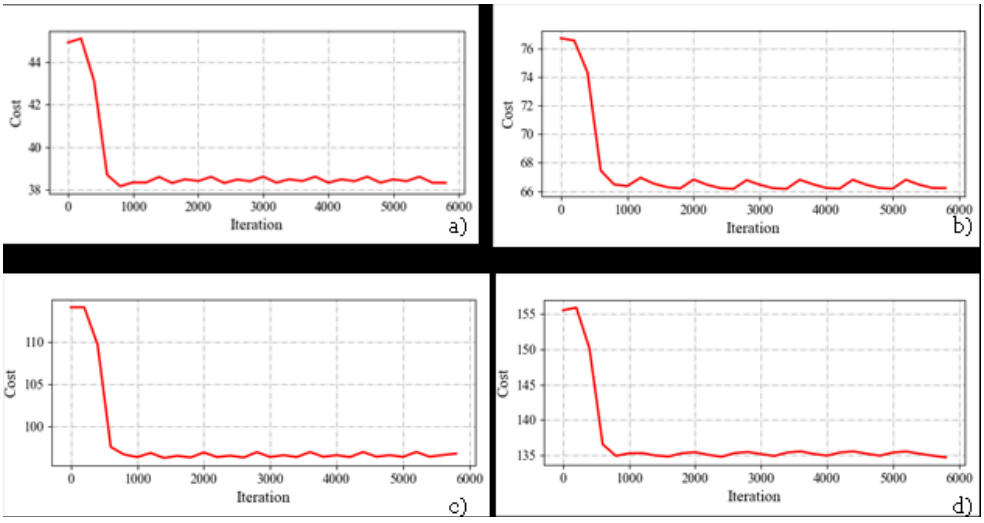
Network Model Verification

The convergence of the model with different hyperparameters have been verified in (Cui et al., 2020). Then we experimentally verified the convergence of the model with different environment parameters, such as different numbers of queues (QN) and different number of clusters (CN).

1. Convergence under different QN and a fixed CN

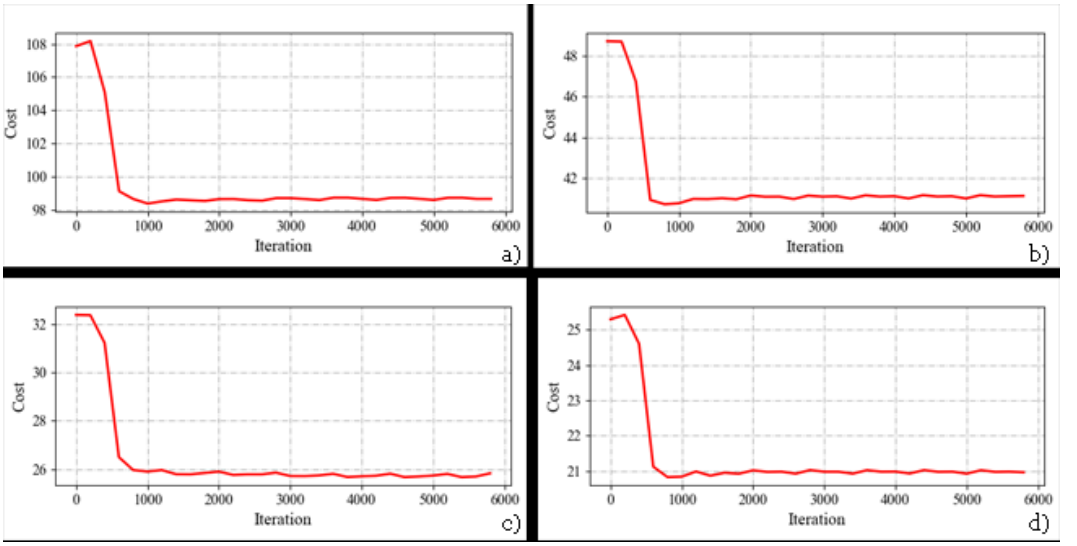
Firstly, we examined the convergence of the MHDNNL algorithm under different numbers of queues and a fixed number of clusters. In this experiment, the CN was to 5, and the QN was set to 4, 6, 8, and 10, respectively. The results of this experiment are shown in Figure 3.

