International Journal for Housing Science and Its Applications
Publish August 4, 2025. Volume 46, Issue 3 Pages 2257-2269

https://doi.org/10.70517/ijhsa463189

# Deep Reinforcement Learning-based Co-Optimization Method for Energy Consumption and Arithmetic Power in Cloud Computing Data Centers

**Xiangwen Wang[1] and Yixuan Wang[2,*]**
[1] Lanzhou Dechangtai Information Technology Co., Ltd., Gansu, 730070, China
[2] SAT School of Intelligent Engineering, Xi'an Jiaotong Liverpool University, Suzhou, Jiangsu, 730070, China
Corresponding authors: (e-mail: yixuanwang2025@126.com).

**Abstract** As the scale of data centers continues to expand, they face the challenge of rapidly increasing data volume. To solve the problem, this study constructs a system model based on cloud computing data center scenario, designs a task scheduling model using the improved Double DQN algorithm, and proposes a co-optimization method for energy consumption and arithmetic power in cloud computing data centers. Through simulation experiments on Cloud Sim cloud simulation platform, it is found that this paper's method has smaller energy consumption compared with other algorithms, and the energy consumption values are reduced by 23.92% and 17.62% compared with the Q-learning algorithm and the Q-learning(λ) algorithm with different numbers of virtual machines, and it has a faster convergence speed. Meanwhile, this paper's method performs better in reward value, average latency and load balancing, and the average latency is reduced by 30.30%~53.33% and 67.76%~90.59% than the comparison method in regular traffic and high traffic environment. The results show that the optimized Double DQN algorithm in this paper can effectively reduce the energy consumption and latency of cloud computing data centers, and has some practical value in the co-optimization of energy consumption and arithmetic power.

**Index Terms** deep reinforcement learning, Double DQN algorithm, co-optimization, cloud computing data center

## I. Introduction

With the acceleration of digital transformation, cloud computing data centers, as the infrastructure of the information age, assume the core functions of data storage, processing and transmission [1], [2]. However, its huge energy consumption and complex power load management problems are becoming more and more prominent, and have become a key factor restricting sustainable development [3], [4]. The energy consumption of cloud computing data centers mainly comes from IT equipment, cooling systems, power supply and distribution systems, and other auxiliary facilities [5]. Among them, servers and storage devices are the big head of energy consumption, accounting for more than 40% of the total energy consumption, while the cooling system, which maintains the appropriate temperature, follows closely behind with about 35% of energy consumption [6]-[8]. With the explosive growth of data volume and the increased demand for computing power, the energy consumption of data centers continues to increase and has become an important part of global energy consumption [9]-[11]. In order to solve the problem of energy consumption in data centers, deep reinforcement learning is gradually being applied [12].

Deep reinforcement learning, is a technique that combines the perceptual ability of deep learning and the decision-making ability of reinforcement learning [13], [14]. It trains models to autonomously learn and make optimal decisions in a given environment by simulating the human decision-making process of obtaining reward and punishment signals [15]. This method has made important breakthroughs in many fields and has been widely used in robot control and autonomous driving [16], [17]. In the optimization of energy consumption in cloud computing data centers, its core idea is to enable the data system to act like an intelligent body and learn the optimal strategy through continuous interaction with the environment, so as to achieve the improvement of data center performance [18]-[20].

Based on the analysis of cloud computing data centers and their basic architectures, the study constructs a system model containing load module, cloud environment module, monitoring module, task processing module, and scheduling module, and optimizes the energy consumption and arithmetic power of the data centers with the objectives of latency, energy consumption, and load balancing. Subsequently, the state space, action space and reward function are designed, and the optimized Double DQN algorithm is used to build the task scheduling model, and the utilization efficiency of experience data is improved by improving the experience replay unit. Then, simulation experiments using Cloud Smi 4.0 are conducted to compare the energy consumption of this algorithm with other algorithms under different numbers of virtual machines and iterations, and the distribution of the number of physical machines of each algorithm under different CPU utilization rates, to explore the effect of energy consumption optimization of the proposed method. Finally, the reward value, average delay and load balance degree of different algorithms are compared under two scenarios of traffic intensity of 50% and 100% to evaluate the arithmetic power optimization effect of the proposed method on cloud computing data centers.

## II. Cloud computing data centers and their architecture

With the rapid progress of information technology and the dramatic increase in the volume of data, traditional data management and storage methods are no longer able to cope with the demands of large-scale and efficient data processing. The emergence of cloud computing technology, with its elasticity, scalability, and pay-as-you-go characteristics, provides a new solution and becomes an ideal platform for building a modern big data center. Cloud computing uses virtualization technology to dynamically deploy services to data center resources and provide users with the required services. The basic architecture of cloud computing data center is shown in Figure 1, including infrastructure layer, platform layer and application layer. Cloud computing-based big data centers use virtualization technology to flexibly schedule computing, storage and network resources to optimize resource allocation and utilization efficiency.

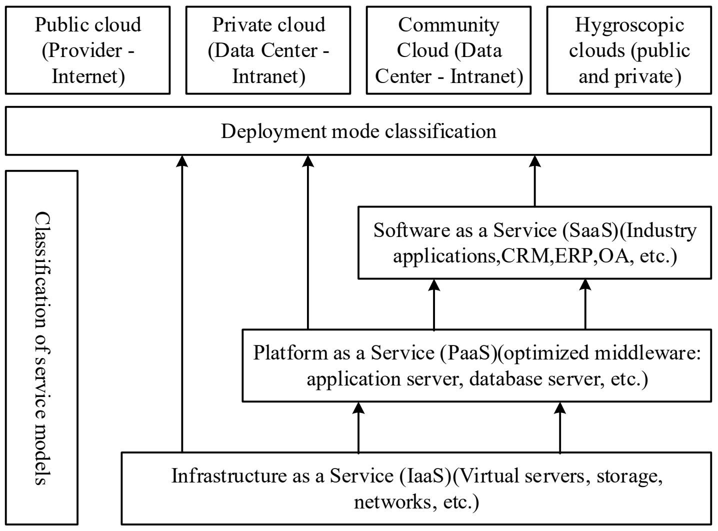

Public cloud (Provider - Internet)
Private cloud (Data Center - Intranet)
Community Cloud (Data Center - Intranet)
Hygroscopic clouds (public and private)
Deployment mode classification
Classification of service models
Software as a Service (SaaS)(Industry applications,CRM,ERP,OA, etc.)
Platform as a Service (PaaS)(optimized middleware: application server, database server, etc.)
Infrastructure as a Service (IaaS)(Virtual servers, storage, networks, etc.)

