

Adaptive scheduling strategy for cloud computing resources based on Q-learning algorithm

Jiaxing Xu*, Xiaofei Gao, and Ningyuan Gao

PipeChina Digital Co., Ltd, Beijing 100013, PR China

Received: 7 July 2025 / Accepted: 24 November 2025

Abstract. Given the frequently varying loads of virtual machines and swift variations in resource demand in a cloud computing environment, this article introduces an adaptive resource scheduling strategy model based on Q-learning. The model develops a refined state and action space through real-time monitoring of virtual machine resource states, and an optimized reward mechanism that dynamically adjusts the resource allocation strategies. The reinforcement learning algorithm produces the optimal scheduling strategy through ongoing learning and adjustments to improve resource utilization, load balancing and task execution efficiency. Experimental results illustrate that the CPU utilization of virtual machines implementing the adaptive resource scheduling strategy using Q-learning achieves 0.87; memory utilization achieved 0.72; bandwidth utilization achieves 0.74; and the resource utilization curve is reasonably stable. This demonstrates that this model improves the efficiency of resource scheduling for virtual machines in a cloud computing environment, and ensures resources are efficiently and smoothly allocated.

Keywords: Cloud computing environment / resource management / Q-learning algorithm / adaptive resource scheduling / load balancing

1 Introduction

Cloud computing technology has emerged as a fundamental component of contemporary information infrastructure, pervading sectors like finance, healthcare, and manufacturing to facilitate digital transformation. Its on-demand resource provisioning and elastic scaling mechanisms have greatly decreased IT deployment costs for enterprises and improved service agility, becoming an architectural asset for global digitalization [1]. However, the dynamic characteristics of cloud computing environments produce persistent, serious challenges in this regard. For example, virtual machine (VM) loads can vary significantly depending on the task, with different users (and services) having different workload intensities, and user requests and applications may quickly transition between workloads, creating surges and peaking resource demands. These factors contribute to inefficient resource utilization that delays or degrades system response times, detrimental service quality, and impacts on user satisfaction and experience.

Existing studies predominantly utilize rules or statistical models as the means of the allocative resources. In the early phase of research, Esh M et al. [2] created a model

grounded in time-series prediction. The model modeled the plans by studying the historical load data and their moving average. However, the initial research on time-series did not perform well in a dynamic environment with changing demands. Torres JF et al. [3] looked at the positive characteristics of a deep learning approach for time-series prediction, although they lacked explainable depth or real-time predictive capability. Hewamalage H et al. [4] implemented recurrent neural networks for time-series predictions but faced challenges such as training complexity and Gradient vanishing problems. A different branch of studies employed hybrid genetic algorithms on task allocation problems for optimizing resource scheduling to maximize efficiencies around resource allocations; yet the optimization processes took considerable time in a rapidly changing dynamic environment [5–7]. Some scholars used the Linear Regression (LR) methods to predict resource demand, thus reducing the frequency of virtual machine migrations. However, due to the model being entirely dependent on historical data, these types of models were not effective at applying itself to higher frequency changes in load demand [8,9]. Other studies have attempted to enhance task allocations with a combination of dynamic scheduling rules with only a fraction of the resource waste getting alleviated, and still lacked immediacy and accuracy [10,11]. Even though this research has produced some results, traditional approaches

* e-mail: xujx_pipechina@163.com

frequently struggle to strike a balance between efficiency and performance in cloud computing scenarios with greater dynamism and complexity.

To deal with the issue of real-time adaptability in resource scheduling, reinforcement learning has become a hot research topic. Talaat F M, Bhattacharya P, AlQerm, I [12–14] presented models for resource allocation optimization based on Q-learning, which improved resource utilization and load balancing. Some studies have [15–17] approached the scheduling of multi-task resources based on distributed multiagent reinforcement learning with good scalability in multi-user scenarios. Other works have [18,19] used a priority-based Q-learning model to adjust the resource requirements of high-priority tasks. However, their state-space complexity resulted in increased training costs. Other works have applied the DDPG (Deep Deterministic Policy Gradient) algorithm to optimize continuous resource allocation with some success in reducing system overload [20–22], still requiring significant computational power and imposing a strict design requirement on the rewards mechanism. These studies still confront the same issues of low-efficiency model training, poorly designed incentive mechanisms, and complex state space partitioning in practice.

This paper proposes a Q-learning-based adaptive resource scheduling strategy to overcome these limitations. Q-learning is a model-free reinforcement learning algorithm that is exceptionally well-suited for decision-making in uncertain environments. The motivation behind the research will be the goal of maintaining real-time adaptability, achieving balanced multi-objective optimization, or improving stability in the context of existing cloud computing resource scheduling mechanisms subject to dynamic load variation and dynamic resource demand. The research object is the virtual machine resource scheduling mechanism in cloud-computing environments. The goal of the research will be to quantitatively improve some core performance objectives by increasing CPU utilization to a minimum of at least 0.85, increasing memory utilization to greater than 0.70, increasing bandwidth utilization to greater than 0.72, reducing the imbalance load factor to less than 0.2, and reducing the average task completion time by greater than 20% compared to traditional models. To meet that goal, the research tasks will include: 1) constructing a fine granularity or discrete state space based on monitoring the status of a VM in real-time (resource state includes CPU, memory, and bandwidth); 2) designing a discrete action space with respect to the resource and VM migration; 3) optimizing the reward mechanism to balance resource utilization, load balancing, and task response time; and 4) implementing and testing the Q-learning algorithm with dynamically tuned parameters through comparative experiments.

2 Q-learning adaptive resource scheduling strategy

2.1 Virtual machine load monitoring

In the section regarding virtual machine load monitoring and analysis section [23–25], this model is based on the OpenStack cloud computing platform, which serves in the real-time monitoring of the virtual machine resources

focused on their dynamic changes over time by CPU, memory and bandwidth. Load information is collected from loaded systems by Prometheus [26–28] and Grafana [29] to examine the changing patterns in fluctuations and demands of the resources. The use of Prometheus and Grafana provides very efficient use of data in regard to timeliness and accuracy by configuring the systems accordingly. In term of actual data timeliness, Prometheus uses an active pull mode to collect the resource metrics for the virtual machine, and does this at a 10 s data sample that can be adjusted dynamically depending on the load changes experienced. In terms of data accuracy, Prometheus reads the resources that are being used from the operating system via the kernel-level data through a Node Exporter that is deployed in the virtual machine, therefore, reducing or eliminating loss of data occurring, if converted in the mid-data forwarding stage. Grafana merely renders the source metrics being stored from Prometheus, and does not mutate the actual data, which you will ultimately render, therefore accuracy is assured. This data indicates the changes in resources due to load, can be utilized to make decisions in regard to resource scheduling from past data, but can also help predict changes in resources demands through real-time analysis to inform a basis for dynamically adjusting the scheduling strategy.

In the establishment and segmentation of state space, the model employs resource utilization (CPU utilization, memory usage, etc.) as state inputs, and segments the system load state into three discrete intervals (low, medium, and high) based on resource utilization. This both simplifies the representation of the system state and allows the Q-learning algorithm to schedule accurately where decisions are based on resource demand. Each state indicates the resource demand and load level of the virtual machine's resources, allowing the algorithm to make highly effective decisions in complicated resource management environments.