Figure 3. Convergence of MHDNNL algorithm versus QN with CN=5. (a) QN=4; (b) QN=6; (c) QN=8; (d) QN=10



It can be seen from Figure 3 that the proposed MHDNNL algorithm tends to converge eventually, and the cost increases with the increase of QN. In this experiment, the CN is fixed to 5. An increase in QN means an increase in the number of tasks, so that the competition for cloud resources (such as CPU, bandwidth, etc.) by tasks becomes more intense. Eventually, the cloud resources allocated to each task will be reduced, leading to an increase in cost.

Figure 4. Convergence of MHDNNL algorithm versus CN with QN=5. (a) CN=3;(b) CN=6; (c) CN=9; (d) CN=12



2. Convergence under different CN and a fixed QN

Then, we examined the convergence of the MHDNNL algorithm under different numbers of clusters and the same number of queues. In this experiment, the QN was to 5, and the CN was set to 3, 6, 9, and 12, respectively. The results of this experiment are shown in Figure 4.

It can also be seen from Figure 4 that the proposed MHDNNL algorithm tends to converge eventually, but the cost decreases with the increase of CN. In this experiment, the QN is fixed to 5, which means the tasks is fixed, too. An increase in CN means an increase of cloud resources (such as CPU, bandwidth, etc.). Eventually, the cloud resources allocated to each task will be increased, leading to an reduce in cost.

Algorithm Comparison

We compared the optimization scheduling effect of MHDNNL algorithm with several benchmark algorithms versus QN and CN. The benchmark algorithms include Random, Round-Robin (RR), and MoPSO.

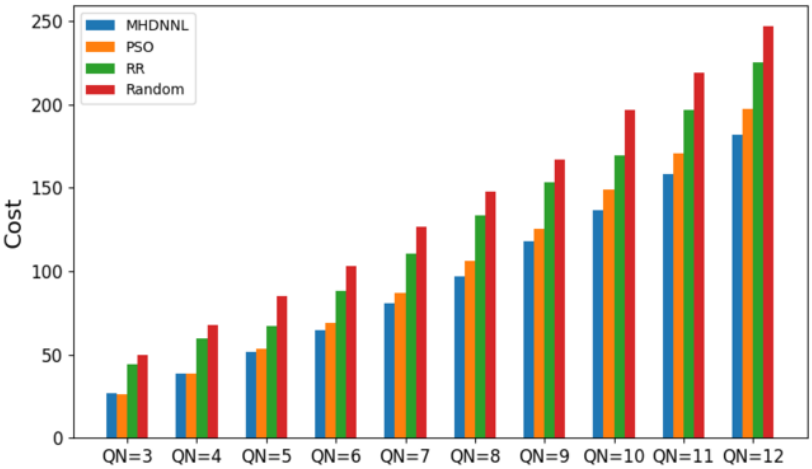
1. Performance comparison with different QN and a fixed CN

Firstly, we compared the proposed MHDNNL algorithm with benchmark algorithms for different QN with a fixed CN. In the experiments, the CN was fixed at 5, and the QN was set from 3 to 12. The results of this experiment are shown in Figure 5.

It can be seen from Figure 5 that the cost of all algorithm increases with the increase of QN. In this experiment, the proposed MHDNNL algorithm performed better than RR and random algorithms.

When the QN is small, the optimization results of MHDNNL algorithm are close to those of MoPSO algorithm. However, when the QN is 5 or more, the costs of the MHDNNL algorithm are better than those of MoPSO algorithm. The CN is fixed means the resources are limited. As the QN increases, the task's competition for resources will become more intense. In this case, the MHDNNL algorithm shows better TS performance than the MoPSO algorithm.

Figure 5. Algorithm performance comparison versus QN with CN=5

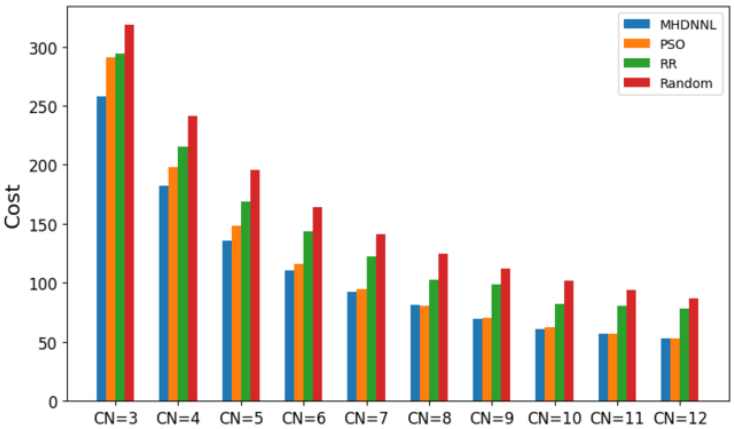


2. Performance comparison with different CN and a fixed QN

Then, we compared the proposed MHDNNL algorithm with benchmark algorithms for different CN with a fixed QN. In the experiments, the QN was fixed at 10, and the CN was set from 3 to 12. The results of this experiment are shown in Figure 6.

