Journal of Computer Science

Research Article

MCWDRL: Multi-Cloud Workflow Scheduling Using Deep Reinforcement Learning and Improved Workflow Segmentation for Multi-Cloud Environments

S. Gowri[1] and A. Sumathi's[2]

[1]*Department of Computer Science, KPR College of Arts Science and Research, Arasur Coimbatore, Tamil Nadu, India*
[2]*Department of AI&ML, KPR College of Arts Science and Research, Arasur Coimbatore, Tamil Nadu, India*

Article history
Received: 22-12-2025
Revised: 27-05-2026
Accepted: 12-06-2026

Corresponding Author:
S. Gowri
Department of Computer Science, KPR College of Arts Science and Research, Arasur Coimbatore, Tamil Nadu, India
Email: sgowri83@gmail.com

Abstract: The rapid growth of cloud computing has led to complex workflow scheduling challenges in multi-cloud environments, where efficient resource utilization, minimized makespan, and reduced costs are paramount. Traditional scheduling algorithms often struggle to adapt to the dynamic nature of cloud resources and workflows, leading to suboptimal performance. To overcome this issue, the paper proposed a novel Multi-Cloud Workflow Scheduling using Deep Reinforcement Learning and Improved Workflow Segmentation (MCWDRL) approach that addresses these challenges by synergistically combining deep reinforcement learning, workflow segmentation, and scheduling techniques. MCWDRL optimizes workflow execution by partitioning workflows into manageable segments, leveraging deep reinforcement learning to adapt to changing cloud environments, and implementing a scheduling policy that minimizes makespan and resource cost. By integrating these components, MCWDRL offers a robust and adaptive solution for workflow scheduling in multi-cloud environments, outperforming existing algorithms and demonstrating significant improvements in workflow execution efficiency and resource utilization.

Keywords: Multi-Cloud Computing, Makespan, Resource Utilization, Resource Cost, Scheduling

Introduction

Cloud computing represents a dynamic concept that enables broad availability of a shared pool of computing resources like servers, storage, applications, etc., via the Internet. This model enables individuals and organizations to tap into scalable, flexible, and cost-effective computing resources, eliminating the need for upfront capital expenditures on hardware and software (Younis et al., 2024; Sami et al., 2019; Wang and Deng, 2023). Cloud computing has transformed the way data is stored, processed, and managed, offering numerous benefits, including increased agility, improved collaboration, and enhanced data security. The cloud computing model is typically categorized into three service models: IaaS, PaaS, and SaaS. As a result, cloud computing has become an essential component of modern computing, enabling businesses and individuals to leverage the power of technology to drive innovation, improve efficiency, and reduce costs (Younge et al., 2010; Lee et al., 2015).

The process of scheduling cloud computing tasks entails assigning these tasks to various resources in the cloud, such as virtual machines, containers, and/or actual physical servers. Some of the objectives are resource utilization, minimizing the makespan time of tasks, and increasing system throughput while adhering to some constraints. Cloud task scheduling involves predicting the required resources for tasks, identifying available resources, selecting the most suitable resources, and allocating tasks to those resources, taking into account factors such as task priority, resource availability, cost, and Quality of Service (QoS) requirements (Maryam et al., 2018; Ferrer et al., 2016). Effective cloud task scheduling is crucial to ensure efficient resource utilization, minimize latency, and maximize throughput, and various scheduling algorithms and techniques, such as heuristic, meta-heuristic, and machine learning-based approaches, are employed to achieve these objectives in cloud computing environments (Mao et al., 2016).

DOI: 10.3844/jcssp.2026.1949.1958

Multi-cloud task scheduling using Deep Reinforcement Learning (DRL) is an emerging approach that leverages the strengths of DRL to optimize task scheduling across multiple cloud environments (Mao et al., 2019; Imran et al., 2020). In this paradigm, a DRL agent learns to schedule tasks across multiple cloud providers, taking into account factors such as resource availability, pricing, latency, and QoS requirements, to optimize objectives like cost, makespan, and energy consumption. The DRL agent interacts with the multi-cloud environment, observing the current state of resources and tasks, and learns to make decisions that maximize the cumulative reward, which is typically defined based on the optimization objectives. By leveraging techniques like Deep Q-Networks (DQN), Policy Gradient Methods, or Actor-Critic Methods, the DRL agent can learn complex scheduling policies that adapt to changing cloud environments, improving overall efficiency, scalability, and reliability (Ardagna et al., 2014; Addis et al., 2013).

