Journal of Science and Transport Technology
Journal homepage: https://jstt.vn/index.php/en

Reinforcement Learning-Based Framework for Optimal Task Scheduling in Cloud Computing

KrishnaRao Patwari[1,*], Raghvendra Kumar[2], J.S.V.R.S. Sastry[3], J.S. Ishaani Priyadarshini[4], Tran Ha Thanh[5], Vu Trong Hieu[6]

[1]Department of Computer Science and Engineering, Gandhi Institute of Engineering and Technology University, Gunupur, Odisha 765022, India

[2]Department of Computer Science and Engineering, Gandhi Institute of Engineering and Technology University Gunupur, Odisha and 765022, India

[3]Department of Computer Science and Engineering, Gandhi Institute of Technology and Management University, Hyderabad, Telangana 502329, India

[4]Computer Science Department, Edmonds College, Washington and 98020 USA

[5]University of Transport Technology, Hanoi 100000, Vietnam

[6]Geotechnical and Artificial Intelligence research group, University of Transport Technology, Hanoi 100000, Vietnam

Article info
Type of article:
Original research paper

DOI:
https://doi.org/10.58845/jstt.utt.2026.en.6.1.243-266

***Corresponding author:**
Email address:
Krishnarao.patwari@giet.edu

Received: 26/11/2025
Received in Revised Form: 30/01/2026
Accepted: 01/03/2026

Abstract: Cloud computing enables the execution of large-scale computing tasks in a pay-per-use manner, allowing users worldwide to submit diverse workloads to cloud infrastructures. In this context, efficient task scheduling plays a crucial role in ensuring quality of service and optimal resource utilization. Existing cloud task scheduling approaches are largely based on heuristic or learning-based methods; however, heuristic approaches lack adaptability to dynamic runtime conditions.

To address these limitations, this paper proposes a reinforcement learning–based framework for optimal task scheduling in cloud computing. The scheduling problem is modeled as a Markov Decision Process (MDP) and solved using a reinforcement learning–based optimal task scheduling algorithm (RL-OTS) that learns scheduling policies through continuous interaction between the environment and the decision-making agent. The proposed framework dynamically selects scheduling actions based on the observed system state to improve scheduling efficiency under varying workload conditions.

An empirical evaluation conducted using heterogeneous workloads of 2000, 5000, and 10000 jobs demonstrates that the proposed RL-OTS algorithm consistently outperforms several state-of-the-art scheduling methods, achieving success rates of 81.10%, 80.10%, and 80.20%, respectively. These results highlight the effectiveness of reinforcement learning for adaptive and intelligent task scheduling in cloud computing environments.

Keywords: Cloud Computing, Task Scheduling, Reinforcement Learning, Resource Provisioning, Resource Optimization.

1. Introduction

Technologies like cloud computing also enable a computing environment where it is possible to execute jobs given by users across the globe. There might be millions of users across the world who wants to perform their jobs by cloud infrastructure [1]. Provided this complexity and heterogeneity in terms of workload and also resources, it is essential to have mechanisms for efficient task scheduling. Task scheduling is when an algorithm makes appropriate decisions in scheduling a given set of jobs in the available resources in the cloud infrastructure. There has been continuous research on task scheduling. Many heuristic approaches could perform task scheduling based on specific procedures. However, heuristics-based approaches lack dynamism in understanding the runtime situations and making well-informed decisions [2]. In highly dynamic cloud environments, where workloads and resource availability change continuously, existing heuristic and static scheduling approaches fail to adapt in real time, leading to suboptimal scheduling decisions and degraded system performance. To overcome this problem, learning-based approaches came into existence in machine learning and deep learning or, in other words, artificial intelligence.

Learning-based approaches are found efficient as they can understand the runtime environment, including its dynamic changes while making task scheduling decisions. Mainly reinforcement learning is very important and valuable to meet such runtime complex situations while making optimal task scheduling decisions. In this work, task scheduling is modeled as a reinforcement learning problem using Q-learning principles, where a deep neural network (DNN) is employed to approximate the action–value function. The learning process is guided by a reward function designed to reflect scheduling objectives such as task completion efficiency and deadline satisfaction. The actor and environment interaction has state space and action space with an iterative process toward making more appropriate scheduling decisions. Reinforcement learning is a widely used approach. There are many existing methods based on learning-based phenomena.

Several learning-based approaches have been explored for cloud task scheduling to improve adaptability and decision-making under dynamic conditions. However, many existing methods rely on simplified assumptions or lack scalability when handling heterogeneous workloads and rapidly changing resource availability. These limitations highlight the need for a more robust and adaptive reinforcement learning–based scheduling framework that can effectively learn optimal scheduling policies in complex cloud environments. From the literature, it was observed that there is a need for optimal task scheduling in cloud computing by exploiting reinforcement learning-based approaches. The paper's contributions could potentially answer several research questions, such as: What is reinforcement learning? How can reinforcement learning be applied for efficient task scheduling in a cloud environment? What is the performance level of a reinforcement learning-based approach compared to existing heuristic or other learning-based approaches? This study aims to address the challenge of optimal task scheduling in dynamic cloud environments, where heterogeneous workloads and fluctuating resource availability demand adaptive and real-time scheduling mechanisms. From an infrastructure systems perspective, efficient cloud task scheduling is a critical enabler for large-scale digital infrastructure services. Modern infrastructure platforms—such as intelligent transportation systems (ITS), traffic management backends, urban sensing infrastructures, and infrastructure IoT monitoring systems—rely on cloud and edge–cloud computing environments to process continuous streams of data in real time. In such systems, delays or inefficient resource allocation at the scheduling level can directly impact service reliability,

responsiveness, and operational safety. Therefore, adaptive and learning-based scheduling mechanisms are increasingly required to support infrastructure services that operate under dynamic workloads, strict latency requirements, and geographically distributed computing resources. Our contributions to this paper are as follows. The performance of the proposed scheduling approach is evaluated using key metrics including success rate, scheduling efficiency, and workload handling capability under varying job sizes. These performance improvements demonstrate the practical applicability of reinforcement learning–based scheduling for improving adaptability and efficiency in real-world cloud computing systems.

We proposed a reinforcement learning-based framework for optimal task scheduling.

We proposed an algorithm known as Reinforcement learning-based Optimal Task Scheduling (RL-OTS), which requires action space and state space with the interaction between environment and actor to make well-informed decisions in task scheduling.

We evaluated our methodology with a prototype application using different workloads for empirical study.

Key Contributions:

Unlike existing deep reinforcement learning–based schedulers, the proposed RL-OTS framework introduces a lightweight Deep Q-Learning formulation with a resource-aware state design and cost-driven reward modeling that directly captures execution time, waiting time, and resource utilization. RL-OTS is designed under minimal deployment assumptions, does not require workload prediction or labeled data, and learns scheduling policies online, making it suitable for highly dynamic cloud environments.

The remainder of the paper is structured as follows. Section 2 reviews prior works related to task scheduling in cloud computing. Section 3 presents the proposed methodology, which includes the problem definition, our framework, and the proposed algorithm for efficient task scheduling in the cloud. Section 4 presents our experimental results related to task scheduling. Section 5 provides a discussion of the proposed research besides giving limitations of the study. Section 6 concludes our work and provides scope for future research.

2. Related Work

The section reviews prior works on task scheduling in cloud computing based on learning-based phenomena. Sohrabi et al. [1] proposed a hybrid approach combining Reinforcement Learning (RL) with Genetic Algorithms (GA) to optimize makespan, usage, and load balancing in scientific workflows. While this integration showed performance improvements, it relies heavily on predefined heuristics and lacks the flexibility to adapt to dynamic and unpredictable cloud environments in real time. Moreover, the computational overhead introduced by combining RL with GA can hinder its scalability in large-scale systems. Liao et al. [2] Multi-task Deep Reinforcement Learning (MDTS) is used to enhance parallel Task Scheduling in the Internet of Vehicles, providing better results under different conditions. Yu et al. [3] addressed task planning in Human-Robot Collaboration by proposing a novel model based on a chessboard structure and implementing a Deep Q-Network Multi-Agent Reinforcement Learning (DQN-MARL) framework. Although this approach is effective in controlled and structured environments, it falls short in generalizing to the inherently dynamic, multi-tenant, and resource-constrained nature of cloud systems. Additionally, the reliance on a fixed game-like strategy limits its adaptability to diverse real-world scheduling challenges, such as heterogeneous workloads and fluctuating demand. Zhu et al. [4] Deep reinforcement learning techniques like Sharer are used to solve resource scheduling problems in cloud manufacturing and offer efficient solutions for various scenarios Garg et al. [5] focused on energy efficiency by designing a supervised neural network-based autonomous scheduler for cloud data centers. While this

approach reduces energy consumption, it is built on supervised learning, which necessitates large volumes of labeled training data. In dynamic cloud environments where job characteristics constantly change, maintaining updated labeled datasets is impractical. Furthermore, the model's dependence on static training data reduces its ability to adapt to new patterns or unexpected workloads.

Guo et al. [6] A deep reinforcement learning system for cloud resource management called DeepRM Plus performs 375% faster at convergence than DeepRM. The method provides a basic framework for effective solutions and demonstrates the potential of combining deep reinforcement learning and imitation learning for challenging resource scheduling issues. Additional resource management-related reinforcement learning techniques might be investigated in future research. Hakami et al. [7] online reinforcement learning, a proactive method of job scheduling, reduces response times and boosts resource efficiency in changing cloud demands. The suggested algorithm works better than the current one. Ding et al. [8] Using a centralized dispatcher and a scheduler based on Q-learning, the QEEC framework tackles energy efficiency in cloud task scheduling. It is preferable to other policies, as demonstrated by experiments. Future research will focus on scalability and other heuristics. Asghari et al. [9] suggested a multi-agent system based on Q-learning that optimizes cloud job scheduling and resource provisioning, increasing resource efficiency and cutting expenses. Future research can investigate more methods of enhancement. Gazori et al. [10] IoT applications that rely on fog confront difficulties managing growing volumes of data. Techniques for Transfer Learning (TL) and Prioritized Experience Replay (PER) may be investigated in future research for advancement. Creating a system for rescheduling tasks that need long wait periods is also recommended.