It can be seen Figure 6 that the cost of all algorithm decreases with the increase of CN. In this experiment, the proposed MHDNNL algorithm also performed better than RR and random algorithms.

Figure 6. Algorithm performance comparison versus CN with QN=10



When the CN is small, the optimization results of MHDNNL algorithm are better than those of MoPSO algorithm. However, when the CN is 9 or more, the costs of the MHDNNL algorithm are close to those of MoPSO algorithm. Similarly, the QN is fixed means the tasks is fixed. As the CN increases, the resources will be more abundant, so the task's competition for resources will become less intense. In summary, in an environment with fierce resource competition, the MHDNNL algorithm for TS can perform better than the benchmark algorithms.

CONCLUSION

Based on MHDNNL, we proposed a cloud TS optimization scheduling algorithm which solves the MOO problem of MQMC scheduling in cloud computing well. The proposed TS model is trained by cooperating with multiple heterogeneous DNNs and by drawing on the memory replay mechanism of DRL, effectively improves the convergence speed and results, and generates a optimal scheduling strategy to minimize the system energy consumption and task latency. The proposed can better adapt to large-scale MOO problems, and the optimization results are better than heuristic algorithms such as MoPSO.

Although we have done some research on the TS problem of MQMC scheduling in cloud computing, and achieved some research results, there are still some problems for further research and improvement, including the following issues: (1) The MHDNNL algorithm we proposed is with a fixed number of clusters. It should be able to effectively predict the user workloads and realize the dynamic adjustment of the number of clusters. (2) The scale of cloud data centers becomes more and more large, the single scheduling model has limited performance; multi-model collaborative scheduling will therefore be a fruitful direction for future research.

ACKNOWLEDGMENT

This research was supported by National Natural Science Foundation of China, Guangdong basic and applied basic research foundation, Maoming Science and Technology Project. Zhiping Peng and Delong Cui are corresponding authors.

CONFLICT OF INTEREST

The authors of this publication declare there is no conflict of interest.

FUNDING AGENCY

This work was sponsored by National Natural Science Foundation of China [61672174, 61772145]; Guangdong basic and applied basic research foundation [2020A1515010727, 2021A1515012252]; Maoming Science and Technology Project [mmkj2020008].

REFERENCES

Al-Maytami, B. A., Fan, P., Hussain, A., Baker, T., & Liatsis, P. (2019). A Task Scheduling Algorithm With Improved Makespan Based on Prediction of Tasks Computation Time algorithm for Cloud Computing. *IEEE Access: Practical Innovations, Open Solutions*, *7*, 160916–160926. doi:10.1109/ACCESS.2019.2948704

Alsadie, D. (2021). A Metaheuristic Framework for Dynamic Virtual Machine Allocation With Optimized Task Scheduling in Cloud Data Centers. *IEEE Access: Practical Innovations, Open Solutions*, *9*, 74218–74233. doi:10.1109/ACCESS.2021.3077901

Bitsakos, C., Konstantinou, I., & Koziris, N. (2018). DERP: A deep reinforcement learning cloud system for elastic resource provisioning. In *Proceeding of IEEE International Conference on Cloud Computing Technology and Science* (pp. 21–29). IEEE. doi:10.1109/CloudCom2018.2018.00020

Chen, X., Cheng, L., Liu, C., Liu, Q., Liu, J., Mao, Y., & Murphy, J. (2020). A WOA-based optimization approach for task scheduling in cloud computing systems. *IEEE Systems Journal*, *14*(3), 3117–3128. doi:10.1109/JSYST.2019.2960088

Cheng, M., Ji, L., & Nazarian, S. (2018). DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers. In *Proceedings of 2018 23rd Asia and South Pacific Design Automation Conference* (pp. 129–134). IEEE. doi:10.1109/ASPDAC.2018.8297294