Figure 1: The basic architecture of the cloud computing data center

### *II. A.Infrastructure layer*

In the cloud-based big data center architecture, the infrastructure layer (IaaS) is the most core part, which provides virtualization of physical resources such as computing, storage, and network for the data center. The infrastructure layer mainly includes cloud servers, virtual storage, network devices and their management tools, which abstract physical resources into flexible computing and storage units through virtualization technology. Cloud servers provide on-demand computing capability, which can dynamically allocate computing resources according to demand, thus avoiding resource wastage and improving resource utilization efficiency. Virtual storage improves data access speed and fault tolerance by distributing data across multiple storage nodes through distributed storage technology. The network layer realizes flexible network configuration and management through software-defined network (SDN) technology to ensure efficient connectivity within and outside the data center.

### *II. B.Platform layer*

In the cloud-based big data center architecture, the platform layer (PaaS) carries the core function of big data processing and is an important bridge between the infrastructure layer and the application layer. The platform layer mainly provides distributed computing frameworks, data processing platforms, and service-oriented development tools to support efficient storage, processing, and analysis of big data. Common technologies are distributed file systems and computing frameworks. These platforms are capable of decentralized storage and parallel processing of massive data, greatly improving data processing efficiency and computing power. The platform layer also provides data warehouse, data analysis and visualization tools to support in-depth analysis and real-time monitoring of data. Through the efficient operation of the platform layer, enterprises and organizations are able to respond to the challenges of the big data environment with lower costs and greater flexibility, ensuring efficient collaboration of data flow, processing and application services.

### *II. C.Application layer*

In a cloud-based big data center architecture, the application layer (SaaS) is a key component in realizing end-user requirements, and it directly provides application services based on big data analytics for enterprises and organizations. The application layer helps users extract valuable information from massive data by providing a variety of industry-specific solutions, such as business intelligence (BI), data mining, precision marketing, risk management, etc., and enables in-depth analysis through data visualization tools. This layer not only supports real-time processing of big data, but also provides personalized services for different business needs, ensuring that data-driven decisions are more accurate and efficient. The application layer also simplifies the technical deployment of enterprises in data processing, storage and analysis by integrating different cloud services, lowering the technical threshold and enabling non-technical users to easily access and use data through an intuitive interface.

## III. Cloud computing data center energy consumption and arithmetic power co-optimization methods

Cloud computing data centers allow users to use resources or services free from space and time constraints, saving the time and cost of building resources and services locally. However, users' requests become more and more complex and the number of requests explodes, which leads to the resources of data centers becoming more and more strained. Therefore, in this paper, we combine deep reinforcement learning methods to investigate the co-optimization method of energy consumption and arithmetic power in cloud computing data centers.

### *III. A. System Modeling and Problem Definition*

The system model studied in this paper mainly contains load module, cloud environment module, monitoring module, task processing module and scheduling module. According to the system model, the task scheduling problem in a cloud environment can be summarized as executing $a$ tasks on $n$ virtual machines (VMs) according to a certain scheduling policy and optimizing the scheduling results under multi-objective constraints.

#### III. A. 1) Load module

Workloads, i.e., resource requests such as CPU, memory, and disk storage, are submitted by users to the cloud data center, and each request is considered as a complete task.

Suppose the user submits $a$ mutually independent tasks, each requiring $d$ types of resources, and the task arrives at the $i(i=1,2,\cdots,a)$ th task Task, the resource request vector is represented as $r_i^{request}=\left\{r_{i,1}^{request},r_{i,2}^{request},\cdots,r_{i,d}^{request}\right\}$, and the $k$ instances contained in $Task_i$ are denoted as $TI_i=\left\{TI_{i,1},TI_{i,2},\cdots,TI_{i,k}\right\}$. Consider 3 types of resources, CPU, memory and disk storage, which in turn can be formally defined as a hexadecimal group of tasks:

$$Task_i=\left\{T_i^{sub},T_i^{dur},num_i,cpu_i,mem_i,disk_i\right\} \tag{1}$$

where $T_i^{sub}$ denotes the submission time of $Task_i$, $T_i^{dur}$ denotes the expected computation time of $Task_i$, $num_i$ denotes the number of task instances contained in $Task_i$, $cpu_i$ denotes the number of CPU cores required to process $Task_i$, $mem_i$ denotes the size of memory required to process $Task_i$, and $disk_i$ denotes the size of disk storage required to process $Task_i$.

#### III. A. 2) Cloud Environment Module

In this paper, we consider a cloud data center based on an Infrastructure as a Service (IaaS) model, consisting of a VM model, a delay model, and an energy consumption model.

(1) VM model

Cloud data centers use virtualization technology to isolate, share and manage resources and contain $n$ VMs, denoted as $VMs=\left\{VM_1,VM_2,\cdots,VM_n\right\}$. Assuming that there are $d$ different types of resources in the VM, the total resources of the $j(j=1,2,\cdots,n)$ th VM $VM_j$ are denoted as $r_j^{total}=\left\{r_{j,1}^{total},r_{j,2}^{total},\cdots,r_{j,d}^{total}\right\}$, $VM_j$ has used resources denoted as $r_j^{usage}=\left\{r_{j,1}^{usage},r_{j,2}^{usage},\cdots,r_{j,d}^{usage}\right\}$, $VM_j$ with resource utilization expressed as

$u_j = \left\{ r_{j,1}^{usage} / r_{j,1}^{total}, r_{j,2}^{usage} / r_{j,2}^{total}, \cdots, r_{j,d}^{usage} / r_{j,d}^{total} \right\}$ , $VM_j$ current available resources are represented as $r_j^{surlpus} = \left\{ r_{j,1}^{total} - r_{j,1}^{usage}, r_{j,2}^{total} - r_{j,2}^{usage}, cdots, r_{j,d}^{total} - r_{j,d}^{usage} \right\}$. Consider 3 types of resources, CPU, memory and disk storage, which in turn can be formally defined as a quintuple of VMs:

$$VM_j = \left\{ cpu\_cap_j, mem\_cap_j, disk\_cap_j, P_j^s, P_j^{\max} \right\} \tag{2}$$

where $cpu\_cap_j$ denotes the CPU capacity of $VM_j$, $mem\_cap_j$ denotes the memory capacity of $VM_j$, $disk\_cap_j$ denotes the disk storage capacity of $VM_j$, $P_j^s$ denotes the static power of $VM_j$, and $P_j^{\max}$ denotes the maximum power of $VM_j$.