Tong et al. [11] state that in cloud computing, task scheduling is essential to overall performance. Experiments in WorkflowSim validate the superior learning and efficiency of the proposed deep Q-learning algorithm, DQTS. Creating a model of energy use is one of the upcoming tasks. Yaghmaee et al. [12] for large computations, cloud computing, a distributed environment, improve efficiency. Regarding job scheduling, resource provisioning, and overall user service quality, the suggested RL-based architecture performs admirably. Karri et al. [13] Effective techniques are needed due to the intricacy of task scheduling. By outperforming existing algorithms, Deep Q-learning-based DRLBTSA lowers makespan, SLA violations, and energy usage. Jia et al. [14] presented a unique edge computing work scheduling method based on DRL. The success of the proposed approach is demonstrated by the significant improvement in energy usage and SLA violation compared to baselines. Sohrabi et al. [15] presented a hybrid approach for cloud resource management called MO-CRAML, which combines reinforcement learning and coral reef optimization. Its better utilization and load-balancing performance are shown by the experiments.

Sayyed et al. [16] Enhanced Multi-Verse Optimizer (EMVO), which outperforms Particle Swarm Optimization (PSO) and Multi-Verse Optimizer (MVO) in terms of makespan time and resource consumption as a superior task scheduler in cloud computing. Gari et al. [17] it's the potential for auto-scaling in cloud computing; reinforcement learning (RL) provides flexible, dynamic resource management. Future research is discussed, and RL-based proposals are compared in the survey. Hu et al. [18] presented a Deep RL-based algorithm that optimizes energy and latency for task offloading in the Power Internet of Things (PIoT). The results show improved system utility. The algorithm can be deployed with ease. Li et al. [19] investigated the use of deep reinforcement learning (DRL) to save energy in cloud computing and smart grids. It presents incredibly efficient DRL hardware with stochastic computing. Liu et al. [20] looked into cloud robotics and suggested a resource allocation plan based on reinforcement

learning (RL). Numerical studies show that RL achieves higher autonomy, efficiency, and utility levels than Greedy Allocation (GA). For practical applications, further research will focus on accelerating convergence and improving utility functions.

Wu et al. [21] investigated to facilitate real-time data-driven changes; this work presents a task-scheduling technique for UAV clusters that is based on reinforcement learning. Real-time learning appropriateness is emphasized, and channel allocation difficulties are addressed. Wahab et al. [22] presented BigTrust Scheduling, which minimizes costs and makespan while optimizing extensive data job scheduling in cloud computing. The proposed method shows promise. Mei et al. [23] tackled the difficulties associated with cloud computing task scheduling, presenting DDQN-TS as a unique approach and showcasing effective QoS performance and adherence. Peng et al. [24] proposed a DRL-based approach to handle conflicts in cloud service optimization between energy costs and service quality. Subsequent research delves into intricate scheduling issues and the amalgamation of load predictions. Guo et al. [25] aimed to optimize energy consumption and meet latency constraints by introducing a job scheduling method for Vehicular Edge Computing (VEC) networks enabled by imitation learning. The suggested approach exceeds benchmarks by more than 50% when imitation learning is used alongside expert demonstrations.

Kimpan et al. [26] surpassed other algorithms in various performance measures and datasets by implementing a multi-objective task scheduling strategy for cloud computing that integrates ABC and Q-learning for optimization. Jasser et al. [27] presented DRLPPSO, a scheduling technique that combines Parallel Particle Swarm Optimization (PPSO) and Deep Reinforcement Learning (DRL) to improve cloud load balancing efficiency. It shows increased speed, accuracy, and rewards. Future research will compare this algorithm to others and investigate a hybrid resource management strategy. Yan et al. [28] focused on automated guided vehicle-based cloud manufacturing (CM) production scheduling in real-time, using a distributed dueling deep Q network (D3QN) for optimization. The outcomes demonstrate the effectiveness of D3QN and the possibility for better real-time decision-making in CM. The goal of future research is to investigate distributed real-time scheduling with digital twins while taking CM uncertainties into account. Li et al. [29] presented a DT-enabled MEC network for mobile applications that operate in real-time. It efficiently reduces latency by integrating DNN inference tasks with an A3C-based resource allocation approach. The suggested technique works well in continuous DNN task settings, outperforming AC and DQN. Ma et al. [30] analyzed the role of IoE in real-time applications and suggested a DRL-TSS model for effective scheduling in edge-cloud Internet of Things systems. Superior performance and efficiency are demonstrated via evaluation. Subsequent research endeavors are geared toward improving deep learning models for IoE scheduling.

Sudheer et al. [31] proposed that DRLBTSA effectively optimizes resource allocation while minimizing makespan, violations, and energy usage. Cloud jobs are unpredictable, which makes scheduling them challenging. Sadatdiynov et al. [32] assisted in overcoming the constraints of managing SMD data by improving SMD capabilities through task offloading and bringing computing closer together. An overview is given of several optimization strategies for efficient compute offloading. Chen et al. [33] implemented hierarchical edge clouds to improve the quality of task services and the usage of resources. JNNHSP combines heuristic and neural network techniques to achieve scheduling efficiency that is superior to alternatives. Zhou et al. [34] prospered resource management as a prerequisite for cloud computing in Web 2.0. Deep reinforcement learning is a powerful tool for solving complex scheduling

problems. Wang et al. [35], although it confronts server overload, edge/fog computing satisfies IoT needs. DRLIS, a DRL-based scheduler, outperforms alternatives and maximizes load balancing and IoT app response time.

Tejer et al. [36] planned for tasks that are crucial in many real-life circumstances. Machine learning techniques like reinforcement learning are replacing older approaches. For robustness and efficacy, parameter tuning is vital. Wu et al. [37] reduced latency STIN's satellite edge computing promises to deliver IoT services. By offloading tasks during service transfer, the DRL-based approach seeks to address related problems. Salehnia et al. [38] (2024), to communicate with one another and collect data for various purposes, Internet of Things devices utilize unique addressing. Edge Computing links the cloud with the Internet of Things, reducing delays.

Table 1 shows a summary of the literature findings. From the literature, it was observed that the learning-based approaches have mechanisms to learn runtime situations and make appropriate decisions. Reinforcement Learning (RL) is more suitable for task scheduling in cloud computing. Several existing works [1], [3], [5] highlight improvements in task scheduling; however, they often rely on static assumptions or structured environments.

Comparison with Closest Deep Reinforcement Learning Schedulers

Compared to DDQN-TS, which optimizes response time using dual Q-networks under fixed QoS assumptions, RL-OTS employs a single Deep Q-Network with a resource-aware state representation and cost-based reward function, enabling adaptive scheduling under heterogeneous and fluctuating cloud workloads.

Table 1. Summary of literature findings

Reference	Method	Advantages	Limitation
[1]	RL	Improved makespan, load balancing, and resource utilization, besides being deadline-aware	In the future, authors intended to use multiple agents-based methods.
[3]	Q-learning-based method DQN-MARL	Optimal task scheduling	Needs to be improved with more complicated tasks with multi-agent coordination
[5]	ML approach	Energy efficiency, improved makespan, and reduced costs	Workflow scheduling is yet to be explored.
[9]	RL	Optimal resource utilization, improving makespan, energy efficiency, and cost reduction	Speeding up convergence with game theory and neural networks is to be carried out.
[10]	DRL	Energy conservation, better resource utilization, a decrease of state-action space, and getting rid of SPoF issues.	The prioritized Experience Replay technique is yet to be explored to improve the RL process.

Table 1. (continued)

Reference	Method	Advantages	Limitation
[15]	ML with long–term	Improves load balancing, resource utilization	Improving the other related parameters, such as the user cost model and the energy consumption of cloud resources, can be considered another future work using coral reef optimization for cloud resource management.
[23]	DRL-based DDQN-TS method	Action Q-value is optimized, and acceleration of convergence besides the lowest average task response.	Authors intend to apply their framework to complex cloud environments in the future
[26]	MOABCQ approach	Makespan reduction, load balancing, cost reduction, and improving throughput	In the future, authors intend to improve their model for edge-cloud environments.
[28]	DRL and a multitask generative hyperheuristic approach	A distributed approach to scheduling improves learning efficiency and collaboration among agents, leading to scheduling efficiency	In the future, authors intend to improve their framework by considering digital twins in the learning process.
[30]	Deep-Reinforcement Learning-enhanced Two-Stage Scheduling (DRL-TSS)	Better approximation ratio, improved makespan, support for edge-cloud	The authors intend to improve their methodology with deep learning-based scheduling optimization

In contrast to DRLBTSA, which integrates deep reinforcement learning with heuristic task-binding strategies and requires predefined SLA constraints, RL-OTS learns scheduling decisions directly from environment feedback without relying on heuristic bias or prior workload labeling.

Unlike MOABCQ, which combines reinforcement learning with swarm-based optimization and introduces additional computational overhead, RL-OTS follows a lightweight learning setup with fewer hyperparameters, making it more suitable for scalable and real-time cloud scheduling scenarios.

RL-OTS dynamically adapts to heterogeneous workloads and resource fluctuations, overcoming the rigid learning frameworks and limited real-time adaptability of earlier methods.

While prior works achieved significant improvements in task scheduling, they often relied on either static assumptions or required labeled data, limiting adaptability. In contrast, RL-OTS

autonomously learns from the environment, handles dynamic workloads, and exhibits superior adaptability without requiring pre-labeled data.

3. Proposed Methodology

This section presents our proposed methodology for reinforcement learning-based task scheduling in cloud computing. It throws light on the problem definition, the proposed framework, the proposed algorithm, and the evaluation methodology.

Provided a set of tasks obtained from various users across the globe to the cloud infrastructure, developing a based framework for understanding the runtime situation and performing optimal scheduling of the given tasks is the challenging problem considered.

3.1. Proposed Framework

The proposed optimal task scheduling approach is based on RL. The system model is illustrated in Fig. 1. In a cloud computing environment, jobs or tasks arrive from users across the globe, and those tasks are to be scheduled in cloud infrastructure. Scheduling the tasks with the highest efficiency is a significant problem to be solved. There might be millions of users sending jobs to cloud infrastructure for execution.