In the action space design, the model identifies which resource-scheduling actions the system can perform [30–32]. Examples of resource-scheduling actions are to change a virtual machine's resource quota or migrate a virtual machine to improve the resource configuration. Each action relates to a transition from one resource configuration to another - allowing a richer selection space for the Q-learning algorithm [33]. The Q-learning algorithm can then learn and select the best scheduling solution. An adequately designed action space allows the Q-learning algorithm to find the optimal solution with multiple schemes and to optimize the efficiency of resource allocation using cloud computing resources. This research paper applies reinforcement learning to realize resource scheduling. In the system, the scheduling strategy is determined by the current state and modified with the reward from the feedback of the system. The Q value continues to be modified through feedback from the environment to improve resource scheduling states. The Q-learning algorithm will find the optimal scheduling strategy during the learning process, thereby allocating resources and executing tasks effectively under varying load conditions. As the learning advances, the model increasingly converges to the optimal strategy, thereby

optimizing the resource utilization, balancing the load, and minimizing the task completion time.

The rewards mechanism [34,35] has a direct impact on the scheduling decision effect. The design of the rewards function considers the impact of three important factors - resource utilization, load balancing, and response time - upon the scheduling strategy. Strategies with high resource utilization efficiency and good load balancing effects are provided with positive rewards while strategies with poor allocation of resources are punished with negative rewards. The rewards mechanism therefore steers the algorithms towards selecting strategies that lead to improved system performance, reduced latency, and increased resource allocation efficiency, allowing the system to maintain a very high efficiency and a stable state despite fluctuations in load and changes in resource demand. This paper combines monitoring of virtual machine loads, splitting the state space, action space design, implementation of the Q-learning algorithm, and rewards mechanism optimization allowing adaptive resource scheduling that dramatically improves resource utilization, load balancing and system stability while providing an effectively automated resource management solution in a cloud computing environment. The paper will monitor the resources used by the virtual machines, and define the formula of virtual machine resource utilization:

$$R_i = \frac{U_i}{C_i}. \quad (1)$$

Among them, R_i represents the resource utilization of the i th virtual machine; U_i is the actual amount of resources used by the virtual machine; C_i is the total amount of resources allocated to the virtual machine. This formula calculates the virtual machine resource occupancy ratio to support load prediction and decision-making. In resource utilization monitoring, directly using raw data may be affected by short-term fluctuations. To this end, the exponentially weighted moving average method is applied for smoothing.

$$\widehat{R}_i(t) = \alpha R_i(t) + (1-\alpha)\widehat{R}_i(t-1). \quad (2)$$

Among them, $\widehat{R}_i(t)$ is the smoothed resource utilization at time t , and $\alpha \in (0,1)$ is the smoothing coefficient. This formula is used to reduce data fluctuations, extract long-term trends, and facilitate state decision-making of the Q-learning algorithm.

2.2 Definition and division of state space

When developing an adaptive resource scheduling strategy based on Q-learning, it is important to track resource usage by the virtual machines and derive important metrics to use as state inputs. Because each virtual machine's resource consumption impacts the overall system load, classifying the resource metrics in detailed and meaningful increments will assist in measuring load variations in real-time.

Resource usage and load metrics are quantified with a limited number of categories. CPU and memory usage is classified into three increasing categories: low, medium, and high. For CPU usage, the increasing class intervals are:

low = 0–40%, medium = 40%–70%, and high = 70%–100%. Memory usage is also categorized in a similar fashion into increasing low, medium, and high load classifications. This provides a clear representation of the resource complexity, which in turn allows the state space to be more manageable, while facilitating a responsive mechanism to changing load conditions. The state space is separated between competition and cooperation of resources between virtual machines. For example, a virtual machine that is utilizing a larger amount of CPU resources may influence resource allocation for other virtual machines on the same physical host, therefore changing the resource requirements for other virtual machines.

The model is intended to integrate the joint state of multiple virtual machines and normalize load across a group, allowing the Q-learning algorithm to learn that [36], when interacting with each individual resource, virtual machines make accurate scheduling decisions. Each discrete state corresponds uniquely to the current system's resource usage and current load-balancing status so that the Q-learning algorithm can accurately evaluate the effect of resource allocations. The scheduling action is then selected based on the current state to achieve the optimal resource allocation. This splitting method can facilitate scheduling in more complex problems and improve the adaptability of the system to dynamic resource demands and performance changes.

The state representation formula is:

$$S(t) = \{U_i(t) \mid i \in [1, n]\}. \quad (3)$$

$S(t)$ is the overall state of the system at time t , including the resource utilization $U_i(t)$ of all virtual machines.

Adjusting dynamic thresholds uses an intuitive sliding window algorithm that refreshes the threshold every 30 min based on the variances in task load fluctuations and the presence of resource bottlenecks within that window. As an example, if a resource type is consistently in the "high load but no performance bottlenecks state" (in this example, a compute-intensive task is 82% CPU and produces no task response delays), the resources high-load threshold is adjusted automatically upwards by 5% (per instance, no single upward adjustment can exceed 10% of the threshold). Conversely, if the load is moderate and loading the resources multiplies, the moderate-load threshold is reduced 3%. High-load and moderate-load thresholds are adjusted so resources are consistently operating under appropriate load levels relative to the demands of the situation. Additionally, users will have the opportunity to manually configure a custom threshold in the OpenStack platform console to customize business scenario needs, bolstering the second principle of flexibility and adaptability in state space partitioning.

2.3 Design of action space

The Q-learning scheduling strategy's efficiency is influenced by how the action space is designed. The model can define scheduling actions by specifying adjustments in resource allocation, increasing or decreasing resource allocations, or migrating virtual machines to deal with

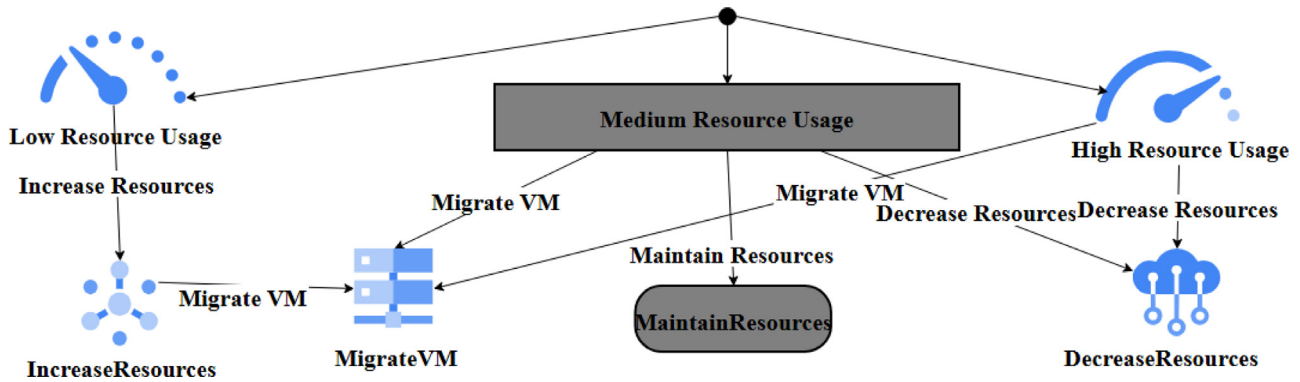


Fig. 1. Virtual machine resource scheduling state and action space distribution diagram.

dynamic resource demands and changes in load. The model uses an action of adding resources when the workload using the virtual machine books exceeds a defined quota, uses reducing resources when the resource utilization is below a defined threshold, farms virtual machines to combine loads and reduce the resource count rather than incur a resource load on a single approach (i.e., improving load balance). The action space design is directly influenced by how the state space is divided. When the model performs a scheduling action, the action itself will change the configuration for specific resources, allowing the Q-learning model to adjust resources when there are changes to the load. The scale and accuracy of the action space are both determining factors of the scheduling effect. A well-designed action space will improve the learning efficiency of Q-learning, as it will allow the system to better select scheduling actions and make an adjustment to preserve resources and allocate resources in an optimal manner. The reward mechanism provides feedback, reinforcement, decision optimization, improves resource utilization and load balance, and contributes to both system stability and operation efficiency. Figure 1 illustrates the distribution of state and action spaces for virtual machine resource scheduling.

