International Journal of Scientific Research and Engineering Development-– Volume 8 Issue 4, July Year 2025
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Workflow Scheduling in Cloud Computing Using Reinforcement Learning

Sriperambuduri Vinay Kumar[1], Peddapally Srividya[2], Shanigarapu Vaishnavi[3]
[1,2,3]Vasavi College of Engineering, Hyderabad, India
s.vinaykumar@staff.vce.ac.in[1], peddapallysrividya@gmail.com[2], vaishnavishanigarapu3004@gmail.com[3]

Abstract
In this paper, we introduce an RL-based workflow scheduling framework that combines a customized Q-learning algorithm with a novel state encoding method. The state representation captures task urgency, cost-effectiveness of VMs, and executability, allowing the scheduler to make more situational decisions. Differing from common RL applications, our model updates Q-values perpetually through leveraging deadline-aware reward signals and load-balancing feedback. Scientific workflows in real-world scenarios are simulated by Workflow Sim for testing our solution's effectiveness. The major contributions of this work include adaptive task-to-resource mapping enabled by the learned state vectors and integrated intelligent learning in workflow execution for improved make span and scalability.

Keywords: Workflow Scheduling, Cloud Computing, Reinforcement Learning, Q-Learning, Directed Acyclic Graph, WorkflowSim, Task Scheduling, Resource Optimization

---

1. Introduction
The sudden spike in the industrial and scientific applications has drastically increased the scale and complexity of the computational workflows. Such workflows, which are typically executed in distributed cloud computing platforms, are commonly represented as Directed Acyclic Graphs (DAGs) where tasks depend on one another and must be scheduled properly. Optimal scheduling is essential in terms of minimizing execution time (makespan), achieving maximum resource utilization, and satisfying Quality of Service (QoS) requirements. Traditional scheduling algorithms, such as heuristic and metaheuristic methods, are rigid, are likely to depend on static rules, and lack learning from previous scheduling outcomes—thus being less than ideal for dynamic and heterogeneous environments. Breakthroughs in the domain have recently focused on artificial intelligence, and in particular Reinforcement Learning (RL), for intelligent and responsive scheduling.

2. Literature Survey
Workflow scheduling on cloud infrastructures has, in the last few years, made immense leaps. Many approaches have been proposed to maximize scheduling speed, dependability, energy efficiency, and cost minimization. Minimizing the makespan by carefully reordering jobs and allocating them to heterogeneous systems and thus reducing total workflow execution time is one of the well-established research directions [1]. Likewise, other research has focused on energy efficiency and system reliability through adaptive and fault-tolerant scheduling mechanisms that are capable of recovery from failures or dynamic task reassignment [2].
To better handle time and budget constraints, certain researchers have suggested algorithms that consider both energy consumption and cost simultaneously, such that tasks are performed within deadlines without exceeding budgets [16]. To handle complex multi-cloud systems, where hardware is dispersed and shared, cost-efficient and reliability-aware scheduling techniques have been suggested to reduce failures and enhance availability [3].
In recent years, artificial intelligence, particularly reinforcement learning (RL), has emerged as a strong method for intelligent and dynamic scheduling. Methods like Deep Q-Learning (DQL) have been used to carry out task-to-resource assignment in an automated manner, enabling systems to learn scheduling policies over time [11]. Certain methods are focused on energy-aware RL models, which are designed specifically for scientific workflows with high computational and communication demands [14].
Besides, multi-objective optimization has also been addressed through the inclusion of multi-criteria decision-making (MCDM) techniques. These help schedulers balance between multiple factors such as execution time, cost, priority of tasks, availability of resources, and latency, thereby making more rational and adaptable decisions [13]. Further work has built upon this by employing multi-agent reinforcement learning (MARL) architectures, where multiple agents

collaborate or compete to improve global scheduling performance [9]. At the same time, hardware-efficient Q-learning implementations have opened doors for real-time scheduling in embedded systems and thereby facilitated RL for edge and IoT-based systems [10].