(2) Delay modeling

Firstly, the system latency is explained: during a single cloud service, the system latency is initially 0, and it increases accordingly as the cloud service proceeds. Secondly, the task-related latency is explained: according to the load module, each task contains $k$ instances, and the system makes a scheduling decision to execute the instance $TI_{i,b}(b = 1, 2, \cdots, k)$ of the $i$ th task $Task_i$ on the $j$ th VM node $VM_j$. The execution completion flag is returned by $Task_i$ only after all instances contained in $Task_i$ have been executed. The execution start time for all instances of $Task_i$ is denoted as $T_{TI_i}^{start} = \left\{ T_{i,1}^{start}, T_{i,2}^{start}, \cdots, T_{i,k}^{start} \right\}$, and the execution end time for $Task_i$ the execution end time of all instances is denoted as $T_{TI_i}^{finish} = \left\{ T_{i,1}^{finish}, T_{i,2}^{finish}, \cdots, T_{i,k}^{finish} \right\}$. Based on the above theory, the starting time of $Task_i$ is the minimum starting time of all the instances it contains, denoted as:

$$T_i^{start} = Min\left\{ T_{i,1}^{start}, T_{i,2}^{start}, \cdots, T_{i,k}^{start} \right\} \tag{3}$$

The end time of $Task_i$ is then the maximum end time of all instances it contains, denoted:

$$T_i^{finish} = Max\left\{ T_{i,1}^{finish}, T_{i,2}^{finish}, \cdots, T_{i,k}^{finish} \right\} \tag{4}$$

The time required for a complete cloud service is equal to the maximum completion time (Makespan) of all $a$ tasks, Makespan refers to the computation completion time of the last task and is also equal to the system latency at this point, denoted as:

$$Makespan = Max\left\{ T_1^{inish}, T_2^{finish}, \cdots, T_a^{finish} \right\} \tag{5}$$

(3) Energy consumption model

The energy consumption of a cloud data center is mainly generated by the VMs deployed on it, including both dynamic and static energy consumption. In addition, the disk storage required by the tasks in this paper's study is negligible compared to the disk capacity of the VMs, so the power of $VM_j$ can be estimated by the following linear model:

$$P_j = P_j^s + c_j \times (U_j^{cpu} + U_j^{mem}) \tag{6}$$

where $P_j^s$ denotes the static power of $VM_j$, $c_j \times (U_j^{cpu} + U_j^{mem})$ denotes the dynamic power of $VM_j$, $U_j^{cpu}$ denotes the CPU utilization of $VM_j$, $U_j^{mem}$ denotes the memory utilization of $VM_j$. $c_j$ denotes the influencing factors of CPU utilization and memory utilization, which are denoted as:

$$c_j = P_j^{\max} - P_j^s \tag{7}$$

According to the load module, each task contains $k$ instances, and each instance produces a certain amount of energy consumption when it executes on a VM node, and the energy consumption produced by $Task_i$ is the sum of the energy consumption produced by its instances. Then the energy consumption generated by an instance $TI_i$ of $Task_i$ executing on $VM_j$ is specifically denoted as:

$$E_{i.b}^j = \int_{T_{i.b}^{s\tan}}^{T_{i.b}^{finish}} P_j \tag{8}$$

Then the total energy consumption generated by processing $a$ tasks is specified as:

$$E_{total} = \sum_{j=1}^{n} \sum_{i=1}^{a} \sum_{b=1}^{k} E_{i,b}^{j} \tag{9}$$

### III. A. 3) Monitoring module

The monitoring module includes a task monitor and a VM monitor, whose main role is to monitor changes in the cluster environment. The task monitor is responsible for monitoring information such as resource requirements of tasks in the cluster, remaining instances of tasks, and task completion, while the VM monitor is responsible for monitoring information such as power and resource utilization of VMs.

### III. A. 4) Task processing module

Design of a task processing module based on an access control strategy and a prioritization strategy, which preprocesses the queue of tasks submitted by the user and filters out high-quality tasks to be provided to the scheduling module for processing, in order to reduce the overhead incurred in the subsequent training process.

(1) Task Access

In each time step, the available resource information $r_j^{surlpus}$ of $VM_j$ and the resource request information $r_i^{request}$ of $Task_i$ are accessed in real time by the monitoring module, and the resource evaluation is performed for each task and VM in turn. If there exists a VM that can accommodate the task, i.e., the available resources of $VM_j$ are not less than the required resources of $Task_i$, it can be expressed as follows:

$$r_{i,v}^{request} \le r_{j,v}^{surlpus}, \forall v \in (1, 2, \cdots, d) \tag{10}$$

If the above condition is satisfied then $Task_i$ is added to the pre-screening task queue and only tasks that can be executed immediately are screened. For tasks that do not satisfy the above conditions, they wait for re-evaluation at the next time step. Such tasks can be processed only after the VM releases sufficient resources.

(2) Prioritization

On the basis of the task admission policy, consider prioritizing the pre-screened task queue. The prioritization of $Task_i$ is defined as follows:

$$rank_i = \eta \times \log T_i^{dur} + (1-\eta)\log num_i \tag{11}$$

$$s.t. \eta \in [0,1] \tag{12}$$

where the smaller the value of $rank_i$, the higher the ranking of $Task_i$, i.e., the higher the degree of priority, $T_i^{dur}$ denotes the desired computation time of $Task_i$, and $num_i$ denotes the number of instances of the task contained in $Task_i$. The value of $\eta$ represents the extent to which the task duration and the number of instances contained in the task affect the prioritization. When $\eta$ is large, the priority will focus on shorter task durations, while when $\eta$ is small, $1-\eta$ is large, and the priority will focus on fewer number of instances contained in the task.