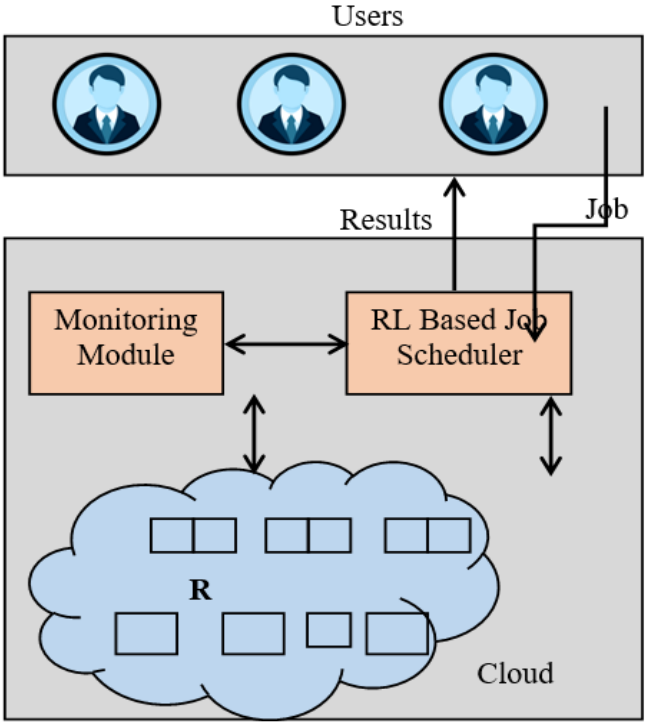


Fig. 1. Our system model for optimal task scheduling

It is crucial to ensure that all the tasks are correctly scheduled and executed as expected. Towards this end, there are many heuristics-based approaches that could provide scheduling capabilities. However, with the emergence of AI, learning-based approaches have come into existence. Learning-based approaches are capable of understanding the runtime environment and availability of resources before making scheduling decisions.

Reinforcement learning is one of the machine learning techniques in which an agent will monitor the environment and learn from time to time to make well-informed decisions toward task scheduling. In other words, the proposed system for optimal task scheduling is based on RL.

RL-based job scheduling is responsible for executing the underlying mechanism of our framework and ensuring that the tasks are scheduled properly and executed as per the user expectations.

The cloud infrastructure has resources available. Based on the freely available resources and the number of tasks in hand deciding on task

scheduling is very dynamic. This is the reason heuristic algorithms fail to achieve optimal performance.

For the same reason, we proposed a learning-based approach based on RL toward efficient task scheduling. The proposed system has a monitoring module that monitors the cloud resources and coordinates with the job scheduler to help it in optimal scheduling.

In the process of RL, the cloud computing environment is monitored and an agent obtains its state before making a scheduling decision. Based on the state of the environment and the availability of resources besides the jobs in hand, there is an action taken by the agent, and it takes feedback for the action. Therefore, it is an iterative process between the agent and the environment with the action and state space for optimal scheduling decisions. Fig. 2 illustrates the process involved in reinforcement learning.

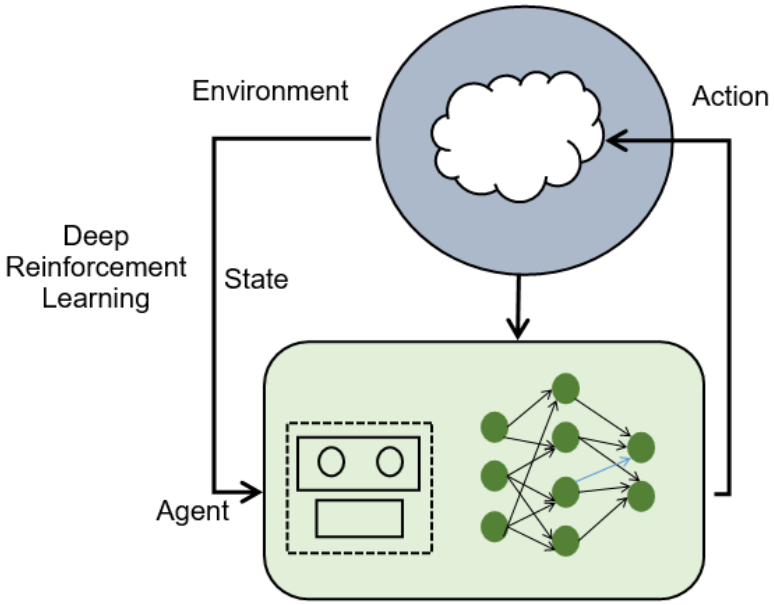


Fig. 2. Illustrate the process of reinforcement learning

Table 2. Notations Used in the Proposed System

Symbol	Meaning
S	State space representing resource availability and task assignments
A	Action space representing scheduling decisions
R	Reward function capturing scheduling efficiency
π(a	s)
γ	Discount factor for future rewards
Q(s, a)	Value function of taking action a in state s
Y	Transition probability from current to next state

RL is preferred in the proposed metrology for optimal task scheduling due to its ability to understand a very dynamic runtime environment with heterogeneous tasks and heterogeneous resources available in the cloud infrastructure. A continuous process in which an agent makes an action and the environment gives feedback leads to optimal task-scheduling decisions. Table 2 shows important notations.

3.2. RL Based Approach for Task Scheduling

Reinforcement learning (RL) aims to learn an optimal decision policy by enabling an agent to interact with its environment and improve its actions through trial-and-error feedback. In this work, cloud task scheduling is formulated as a Markov Decision Process (MDP) and solved using

a Deep Q-Network (DQN). Although the learning objective follows Q-learning principles, traditional tabular Q-learning is not suitable for cloud scheduling because the state–action space is large, continuous, and highly dynamic. Therefore, the action–value function $Q(s,a)$ is approximated using a deep neural network, which corresponds to Deep Q-Learning (DQN) in our implementation.

Let $V = \{vm_1, vm_2, \dots, vm_m\}$ denote the set of available virtual machines (VMs), and let $E = \{e_1, e_2, \dots, e_k\}$ denote the set of incoming tasks/jobs submitted by users. At each time step t, the scheduler observes the system state $s_t \in S$, which summarizes the task queue and VM resource availability, and selects an action $a_t \in A$, where an action represents assigning a task to a feasible VM according to the scheduling policy $\pi(a \mid s)$. After executing a_t, the environment transitions to the next state s_{t+1} and returns an immediate reward r_t, defined in this work as a negative cost reflecting scheduling objectives (e.g., execution time, waiting time, and resource usage). The learning goal is to identify an optimal policy π^* that maximizes the expected cumulative discounted reward (equivalently, minimizes long-term scheduling cost). Based on this formulation, the following subsections define the state space, action space, and reward function used in RL-OTS.

3.2.1. State Space

At each decision step t, the environment state $s_t \in S$represents the current cloud scheduling condition and is encoded as a feature vector that summarizes both task-level and VM-level information. Specifically, s_t includes: (i) the resource requirements of the selected (or arriving) task e_i (e.g., required CPU, RAM, bandwidth, and storage), and (ii) the current available resources of each virtual machine $vm_j \in V$ (available CPU, RAM, bandwidth, and storage), along with the current allocation/queue status if applicable.

A VM vm_j is considered feasible for task e_i only if its available resources satisfy the task's constraints (defined later in this section). The DQN observes s_t and uses it to estimate $Q(s_t, a)$ for candidate scheduling actions, enabling the scheduler to select an appropriate VM under dynamic workload and resource conditions.

3.2.2. Action Space

At each scheduling step t, the agent selects an action $a_t \in A$ that represents assigning the current task e_i to one of the available virtual machines. Therefore, the action space is a discrete set of VM-selection actions, i.e.,

$A = \{a^{(1)}, a^{(2)}, \dots, a^{(m)}\}$,

where action $a^{(j)}$ denotes "assign task e_i to VM vm_j".

For implementation, this selection can be represented as a one-hot vector of length m, where the j-th element is 1 if vm_j is selected and all other elements are 0. For example, $(0,1,0,0,\dots,0)$ indicates that the task is assigned to the second VM. Actions that violate feasibility constraints (i.e., insufficient CPU/RAM/bandwidth/storage) are excluded from the valid action set at time t, ensuring that only resource-satisfying assignments are considered during decision-making.

3.2.3. Reward

We define the immediate reward at decision step t as a negative scheduling cost so that maximizing cumulative reward is equivalent to minimizing long-term scheduling cost. When task e_i is assigned to virtual machine vm_j at time step t, the reward is

$$r_t = -C(e_i, vm_j). \tag{1}$$

The instantaneous cost $C(e_i, vm_j)$ combines time-related cost and resource-pressure cost as

$$C(e_i, vm_j) = \alpha U_j (T_{ij} + W_{ij}) + \sum_{k \in \{\mathrm{CPU,RAM,BW,ST}\}} \beta_k \phi_k(e_i, vm_j), \tag{2}$$

where U_j is the unit cost of VM vm_j (currency per second), T_{ij} is the execution time of task e_i on VM

vm_j (seconds), and W_{ij} is the waiting/queueing time before the task begins execution on vm_j (seconds). The term $\alpha$is a scaling factor to keep the reward magnitude numerically stable. The resource-pressure term $\phi_k(e_i, vm_j)$ is defined as the normalized demand ratio

$$\phi_k(e_i, vm_j) = \frac{req_k(e_i)}{avail_k(vm_j)}, \quad (3)$$

where $req_k(e_i)$ is the amount of resource type k requested by task e_i, and $avail_k(vm_j)$ is the currently available amount of the same resource on VM vm_j. For clarity of units: req_{CPU} and $avail_{\text{CPU}}$ are measured in CPU cores (or normalized vCPU units), req_{RAM} and $avail_{\text{RAM}}$ are in GB, req_{BW} and $avail_{\text{BW}}$ are in Mbps, and req_{ST} and $avail_{\text{ST}}$ are in GB. The weights β_k are non-negative coefficients that control the relative importance of each resource type. All resource ratios are dimensionless and are normalized to avoid domination of the learning signal by any single component.

The DQN learns the action–value function $Q(s, a)$ that maximizes the expected discounted return:

$$Q^*(s,a) = \mathbb{E}\left[r_t + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a\right], \quad (4)$$

where $\gamma \in (0,1)$ is the discount factor and the transition from s_t to s_{t+1} is determined by the scheduling environment.