Figure 1 illustrates the multi-cloud task scheduling process using DRL. Cloud users submit tasks to the task queue, which are retrieved by the DRL agent. The agent interacts with the cloud system to get the current state of resources, tasks, and QoS, and takes an action to schedule tasks to the clouds. The cloud system executes the tasks and provides a reward to the agent, which updates its policy using a DRL algorithm (Vullam et al., 2025; Zhao et al., 2021). The agent generates an optimized schedule, which is sent to cloud providers for execution, enabling efficient and adaptive task scheduling across multiple clouds.

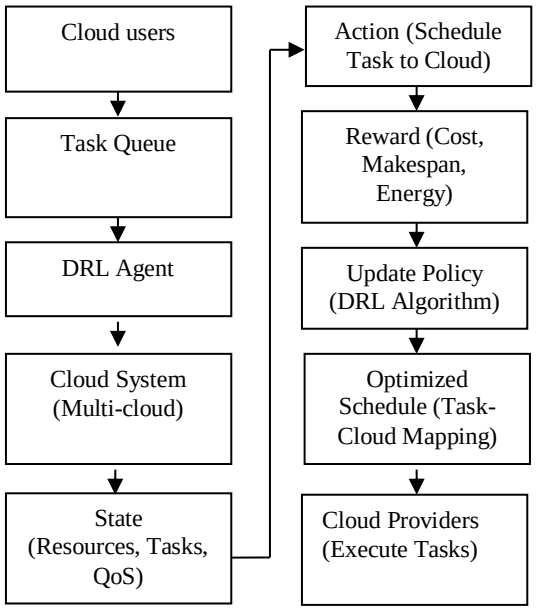

Fig. 1: Multi-Cloud Task Scheduling using DRL

This paper proposes MCWDRL, a novel Multi-Cloud Workflow Scheduling model that combines Deep Reinforcement Learning with Improved Workflow Segmentation for efficient task scheduling across multi-cloud environments. MCWDRL differs from standard scheduling methods through its adaptive learning capability, workflow partitioning strategy, and optimized scheduling policy tailored to multi-cloud complexities, allowing it to effectively address complex workflow scheduling problems with multiple constraints and objectives. The proposed MCWDRL method adopts the advantages of deep reinforcement learning and workflow splitting techniques to implement scheduling and resource management in a multi-cloud setting, thereby minimizing the makespan, costs, and energy usage while ensuring efficient resource management.

Literature Review

Tong et al. (2020) introduced a fresh AI algorithm, DQTS, which leverages the power of Q-learning and DNNs to tackle DAG tasks in cloud computing environments. By integrating DQL into task scheduling, their approach enables efficient handling of complex workflows. Experiments conducted using WorkflowSim demonstrate the effectiveness of DQTS in optimizing makespan and load balance, outperforming existing methods.

Ajmera and Tewari (2021) proposed an energy-efficient Virtual Machine Scheduling approach, VMS-MCSA, leveraging an MCSA inspired by the Artificial Immune System. By adapting CSA operators for discrete optimization, their approach tackles dynamic VM scheduling challenges. Additionally, a VM-consolidation model enables constraint-based migration, optimizing energy efficiency in cloud environments.

Xia et al. (2022) introduced an initialization scheduling series method that considers task data volume when initializing VM instances, achieving a balance between makespan and energy consumption. Their approach leverages traditional crossover and mutation operators for early exploration, while preserving favorable gene blocks (longest common subsequences) from elite individuals in later stages, reducing disruption during evolution and optimizing task scheduling.

Mangalampalli et al. (2023) provide a summary of cloud computing, highlighting its revolutionary impact on the IT industry and beyond, offering a pay-as-you-go model for diverse services. They discuss service models, deployment models, and virtualization techniques, emphasizing the crucial role of task schedulers for seamless cloud services. They discussed a priority-based task scheduling algorithm, leveraging the Chaotic Social Spider Algorithm, and simulated it on CloudSim, demonstrating its effectiveness in optimizing task scheduling in cloud surroundings.