### III. A. 5) Movement control module

In each time step, the scheduling module receives inputs from the monitoring module and uses a fully connected neural network as the brain of the Agent to pass the environment information through the neural network, after which it outputs a VM-task allocation scheme, and finally the scheduling module pushes the task to the appropriate buffer queue and assigns it to a specific VM for execution.

### III. A. 6) Problem definition

Latency, energy consumption, and load balancing are the core metrics to measure the performance of scheduling algorithms, which are used in this paper as the objectives of co-optimization of energy consumption and arithmetic in data centers, so the specific optimization objective, i.e., minimizing the weighted sum of the maximum completion time and total energy consumption of all the tasks, and at the same time, achieving good load balancing.

The optimal maximum completion time of $a$ tasks for a single cloud service is denoted as:

$$f_1(x) = \min(Makespan) \tag{13}$$

The total energy consumption generated by $a$ tasks of a single cloud service is optimally expressed as:

$$f_2(x) = \min(E_{total}) \tag{14}$$

For load balancing, this is expressed by minimizing the standard deviation of resource utilization across VMs:

$$f_3(x) = \min\left(\sqrt{\frac{\sum_{j=1}^{n}(u_j-\mu)^2}{n}}\right) \tag{15}$$

where $\mu$ denotes the mean of all VM resource utilization.

Based on the above definition, the multi-objective function is expressed as:

$$F(x) = [f_1(x), f_2(x), f_3(x)] \tag{16}$$

Considering that the maximum completion time and the total energy consumption of the tasks may have large differences in data size, the data are normalized using a logarithmic approach. Therefore, the final multi-objective optimization function and its constraints are expressed as:

$$F = \alpha_1 \times \log f_1(x) + \alpha_2 \times \log f_2(x) + \alpha_3 \times \log f_3(x) \tag{17}$$

$$s.t. \alpha_1 + \alpha_2 + \alpha_3 = 1, \forall \alpha_1, \alpha_2, \alpha_3 \in [0,1] \tag{18}$$

$$r_{i,1}^{request} \le r_{j,1}^{surlpus}, \forall i \in (1,2,\cdots,a), j \in (1,2,\cdots,n) \tag{19}$$

$$r_{i,2}^{request} \le r_{j,2}^{surlpus}, \forall i \in (1,2,\cdots,a), j \in (1,2,\cdots,n) \tag{20}$$

$$r_{i,3}^{request} \le r_{j,3}^{surlpus}, \forall i \in (1,2,\cdots,a), j \in (1,2,\cdots,n) \tag{21}$$

where constraint (18) represents the weighted value coefficients of the objective function, and $\alpha_1, \alpha_2$ and $\alpha_3$ represent the optimization focus of the task scheduling strategy. When $\alpha_1$ is large, the optimization direction will focus on the maximum completion time of the task, whereas when $\alpha_2$ and $\alpha_3$ are large, the optimization direction will focus on the total energy consumption produced by the task and load balancing. Constraint (19) indicates that the CPU required by a task should not be larger than the available CPU of the VM serving it. Constraint (20) indicates that the memory required by a task should not be larger than the available memory of the VM serving it. Constraint (21) indicates that the storage required by the task should be no greater than the available storage of the VM serving it.

## III. B. Double DQN based optimization design

### III. B. 1) Enhanced Learning Intelligentsia Learning

Reinforcement learning intelligences for learning contain three important components: state space, action space, and reward function.

(1) State Space

The state space reflects the allocation and utilization of tasks and virtual machines in the current state. For the state of a data center VM resource, the completion time of its currently assigned tasks can be calculated, with each machine having its own completion time $CT_k$. A global maximum completion time makespan is also recorded, which can be used as $V_{current} = \{v_1, v_2, v_3, ..., v_m, makespan\}$ represents the machine state. For the task state, the intelligent body only needs to focus on the queue of tasks to be scheduled, which can be represented by $T_{wait} = \{t_1, t_2, t_3, ...\}$. Therefore, $S = \{S_1, S_2, S_3, ...\}$ to denote the state space set where $S_i = (V_{current}, T_{wait})$.

(2) Action Space

The action space reflects the behavior of the scheduler in assigning tasks to VMs. For a datacenter with $m$ VM resources, the action space for each task scheduling is the set of these $m$ VM numbers, which can be represented as the set of action spaces by $A = \{1, 2, 3, ...., m\}$ to denote the set of action spaces. The criterion for an intelligent body to select an action to execute is to make the value of makespan minimized.

(3) Reward function

The reward function is used to describe the immediate reward value for moving from one state to another after taking an action. It is divided into two parts as follows:

In the first part, the makespan does not grow after the intelligent body executes the task scheduling action, and the reward function design in this case is shown in Equation (22):

$$R_1 = -(CT_{sta}(S') - CT_{sta}(S)) * \rho_1 if (MS(S') = MS(S)) \tag{22}$$

The $CT_{var}(S')$ is the variance of the machine completion time after executing an action to enter the $S'$ state during task scheduling, and the $\rho_1$ is a constant used to regulate the reward value to the appropriate range. The design of the reward $R_1$ helps to promote cluster load balancing when the

In the second part, makespan grows after the intelligence performs the task scheduling action, and the design of the reward function in this case is shown in Equation (23):

$$R_2 = -(MS(S') - MS(S)) * \rho_2 if(MS(S') > MS(S)) \tag{23}$$

The $MS(S')$ is the makespan to enter the $S'$ state after executing the action during the task scheduling process, and the $\rho_2$ is used to regulate the reward value to the appropriate range.

So the reward functions are combined as shown in equation (24):

$$R = \begin{cases} -(CT_{sta}(S') - CT_{sta}(S)) * \rho_1 & if(MS(S') = MS(S)) \\ -(MS(S') - MS(S)) * \rho_2 & if(MS(S') > MS(S)) \end{cases} \tag{24}$$

**III. B. 2) Double DQN model design**

Although DQN overcomes the limitation of Q-learning's Q-table when performing task scheduling, the action selection strategy it adopts still maximizes the Q-value $max_{a'} Q(s', a', \theta)$, which leads to the maximization bias problem, making the estimated action value biased. When training the network, the action value estimation is related to the network weights $\theta$, and when the weights change, the action value estimation also changes. In reinforcement learning, the objective value is computed as:

$$y_t = r_t + \gamma * \max_{a'} Q(s_{t+1}, a', \theta) \tag{25}$$

where $y_t = r_t + \gamma * max_{a'} Q(s_{t+1}, a', \theta)$ is the value of the reward based on the actual observation, $\max_{a'} Q(s_{t+1}, a', \theta)$ is the estimate made at $s_{t+1}$ based on the $Q$ network. The use of the $\max_{a'} Q(s_{t+1}, a', \theta)$ strategy to maximize the $Q$ value when calculating the target value is therefore prone to cause overestimation problems.