3.3. The Proposed Algorithm

Algorithm: Learning based Optimal Task Scheduling (RL-OTS)
Inputs:
A set of tasks denoted as E = $\{e_1, e_2 \ldots . e_n\}$
A set of VMs denoted as V = $\{v_1, v_2 \ldots . v_n\}$
Output: Tasks are scheduled in the cloud with optimization

1. Q(s, a)←initial state and action space
2. For each episode
3. s←ObtainState()
4. a←ChooseAction(s, policy)
5. For each step t in episode e
6. a←ChooseAction(t, s, policy)
7. r←ObtainReward()
8. s'←UpdateState(s)
9. Verify optimal value function using Eq. 4
10. Perform Q-table approximation using DQN
11. Apply resource criterion using Eq. 5
12. policy'←SchedulingPolicyUpdate(policy) //using Eq. 6
13. minimalReward←ComputeMinimalReward()
14. Update action a←a'
15. Update state s←s'
16. End For
17. End For
18. End

Algorithm 1. Learning-based Optimal Task Scheduling (RL-OTS)

We proposed an algorithm known as Reinforcement Learning based Optimal Task Scheduling (RL-OTS) which has required action space and state space with the interaction between environment and actor towards making well informed decisions in task scheduling.

Algorithm 1 (RL-OTS) is designed to optimize the scheduling of tasks in a cloud computing environment. The algorithm operates by leveraging a set of tasks (E) and a set of virtual machines (V). The core of the algorithm is a reinforcement learning approach that iterates through episodes, where each episode consists of multiple steps. At the beginning of each episode, the algorithm obtains the current state (s) and chooses an action (a) based on a policy. For each step within an episode, the algorithm selects an action, obtains a reward (r), updates the state (s'), and verifies the optimal value function using a specified equation (Eq. 4). A deep neural network (DNN) is employed to approximate the Q-table, which is a fundamental structure in reinforcement learning that captures the expected rewards for each state-action pair. The algorithm also applies a resource criterion using another equation (Eq. 5), which is likely related to the allocation of resources based on the state and action. Following this, the scheduling policy is updated (Eq. 6), which reflects the learning process and adaptation to the environment. The minimal reward is computed, which could serve as a baseline or threshold for the reward function. The actions and states are then updated accordingly. The process continues iteratively, with the algorithm learning from the rewards and state transitions to improve the task scheduling policy. The ultimate goal is to find an optimal scheduling policy that maximizes the rewards, which could be related to efficiency, resource utilization, or other performance metrics. In summary, the RL-OTS algorithm uses reinforcement learning techniques, including a deep neural network for Q-table approximation, to optimize task scheduling in cloud computing. It operates through iterative learning episodes, updating the policy based on rewards and state transitions to achieve an optimal scheduling strategy.

3.4. Implementation Details (Deep Q-Network Configuration and Training)

RL-OTS is implemented using a Deep Q-Network (DQN) rather than tabular Q-learning. The term “Q-table” is used only in a conceptual sense to denote the action–value function Q(s,a). In the actual implementation, Q(s,a) is approximated by a deep neural network because the cloud scheduling state–action space is high-dimensional, continuous, and changes dynamically with workload and resource availability.

The Q-function approximator is implemented as a multilayer perceptron (MLP) with fully connected layers. The input to the network is the state vector s_t, which encodes the current task requirements and the available resources of the candidate virtual machines. The network contains three hidden layers with 128, 64, and 32 neurons, respectively. ReLU activation is used in all hidden layers, and the output layer uses a linear activation to produce a Q-value for each discrete VM-selection action.

Training is performed by minimizing the mean squared error between the predicted Q-values and the target Q-values computed from the Bellman optimality update. The Adam optimizer is used with a learning rate of 0.001. The discount factor is set to $\gamma = 0.95$ to balance short-term scheduling gains with long-term cumulative improvements.

Action selection follows an ε-greedy exploration strategy. The exploration rate ε is initialized at 1.0 and decays gradually to 0.1 across training, enabling the agent to explore scheduling options early and increasingly exploit learned scheduling policies as training progresses. To stabilize learning and reduce correlations between consecutive updates, experience replay is employed. Transitions (s_t, a_t, r_t, s_{t+1}) are stored in a replay buffer, and the DQN parameters are updated using mini-batches of size 64 sampled

from this buffer.

The agent is trained for 500 episodes for each workload configuration. Each episode consists of scheduling tasks sequentially until all tasks in the workload are allocated and executed, or a predefined termination condition is reached. If a target network is used for additional stability, it is updated periodically to compute more stable targets during training; otherwise, targets are computed directly from the current network. These reported settings are provided to ensure reproducibility and to clearly distinguish RL-OTS as a DQN-based deep reinforcement learning approach rather than a tabular Q-learning method.

4. Experimental Results

This section presents the results of our empirical study with the prototype application, considering different workloads for task scheduling in the cloud. The empirical study makes various observations to evaluate the performance of the proposed task scheduling algorithm and compare it with the state of the art.

Each experiment was repeated 10 times using different random seeds to account for variability in learning and workload generation. For each workload setting, we report the mean and standard deviation of the success rate across runs. Since confidence intervals are not explicitly tabulated in this manuscript, we report variability using standard deviation rather than claiming 95% confidence intervals.

4.1. Datasets and Environment

Experiments are made with different workloads. The empirical study involved 2000 jobs in the first experiment, 5000 jobs in the second experiment, and 10,000 jobs in the final experiment. In each experiment given number of tasks are scheduled with the help of the proposed algorithm.

The reinforcement learning environment is modeled as a simulated cloud computing system in which the scheduler acts as an agent interacting with the cloud infrastructure. The state space represents the current status of the system, including the task queue characteristics, virtual machine availability, and resource utilization levels.

The action space consists of selecting an appropriate virtual machine for executing an incoming task based on the observed system state. Upon executing an action, the environment transitions to a new state depending on task execution and resource allocation outcomes.

The reward signal is computed based on scheduling objectives such as task completion efficiency, deadline satisfaction, and effective resource utilization, guiding the learning agent toward optimal scheduling decisions.

An episode consists of scheduling a sequence of tasks until all tasks in the workload are processed or a predefined termination condition is met. This environment setup enables the agent to learn adaptive scheduling policies under dynamically changing workload conditions.

4.2. Parameters

Observations are made concerning metrics like finish time, utilization rate, and success rate of the proposed and existing state-of-the-art algorithms.

Utilization rate measures how heavily the cloud resources are loaded over the scheduling horizon. It is defined as the fraction of used capacity relative to total capacity, aggregated across VMs and time:

$$UR = \frac{\sum_{t-1}^{H} \sum_{j-1}^{m} \mathrm{UsedRes}(vm_j, t)}{\sum_{t-1}^{H} \sum_{j-1}^{m} \mathrm{TotalRes}(vm_j)}. \quad (5)$$

Here, H is the number of scheduling steps in an episode, m is the number of VMs, $\mathrm{TotalRes}(vm_j)$ is the available capacity of VM vm_j, and $\mathrm{UsedRes}(vm_j, t)$ is the allocated/consumed capacity at time t. In this paper, UsedRes and TotalRes are computed as a normalized aggregation of CPU, RAM, bandwidth, and storage usage, so UR is dimensionless and lies in $[0, 1]$. Lower utilization values in our results indicate reduced resource contention and better load distribution (less saturation), which is

preferable for meeting deadlines and improving success rate under heterogeneous workloads.

4.3. Experimental Results

This section presents the experimental results of our empirical study with different workloads. Each workload has a specific number of jobs, as mentioned earlier, and the observations are made to analyze the performance of the proposed method.

The experiments were conducted on a machine with an Intel Core i7 processor, 16 GB RAM, and Windows 10 OS. A cloud simulation environment was built using Python with OpenAI Gym toolkit for task scheduling simulations. Synthetic workload datasets of 2000, 5000, and 10000 jobs were generated, simulating diverse user and resource profiles to reflect real-world cloud scenarios.

Table 3. Performance of task scheduling methods with 2000 jobs workload

Performance Measure	Reward	Success Rate	Utilization Rate	Finish Time
Random policy	353082	0.479	0.344	60.54
Round-Robin policy	350367	0.471	0.346	60.36
Earliest policy	363007	0.522	0.332	60.42
DQN Policy (Proposed)	428305	0.811	0.233	63.87

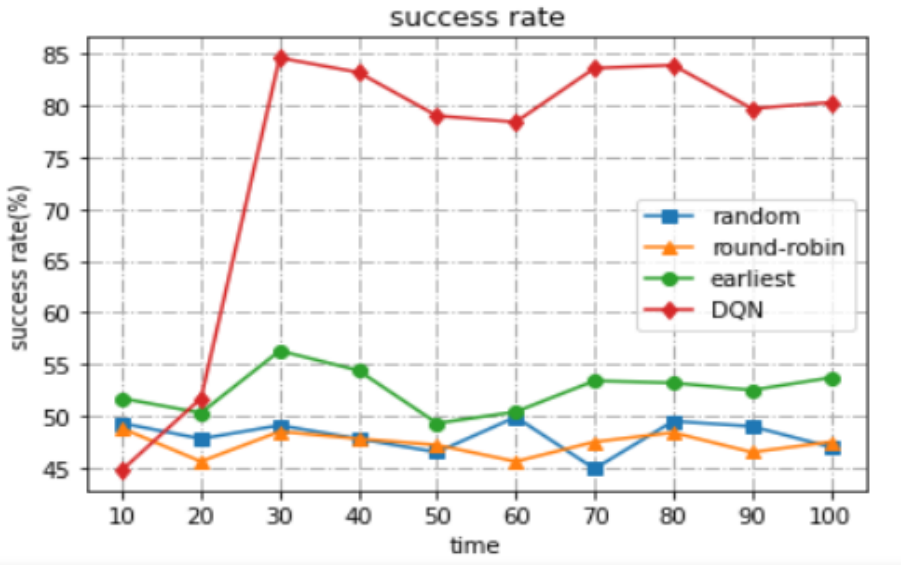


Fig. 3. Success rate exhibited by the proposed state of the algorithms with 2000 jobs workload

As presented in Table 3, the task scheduling performance with 2000 jobs workload ss provided for different scheduling algorithms.