The distribution of the virtual machine resource scheduling states and action spaces is provided in Figure 1, which demonstrates distinct connections between each state of the virtual machine resource utilization and the corresponding resource scheduling action in the Q-learning-based adaptive resource scheduling strategy. It divides the virtual machine resource utilization into three distinct states low, medium, and high and associates different types of resource scheduling action options with each state. When resource utilization reaches “low,” the system can either choose to “increase resources” or “migrate virtual machines” in order to improve the efficiency of the resource utilization for dynamic VM resource allocation and avoid having resources that sit idle. When resource utilization reaches a state of “high,” the system can choose to either “reduce resources” or “migrate virtual machines” to relieve resource overload in order to maintain stability of the system. When resource utilization reaches a state of “medium,” the system chooses to “maintain resources” to keep the existing resource configuration and achieve dynamic resource balance.

2.4 Q-learning algorithm implementation

The Q-learning algorithm, which is founded on value functions, is a sort of reinforcement learning algorithm that is predominantly used to make actions in the context of dynamic and uncertain environments. A cloud computing environment can be described as dynamic because the load on virtual machines and resource requirements are variable in nature. Static scheduling agents have difficulty adapting to rapidly changing demands. Q-learning can successfully address the scheduling problem in cloud computing through ongoing optimization of decisions, which happens by virtue of interaction with the environment. The central idea is to learn and refine the optimal resource scheduling policy iteratively, while also continuing to improve resource utilization in the system and the performance of the system as a whole.

For Q-learning to be implemented, the first step is to define the state space and action space related to relevant periods in a cloud computing environment. The components of the state space are typically tied to what load conditions and resource requirements the virtual machine has. States will involve reference indicators for the virtual machine’s load/resource requirement of such things as CPU utilization, memory consumption rate, bandwidth consumption rate, and so on. The resources in these states are typically encoded discretely, into the resource utilization indicators mentioned previously, for subsequent processing by the Q-learning algorithm, it is standard for each resource utilization state of each virtual machine to be divided into a low, medium, or high state meaning that there are generally three states for each resource that are described to the Q-learning algorithm. In this way, the complex resource scheduling problem is simplified to a discrete state space, enabling the Q-learning algorithm to make appropriate decisions under different states.

The action space is critical for the proposed method’s design. Actions reflect resource scheduling, representing actions taken in a given system state which typically means changing the virtual machine’s resource quota either upwards or downwards or migrating from one node to another in order to provide improved resource allocation. Each action in the scheduling state may lead to changes to the system’s resource allocation. A well-designed action

space will allow Q-learning to select the best resource scheduling strategy. Under high-load conditions, the proposed approach adds resources by using Q-learning to assign more resources or migrate virtual machines. With low-load conditions, it would reduce the resource quota and migrate machines to the lowest loaded server.

With respect to performance, Q-learning can improve scheduling and system performance as a result of the design of the reward mechanism. The learning process employed consists of states, actions and rewards. The mechanism selects actions for performance which is guided by updating Q-values and the action(s) selected are updated based on rewards to the current q-value, and the Bellman equation governs this update.

$$Q(s, a) \leftarrow Q(s, a) + \eta[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]. \quad (4)$$

Among them, $Q(s, a)$ represents the value of selecting action a in state s ; s' is the new state after executing the action; $\max_{a'} Q(s', a')$ represents the maximum Q value in the new state. This formula iteratively updates the strategy to find the optimal scheduling solution. The reward function r is designed based on resource utilization and scheduling efficiency, and the system receives feedback on how to improve the current strategy.

The beginning ϵ value is initialized at 0.9 (reaching between 0 and 1) and is strictly adjusted by a linear decay rule, meaning every 10 rounds trained the ϵ value is decreased by 0.05 until reaching a value of 0.1, in which it maintains the constant value of 0.1. This parameter setting significantly affects the efficiency of model training, with the beginning ϵ value being initially high (of 0.9) allowing the algorithm to fully explore the state-action space of the action space early in training, avoiding premature regression from an optimal policy to a policy that is locally optimal. The ϵ value then linearly decreases as training continues, which relies on the algorithm to select the “best” known optimal policy, thereby speeding up convergence. The ϵ value of 0.1 at the end ensures a small amount of exploration is maintained, to ensure the model can adapt to subtle changes of the environment, later. Alternatively, if the initial ϵ value is too low, then there is not enough exploration, resulting in premature convergence at that state. If the decay is too slow or the final value is too high, then the exploration is less effective which prolongs the cycle of training. The above parameter set allows for a dynamic interplay or balance, when appropriate, between exploration and exploitation exploring effectively to ensure efficient model convergence.

Q-learning also provides good long-term stability and performance. Too many resource rescheduling can yield fluctuations in system performance, no matter how quick the load changes in cloud computing. Q-learning minimizes the frequency of resource tuning while maximizing resource utility. This provides more system stability without more frequent tuning and minimizes overhead. When load changes change on cloud computing resources, Q-learning will ensure that the adaptive scheduling strategies built

with Q-learning action are dynamic and responsive to load change. The proposed algorithm provides additional resources to account for heavy overload and fewer resources when the load is light to eliminate waste. Q-learning selects the best possible action for each state at each decision point in the system, resulting in better resource allocation, improved resource utility and system stability. Simply put, Q-learning provides an adaptive, effective solution to resource scheduling in cloud computing that continually improves performance in a dynamic environment.

2.5 Reward mechanism setting

The reward mechanism is an important part of the Q-learning model and has direct effects on the resource scheduling strategy of the system. The reward function should optimize the use of resources, load balancing, and response time of the system. A strategy that increases resource utilization while decreasing idle time of resources should receive a positive reward, and the strategy would be penalized for either not utilizing the resource or over-utilizing the resource. For load balancing, in order to avoid overload of resources and idleness of virtual machines, the reward function can assign positive rewards and penalties based on how imbalanced the load is, allowing for more optimal resource allocation across the environment.

Response time optimization: the reward mechanism should aim to lower the response time and waiting time of the task, initiating responses to user needs quickly. If the response time is low and the waiting time is low, the user receives a positive reward, while a penalty would be targeted toward long response times to encourage responses while optimizing scheduling strategies. To ensure stability and long-term performance, the reward mechanism can also place penalties on excessive reconfiguring so that the system does not incur unnecessary overhead by continually adjusting the resources.