While these studies are fantastic advancements, most of them are either single-objective oriented, static encoding of the environment-dependent, or static infrastructure scenario-based. Our approach addresses these constraints by integrating deadline-conscious reward modeling, a customized state encoding process, and adaptive Q-value updates that learn to cope with the dynamic nature of cloud environments. Unlike most traditional RL approaches based on static environments or pre-trained models, our scheduler learns in real-time as workflows are running, allowing it to respond to task priority adjustments and VM availability on-the-fly.

Further, our method is a bridge between current Q-learning methods and recent developments in deep reinforcement learning (DRL), where future enhancement can plausibly employ neural approximators in place of the Q-table to handle larger and more complex state-action spaces. Although scalable opportunities exist with deep RL models, these have been conventional to require extensive amounts of training time and resources. Comparatively, our Q-learning approach offers a basic but effective model with realistic learning rate that is preferable in simulation-based contexts such as WorkflowSim.

3. Proposed Methodology

Inclusion of Reinforcement Learning (RL), allowing adaptive decision-making for workflow task assignments, constitutes the foundation of the proposed scheduling framework. The overview of key algorithms and their roles in the system is detailed below.

3.1 Q-Learning

A Q-learning model-free reinforcement learning algorithm estimates a Q-value for all state-action pairs in an effort to figure out the optimal action-selection strategy. The Bellman equation is utilized iteratively by the agent in updating its Q-values:

$$\mathbf{Q\ (s,\ a) \leftarrow Q\ (\ s,\ a) + \alpha[r + \gamma\ max_{a'}\ Q(s', a') - Q(s,\ a)]}$$

Where:

- s: current situation
- s′: next state
- r: reward given
- a: action executed
- α: learning rate
- γ: discount factor

For workflow scheduling, the operation is similar to assigning a specific task to a resource, while the state stores the current system state and the cloud environment.

3.2 State Encoding

A special encoder is employed to map the current system state into a numerical vector. The features include:

- The total number of tasks
- VM priority based on cost per bandwidth (**VmPriority=vmcost/maxcost**)
- Task priority according to task length and VM processing power (**TaskPriority=tasklength/vmMips**)
- Task deadline is considered while assigning rewards; execution time is computed by dividing task length by VM MIPS (**Deadline=tasklength*deadlinebufferfactor**)
- The cost-effectiveness of the VM, normalized by the maximum cost of all VMs available

The agent is able to generalise across various workflows and cloud setups due to this encoded vector.

3.3 Scheduling Methodology

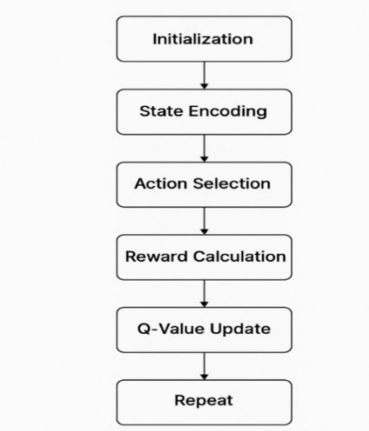

Fig 3.3.1 Workflow Scheduler flowchart

The scheduling model's architecture and behavior operate in an episodic manner, with each episode being the planning and execution of a complete workflow. The RL agent engages with a simulated cloud environment, monitors the outcome of its decisions, and learns to improve its scheduling policy over time through reward feedback.

Step 1: Initialisation of the Workflow

The simulator runs a selected workflow from the benchmark data set. Task attributes such as communication requirements, data dependencies, and execution time are initialized.

Step 2: State Encoding

The system's current condition is expressed numerically for every possible task and virtual machine combination. This includes the predicted execution time, deadline, task priority (based on computationally length and MIPS of the VM), and VM priority (based on cost per bandwidth).

Step 3: Selecting an Action

To decide whether to investigate a random task-to-VM mapping or take advantage of the one with the highest expected reward (Q-value), the DRL agent uses an ε-decay policy. This strikes a balance between utilising established tactics and picking up new ones.