Bootstrap can be avoided and the problem of overestimation can be mitigated by utilizing a target network. The target network is to construct a second network $Q(s, a, \theta^-)$ with the same structure as the original network $Q(s, a, \theta)$ but with different parameters, where $\theta^- \neq \theta$, the original network is called the evaluation network. The target network $Q(s, a, \theta^-)$ is used to calculate the TD target value. Then:

$$y_t = r_t + \gamma * \max_{a'} Q(s_{t+1}, a', \theta^-) \tag{26}$$

The target network avoids bootstrapping by delaying the update of network parameters to upset the data correlation, but it still does not eradicate the overestimation problem caused by the maximization bias, and the only way to solve the overestimation problem is to improve the design in the training algorithm of the DQN. Using the learning model of Double DQN to improve the performance of cloud data center task scheduling, Double DQN splits the maximization into two steps when calculating the target Q-value: selecting and finding the value. Instead of finding the maximum Q value in each action directly inside the target Q network, the evaluation Q network is utilized to find the action $a^*$ that produces the maximum Q value, and then the target network is used to compute the Q value of $a^*$.

Selection: based on the state $s_{t+1}$, an action $a^*$ is selected using the evaluation network, i.e., $a^*$ corresponds to the largest $Q$ value in the evaluation network:

$$a^* = \arg\max_{a \in A} Q(s_{t+1}, a, \theta) \tag{27}$$

Finding the value: calculate the value of $(s_{t+1}, a^*)$ using a target network, i.e., use the target network to find the target value corresponding to the action $a^*$:

$$y_t = r_t + \gamma Q(s_{t+1}, a^*, \theta^-) \tag{28}$$

Since the evaluation network and the target network are independent of each other and $a^*$ is not necessarily the target network $\theta^-$ parameters under $\arg\max Q$ have the following inequality, bias can be eliminated and network overestimation can be avoided:

$$Q(s_{t+1}, a^*, \theta^-) \leq \max_{a \in A} Q(s_{t+1}, a, \theta^-) \tag{29}$$

Therefore the target Q value in Double DQN is calculated as:

$$y_t = r_t + \gamma Q(s_{t+1}, \arg\max_{a\in A} Q(s_{t+1}, a, \theta), \theta^-) \tag{30}$$

During the learning process of Double DQN algorithm, the intelligent body observes the state of the cloud data center environment, reads the virtual machine and the task, and then the evaluation network generates the appropriate action, i.e., the mapping of the task to the virtual machine, and stores $(s,a,r,s')$ into the experience playback memory unit. Then determine whether it is the termination state of an event, if it is the termination state, the TD target is $r$, otherwise use the target network to compute the TD target value. $y_t$ is the target value fitted by the network and $Q(s_t,a_t,\theta)$ is the estimated value. The network is trained by minimizing the mean square deviation between two Q values. The minimized mean square error (MSE) loss function is shown in equation (31):

$$L(\theta) = E\left[(y_t - Q(s_t,a_t,\theta))^2\right] \tag{31}$$

The gradient descent method is then performed to update the network and compute the gradient of the loss function associated with the parameter $\theta$:

$$\nabla\theta = \frac{\partial L(\theta)}{\partial\theta} = E\left[(y_t - Q(s,a,\theta)\frac{Q(s,a,\theta)}{\partial\theta})\right] \tag{32}$$

Update the evaluation network parameters $\theta = \theta + \nabla\theta$, and wait for a certain number of steps $c$ apart before updating the TD target network parameters once so that $\theta^- = \theta$.

### III. B. 3) Experience Replay Unit Design and Improvement

To address the lack of data, sparsity of rewards, and strong correlation of data in the training process of deep reinforcement learning. Experience replay is used to train the network, storing the data of the intelligences interacting with the environment into the Replay Buffer, and randomly sampling from the Replay Buffer for training in subsequent learning to improve the efficiency of experience data utilization. In order to guarantee the effectiveness and efficiency of learning, the empirical replay unit in Double DQN is updated and designed to better serve the training of the network.

For the data $(s,a,r,s')$ put into the empirical replay unit, it carries two pieces of information that can be used to signify the value of the data. One is the TD error $\delta = y_t - Q(s_t,a,\theta)$, which is trained with the goal of making the expectation of $\delta^2$ as small as possible in order to adjust the parameters of the neural network so that it is able to more accurately predict the value of the next state. Thus $\delta$ can be somewhat fully responsive to the state. Another information is utilized: reward $r$, which is the reward value obtained by taking action $a$ in state $s$ and entering the next state $s'$, the higher the reward $r$ indicates the more valuable this action is. So during the training of the algorithm, the TD error

When the empirical data $(s,a,r,s')$ passes through the control unit and enters the Replay Buffer, a flag bit $\tau_{len}$ will be added to it to become $(s,a,r,s',\tau_{len})$. The reward $r$ and $\delta$ are used to determine the value of $\tau_{len}$. The larger the reward, the slower the makespan grows, indicating that the actions performed in that state make the cluster load more balanced.The smaller the absolute value of the TD error, the more accurate the neural network prediction. Therefore, the value of $\tau_{len}$ can be calculated by equation (33):

$$\tau_{len} = L_{ini} + r*\alpha_1 - |\delta|*\alpha_2 \tag{33}$$

where $L_{ini}$ is the initial life cycle set for each empirical data. The rewards $r$ and $|\delta|$ are used to be used to adjust the values of $\tau_{len}$. $\alpha_1$ and $\alpha_2$ are used to adjust $r$ and $|\delta|$ to the appropriate range. Thus the larger the reward, the smaller the error, then the value of $\tau_{len}$.