When designing the reward mechanism, a discount factor should be incorporated to measure the relationship between rewards that are received immediately and rewards that are received at some future point in time. The discount factor allows the system to make choices that provide long-term benefits rather than receiving immediate returns, thereby allowing the algorithm to avoid local optima while providing an efficient resource scheduling mechanism. The weights of the reward mechanism should be adjustable to match actual application scenarios, thereby supporting flexible resource scheduling characteristics that enable corresponding rewards to be continuously improved in this changing cloud computing environment. The reward mechanism provides the Q-learning structure the advantage of improving the efficiency and stability of resource scheduling by directing and guiding the learning process. Based on multidimensional feedback comprising resource utilization, load balancing, and response time, decisions are constantly optimized, which ensure that resource scheduling adapts and dynamically evolves in a cloud computing environment to improve overall performance. The reward function to advance the optimization of the Q-learning strategy has been defined as follows:

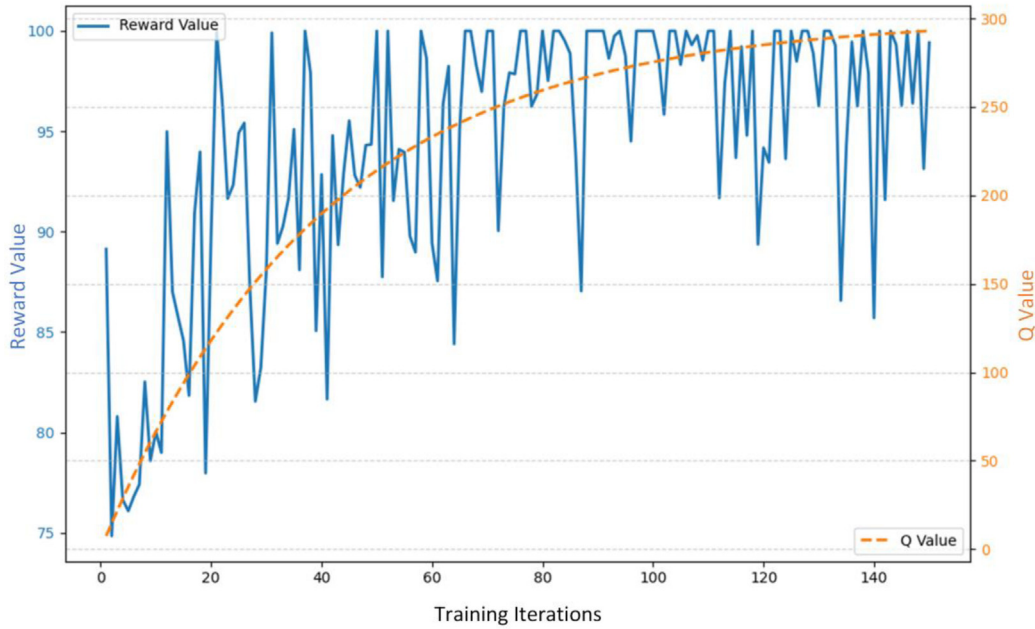


Fig. 2. Reward value and Q value change with training iterations.

$$r = \omega_1 U_{\text{avg}} - \omega_2 D. \quad (5)$$

Among them, U_{avg} is the average resource utilization of the system; D is the load imbalance between virtual machines; ω_1 and ω_2 are weight factors that measure the impact of resource utilization and delay on rewards, respectively. The experiment was based on the unified environment configuration of the original QARS model, keeping other parameters unchanged. To prioritize the execution efficiency of high-priority tasks, the original reward function introduces a task priority weight parameter (ranging from 0 to 0.4, satisfying $\omega_1 + \omega_2 + \omega_3 = 1$) and the average response time of high-priority tasks, T_{high} . The revised reward function is

$$r = \omega_1 U_{\text{avg}} - \omega_2 D + \omega_3 T_{\text{high}}. \quad (6)$$

T_{high} is calculated by calculating the average time from submission to completion of all high-priority tasks per unit time. When the execution delay of a high-priority task is high, the penalty term corresponding to ω_3 is used to reduce the overall reward value; otherwise, the penalty is reduced.

Figure 2 illustrates the changes in the reward value and Q value of the model in this paper during training.

As shown in Figure 2, the reward value and Q-value change trends of the model in this paper are displayed using a double Y-axis. The horizontal axis is the number of training iterations (1 to 150 rounds); the left Y-axis is the reward value (the initial range is 75 to 100, and it remains between 80 and 100 after stabilization); the right Y-axis is the Q value (converging from close to 0 and finally close to 300). The reward value curve shows the strategy's optimization effect, and the Q value curve verifies the model's learning process and convergence. Figure 3 is a specific flowchart of the model in this paper.

Figure 3 shows the detailed process of the adaptive resource scheduling strategy based on Q-learning. First, the system obtains the load and delay data for the virtual machine from the resource monitoring module and tracks and analyzes them in real time. Then, the system partitions the state space according to the predefined strategy and maps information such as resource usage and load level to recognizable states. Then, the system designs the action space, which includes various resource-scheduling decisions. Next, the Q-learning algorithm learns the actions in each state and adjusts its strategy based on the feedback from each action. The reward mechanism is calculated based on the actual situation, and the updated strategy is returned to the system to optimize resource scheduling. Finally, the system achieves efficient resource scheduling through the scheduling optimization module, while performing performance monitoring and evaluation to ensure the adaptability and stability of the scheduling strategy across different scenarios.

3 System evaluation

3.1 Resource utilization

The experiment is being conducted on the OpenStack cloud platform. The cluster of physical servers is composed of five identical machines, each with an Intel Xeon Gold 6330 CPU, 128GB of DDR4-3200 memory, a 2 TB SSD (RAID 5) and 10 Gbps of network bandwidth. The group of virtual machines utilizes the same configuration, and we deploy 20 virtual machines with identical specifications. Every virtual machine is allocated 2 CPU cores, 4 GB of memory, 50 GB of storage, and 1 Gbps of network bandwidth on initial VM boot. The operating system is Ubuntu Server 20.04 LTS, kernel version 5.4.0-150-generic. The workload

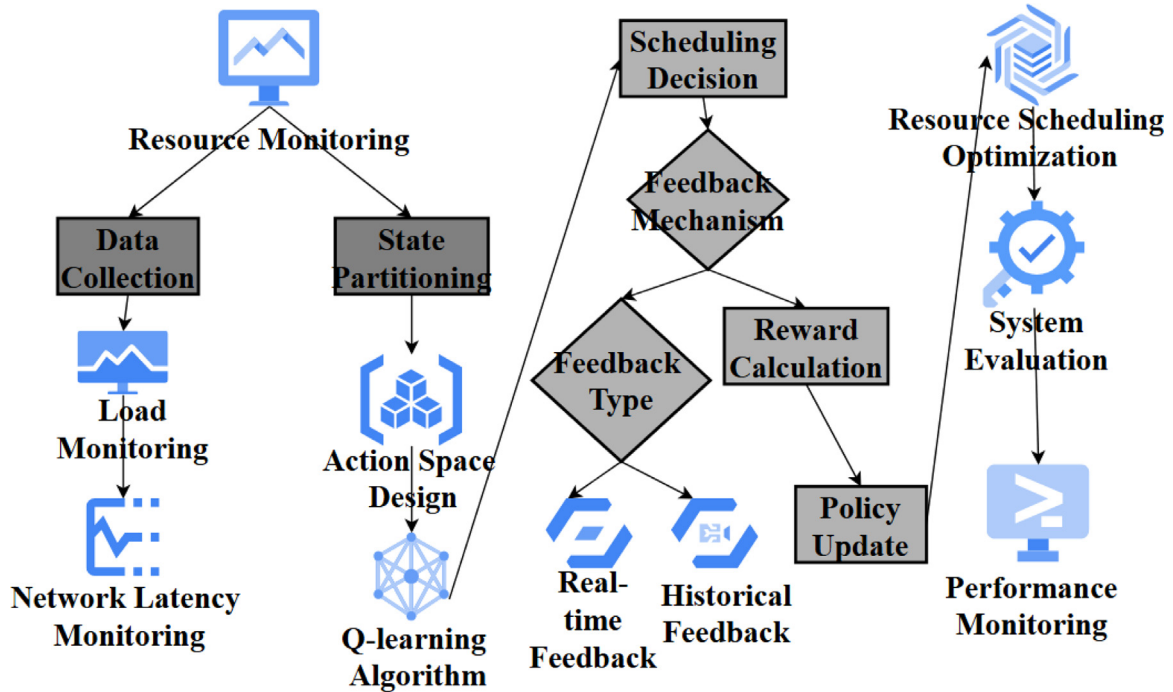


Fig. 3. Flowchart of adaptive resource scheduling strategy based on Q-learning.

used for the experiment is a mixed workload set generated by the CloudSim tool, and consists of three typical cloud tasks: web services, data calculations, and file transfers. There are a total of 1000 functions, and the task submission interval is uniformly randomly distributed in the range of 1 to 5 min. The experiment length is fixed to 25 h. The data collection tools used are Prometheus 2.45.0 and Grafana 10.1.0, with the data sampling interval set to 10 s. This is a repeated measure design, and all models are run independently for 3 times, while maintaining the same hardware, software, and load conditions. The average value is taken as the final experimental result to eliminate the interference of environmental differences in the comparison of resource utilization.