Step 4: Assigning Tasks

Depending on the action selected in the last step, the chosen task is assigned to the chosen virtual machine. The decision of scheduling is simulated by the environment.

Step 5: Calculating the Reward

Three measurements—makespan, deadline, and energy usage—are employed to measure a reward after an execution simulation. For motivating the agent towards selecting speedy and energy-efficient mappings, the reward is designed to be the negative product of the task priority and VM priority as well as including a latency penalty.

$$\textbf{Reward=(deadline-executiontime)*0.1+vmcost}$$

Step 6: Update the Q-value

The Q-value of the state-action pair is updated using the Bellman equation and the provided reward. Through this process, the agent learns incrementally what task-to-VM mappings work best. As needed, they are loaded and stored.

Step 7: Repeat

The agent repeatedly follows steps 2 through 6 for each unscheduled task until the workflow completes. Scheduling performance increases and policy becomes better as the process continues.

3.4 Reward System

The reward function is designed to incentivize the agent to minimize energy usage, latency, and makespan. It is computed as the negative of the product of latency, task priority, and virtual machine priority.

The task length to virtual machine's MIPS rating, i.e., how computationally expensive a task is with respect to processing speed, is utilized for task priority determination. System maximum virtual machine cost to the selected virtual machine's cost per bandwidth, which is the VM's cost or energy efficiency, is utilized for determining VM priority. VM lateness penalty is employed to complete tasks early.

These three components collaborate to penalize inefficient task-to-VM assignments and encourage computationally and economically optimal decisions. A less optimal assignment is represented by a lower (more negative) value, which instructs the Q-learning algorithm to make improved scheduling decisions in the future.

3.5 Training Configuration

- Learning Rate: Initializes at 0.9 and anneals to 0.1
- Discount Factor (γ): 0.5
- Replay Memory Size: 1000 transitions
- Batch Size: 128
- Exploration Rate (ε): Decreases from 1.0 to 0.05 over time

4. Experimental Setup

Employing WorkflowSim, an extension of the CloudSim toolkit specifically designed for workflow-based simulations, the proposed scheduling framework is designed to operate within a simulated cloud environment. WorkflowSim is well-suited to simulate scientific workflows in varied environments since it can handle advanced features such as task dependencies, data transfers, and job scheduling.

4.1 Representation of Workflow

Workflows are represented as Directed Acyclic Graphs (DAGs), with edges representing data dependencies and each node representing a distinct computational task. The child task can start execution only when all of its parent tasks have finished and their data have been passed.

4.2 Dataset Utilised

We utilized the Pegasus Workflow Management System's benchmark scientific workflows. Gene2Life, Floodplain, are some of the small-scale workflows (5–10 tasks).
Medium-scale workflows: Inspiral, CyberShake, Montag, and Epigenomics
The computational characteristics of the workflows vary according to the number of tasks, organization, parallelism level, and data exchange requirements. As a diverse testbed for scheduling performance evaluation on different workflow classes, they provide a varied testbed.

4.3 Environment of Computing

Heterogeneous nodes of different CPU capacities (expressed in MIPS), memory setup, and bandwidth limitation constitute the cloud infrastructure simulated. For simulating real-world cloud scenarios, we have nodes of different capacities:
Network bandwidth: 10 MB/s between nodes
Node capacities: 4 MIPS, 8 MIPS, and 16 MIPS
Because of this heterogeneity, the scheduling algorithm can balance the load of the system while learning optimal task placements that exploit speedy nodes for critical tasks.