As presented in Fig. 3, the success rate of the proposed and existing methods is provided. The results revealed that the proposed DQN based method could achieve highest performance.

The experimental results with 2000 jobs workload are presented in Fig. 4. The reward of the proposed method for task scheduling is found highest when compared with existing methods. Concerning success rate, RL-OTS achieved a 81.10% task completion rate, outperforming the nearest method. The proposed DQN-based task scheduling methodology could also optimize resource utilization. Therefore, its resource consumption is the least among the task scheduling methods. Regarding finish time, interestingly, the proposed method exhibited overhead when compared with state-of-the-art methods. The rationale is that the proposed algorithm strives to achieve optimal task scheduling and resource utilization.

Table 4 shows the task scheduling performance with 5000 jobs workload provided for different scheduling algorithms.

(a) Reward (b) Success Rate

(c) Utilization Time (d) Finish Time

Fig. 4. Results of experiments with 2000 jobs

Table 4. Performance of task scheduling methods with 5000 jobs workload

Performance Measure	Reward	success_rate	UtilizationRate	Finish Time
Random policy	657691	0.461	0.381	105.24
Round-Robin policy	658899	0.464	0.38	104.73
Earliest policy	683697	0.521	0.363	104.7
DQN policy	802792	0.801	0.267	105.95

As presented in Fig. 5, the success rate of the proposed and existing methods is provided. The results revealed that the proposed DQN-based method could achieve the highest performance with a 5000 jobs workload.

The experimental results with a 5000 jobs workload are presented in Fig. 6. The reward of the proposed method for task scheduling is found to be highest when compared with existing methods. Regarding success rate, RL-OTS achieved a 81.11% task completion rate, outperforming the nearest method. The proposed DQN-based task scheduling methodology could also optimize resource utilization. Therefore, its resource consumption is the least among the task scheduling methods. Regarding finish time, interestingly, the proposed method exhibited overhead when compared with state-of-the-art methods. The rationale is that the proposed algorithm strives to achieve optimal task scheduling and resource utilization.

Table 5 shows the task scheduling performance with 10000 jobs workload provided for different scheduling algorithms.

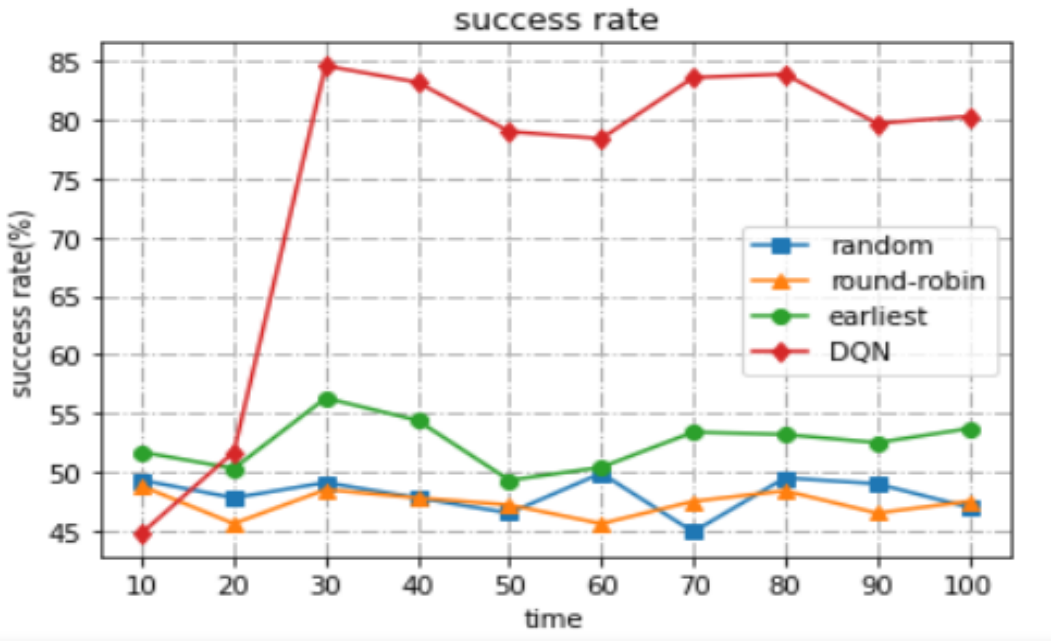
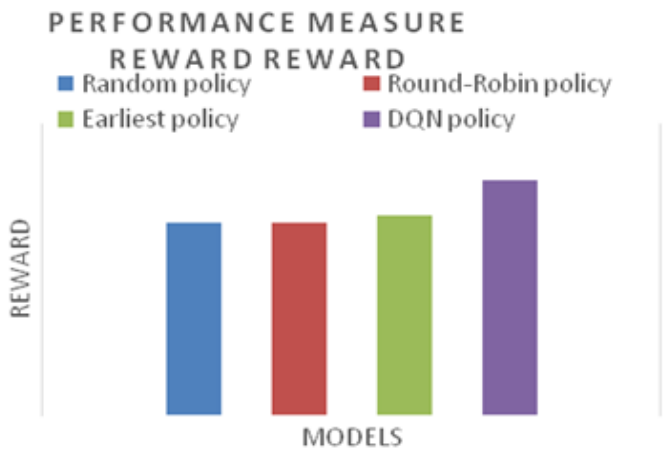
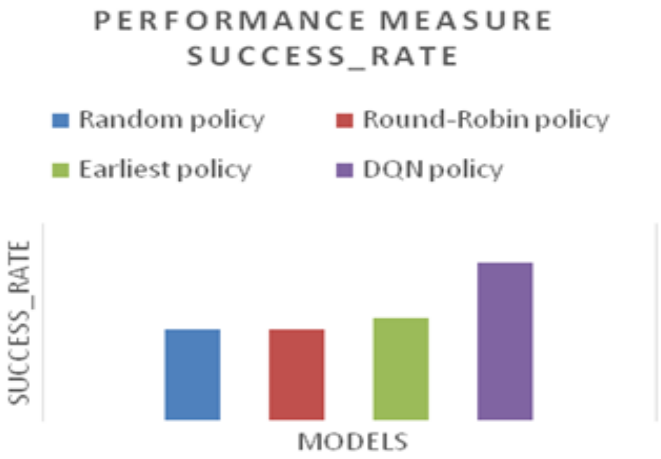


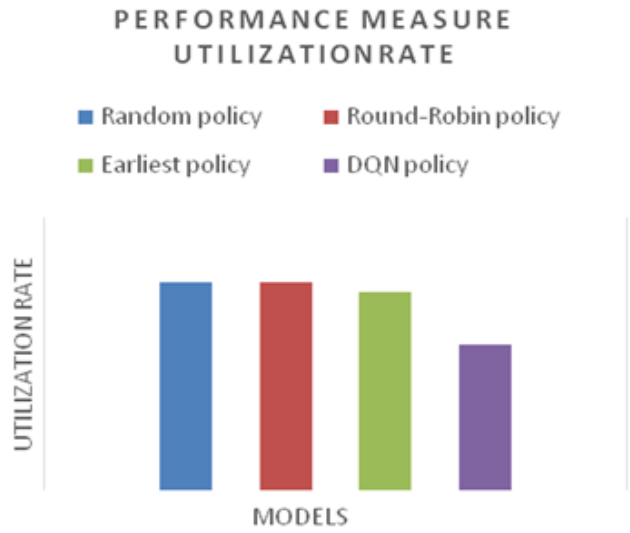
Fig. 5. Success rate exhibited by the proposed state of the algorithms with 5000 jobs workload



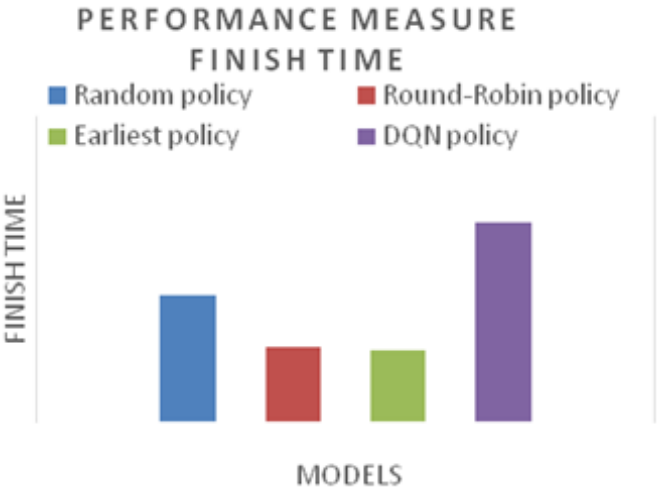
(a) Reward



(b) Success Rate



(c) Utilization Time



(d) Finish Time

Fig. 6. Results of experiments with 5000 jobs

Table 5. Performance of task scheduling methods with 10000 jobs workload

Performance Measure	Reward	success_rate	UtilizationRate	Finish Time
Random policy	347380	0.473	0.383	54.21
Round-Robin policy	345934	0.467	0.383	54.18
Earliest policy	359101	0.526	0.364	54.17
DQN policy	421948	0.802	0.262	55.81

Fig. 7 presents the success rate of the proposed and existing methods. The results revealed that the proposed DQN-based method could achieve the highest performance with a 10,000-job workload

The experimental results with a workload of 10,000 jobs are presented in Fig. 8. The reward of the proposed method for task scheduling is found to be highest when compared with existing methods. Regarding success rate, RL-OTS achieved a 81.20% task completion rate, outperforming the nearest method. The proposed DQN-based task scheduling methodology could also optimize resource utilization. Therefore, its resource consumption is the least among the task scheduling methods. Regarding finish time, interestingly, the proposed method exhibited overhead when compared with state-of-the-art methods. The rationale is that the proposed algorithm strives to achieve optimal task scheduling and resource utilization.

This section presents a comparative analysis of the results of the proposed method and those of other baseline methods. It also analyses the performance of different state-of-the-art methods and compares the results with those of the existing method.

4.4. Comparative Analysis

This section presents a comparative analysis of the results of the proposed method and those of other baseline methods. It also analyses the performance of different state-of-the-art methods and compares the results with those of the existing method.

As presented in Table 6, the performance of the proposed algorithm is compared with baseline methods in terms of success rate in scheduling a given set of jobs.