Coello, C., Brambila, S. G., Gamboa, J. F., Ma, G., & Gómez, R. H. (2019). Evolutionary multiobjective optimization: Open research areas and some challenges lying ahead. *Complex & Intelligent Systems*, *6*(1), 221–236. doi:10.1007/s40747-019-0113-4

Cui, D., Lin, J., Peng, Z., Li, Q., He, J., Yuan, Y., & Guo, M. (2020). Cloud Workflow Task and Virtualized Resource Collaborative Adaptive Scheduling Algorithm Based on Distributed Deep Learning. In *Proceeding of 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications*. IEEE. doi:10.1109/AEECA49918.2020.9213622

Duan, H., Chen, C., Min, G., & Wu, Y. (2017). Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems. *Future Generation Computer Systems*, *74*, 142–150. doi:10.1016/j.future.2016.02.016

Gao, Y., Wang, Y., Gupta, S. K., & Pedram, M. (2013). An energy and deadline aware resource provisioning, scheduling and optimization framework for cloud systems. In *Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis* (pp. 1-10). IEEE. doi:10.1109/CODES-ISSS.2013.6659018

Geng, S., Wu, D., Wang, P., & Cai, X. (2020). Many-Objective Cloud Task Scheduling. *IEEE Access: Practical Innovations, Open Solutions*, *8*, 79079–79088. doi:10.1109/ACCESS.2020.2990500

Hasan Shuvo, M. N., Shahriar Maswood, M. M., & Alharbi, A. G. (2020). LSRU: a novel deep learning based hybrid method to predict the workload of virtual machines in cloud data center. In *Proceedings of 2020 IEEE Region 10 Symposium* (pp. 1604–1607). doi:10.1109/TENSYMP50017.2020.9230799

Huang, L., Feng, X., Zhang, L., Qian, L., & Wu, Y. (2019). Multi-server multi-user multi-task computation offloading for mobile edge computing networks. *Sensors (Basel)*, *19*(6), 1446. doi:10.3390/s19061446 PMID:30909657

Li, H., Ji, L., Wang, Y., Nazarian, S., & Wang, Y. (2017). Fast and energy-aware resource provisioning and task scheduling for cloud systems. In *Proceedings of the 18th International Symposium on Quality Electronic Design* (pp. 174–179). IEEE. doi:10.1109/ISQED.2017.7918312

Ma, W., Wang, R., Gu, Y., Meng, Q., & Wu, Y. (2021). Multi-objective microservice deployment optimization via a knowledge-driven evolutionary algorithm. *Complex & Intelligent Systems*, *7*(3), 1153–1171. doi:10.1007/s40747-020-00180-1

Madni, S. H. H., Latiff, M. S. A., Coulibaly, Y., & Abdulhamid, S. M. (2017). Recent advancements in resource allocation techniques for cloud computing environment: A systematic review. *Cluster Computing*, *20*(3), 2489–2533. doi:10.1007/s10586-016-0684-4

Miettinen, A. P., & Nurminen, J. K. (2010). Energy efficiency of mobile clients in cloud computing. In *Proceedings of the 2nd USENIX Conference HotCloud* (pp. 22–25), USENIX.

Nabi, S., Lbrahim, M., & Jimenez, J. M. (2021). DRALBA: Dynamic and Resource Aware Load Balanced Scheduling Approach for Cloud Computing. *IEEE Access: Practical Innovations, Open Solutions*, *9*, 61283–61297. doi:10.1109/ACCESS.2021.3074145

Peng, Z., Cui, D., Zuo, J., Li, Q., Xu, B., & Lin, W. (2015). Random task scheduling scheme based on reinforcement learning in cloud computing. *Cluster Computing*, *18*(4), 1595–1607. doi:10.1007/s10586-015-0484-2

Peng, Z., Cui, D., Zuo, J., & Lin, W. (2015). Research on cloud computing resources provisioning based on reinforcement learning. *Mathematical Problems in Engineering*, *916418*, 1–12. Advance online publication. doi:10.1155/2015/916418

Rangra, A., Sehgal, V.K., & Shukla, S. (2019). A Novel Approach of Cloud Based Scheduling Using Deep-Learning Approach in E-Commerce Domain. *International Journal of Information System Modeling and Design, 10*(3), 59–75. 10.4018/IJISMD.2019070104