4.4 Architecture

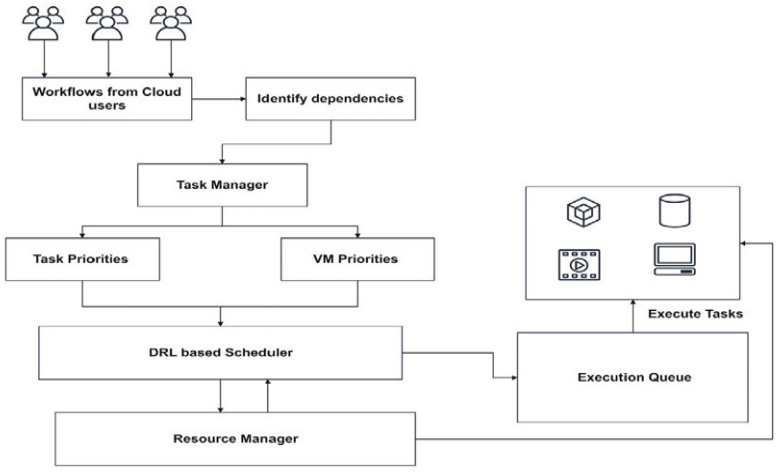

Fig 4.4.1 Architecture of the Proposed System

4.5 Working of the approach

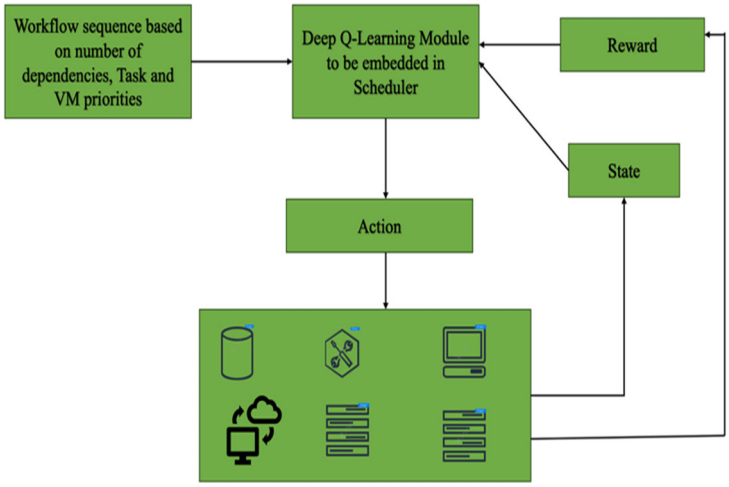

Fig 4.5.1 Functioning of Proposed Approach

5. Results

We tested a variety of scientific workflows that were simulated in WorkflowSim for measuring the effectiveness of the proposed scheduling framework. Round Robin, Min-Min, and Random are utilized as baseline heuristics to compare the performance of the Reinforcement Learning (RL)-based scheduler.

5.1 Workflow Setups

We utilized real-world workloads from the Pegasus Workflow Management System, involving both small and medium-size workloads:

Workflow	Number of Tasks	Type
Floodplain	7	Small
Montage	25	Medium
CyberShake	30	Medium
Inspiral	30	Medium
Epigenomics	24	Medium

5.2 Makespan Reduction

The RL-based scheduler outperformed conventional methods consistently in terms of makespan:

i. RL Vs Random

Workflow	Random Makespan (s)	RL Makespan (s)	Improvement (%)
Inspiral	7571.21	5528.89	26.97%
Floodplain	252031.69	144900.21	42.49%
CyberShake	893.65	465.44	47.92%
Montage	741.31	673.34	9.17%
Epigenomics	20990.28	8156.40	61.13%

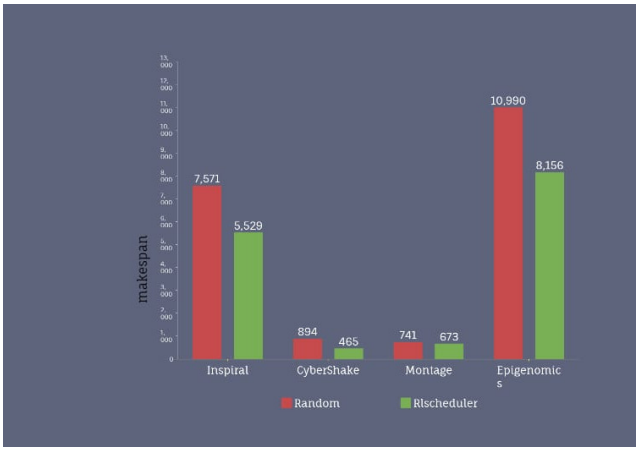

Fig 5.2.1 Bar Chart Comparing the Performance of Reinforcement Learning-Based Scheduler and Random Scheduling Algorithm