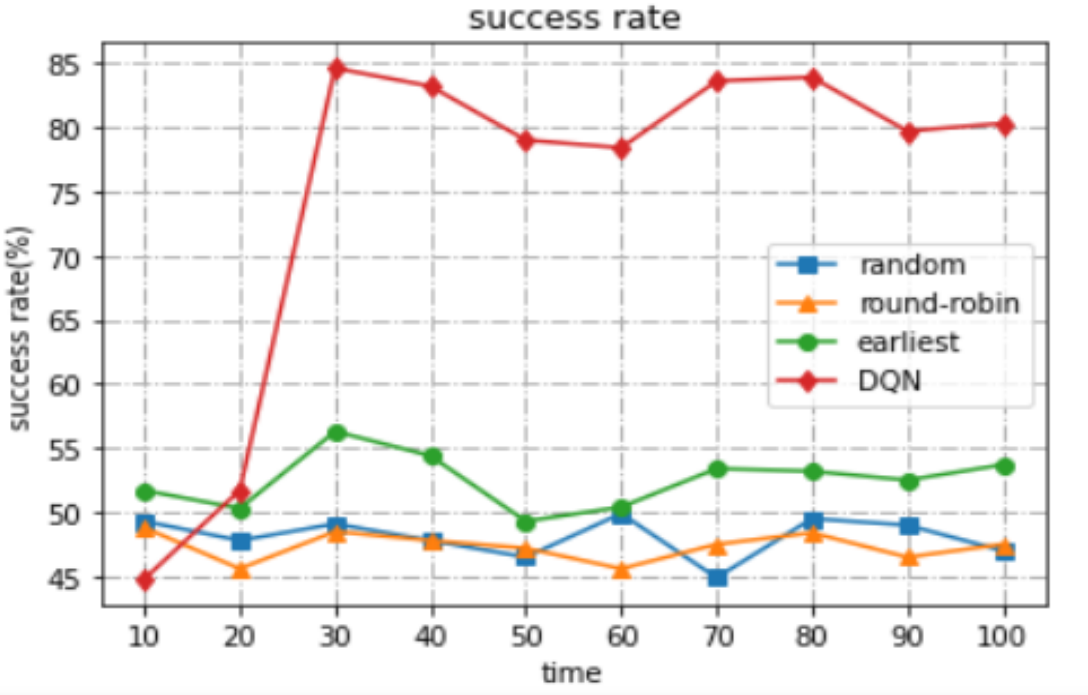
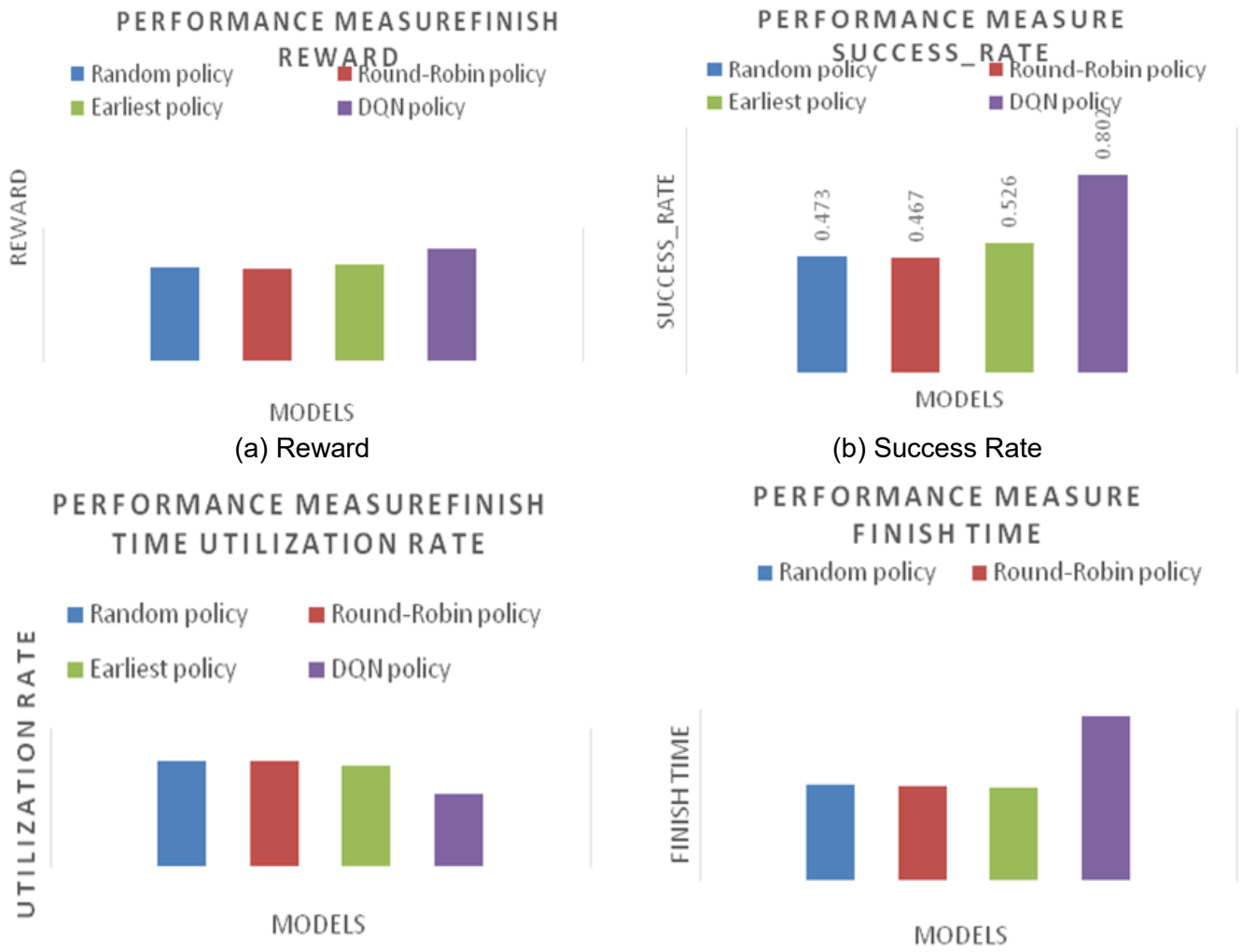


Fig. 7. Success rate exhibited by the proposed state of the algorithms with 10000 jobs workload



(a) Reward (b) Success Rate

(c) Utilization Time (d) Finish Time

Fig. 8. Results of experiments with 10000 jobs

Table 6. Shows performance comparison with baseline methods

SCHEDULING METHOD	SUCCESS RATE (%)		
	2000 JOBS	5000 JOBS	10000 JOBS
Random Policy	0.479	0.461	0.473
Round-Robin Policy	0.471	0.464	0.467
Earliest Policy	0.522	0.521	0.526
DQN Policy (Proposed)	0.811	0.801	0.802

As presented in Fig. 9, the performance of the proposed method is compared against baseline models in terms of success rate. The results showed that the proposed method's success rate is consistently higher with all the workloads like 2000 jobs, 5000 jobs, and 10,000 jobs, respectively. With a 10,000 jobs workload, the random policy achieved a 47.30% success rate, the round-robin policy 46.70%, the earliest policy 52.60%, and the proposed DQN policy had an 80.20% success rate. This reveals that the proposed method outperforms the baseline models with the highest success rate.

As presented in Table 6, the success rate of the proposed method is compared against the state-of-the-art methods. To further strengthen our findings, we conducted multiple runs (10 repetitions) for each workload, computing the mean and standard deviation of the success rates. The results consistently demonstrated that RL-

OTS achieves superior performance with minimal variance, validating the statistical significance of improvements.

As shown in Fig. 10, various scheduling methods were analyzed for their success rates. The success rate is defined as the ratio of the total number of submitted tasks to the number of tasks successfully scheduled and executed in the cloud environment. A higher success rate indicates better performance. Different scheduling methods exhibited varying levels of success rates based on their internal functionalities in the scheduling process. The PSO model achieved a 66.22% success rate, the MBO model achieved 65.38%, the MOPSO model achieved 73.64%, the DRLBTSA model achieved 67.83%, the MOABCQ model achieved 79.20%, and the baseline DNN model achieved 67.50%, while the proposed model achieved an 80.20% success rate. The results show that the proposed method outperformed many existing methods in terms of success rate. Therefore, the proposed method can be utilized in cloud environments for task scheduling.