## IV. Experimental evaluation and analysis

### *IV. A. Experimental setup*

In this paper, we use Cloud Smi 4.0 to conduct simulation experiments on the resource dynamic provisioning problem in cloud computing data centers, and test the efficiency of the related methods in reducing the total energy consumption of cloud computing centers by testing the optimized Double DQN algorithm and the comparison algorithm in this paper. Physical machine utilization and number of virtual machines are taken as input parameters in the experiment. The simulation experiments are conducted with 300 physical machines of three types, namely, HP Pro Lanti ML110G5, HP Pro Lanti DL360G7, and HP Pro-Liant DL360Gen9, and their types are evenly distributed. Meanwhile, in order to ensure the authenticity of the results, this paper selects the CPU utilization data of 200 real virtual machines provided by Planet Lab in one day. PSO, Greedy, Q-learning, and Q-learning(λ) are selected as comparison methods for the experiment.

### *IV. B. Energy consumption optimization effect*

This paper compares the energy consumption of the optimized Double DQN algorithm and other algorithms in the resource allocation problem in cloud computing data centers, Fig. 2 shows the specific energy consumption values of each algorithm at different number of virtual machines. With the increase of the number of virtual machines, the optimized Double DQN algorithm has the smallest change in energy consumption, which is 23.92% lower than that of Q- learning algorithm when the number of virtual machines is 300, 17.62% lower than that of Q-learning(λ) algorithm, and also has a significant decrease in energy consumption compared to Greedy and PSO algorithms.

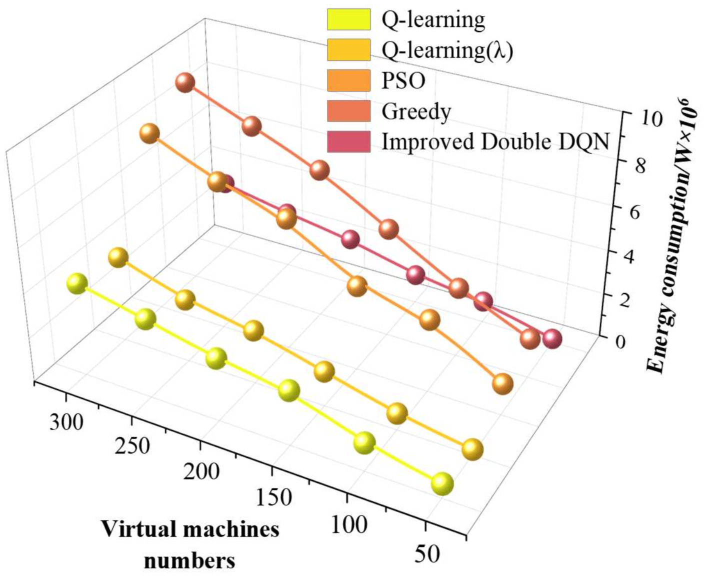


Figure 2: The specific energy consumption of each algorithm in different number virtual machines

Fig. 3 shows the comparison of energy consumption of various algorithms. The optimized Double DQN algorithm has faster convergence speed compared to Q-learning, Q-learning(λ), PSO and Greedy algorithms, and the optimal convergence value of this algorithm is around 1.5 × 106W, which is smaller than Q-learning, Q-learning(λ), PSO and Greedy algorithms. Therefore, the optimized Double DQN algorithm in cloud computing data centers has both better convergence speed and lower energy consumption.

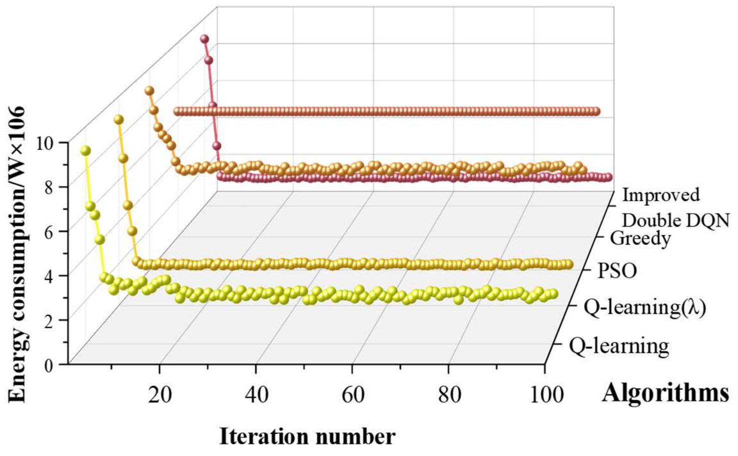


Figure 3: Energy consumption of each algorithm

In this paper, three types of physical machines such as HP Pro Lanti ML110G5, HP Pro-Liant DL360G7, and HP Pro Lanti DL360Gen9 are used, and by referring to the data from Standard PerformanceEvaluation Corporation, it can be seen that their average performance power ratio are 463, 2,999, and 10018, respectively, and the larger value of the average performance-power ratio indicates that the physical machine consumes less energy with the same performance. The distribution of the number of physical machines for each algorithm under different CPU utilization is shown in Table 1. When the CPU utilization is 80%~100%, the optimized Double DQN algorithm is the one that uses the least amount of G5 and Gen9 uses the most amount. When the CPU utilization is 0~20%, the Optimized Double DQN algorithm is using the maximum number of G5 as 88, while the minimum number of Gen9 is 57. The more the number of G9, the higher the efficiency of the cloud computing center and the lower the energy consumption, while the opposite is true for G5. Therefore, in the cloud computing data center energy optimization problem, the optimized Double DQN algorithm produces a placement strategy that is realistic and it produces the least total energy consumption. Similarly, the Q-learning(λ) algorithm is better than the Q-learning algorithm, the PSO algorithm is better than the Greedy algorithm, and the Greedy algorithm is the worst.