ii. RL Vs FCFS

Workflow	**FCFS Makespan (s)**	**RL Makespan (s)**	**Improvement (%)**
Inspiral	4542.61	4097.15	9.82%
Floodplain	117000.21	105600.52	9.77%
CyberShake	266.68	254.32	4.63%
Montage	140.19	137.17	2.15%
Epigenomics	1375.22	1000.52	27.27%

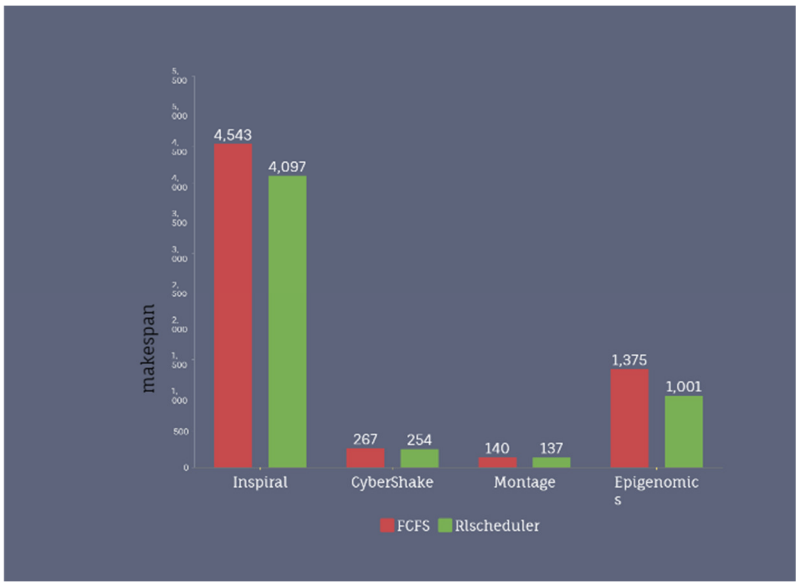

Fig 5.2.2 Bar Chart Comparing the Performance of Reinforcement Learning-Based Scheduler and FCFS Scheduling Algorithm

5.3 Evolution of Rewards

There was a significant increase in cumulative reward in the learning curve over 300,000 training episodes, reflecting improved scheduling efficiency. The RL agent allocated parallel jobs to medium-tier resources and learned to assign high-performance nodes for high-priority tasks.

5.4 Qualitative Findings

- During the early stages of the execution process, the agent learned to prioritize tasks with higher dependencies.

- By allocating parallel tasks across nodes, it exploited CPU cores and unused bandwidth.
- By co-locating dependent tasks, the scheduler reduced data transfer latency for communication-intensive workflows, like epigenomics.

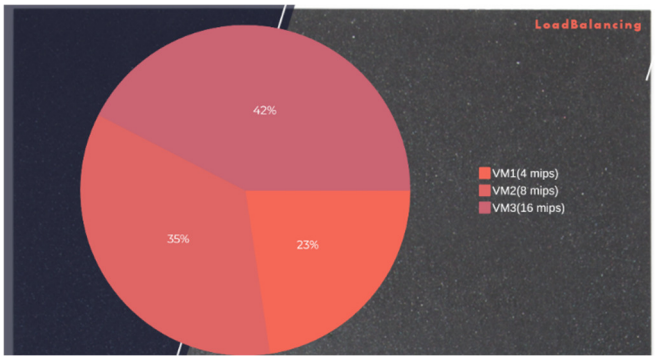

Fig 5.4.1 Pie Chart representing Assignment Scheduling Across Virtual Machines During RL-Based Scheduling (Load Balancing Achieved)