4.5. Training Duration and Stability Analysis

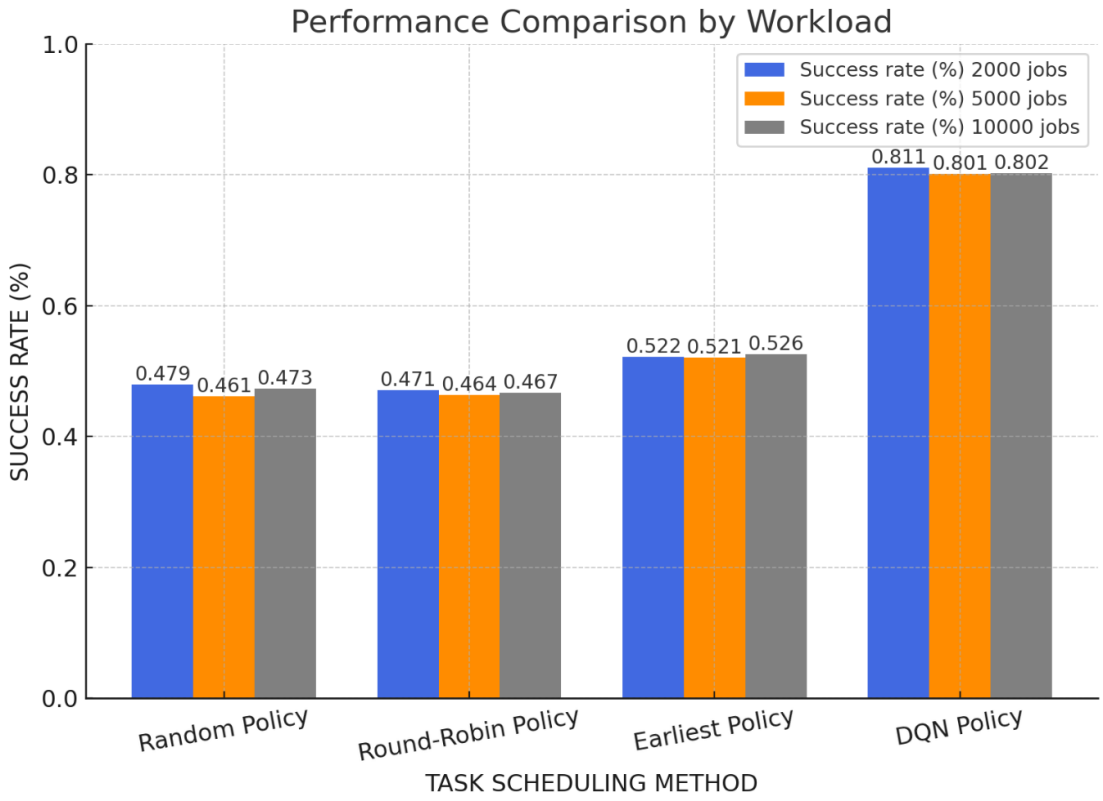


Fig. 9. Performance comparison between proposed and baseline methods

Table 7. Shows performance comparison with state-of-the-art methods

Author & Year	Proposed Work	Methods	Success Rate (%)
Prity et al., 2023	Nature inspired methodology	PSO	0.6622
Hassan et al., 2022	Hybrid approach	MBO	0.6538
Lipsa et al., 2023	Priority based approach	MOPSO	0.7364
Mangalampalli et al., 2023	Proposed DRLBTSA	DRLBTSA	0.67836
Boonhatai et al., 2022	Multi-objective task scheduling optimization	MOABCQ	0.792
Siyu et al., 2023	DT-aided MEC network	DNN	0.675
DQN Policy (Proposed)	RL based approach	RL-OTS	0.802

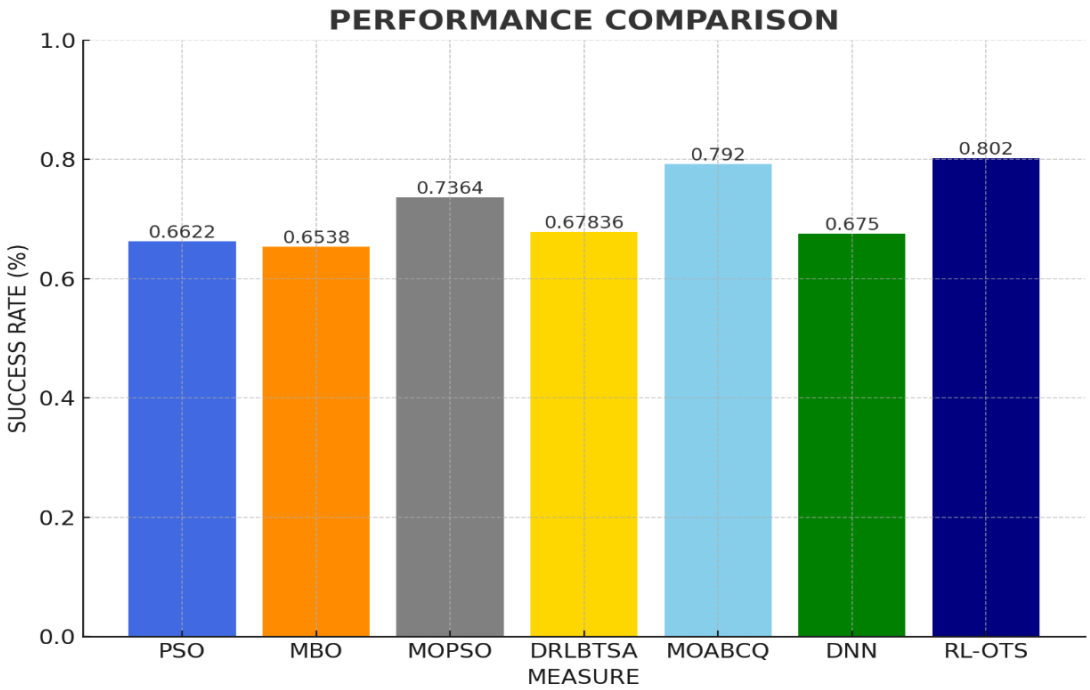


Fig. 10. Performance comparison between proposed and state-of-the-art methods

The RL-OTS model was trained over multiple learning episodes until convergence was observed. In practice, stable learning behavior was achieved after a finite number of episodes, after which performance improvements saturated. To reduce training instability and prevent overfitting, an ε-greedy exploration strategy was employed to balance exploration and exploitation during learning.

Additionally, the training process was repeated across multiple independent runs to mitigate randomness in learning and ensure robustness of the obtained results. The consistency of performance across different workloads indicates that the proposed RL-OTS framework learns stable scheduling policies and generalizes effectively under varying cloud workload conditions.

5. Discussion

The proposed framework for optimal task scheduling is based on RL, which analyzes the runtime environment and makes decisions appropriately. The methodology involved in the system exploits action space and the state space while making well-informed decisions in task scheduling. The empirical study involves three different sets of experiments containing varied workloads. The workload diversity includes 2000 jobs, 5000 jobs, and 10,000 jobs. With each workload, the proposed algorithm is evaluated, and observations are made regarding execution time, success rate, and so on. The proposed system is evaluated, and it found that the RL-based approach outperforms many of the existing heuristics-based approaches in optimal task scheduling. The proposed algorithm takes incoming jobs in terms of episodes and performs deep reinforcement learning-based mechanisms toward making optimal decisions for task scheduling. Since the cloud computing environment is very dynamic in nature in terms of workload diversity and resource diversity, the proposed system follows a state and feedback approach where the actor gets feedback from the environment from time to time while making decisions towards task scheduling. This kind of approach is purely learning-based instead of following specific heuristics. Therefore, the proposed RL-OTS framework demonstrates improved adaptability and resource efficiency compared to heuristic-based scheduling approaches under dynamic cloud workloads. While

RL-OTS achieves a higher success rate and higher resource utilization, indicating more efficient use of available cloud resources and reduced idle capacity, it incurs a marginally higher finish time. This behavior is consistent with the cost-based reward formulation, where the scheduler prioritizes overall allocation efficiency and task fairness over aggressive minimization of completion time. This is attributed to the agent's exploration of multiple scheduling possibilities to optimize the overall allocation. The slightly increased finish time is a trade-off for achieving better task assignment, fairness, and resource utilization. Overall, the experimental results indicate that the proposed trade-off between execution time and resource utilization is suitable for infrastructure-oriented cloud systems where efficiency and fairness are prioritized. However, it has certain limitations, as mentioned in section 5.1.

5.1. Limitations

The proposed system has certain limitations.

Though the empirical study found workload diversity, workload volume needs further improvement to match the real-world cloud computing scale.

The proposed algorithm employs a value-based deep reinforcement learning approach. Although effective, performance can potentially be improved further using actor–critic architectures, which may offer better convergence and scalability in large-scale cloud environments.

6. Conclusion and Future Work

RL-based job scheduling is responsible for executing the underlying mechanism of our framework and ensuring that the tasks are scheduled properly and executed as per the user expectations. The cloud infrastructure has resources available. Based on the freely available resources and the number of tasks at hand, deciding on task scheduling is very dynamic. This is the reason heuristic algorithms fail to achieve optimal performance. For the same reason, we proposed a learning-based approach based on RL toward efficient task scheduling. The proposed system has a monitoring module that monitors the cloud resources and coordinates with the job scheduler to help it in optimal scheduling. We proposed an algorithm known as Reinforcement Learning-based Optimal Task Scheduling (RL-OTS), which requires action space and state space with the interaction between environment and actor to make well-informed decisions in task scheduling. With empirical study, it was observed that the proposed algorithm could outperform many state-of-the-art algorithms. The key academic contribution of this work is the formulation of cloud task scheduling as a deep reinforcement learning problem and the demonstration that a Deep Q-Learning–based scheduler can effectively adapt to dynamic workloads and heterogeneous cloud resources. The proposed RL-OTS framework can be deployed in real-world cloud data centers to optimize resource utilization and job scheduling. In the context of smart infrastructure systems, the proposed RL-OTS framework is particularly relevant for cloud- and edge-based platforms supporting intelligent transportation systems, infrastructure IoT monitoring, and large-scale urban sensing applications. These systems require adaptive scheduling to handle time-varying workloads, real-time data streams, and heterogeneous computing resources while maintaining service continuity and reliability. Potential applications include smart cloud management systems, IoT-based cloud services, and edge-cloud hybrid infrastructures, where efficient task allocation is critical for operational efficiency. However, the current evaluation of RL-OTS is conducted in a simulated cloud environment, and factors such as training overhead, real-time deployment constraints, and communication latency have not been fully investigated. In the future, we intend to improve our framework for efficient task scheduling by incorporating optimizations in the reinforcement learning process. Particularly different optimization parameters can be considered. In the future, to address these limitations, we intend to improve our

framework for efficient task scheduling by incorporating optimizations in the reinforcement learning process. For instance, Service Level Agreements (SLAs) are an important parameter to be incorporated. Another direction for future work is to develop a framework for task scheduling in an edge-cloud environment. Another direction for future work is to extend RL-OTS toward edge–cloud architectures commonly used in smart infrastructure systems, where latency-sensitive tasks from transportation networks, IoT sensors, and distributed infrastructure services must be scheduled across edge and cloud resources. With respect to the edge-cloud environment, IoT workflow applications are commonly executed. Scheduling tasks of such applications in an edge-cloud environment can be done in the future. Another important direction for future work is to evaluate the scalability of RL-OTS in large-scale, real-time cloud systems with hundreds of thousands of tasks and dynamically fluctuating resources. Scalability optimizations such as hierarchical RL or federated RL approaches can be explored.