Mangalampalli et al. (2024b) address the challenge of workflow scheduling in cloud computing, where dynamic workflows with varying task dependencies are generated from heterogeneous resources. They propose a novel multi-objective workflow scheduling algorithm using DRL, which calculates the precedence of workflows and VMs based on dependencies and electricity costs, respectively. These priorities are fed to a scheduler that leverages a model to optimize by mapping workflows to suitable VMs.

Yu et al. (2025) highlight the essential role of efficient task scheduling in cloud computing, introducing RL-MOTS, a cutting-edge framework that harnesses DQN to dynamically allocate tasks across virtual machines. Incorporating multi-objective optimization achieves a balance among reducing energy consumption, reducing costs, and QoS, while adapting to real-time resource utilization, task deadlines, and energy metrics in diverse cloud environments, demonstrating robust performance under varying workload conditions.

Gowri and Sumathi (2025) introduced AGOSA, a novel scheduling algorithm tailored for multi-cloud environments, integrating cloud and data center models to optimize task scheduling and resource allocation. Leveraging grasshopper optimization principles, AGOSA efficiently explores the solution space.

Materials and Methods

This section presents a proposed methodology for Multi-Cloud Workflow Scheduling using Deep Reinforcement Learning and Improved Workflow Segmentation (MCWDRL). The complete workflow is shown in Figure 2.

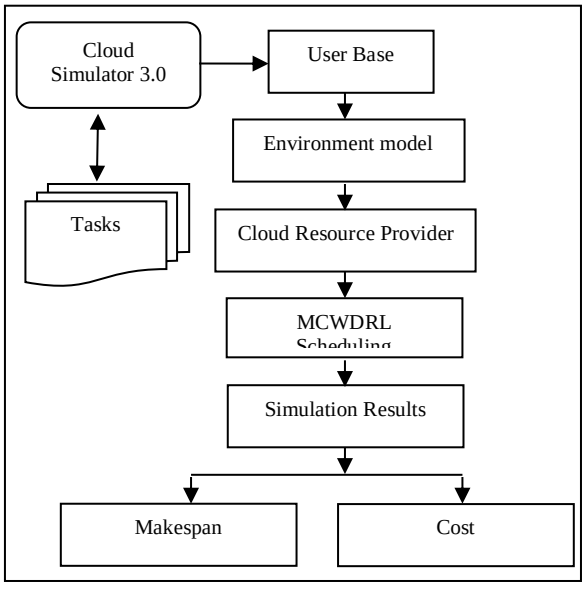

Fig. 2: MCWDRL Flow Diagram

Environment Model

The Environment model simulates a cloud environment using CloudSim 3.0, comprising cloud resources and workflow simulation. The proposed method utilized CloudSim 3.0 to generate a cloud model with multiple data centers and VMs. VM characteristics like MIPS capacity, memory, storage, and bandwidth are defined, along with data center properties like host count, location, and resource allocation policies. The cloud model is configured using CloudSim API, setting up network settings and scheduling policies. The DRL agent interacts with this simulated environment, learning to optimize workflow scheduling decisions.

Cloud Datacenter Creation

The Cloud Datacenter Creation model is a crucial component of the CloudSim 3.0 simulator, responsible for creating and managing cloud datacenters. A datacenter is a collection of computing resources, such as servers, storage, and network devices, that provide cloud services to users. The datacenter creation model involves defining the datacenter's characteristics, such as its architecture, capacity, and configuration, including specifying the number and type of hosts, the number and type of Virtual Machines (VMs), and the network topology. In a multi-cloud environment, the datacenter creation model is extended to support multiple cloud providers, including Microsoft, Google, and Amazon, as pointed out by Gowri and Sumathi (2025). The CloudSim 3.0 simulator provides a range of features and tools to support the creation and management of cloud datacenters, including a graphical user interface and an API for customizing and extending the simulator, allowing the simulation and evaluation of cloud-based applications and services across multiple cloud providers.