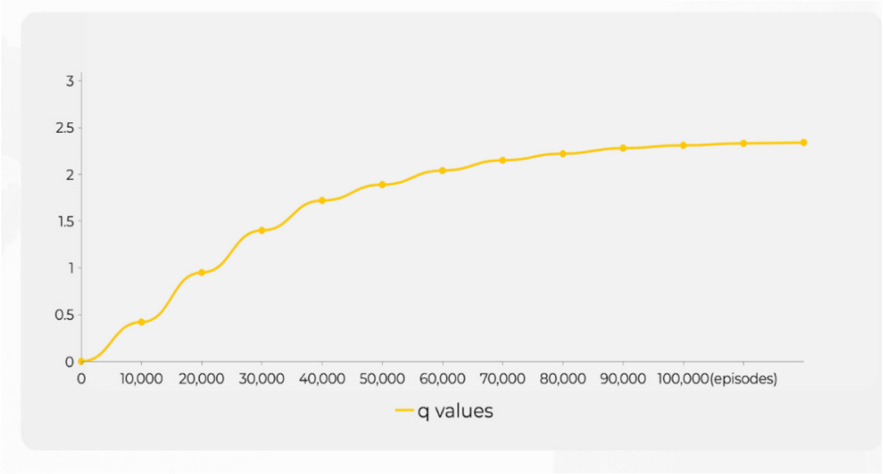

Fig 5.4.2 Line Graph Learning Curve of Q-Value for a Representative State-Action Pair, Reflecting Stability of Learning

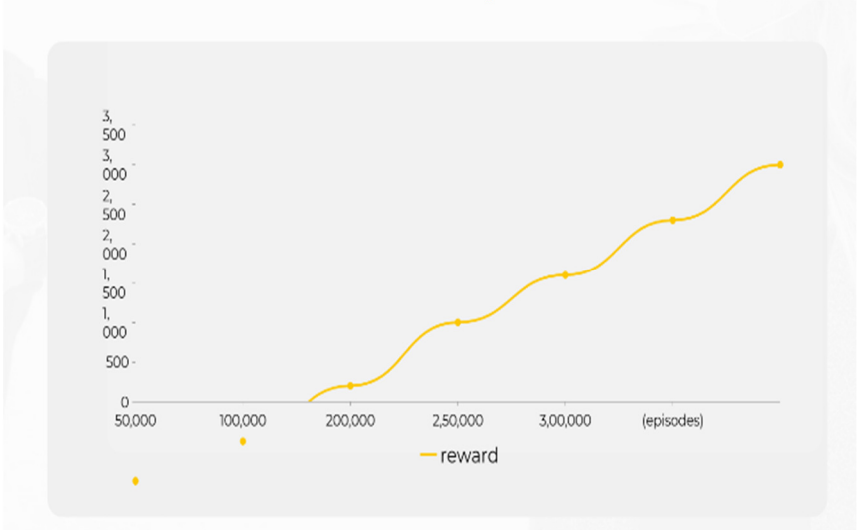

Fig 5.4.3 Line Graph Growth of Cumulative Reward over 300,000 Training Episodes, Illustrating Policy Improvement

6. Conclusion and Future work

In this research, we developed and evaluated a workflow scheduling framework using reinforcement learning for dynamic decision-making in cloud simulations. Compared to traditional scheduling algorithms, the proposed method greatly enhanced makespan minimization by utilizing a Q-learning strategy and representing the scheduling problem as a sequential decision process.

The results indicate how AI-based approaches can be utilized to transcend the limitations of static and rule-based methods in cloud workflow management. The capacity of the system to generalise was confirmed through its high degree of adaptability across a wide range of workflows and resource configurations.

This work can be extended in several ways:

• Employing LSTMs or Recurrent Neural Networks (RNNs) for enhanced capture of task sequences and temporal relationships.
• Adding cost, energy consumption, or deadline fulfillment in multi-objective scheduling.
• Deployment on real cloud infrastructures to experiment under actual workloads and infrastructure variations.
• Changing reward schemes to consider additional performance metrics such as fault tolerance and data locality.