References

[1] A. Asghari, M.K. Sohrabi, F. Yaghmaee. (2021). Task scheduling, resource provisioning, and load balancing on scientific workflows using parallel SARSA reinforcement learning agents and genetic algorithm. *The Journal of Supercomputing*, 77, 2800-2828. https://doi.org/10.1007/s11227-020-03364-1

[2] Q. Qi, L. Zhang, J. Wang, H. Sun, Z. Zhuang, J. Liao, F.R. Yu. (2020). Scalable parallel task scheduling for autonomous driving using multi-task deep reinforcement learning. *IEEE Transactions on Vehicular Technology,* 69(11), 13861-13874. https://doi.org/10.1109/TVT.2020.3029864

[3] T. Yu, J. Huang, Q. Chang. (2021). Optimizing task scheduling in human-robot collaboration with deep multi-agent reinforcement learning. *Journal of Manufacturing Systems*, 60, 487-499. https://doi.org/10.1016/j.jmsy.2021.07.015

[4] H. Zhu, M. Li, Y. Tang, Y. Sun. (2020). A deep-reinforcement-learning-based optimization approach for real-time scheduling in cloud manufacturing. *IEEE Access*, 8, 9987-9997. https://doi.org/10.1109/ACCESS.2020.2964955

[5] M. Sharma, R. Garg. (2020). An artificial neural network based approach for energy efficient task scheduling in cloud data centers. *Sustainable Computing: Informatics and Systems*, 26, 100373. https://doi.org/10.1016/j.suscom.2020.100373

[6] W. Guo, W. Tian, Y. Ye, L. Xu, K. Wu. (2021). Cloud resource scheduling with deep reinforcement learning and imitation learning. *IEEE Internet of Things Journal,* 8(5), 3576-3586. https://doi.org/10.1109/JIOT.2020.3025015

[7] S. Mostafavi, V. Hakami. (2020). A stochastic approximation approach for foresighted task scheduling in cloud computing. *Wireless Personal Communications,* 114, 901-925. https://doi.org/10.1007/s11277-020-07398-9

[8] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, J. Zeng. (2020). Q-learning based dynamic task scheduling for energy-efficient cloud computing. *Future Generation Computer Systems*, 108, 361-371. https://doi.org/10.1016/j.future.2020.02.018

[9] A. Asghari, M.K. Sohrabi, F. Yaghmaee. (2020). Online scheduling of dependent tasks of cloud's workflows to enhance resource utilization and reduce the makespan using multiple reinforcement learning-based agents. *Soft Computing,* 24, 16177-16199. https://doi.org/10.1007/s00500-020-04931-7

[10] P. Gazori, D. Rahbari, M. Nickray. (2020). Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach. *Future Generation Computer Systems,* 110, 1098-1115.https://doi.org/10.1016/j.future.2019.09.060

[11] Z. Tong, H. Chen, X. Deng, K. Li, K. Li. (2020).

A scheduling scheme in the cloud computing environment using deep Q-learning. *Information Sciences,* 512, 1170-1191. https://doi.org/10.1016/j.ins.2019.10.035

[12] A. Asghari, M.K. Sohrabi, F. Yaghmaee. (2020). A cloud resource management framework for multiple online scientific workflows using cooperative reinforcement learning agents. *Computer Networks,* 179, 107340.https://doi.org/10.1016/j.comnet.2020.107340

[13] S. Mangalampalli, G.R. Karri, M. Kumar, O.I. Khalaf, C.A.T. Romero, G.M.A. Sahib. (2024). DRLBTSA: Deep reinforcement learning based task scheduling algorithm in cloud computing. *Multimedia Tools and Applications*, 83, 8359-8387. https://doi.org/10.1007/s11042-023-16008-2

[14] Z. Tang, W. Jia, X. Zhou, W. Yang, Y. You. (2022). Representation and reinforcement learning for task scheduling in edge computing. *IEEE Transactions on Big Data*, 8(3), 795-808. https://doi.org/10.1109/TBDATA.2020.2990558

[15] A. Asghari, M.K. Sohrabi. (2021). Combined use of coral reefs optimization and reinforcement learning for improving resource utilization and load balancing in cloud environments. *Computing,* 103, 1545-1567. https://doi.org/10.1007/s00607-021-00920-2

[16] S.E. Shukri, R. Al-Sayyed, A. Hudaib, S. Mirjalili. (2021). Enhanced multi-verse optimizer for task scheduling in cloud computing environments. *Expert Systems with Applications*, 168, 114230. https://doi.org/10.1016/j.eswa.2020.114230

[17] Y. Garí, D.A. Monge, E. Pacini, C. Mateos, C.G. Garino. (2021). Reinforcement learning-based application autoscaling in the cloud: A survey. *Engineering Applications of Artificial Intelligence*, 102, 104288. https://doi.org/10.1016/j.engappai.2021.104288

[18] J. Hu, Y. Li, G. Zhao, B. Xu, Y. Ni, H. Zhao. (2021). Deep reinforcement learning for task offloading in edge computing assisted power IoT. *IEEE Access,* 9, 93892-93901. https://doi.org/10.1109/ACCESS.2021.3092381

[19] H. Li, R. Cai, N. Liu, X. Lin, Y. Wang. (2018). Deep reinforcement learning: Algorithm, applications, and ultra-low-power implementation. *Nano Communication Networks,* 16, 81-90. https://doi.org/10.1016/j.nancom.2018.02.003

[20] H. Liu, S. Liu, K. Zheng. (2018). A reinforcement learning-based resource allocation scheme for cloud robotics. *IEEE Access*, 6, 17215-17222. https://doi.org/10.1109/ACCESS.2018.2814606

[21] J. Yang, X. You, G. Wu, M.M. Hassan, A. Almogren, J. Guna. (2019). Application of reinforcement learning in UAV clusters task scheduling. *Future Generation Computer Systems*, 95, 140-148. https://doi.org/10.1016/j.future.2018.11.014

[22] G. Rjoub, J. Bentahar, O.A. Wahab. (2020). BigTrustScheduling: Trust-aware big data task scheduling approach in cloud computing environments. *Future Generation Computer Systems,* 110, 1079-1097. https://doi.org/10.1016/j.future.2019.11.019

[23] Z. Tong, F. Ye, B. Liu, J. Cai, J. Mei. (2021). DDQN-TS: A novel bi-objective intelligent scheduling algorithm in the cloud environment. *Neurocomputing*, 455, 419-430. https://doi.org/10.1016/j.neucom.2021.05.070

[24] Z. Peng, J. Lin, D. Cui, Q. Li, J. He. (2020). A multi-objective trade-off framework for cloud resource scheduling based on the deep Q-network algorithm. *Cluster Computing*, 23, 2753-2767. https://doi.org/10.1007/s10586-019-03042-9

[25] X. Wang, Z. Ning, S. Guo, L. Wang. (2022). Imitation learning enabled task scheduling for online vehicular edge computing. *IEEE Transactions on Mobile Computing*, 21(2), 598-611.https://doi.org/10.1109/TMC.2020.3012509

9

[26] B. Kruekaew, W. Kimpan. (2022). Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning. *IEEE Access*, 10, 17803-17818. https://doi.org/10.1109/ACCESS.2022.3149955

[27] A. Pradhan, S.K. Bisoy, S. Kautish, M.B. Jasser, A.W. Mohamed. (2022). Intelligent decision-making of load balancing using deep reinforcement learning and parallel PSO in cloud environment. *IEEE Access*, 10, 76939-76952.https://doi.org/10.1109/ACCESS.2022.3192628

[28] L. Zhang, C. Yang, Y. Yan, Y. Hu. (2022). Distributed real-time scheduling in cloud manufacturing by deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 18(12), 8999-9007. https://doi.org/10.1109/TII.2022.3178410

[29] S. Yuan, Z. Zhang, Q. Li, W. Li, Y. Zhang. (2023). Joint optimization of DNN partition and continuous task scheduling for digital twin-aided MEC network with deep reinforcement learning. *IEEE Access*, 11, 27099-27110. https://doi.org/10.1109/ACCESS.2023.3257342

[30] X. Zhou, W. Liang, K. Yan, W. Li, K.I.-K. Wang, J. Ma, Q. Jin. (2023). Edge-enabled two-stage scheduling based on deep reinforcement learning for Internet of Everything. *IEEE Internet of Things Journal*, 10(4), 3295-3304. https://doi.org/10.1109/JIOT.2022.3179231

[31] S. Mangalampalli, G.R. Karri, M. Kumar, O.I. Khalaf, C.A.T. Romero, G.M.A. Sahib. (2024). DRLBTSA: Deep reinforcement learning based task-scheduling algorithm in cloud computing. *Multimedia Tools and Applications*, 83, 8359-8387. https://doi.org/10.1007/s11042-023-16008-2

[32] K. Sadatdiynov, L. Cui, L. Zhang, J.Z. Huang, S. Salloum, M.S. Mahmud. (2023). A review of optimization methods for computation offloading in edge computing networks. *Digital Communications and Networks*, 9(2), 450-461. https://doi.org/10.1016/j.dcan.2022.03.003

[33] Z. Chen, P. Wei, Y. Li. (2023). Combining neural network-based method with heuristic policy for optimal task scheduling in hierarchical edge cloud. *Digital Communications and Networks*, 9(3), 688-697. https://doi.org/10.1016/j.dcan.2022.04.023

[34] G. Zhou, W. Tian, R. Buyya, R. Xue, L. Song. (2024). Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions. *Artificial Intelligence Review*, 57, 124. https://doi.org/10.1007/s10462-024-10756-9

[35] Z. Wang, M. Goudarzi, M. Gong, R. Buyya. (2024). Deep reinforcement learning-based scheduling for optimizing system load and response time in edge and fog computing environments. *Future Generation Computer Systems*, 152, 55-69. https://doi.org/10.1016/j.future.2023.10.012

[36] M. Tejer, R. Szczepanski, T. Tarczewski. (2024). Robust and efficient task scheduling for robotics applications with reinforcement learning. *Engineering Applications of Artificial Intelligence*, 127, 107300. https://doi.org/10.1016/j.engappai.2023.107300

[37] H. Wu, X. Yang, Z. Bu. (2024). Task offloading with service migration for satellite edge computing: A deep reinforcement learning approach. *IEEE Access*, 12, 25844-25856. https://doi.org/10.1109/ACCESS.2024.3367128

[38] T. Salehnia, A. Montazerolghaem, S. Mirjalili, M.R. Khayyambashi, L. Abualigah. (2024). SDN-based optimal task scheduling method in Fog-IoT network using combination of AO and WOA. *Handbook of Whale Optimization Algorithm (pp. 109–128).* https://doi.org/10.1016/B978-0-32-395365-8.00014-2