In a multi-cloud environment, the datacenter creation model can be extended to consider multiple cloud providers, each with its own set of hosts, VMs, and network topology.

Let $D = (D_1, D_2, \ldots, D_k)$ be a set of k cloud datacenters, where each datacenter D_i is a tuple of (H_i, V_i, N_i, P_i). $H_i = (h_{i1}, h_{i2}, \ldots, h_{imi}$ is a set of m_i hosts in datacenter D_i. $V_i = (v_{i1}, v_{i2}, \ldots, v_{ini}$ is a set of n_i VMs in datacenter Di. N_i is the network topology of datacenter D_i. P_i is the set of resource management policies in datacenter D_i.

The objective function can be formulated as:

$$U_{max} = \sum_{i=1}^{k} \sum_{j=1}^{m_i} \sum_{l=1}^{n_i} x_{ijl} \times (cpu_{il} + mem_{il} + storage_{il} + bw_{il}) \quad (1)$$

Where U_{max} is Utilization maximization.

To minimize the Energy Consumption (EC), it can be formulated as:

DOI: 10.3844/jcssp.2026.1949.1958

$$EC = \sum_{i=1}^{k} \sum_{j=1}^{m_i} E_{ij} \quad (2)$$

To minimize the cost can be formulated as:

$$Cost = \sum_{i=1}^{k} \sum_{j=1}^{m_i} C_{ij} \quad (3)$$

The Computing Capacity Constraint ensures that the aggregate CPU demand of all VMs assigned to a host does not surpass the host's CPU capacity, preventing over-provisioning and guaranteeing sufficient processing resources for all VMs, thereby maintaining optimal performance and preventing potential bottlenecks in the cloud datacenter. For each host *j* in datacenter *i*, and VM index *l* is defined as:

$$\sum_{l=1}^{n_i} x_{ijl} \times cpu_{il} \leq cpu_{ij} \quad (4)$$

This constraint ensures that the total CPU requirement of all VMs assigned to a host does not exceed the CPU capacity of the host.

The Memory Constraint ensures that the aggregate memory demand of all VMs preventing memory over-provisioning and guaranteeing sufficient memory resources for all VMs assigned to a host, which is crucial for maintaining optimal performance, preventing potential bottlenecks, and ensuring the overall reliability and efficiency of the cloud datacenter, by enforcing this constraint, the cloud provider can ensure that each host has sufficient memory to meet the requirements of its assigned VMs, thereby minimizing the risk of memory-related issues and ensuring a high level of service quality. This constraint ensures that the total memory requirement of all VMs assigned to a host does not exceed the memory capacity of the host:

$$\sum_{l=1}^{n_i} x_{ijl} \times mem_{il} \leq mem_{ij} \quad (5)$$

The Cost Constraint guarantees that the aggregate cost of all hosts across multiple datacenters does not exceed the allocated budget *B*, thereby preventing excessive expenditure and guaranteeing cost-effectiveness, this constraint enables cloud providers to manage their infrastructure costs effectively, make informed decisions about resource allocation, and ensure that their cloud deployment remains financially sustainable, by limiting the total cost of hosts, this constraint helps cloud providers balance their resource utilization with budgetary constraints, ensuring that they can deliver reliable and efficient services to their customers while maintaining a healthy bottom line:

$$\sum_{l=1}^{k} \sum_{j=1}^{m_i} C_{ij} \leq B \quad (6)$$

The Cloud Datacenter Creation model in CloudSim 3.0 simulator creates and manages cloud datacenters across multiple providers, defining characteristics like architecture, capacity, and configuration. It aims to maximize utilization, minimize energy consumption, and reduce cost, subject to computing capacity, memory, and budget constraints. The model ensures sufficient CPU and memory resources for VMs, prevents over-provisioning, and manages infrastructure costs effectively, ensuring reliable services and a healthy bottom line.