Evaluating resource utilization is one of the main performance indicators in this evaluation. The proposed Q-learning adaptive resource scheduling strategy enables the system to dynamically alter resource quotas for virtual machines based on continuously monitored resource consumption information, such as CPU, memory, and bandwidth. For evaluation purposes, the system automatically collects resource usage data from each virtual machine in real time, compares the virtual machine's actual resource usage value with the maximum resource quota for the virtual machine, and then calculates utilization efficiencies for each resource.

As shown in the iterative process, in the process of evaluation, the system learns the trend of resources utilization and finds which resource allocation method can enhance the global efficiency. If the CPU or memory utilization of some virtual machines is low, the resources allocation of these virtual machines will be adjusted by Q-learning strategy to release the occupied resources and reassign them to other virtual machines which have more

resources needs. After several rounds of learning, the system will gradually optimize the resources scheduling and eliminate the idle and wasteful resources assignment, and then enhance the overall resources utilization. The comparison with the Traditional Model can more reflect the performance and stability of the model in VM resources allocation. Figure 4 Comparison of Resource Utilization of SLR, DDPG, Priority-Based Q-learning Model (PQL), Time Series Prediction-Based Resource Allocation (TSPR) Model, and Model in the Paper (Model of Model in the Paper: Q-learning-Based Adaptive Resource Scheduling Strategy Model, QARS).

As shown in Figure 4, the CPU and memory utilization curve of the proposed model constantly increases during the operation of the project. Other models have the upward trend, but are not very stable. The CPU utilization of the proposed model can achieve 0.87 and the memory utilization can achieve 0.72, which are much higher than other models. Although the bandwidth utilization is affected by the network environment and fluctuates a little, it is still very high, up to 0.74, and the stability is also greatly improved compared with other models. Compared with other models, QARS has much higher utilization and much less fluctuation, which also means that the virtual machine resource allocation performance of QARS is more stable, and the virtual machine resource utilization can be improved.

3.2 Load balancing

Load balancing is the degree to which the resources are equally distributed among the virtual machines. In Q-learning adaptive resource scheduling strategy, the load balancing can be evaluated by the differences of virtual

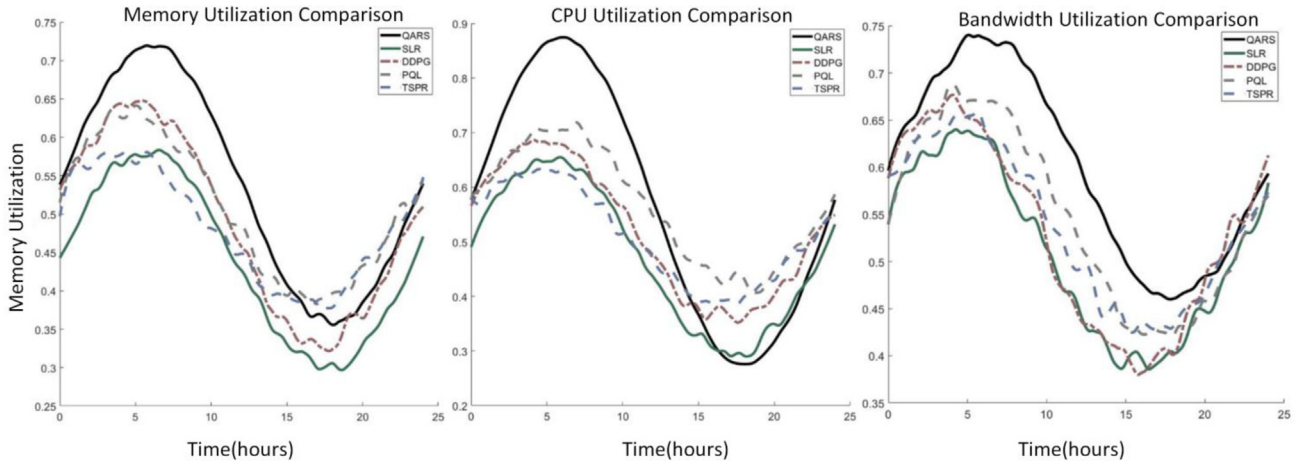


Fig. 4. Resource utilization comparison chart.

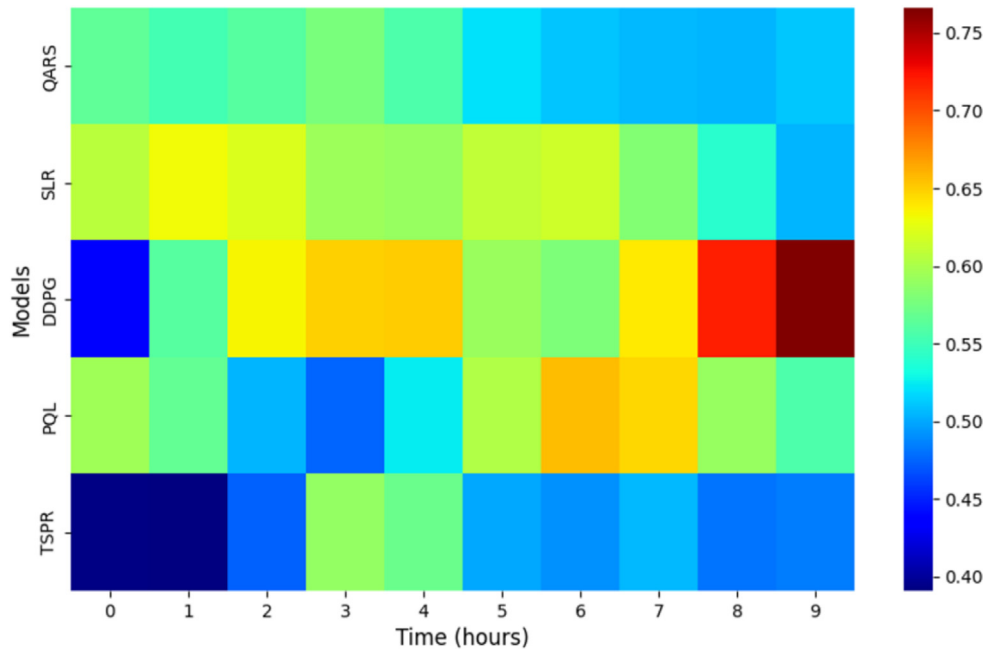


Fig. 5. Model load change diagram.

machine resources usage. In some cases, the system selects each virtual machine and calculates the load difference of other virtual machines based on the virtual machine resources usage information. When the allocation of virtual machine resources is very imbalanced, the scheduling strategy adjusts each virtual machine resources quota automatically based on the output results of Q-learning algorithm to realize the balanced allocation.