Table 1: The physical machines numbers for each algorithm at different CPU utilization rates

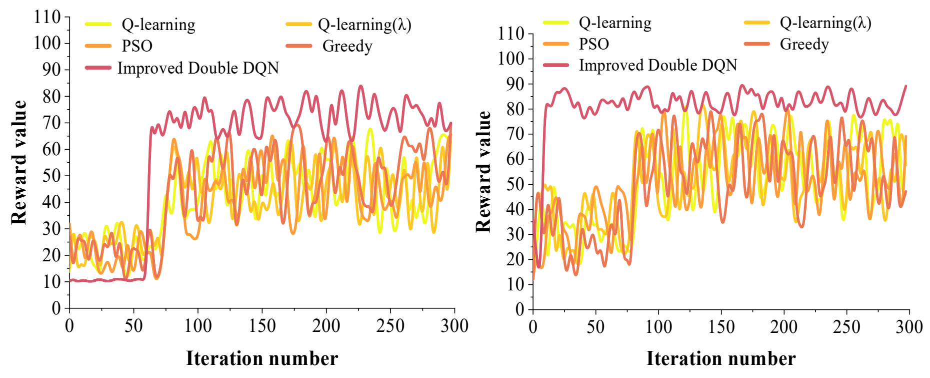
| CPU utilization rates | The physical machines numbers | | | | |
|---|---|---|---|---|---|
| | Q-learning | Q-learning(λ) | PSO | Greedy | Improved Double DQN |
| [80%,100%] G5 | 40 | 2 | 47 | 2 | 10 |
| [80%,100%] G7 | 7 | 8 | 11 | 8 | 11 |
| [80%,100%] Gen9 | 11 | 70 | 10 | 17 | 12 |
| (20%,80%) G5 | 27 | 20 | 23 | 6 | 5 |
| (20%,80%) G7 | 31 | 38 | 24 | 15 | 20 |
| (20%,80%) Gen9 | 30 | 41 | 23 | 21 | 22 |
| [0,20%] G5 | 40 | 5 | 5 | 119 | 88 |
| [0,20%] G7 | 53 | 55 | 52 | 78 | 75 |
| [0,20%] Gen9 | 61 | 61 | 105 | 34 | 57 |
| [80%,100%] Total | 58 | 80 | 68 | 27 | 33 |
| (20%,80%) Total | 88 | 99 | 70 | 42 | 47 |
| [0%,20%] Total | 154 | 121 | 162 | 231 | 220 |
| The physical machines numbers | 300 | 300 | 300 | 300 | 300 |

## ***IV. C. Calculation power optimization effect***

Latency and bandwidth are important metrics for measuring the arithmetic performance of data networks. In order to comprehensively evaluate the arithmetic optimization performance of the proposed algorithms in this paper, this study conducted experiments under two different traffic loading conditions: a regular environment with a traffic intensity of 50%, and a high-traffic stress environment with a traffic intensity of 100%. In these two environments, the network performance of the optimized Double DQN algorithm is tested and compared and analyzed with existing PSO, Greedy, Q-learning, and Q-learning(λ) algorithms.

The comparison of reward values of the algorithms in different environments is shown in Fig. 4. The cumulative reward value of the optimized Double DQN algorithm is always higher than that of the PSO, Greedy, Q-learning, and Q-learning(λ) algorithms in the regular environment with a traffic intensity of 50%. During the iterations, the reward value of Optimized Double DQN algorithm is about 49.65% higher than PSO algorithm, about 35.51% higher than Greedy algorithm, about 44.07% higher than Q-learning algorithm, and about 46.52% higher than Q-learning(λ) algorithm. This shows that the optimized Double DQN algorithm is effective in improving the network quality of service and maintaining a high level of performance in network environments dealing with medium traffic intensity. The performance advantage of the Optimized Double DQN algorithm is even more obvious in a high traffic pressure environment with a traffic intensity of 100%. In addition to the reward value, the optimized Double DQN algorithm converges about 87% faster than the comparison algorithm. This indicates that under high traffic pressure, the Optimized Double DQN algorithm is able to better adapt to network changes and maintain network stability and quality of service.

In order to further analyze and evaluate the performance of the mentioned five algorithms in terms of network arithmetic optimization, this study provides a detailed comparison of the two key metrics considered in the reward function, namely latency and load balancing degree. Latency is defined as the time it takes for a packet to be transmitted from the source node to the target node, and is an important measure of network transmission efficiency; the shorter the latency, the faster the transmission. This study focuses on the average delay of each link in each episode and analyzes it comparatively, the results of the algorithm average delay comparison are shown in Fig. 5.



(a)The flow intensity is 50% (b)The flow intensity is 100%

Figure 4: The comparison of rewards in different environments

The average delay of Optimized Double DQN is 30.30% to 53.33% lower than other algorithms in the regular environment with 50% traffic intensity. When the traffic intensity is 100%, the average latency of Optimized Double DQN is 67.76% to 90.59% lower than other algorithms. The results further illustrate the effectiveness of the Optimized Double DQN algorithm for arithmetic performance optimization in cloud computing data centers.

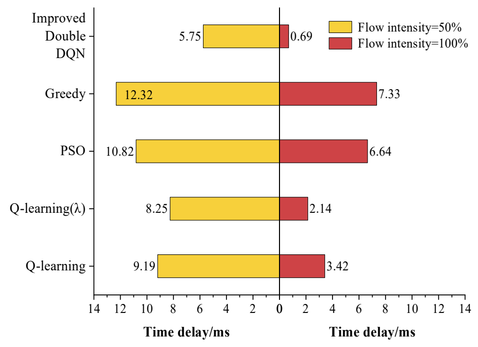
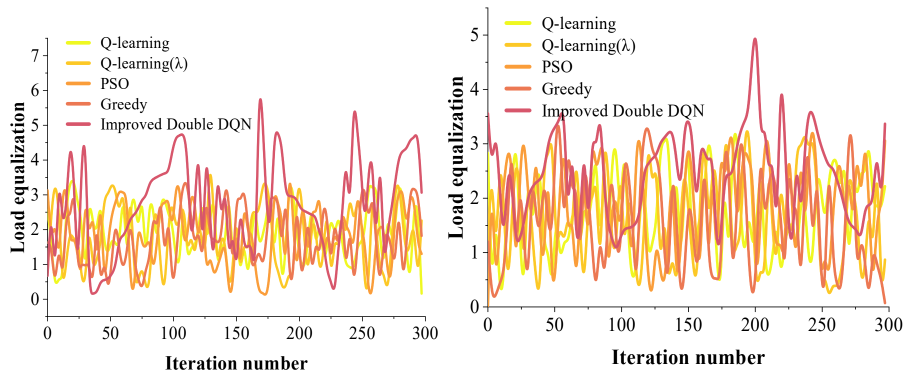


Figure 5: The average delay results of the algorithm

In addition, this study also compares the performance of the algorithms in terms of load balancing, and the comparison of algorithm load balancing degree is shown in Figure 6. When the traffic intensity is 50% and 100%, the load balancing degree of the optimized Double DQN algorithm for 300 iterations is 2.74 and 2.47, respectively, which is higher than that of the comparison algorithms, and the jitter amplitude is smaller, which demonstrates a superior load balancing performance.