References:

[1] Barrett, E., Howley, E., Duggan, J.: Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. Concur. Comput. Pract. Exp. 25(12), 1656–1674 (2013)
[2] Yan, Y., Zhang, B., Guo, J.: An adaptive decision making approach based on reinforcement learning for self-managed cloud applications. In: 2016 IEEE International Conference on Web Services, pp. 720–723 (2016)
[3] Fakhfakh, F., Kacem, H.H., Kacem, A.H.: A provisioning approach of cloud resources for dynamic workflows. In: 8th IEEE International Conference on Cloud Computing, pp. 469– 476. IEEE (2015)
[4] Dong, T., Xue, F., Xiao, C. and Zhang, J., 2021. Workflow scheduling based on deep reinforcement learning in the cloud environment. *Journal of Ambient Intelligence and Humanized Computing*, *12*(12), pp.10823-10835.
[5] Pan, J. and Wei, Y., 2024. A deep reinforcement learning-based scheduling framework for real-time workflows in the cloud environment. *Expert Systems with Applications*, *255*, p.124845.
[6] Wu, Fuhui, et al. "Workflow Scheduling in Cloud: A Survey." The Journal of Supercomputing, vol. 71, no. 9, 13 May 2015, pp. 3373–3418, https://doi.org/10.1007/s11227-015-1438-4.
[7] Mangalampalli, S., Hashmi, S.S., Gupta, A., Karri, G.R., Rajkumar, K.V., Chakrabarti, T., Chakrabarti, P. and Margala, M., 2024. Multi objective prioritized workflow scheduling using deep reinforcement based learning in cloud computing. *IEEE Access*, *12*, pp.5373-5392.
[8] Chen, G., Qi, J., Sun, Y., Hu, X., Dong, Z. and Sun, Y., 2023. A collaborative scheduling method for cloud computing heterogeneous workflows based on deep reinforcement learning. *Future Generation Computer Systems*, *141*, pp.284-297.
[9] Nascimento, A., Olimpio, V., Silva, V., Paes, A. and de Oliveira, D., 2019, May. A reinforcement learning scheduling strategy for parallel cloud-based workflows. In *2019 IEEE international parallel and distributed processing symposium workshops (IPDPSW)* (pp. 817-824). IEEE.
[10] Jayanetti, A., Halgamuge, S. and Buyya, R., 2022. Deep reinforcement learning for energy and time optimized scheduling of precedence-constrained tasks in edge–cloud computing environments. *Future Generation Computer Systems*, *137*, pp.14-30.
[11] Kaur, A., Singh, P., Singh Batth, R. and Peng Lim, C., 2022. Deep-Q learning-based heterogeneous earliest finish time scheduling algorithm for scientific workflows in cloud. *Software: Practice and Experience*, *52*(3), pp.689-709.
[12] Cui, D., Peng, Z., Ke, W., Hong, X. and Zuo, J., 2018. Cloud workflow scheduling algorithm based on reinforcement learning. *International Journal of High Performance Computing and Networking*, *11*(3), pp.181-190.
[13] Qin, Y., Wang, H., Yi, S., Li, X. and Zhai, L., 2021. A multi-objective reinforcement learning algorithm for deadline constrained scientific workflow scheduling in clouds. *Frontiers of Computer Science*, *15*, pp.1-12.
[14] Reddy, P.V. and Reddy, K.G., 2023. An energy efficient RL based workflow scheduling in cloud computing. *Expert Systems with Applications*, *234*, p.121038.
[15] Uma, J., Vivekanandan, P. and Shankar, S., 2022. Optimized intellectual resource scheduling using deep reinforcement Q-learning in cloud computing. *Transactions on Emerging Telecommunications Technologies*, *33*(5), p.e4463.

[16] Wu, Quanwang, et al. "Deadline-Constrained Cost Optimization Approaches for Workflow Scheduling in Clouds." IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 12, 1 Dec. 2017, pp. 3401–3412, https://doi.org/10.1109/tpds.2017.2735400. Accessed 3 May 2023.