Load balancing optimization is achieved gradually via reinforcement learning. When the load of virtual machine is too heavy or too light, select the optimal action to balance the load of different virtual machines based on the information in the space state, namely, the virtual machine resources overloading or idle is avoided, the resources is balancedly allocated to different virtual machines and even the whole cloud computing environment, and then enhance

the overall performance and responsiveness. Load balancing calculation formula is as follows.

$$D = \sqrt{\frac{1}{n} \sum_{i=1}^n (U_i U_{avg})^2} \quad (7)$$

where, U_i is the utilization of virtual machine i ; U_{avg} is the average utilization of system; n is the number of virtual machines. The smaller D value represents the more balanced load.

Figure 5. Load fluctuation of five models (QARS, SLR, DDPG, PQL, TSPR) from 0 h to 9 h.

As shown in Figure 5, the fluctuation of load of QARS model is small, and it is more balanced. The load of QARS model is ranged in 0.45–0.55, and it is almost not fluctuated, so the resources of virtual machine can be

Table 1. Comparison of resource allocation times.

| Model | Average allocation frequency (allocations/hour) | Peak allocation frequency (allocations/hour) | Minimum allocation frequency (allocations/hour) |
|-------|---|--|---|
| QARS | 14 | 16 | 12 |
| SLR | 21 | 25 | 18 |
| DDPG | 17 | 19 | 15 |
| PQL | 19 | 22 | 16 |
| TSPR | 23 | 28 | 20 |

balanced and effectively utilized. However, the fluctuation of other models are large and the load balance is not good, and the resources are not effectively utilized.

3.3 Task completion time

Task completion time is the basis for the evaluation of scheduling efficiency. It is the time from the submission of a task to its completion. Task completion time includes several stages, such as queuing time, scheduling time, and execution time. The mixed workloads in the experiment have an impact on the task completion times of models. This impact is caused by the different amounts of resources required by workloads. Web service tasks are short-lived and highly concurrent, and are sensitive to the responsiveness of CPUs; data computation tasks need continuous availability of high amounts of memory, and the allocation efficiency of memory directly influences execution time; and file transfer tasks depend on stable bandwidth, and changes in bandwidth will cause transfers to be blocked for long periods, resulting in increased transmission latency. Experiments show that, for all three types of tasks, the QARS model obtains better completion times than those of the comparison models. This is because the QARS model can match task type with scheduling strategies for resources dynamically. For web service tasks, it reduces queue time by lowering the response cycle of the CPU resource adjustment. For data computation tasks, it ensures continuous availability of memory so that execution is not blocked by memory reclaim that occurs repeatedly. For file transfer tasks, it adjusts transmission queue priorities in real time according to the utilization of bandwidth so as to reduce the latency caused by the contention of bandwidth. This proves that matching task type with a scheduling strategy for resources is very important to task completion time.

To present the amount of resources allocated by different model strategies, Table 1 shows the number of resource allocations for each model, which are extracted from experimental data under the same conditions.

Average frequency of allocating resources for the proposed model is 14 times in an hour; maximum is 16 times; minimum is 12 times; allocation frequency difference is only 4. It is evident that, in a dynamic cloud computing environment, the model can keep the state of stable and efficient allocating resources, and the fluctuation in the scheduling process is reduced effectively.

Figure 6. Comparison of queuing time, scheduling time and execution time for completion time of each model.

In Figure 6, our model spends 5 s in queuing time, 3 s in scheduling time, and 6 min in execution time, while other models spend more time in these three processes. Our model completes a task with higher efficiency in resource scheduling and execution. In comparison, it takes more seconds in queuing time, more seconds in scheduling time, and more minutes in execution time for SLR, DDPG, PQL, and TSPR models. The TSPR model spends the most time in completing a task, which means its efficiency in resource scheduling and task execution is the lowest. Our model performs better than other models in three processes and completes a task rapidly.

3.4 System stability

System stability evaluation evaluates whether long-term virtual machine resource scheduling will cause system instability due to frequent resources adjustment. Q-learning adaptive resource scheduling strategy can achieve smooth resource adjustment and avoid unnecessary configuration reconfiguration based on historical data and feedback. In the evaluation, system stability is measured by recording the number of virtual machine resource scheduling change, adjustment times and system situations after each adjustment. System has high stability if its virtual machine resource configuration is adjusted infrequently, namely, avoid system instability caused by system scheduling adjustment frequently. The optimized reward mechanism allows Q-learning algorithm to balance virtual machine resource adjustment times and necessity and avoid unnecessary adjustment to reduce overhead, guaranteeing the system stability in long-term operation and avoiding performance fluctuation caused by long-term resource scheduling adjustment. Finally, the long-term performance fluctuation evaluates system stability. The following is the performance fluctuation coefficient formula of this paper:

$$P = \frac{\max(P) - \min(P)}{\bar{P}} \quad (8)$$

$\max(P)$ and $\min(P)$ are the maximum and minimum values of the performance indicators, respectively, and \bar{P} is the average value of the performance indicator. A smaller ΔP means that the system performance is more stable.

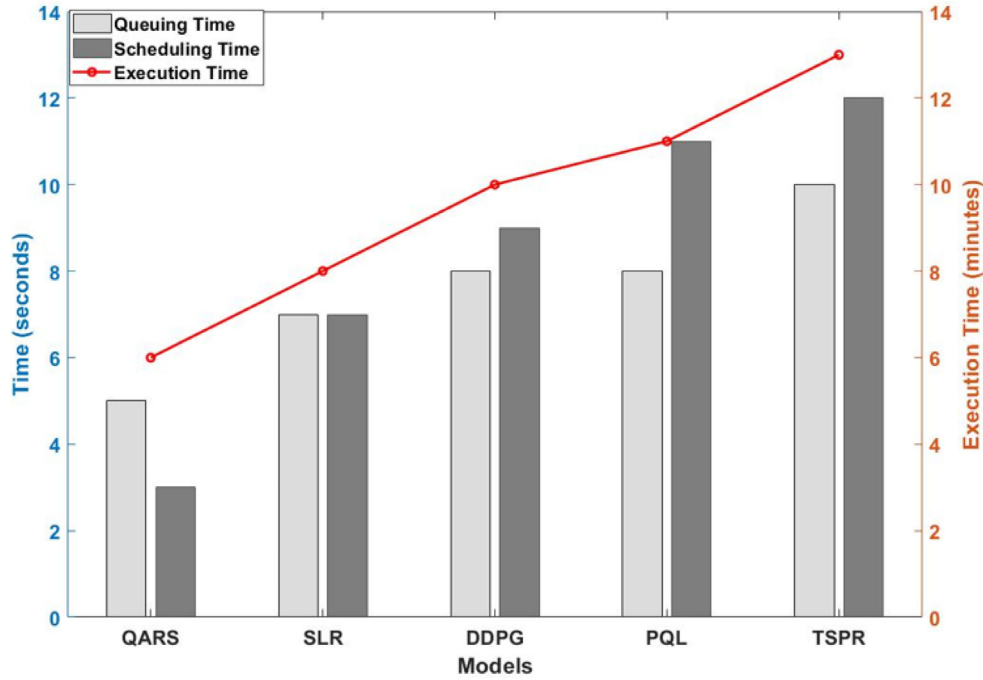


Fig. 6. Model overload ratio under different load environments.