(a)The flow intensity is 50% (b)The flow intensity is 100%

Figure 6: The comparison of the algorithm load equalization

## V. Conclusion

The rapid growth of the number of cloud computing data centers makes its energy consumption problem more and more serious, this paper for this problem, puts forward a deep reinforcement learning based data center energy consumption and arithmetic power co-optimization method, using the improved Double DQN algorithm to optimize the constructed system model. The energy consumption and arithmetic power optimization effect of the proposed method is explored through experimental analysis. The main results are as follows:

(1) Compared with the comparison algorithms, the proposed method in this paper has the smallest energy consumption value under different numbers of VMs, which is 23.92% and 17.62% lower than the Q-learning algorithm and Q-learning(λ) algorithm. In iterative training with the same number of VMs, the optimal convergence value is around 1.5×106W and has the fastest convergence speed. Meanwhile, the results of the distribution of the number of physical machines for each algorithm under different CPU utilization also show that it performs well in data center energy consumption.

(2) The reward value of this paper's method is overall higher than other methods in both regular and high traffic environments, and its convergence speed is about 87% faster in the high traffic environment. Its average latency is reduced by 30.30% to 53.33% and 67.76% to 90.59% in both environments and shows better load balancing. The proposed optimized Double DQN algorithm effectively reduces the network transmission delay and improves the arithmetic performance of cloud computing data centers.

In this paper, we have initially completed the research of cloud resource scheduling method based on deep reinforcement learning, but there are still some shortcomings. Although the cloud environment and edge cloud environment are simulated by the mature Cloud Sim cloud simulation platform, and the simulated scheduling experiments are completed in the corresponding environment, there are more uncertainties in the real cloud environment, which all affect the stability of the algorithm. In the future research, the algorithm proposed in this paper will be applied to real environments, and the algorithm will be gradually improved for the problems, so that it can still work stably and efficiently in real environments.

## References

[1] Katal, A., Dahiya, S., & Choudhury, T. (2023). Energy efficiency in cloud computing data centers: a survey on software technologies. Cluster Computing, 26(3), 1845-1875.

[2] Sharkh, M. A., Kanso, A., Shami, A., & Öhlén, P. (2016). Building a cloud on earth: A study of cloud computing data center simulators. Computer Networks, 108, 78-96.

[3] Helali, L., & Omri, M. N. (2021). A survey of data center consolidation in cloud computing systems. Computer Science Review, 39, 100366.

[4] Edwin, E. B., Umamaheswari, P., & Thanka, M. R. (2019). An efficient and improved multi-objective optimized replication management with dynamic and cost aware strategies in cloud computing data center. Cluster Computing, 22(Suppl 5), 11119-11128.

[5] Uchechukwu, A., Li, K., & Shen, Y. (2014). Energy consumption in cloud computing data centers. International Journal of Cloud Computing and Services Science (IJ-CLOSER), 3(3), 31-48.

[6] Ma, H., & Ding, A. (2022). Method for evaluation on energy consumption of cloud computing data center based on deep reinforcement learning. Electric Power Systems Research, 208, 107899.

[7] Fiandrino, C., Kliazovich, D., Bouvry, P., & Zomaya, A. Y. (2015). Performance and energy efficiency metrics for communication systems of cloud computing data centers. IEEE Transactions on Cloud Computing, 5(4), 738-750.

[8] Ibrahim, H., Aburukba, R. O., & El-Fakih, K. (2018). An integer linear programming model and adaptive genetic algorithm approach to minimize energy consumption of cloud computing data centers. Computers & Electrical Engineering, 67, 551-565.

[9] Ahmed, K. M. U., Bollen, M. H., & Alvarez, M. (2021). A review of data centers energy consumption and reliability modeling. IEEE access, 9, 152536-152563.

[10] El Kafhali, S., & Salah, K. (2018). Modeling and analysis of performance and energy consumption in cloud data centers. Arabian Journal for Science and Engineering, 43(12), 7789-7802.

[11] Nehra, P., & Nagaraju, A. (2019, March). Sustainable energy consumption modeling for cloud data centers. In 2019 IEEE 5th international conference for convergence in technology (I2CT) (pp. 1-4). IEEE.

[12] Mastelic, T., Oleksiak, A., Claussen, H., Brandic, I., Pierson, J. M., & Vasilakos, A. V. (2014). Cloud computing: Survey on energy efficiency. Acm computing surveys (csur), 47(2), 1-36.

[13] Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine, 34(6), 26-38.

[14] Wang, X., Wang, S., Liang, X., Zhao, D., Huang, J., Xu, X., ... & Miao, Q. (2022). Deep reinforcement learning: A survey. IEEE Transactions on Neural Networks and Learning Systems, 35(4), 5064-5078.

[15] Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y. C., & Kim, D. I. (2019). Applications of deep reinforcement learning in communications and networking: A survey. IEEE communications surveys & tutorials, 21(4), 3133-3174.

[16] Fujita, Y., Nagarajan, P., Kataoka, T., & Ishikawa, T. (2021). Chainerrl: A deep reinforcement learning library. Journal of Machine Learning Research, 22(77), 1-14.

[17] Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., & Pérez, P. (2021). Deep reinforcement learning for autonomous driving: A survey. IEEE transactions on intelligent transportation systems, 23(6), 4909-4926.

[18] Arvindhan, M., & Kumar, D. R. (2023). Adaptive Resource Allocation in Cloud Data Centers using Actor-Critical Deep Reinforcement Learning for Optimized Load Balancing. International Journal on Recent and Innovation Trends in Computing and Communication, 11(5s), 310-318.

[19] Simin, W., Lulu, Q., Chunmiao, M., & Weiguo, W. (2023). Research on overall energy consumption optimization method for data center based on deep reinforcement learning. Journal of Intelligent & Fuzzy Systems, 44(5), 7333-7349.

[20] Ran, Y., Zhou, X., Hu, H., & Wen, Y. (2022). Optimizing data center energy efficiency via event-driven deep reinforcement learning. IEEE Transactions on Services Computing, 16(2), 1296-1309.