Rjoub, G., Bentahar, J., Wahab, O. A., & Bataineh, A. S. (2020). Deep and reinforcement learning for automated task scheduling in large:cale cloud computing systems. *Concurrency and Computation*. Advance online publication. doi:10.1002/cpe.5919

Saleh, H., Nashaat, H., Saber, W., & Harb, H. M. (2018). IPSO Task Scheduling Algorithm for Large Scale Data in Cloud Computing Environment. *IEEE Access: Practical Innovations, Open Solutions*, *7*, 5412–5420. doi:10.1109/ACCESS.2018.2890067

Srichandan, S., Kumar, T. A., & Bibhudatta, S. (2018). Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm - sciencedirect. *Future Computing and Informatics Journal*, *3*(2), 210–230. doi:10.1016/j.fcij.2018.03.004

Thein, T., Myo, M. M., Parvin, S., & Gawanmeh, A. (2018). Reinforcement learning based methodology for energy-efficient resource allocation in cloud data centers. *Journal of King Saud University Computer & Information Science*, *32*(10), 1127–1139. doi:10.1016/j.jksuci.2018.11.005

Volodymyr, M., Koray, K., David, S., Rusu, A. A., Joel, V., & Bellemare, M. G. et al.. (2019). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. doi:10.1038/nature14236 PMID:25719670

Wei, Y., Kudenko, D., Liu, S., Pan, L., Wu, L., & Meng, X. (2019). A reinforcement learning based auto-scaling approach for saas providers in dynamic cloud environment. *Mathematical Problems in Engineering*, *5080647*, 1–11. Advance online publication. doi:10.1155/2019/5080647

Xiong, Y., Huang, S., Wu, M., She, J., & Jiang, K. (2019). A Johnson's-Rule-Based Genetic Algorithm for Two-Stage-Task Scheduling Problem in Data-Centers of Cloud Computing. *IEEE Transactions on Cloud Computing*, *7*(3), 597–610. doi:10.1109/TCC.2017.2693187

Yuan, H., Bi, J., & Zhou, M. (2019). Multiqueue Scheduling of Heterogeneous Tasks With Bounded Response Time in Hybrid Green IaaS Clouds. *IEEE Transactions on Industrial Informatics*, *15*(10), 5404–5412. doi:10.1109/TII.2019.2901518

Zhang, P., & Zhou, M. (2018). Dynamic cloud task scheduling based on a two-stage strategy. *IEEE Transactions on Automation Science and Engineering*, *15*(2), 772–783. doi:10.1109/TASE.2017.2693688

Zuo, L., Shu, L., Dong, S., Zhu, C., & Hara, T. (2017). A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. *IEEE Access: Practical Innovations, Open Solutions*, *3*, 2687–2699. doi:10.1109/ACCESS.2015.2508940

Qirui Li received his M.S. degree in Computer Science and Technology in South China University of Technology in 2008, and received the Ph.D. degree in Applied Mathematics from Guangzhou University in 2021.He is studying in Guangzhou University for Ph.D. degree now. He is currently a lecturer at Guangdong University of Petrochemical Technology. He has published more than 10 papers in refereed journals and conference proceedings. His research interests include Artificial Intelligence, cloud computing, image processing and pattern recognition.

Zhiping Peng received his B.S. and M.S. degrees from China University of Petroleum (HuaDong) and Huazhong University of Science and Technology in 1996 and 2001, respectively, and received the Ph.D. degree in Computer Application from South China University of Technology in 2007. He is currently a professor at Guangdong University of Petrochemical Technology. He has published more than 50 papers in refereed journals and conference proceedings. His research interests include cloud computing, machine learning and multi-agent system.

Delong Cui received his M.S. degree in communication and information system in Southwest Jiao tong University in 2008. He is currently a professor at Guangdong University of Petrochemical Technology. He has published more than 20 papers in refereed journals and conference proceedings. His research interests include cloud computing and reinforcement learning.

Jianpeng Lin received the M.S. degree in computer science at Guangdong University of Technology (GDUT) in 2019. He is studying in South China University of Technology for Ph.D. degree. His research interest includes cloud computing, deep reinforcement learning and multi-agent system.

Jieguang He received his Ph.D. in mechanical engineering in Guangdong University of Technology in 2015. He is now a lecturer at Guangdong university of Petrochemical Technology. He has published more than 20 papers in relevant journals and conferences. His research interests include cloud computing, intelligent optimization algorithms, and project scheduling.