Table 2. Comparison of response time performance fluctuation coefficient and mean of different models.

| Model | Average response time (ms) | Max response time (ms) | Min response time (ms) | Coefficient of variation |
|-------|----------------------------|------------------------|------------------------|--------------------------|
| QARS | 52 | 55 | 45 | 0.19 |
| SLR | 80 | 90 | 70 | 0.25 |
| DDPG | 75 | 85 | 65 | 0.27 |
| PQL | 70 | 80 | 60 | 0.29 |
| TSPR | 65 | 75 | 55 | 0.31 |

Response time is a standard indicator for measuring system stability. Table 2 records the response time of each model and calculates the corresponding fluctuation coefficient.

In Table 2, the average response time of the model in this paper is 52 ms; the maximum response time is 55 ms; the minimum response time is 45 ms; and the fluctuation coefficient is 0.19, all of which are better than those of other models. This demonstrates that the model in this paper performs well in terms of response time, that the system is highly stable, can adapt to dynamically changing environments, and exhibits a small fluctuation range. To further verify system stability across models, Table 3 presents energy consumption comparisons among models.

The average energy consumption of the model in this paper is 15.6 kWh; the peak energy consumption is 17.2 kWh; the minimum energy consumption is 14.3 kWh; the energy consumption difference is only 2.9 kWh. This shows that the model's energy consumption fluctuates little during resource scheduling, indicating good efficiency and stability.

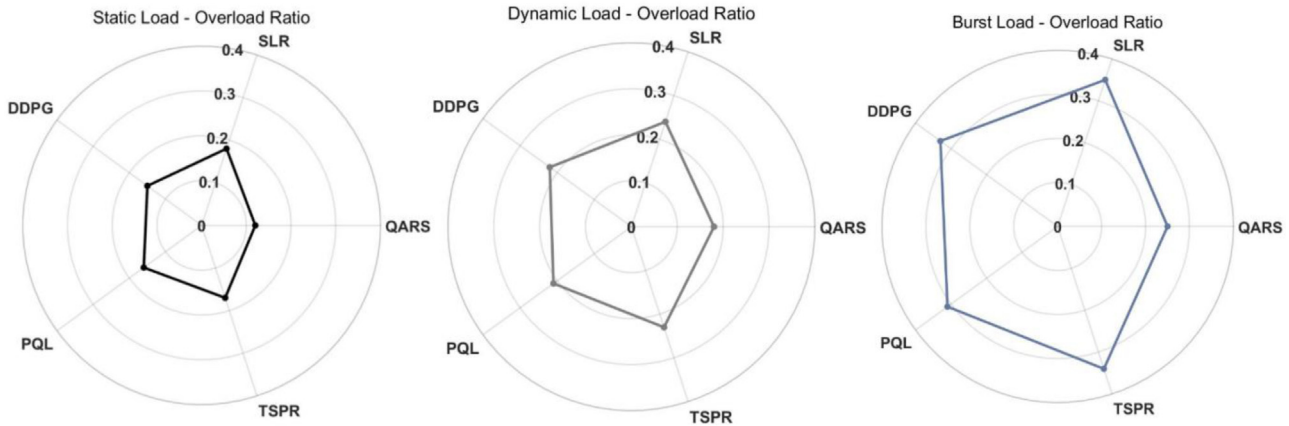
3.5 Resource overload ratio

Resource overload occurs when the virtual machine's resource demand exceeds its allocated resources, rendering it unable to operate normally. In the Q-learning adaptive resource scheduling strategy, the system continuously monitors the virtual machine's resource demand and allocation and dynamically adjusts resource allocation to prevent overload. Through continuous learning and optimization of the Q-learning algorithm, the system can gradually identify and avoid overload, allocate resources in the cloud computing environment rationally, and effectively prevent system crashes or performance degradation caused by overload. The resource overload ratio calculation formula is:

$$R_{\text{overload}} = \frac{\sum_{t=1}^T \mathbf{I}(U_i(t) > 1)}{T}. \quad (9)$$

Table 3. Energy consumption comparison of different scheduling strategies.

| Model | Average energy consumption (kWh) | Peak energy consumption (kWh) | Minimum energy consumption (kWh) |
|-------|----------------------------------|-------------------------------|----------------------------------|
| QARS | 15.6 | 17.2 | 14.3 |
| SLR | 19.5 | 21.8 | 16.9 |
| DDPG | 17.8 | 19.9 | 15.6 |
| PQL | 18.3 | 20.7 | 16.0 |
| TSPR | 20.7 | 23.4 | 18.5 |

**Fig. 7.** Model overload ratio under different load environments.

Among them, $I(U_i(t) > 1)$ is the indicator function when resources are overloaded, and T is the total number of time steps. A smaller R_{overload} means that resource scheduling is more reasonable.

In practical applications, load status and resource overload are closely related, and load status can be divided into three types: static, dynamic, and burst. Static load refers to the system load that remains stable for a long time; dynamic load refers to the system load that changes over time; burst load refers to the system load that increases sharply in a short period of time. Figure 7 shows the resource overload ratio of each model under different load environments.

Figure 7 shows the overload ratios for the five models (QARS, SLR, DDPG, PQL, TSPR) under static, dynamic, and burst load conditions. Under static load, the QARS model has the lowest overload ratio of 0.12, which is significantly lower than that of other models. Under dynamic load, QARS still maintains a low overload ratio (0.18), while other models fluctuate considerably. Under burst load, the overload ratio of QARS is 0.25, indicating that the model's overload ratio remains low across different load conditions, demonstrating strong stability and excellent resource scheduling performance.

4 Conclusions

This paper proposes a Q-learning based adaptive resource scheduling strategy, which can solve the problem of virtual machine fluctuation load and rapidly changing the resource demand in the cloud environment. Compared with the

traditional method SLR, DDPG, PQL and TSPR, the model has obvious advantages in the following indicators: In the aspect of resource utilization, the peak of CPU, memory and bandwidth utilization are 0.87, 0.72 and 0.74, respectively, and the stability of utilization curve is improved; In the aspect of load balancing, the load is always in the range of 0.45–0.55, no obvious fluctuation, and far exceed other models; In the aspect of system stability, the average response time is 52ms, fluctuation coefficient is 0.19, and average energy consumption is 15.6kWh. The fluctuation is small. Through the real-time feedback and continuous optimization of the scheduling strategy, not only the virtual machine resource utilization rate is improved, but also the balanced allocation of resources and the efficient execution of tasks are guaranteed. However, this model still has certain limitations in complex environment and multi-tenant environment. In the complex environment, because the state space is based on CPU, memory and bandwidth utilization and does not take into account the task attributes, it cannot meet the requirements of various kinds of tasks, and there will be a shortage of resources, which will cause the latency of real-time tasks to increase. In the multi-tenant environment, the resource isolation and priority cannot be guaranteed, so it is difficult to guarantee the service level of high-priority tenants, and the resource allocation is uneven, which may cause resource starvation for some tenants. In the future, we can optimize the model to solve more dynamic resource demand, enlarge the state space and insert tenant priority weights into reward function.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

Conflicts of interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Data availability statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Author contribution statement

Jiaxing Xu is responsible for the main writing of the article, Xiaofei Gao is responsible for collecting the data for the article, Ningyuan Gao is responsible for organizing the data of the article.

References

- [1] S. Dinesh, N. Kumar, Machine learning techniques in emerging cloud computing integrated paradigms: a survey and taxonomy, *J. Netw. Comput. Appl.* **205**, 103419 (2022)
- [2] M. Esh, S. Ghosh, Time series analysis-based models for predicting utilization of electronic resources, *J. Electron. Resour. Librariansh.* **35**, 182–194 (2023)
- [3] J.F. Torres, D. Hadjout, A. Sebaa, Deep learning for time series forecasting: a survey, *Big Data.* **9**, 3–21 (2021)
- [4] H. Hewamalage, C. Bergmeir, K. Bandara, Recurrent neural networks for time series forecasting: current status and future directions, *Int. J. Forecast.* **37**, 388–427 (2021)
- [5] X. Fu, Y. Sun, H. Wang, Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm, *Clust. Comput.* **26**, 2479–2488 (2023)
- [6] H. Aziza, S. Krichen, A hybrid genetic algorithm for scientific workflow scheduling in cloud environment, *Neural Comput. Appl.* **32**, 15263–15278 (2020)
- [7] M. Barkaoui, J. Berger, A new hybrid genetic algorithm for the collection scheduling problem for a satellite constellation, *J. Oper. Res. Soc.* **71**, 1390–1410 (2020)
- [8] S. AbdulRahman, H. Tout, A. Mourad, FedMCCS: Multi-criteria client selection model for optimal IoT federated learning, *IEEE Internet Things J.* **8**, 4723–4735 (2020)
- [9] A. Mahmood, J.L. Wang, A time and resource efficient machine learning assisted design of non-fullerene small molecule acceptors for P3HT-based organic solar cells and green solvent selection, *J. Mater. Chem. A.* **9**, 15684–15695 (2021)
- [10] R. Ghafari, F.H. Kabutarkhani, N. Mansouri, Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review, *Clust. Comput.* **25**, 1035–1093 (2022)
- [11] J. Du, B. Jiang, C. Jiang, Gradient and channel aware dynamic scheduling for over-the-air computation in federated edge learning systems, *IEEE J. Sel. Areas Commun.* **41**, 1035–1050 (2023)
- [12] F.M. Talaat, Effective deep Q-networks (EDQN) strategy for resource allocation based on optimized reinforcement learning algorithm, *Multim. Tools Appl.* **81**, 39945–39961 (2022)
- [13] P. Bhattacharya, F. Patel, A. Alabdulatif, A deep-Q learning scheme for secure spectrum allocation and resource management in 6G environment, *IEEE Trans. Netw. Serv. Manag.* **19**, 4989–5005 (2022)
- [14] I. AlQerm, J. Pan, Enhanced online Q-learning scheme for resource allocation with maximum utility and fairness in edge-IoT networks, *IEEE Trans. Netw. Sci. Eng.* **7**, 3074–3086 (2020)
- [15] S. Jiang, Y. Huang, M. Jafari, A distributed multiagent reinforcement learning with graph decomposition approach for large-scale adaptive traffic signal control, *IEEE Trans. Intell. Transp. Syst.* **23**, 14689–14701 (2021)
- [16] Y. Hu, M. Chen, W. Saad, Distributed multiagent meta learning for trajectory design in wireless drone networks, *IEEE J. Sel. Areas Commun.* **39**, 3177–3192 (2021)
- [17] Y. Zhang, B. Di, Z. Zheng, Distributed multi-cloud multi-access edge computing by multiagent reinforcement learning, *IEEE Trans. Wirel. Commun.* **20**, 2565–2578 (2020)
- [18] W. Yuan, Y. Li, H. Zhuang, Prioritized experience replay-based deep q learning: multiple-reward architecture for highway driving decision making, *IEEE Robot. Autom. Mag.* **28**, 21–31 (2021)
- [19] N.A. Shinkafi, L.M. Bello, D.S. Shu'aibu, Priority-based learning automata in Q-learning random access scheme for cellular M2M communications, *ETRI J.* **43**, 787–798 (2021)
- [20] H. Hu, D. Wu, F. Zhou, Intelligent resource allocation for edge-cloud collaborative networks: a hybrid DDPG-D3QN approach, *IEEE Trans. Veh. Technol.* **72**, 10696–10709 (2023)
- [21] J. Chen, H. Xing, Z. Xiao, A DRL agent for jointly optimizing computation offloading and resource allocation in MEC, *IEEE Internet Things J.* **8**, 17508–17524 (2021)
- [22] J. Tian, Q. Liu, H. Zhang, Multiagent deep-reinforcement-learning-based resource allocation for heterogeneous QoS guarantees for vehicular networks, *IEEE Internet Things J.* **9**, 1683–1695 (2021)
- [23] Y. Himeur, A. Alsalemi, F. Bensaali, Recent trends of smart nonintrusive load monitoring in buildings: a review, open challenges, and future directions, *Int. J. Intell. Syst.* **37**, 7124–7179 (2022)
- [24] S. Negi, M.M.S. Rauthan, K.S. Vaisla, CMODLB: an efficient load balancing approach in cloud computing environment, *J. Supercomput.* **77**, 8787–8839 (2021)
- [25] P. Memari, S.S. Mohammadi, F. Jolai, A latency-aware task scheduling algorithm for allocating virtual machines in a cost-effective and time-sensitive fog-cloud architecture, *J. Supercomput.* **78**, 93–122 (2022)
- [26] J. Hu, K. Yang, K. Wang, A blockchain-based reward mechanism for mobile crowdsensing, *IEEE Trans. Comput. Soc. Syst.* **7**, 178–191 (2020)
- [27] Z. Xu, D.K. Jain, S. Neelakandan, Hunger games search optimization with deep learning model for sustainable supply chain management, *Discov. Internet Things.* **3**, 10 (2023)
- [28] Y. Qwareeq, A. Sawwan, J. Wu, Maximum elastic scheduling of virtual machines in general graph cloud data center networks, *Cyber-Phys. Syst.* **10**, 283–301 (2024)
- [29] J. Moeyersons, S. Kerkhove, T. Wauters, Towards cloud-based unobtrusive monitoring in remote multi-vendor environments, *Softw.: Pract. Exp.* **52**, 427–442 (2022)

- [30] Q. Luo, S. Hu, C. Li, Resource scheduling in edge computing: a survey, *IEEE Commun. Surv. Tutor.* **23**, 2131–2165 (2021)
- [31] X. Xiong, K. Zheng, L. Lei, Resource allocation based on deep reinforcement learning in IoT edge computing, *IEEE J. Sel. Areas Commun.* **38**, 1133–1146 (2020)
- [32] H. Yang, J. Zhao, Z. Xiong, Privacy-preserving federated learning for UAV-enabled networks: Learning-based joint scheduling and resource management, *IEEE J. Sel. Areas Commun.* **39**, 3144–3159 (2021)
- [33] J. Wang, H. Zhou, Simplified dynamic modeling methodology for parallel shaft gear transmission system, *Mechanics Industry.* **26**, 28 (2025)
- [34] W. Sun, P. Wang, N. Xu, Dynamic digital twin and distributed incentives for resource allocation in aerial-assisted internet of vehicles, *IEEE Internet Things J.* **9**, 5839–5852 (2021)
- [35] W. Sun, J. Liu, Y. Yue, Joint resource allocation and incentive design for blockchain-based mobile edge computing, *IEEE Trans. Wirel. Commun.* **19**, 6050–6064 (2020)
- [36] Y.S. Kong, S. Abdullah, D. Schramm, S.S.K. Singh, Determining optimal suspension system parameters for spring fatigue life using design of experiment, *Mechanics Industry.* **20**, 621 (2019)

Cite this article as: J. Xu, X. Gao, N. Gao, Adaptive scheduling strategy for cloud computing resources based on Q-learning algorithm, *Mechanics & Industry* **27**, 4 (2026), <https://doi.org/10.1051/meca/2025034>