Multi-Cloud Workflow Scheduling Using Deep Reinforcement Learning and Improved Workflow Segmentation (MCWDRL)

This paper presents a Multi-Cloud Workflow Scheduling approach using Deep Reinforcement Learning and Improved Workflow Segmentation (MCWDRL), a novel methodology that optimizes workflow execution in multi-cloud environments by leveraging deep reinforcement learning, workflow segmentation, and scheduling techniques to minimize makespan and resource cost consumption while ensuring efficient resource utilization. The neural network architecture of the proposed MCWDRL model consists of an input layer that receives workflow and cloud resource state information, followed by two hidden layers, each containing 128 neurons with ReLU activation, and two separate output layers: A policy network with softmax activation to generate action probabilities, and a value network with linear activation to estimate state value. In MCWDRL, the Agent model learns the optimal scheduling policy using DRL, receiving environment state, and selecting actions to maximize performance metrics. The Workflow Module represents the workflow graph, including tasks and dependencies, while the Workflow Segmentation Module partitions the workflow into smaller segments to reduce data transmission overhead. The Agent implements the DRL algorithm, including the policy network, value network, and exploration strategy. The policy network and value network are implemented using Multi-Layer Perceptrons (MLPs) with two hidden layers, each consisting of 128 units with ReLU activation functions. The output layer of the policy network uses a softmax activation function to produce a probability distribution over the action space. The output layer of the value network uses a linear activation function to produce the estimated value. The agent receives the state of the environment (st), selects actions (at), and receives rewards or penalties (rt) based on the performance metrics. S = (s1, s2, …, sn), where si represents the current state of the environment (e.g., resource utilization, task status). Each state siincludes the following attributes: CPU utilization of hosts, memory utilization, bandwidth availability, VM allocation status, task execution status, workflow dependency information, queue length of pending tasks, energy consumption level, and estimated execution cost. Action Space: A = (a1, a2,

…, am), where aj represents the action taken by the agent (e.g., task scheduling, resource allocation). Reward Function: R (st, at) = rt, where rt is the reward or penalty received by the agent for taking action at in state st. Policy Network: π (at|st; θ), where θ represents the policy network parameters. Value Network: V (st, w), where w represents the value network parameters. Figure 3 illustrates a proposed MCWDRL architecture.

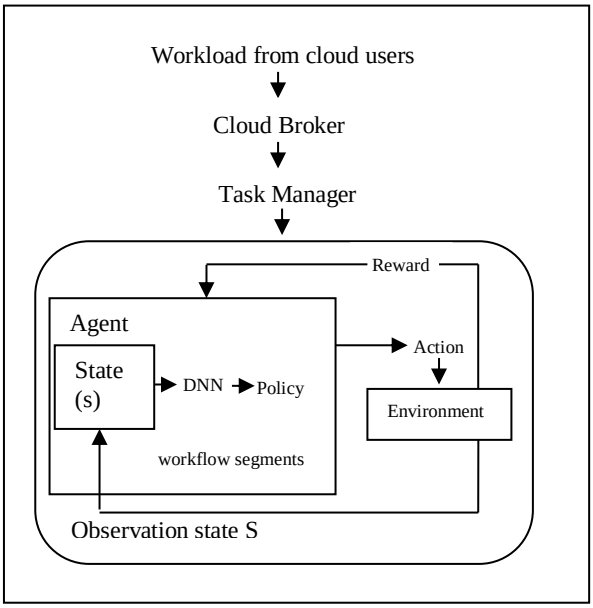

Fig. 3: MCWDRL architecture

The Workflow model is a key part of the MCWDRL approach, representing the workflow graph with tasks, dependencies, and resource needs. The workflow graph is mathematically modeled as:

$$G = (T, E) \quad (7)$$

Where $T = \{t1, t2, ..., tn\}$ represents the set of tasks, and E represents the dependencies between tasks, indicating the order in which tasks must be executed. Additionally, the task requirements are represented as R = (r1, r2, …, rn) where ri specifies the resource requirements of task ti, such as CPU, memory, and bandwidth, providing the agent with the necessary information to make informed decisions about task scheduling and resource allocation, ultimately enabling efficient workflow execution in the multi-cloud environment.

The Workflow Segmentation model plays a vital role in the MCWDRL approach by dividing the workflow into smaller, more manageable segments, thereby enhancing execution efficiency and reducing data transmission overhead. This is achieved through techniques such as graph partitioning, clustering, or community detection, which group tasks with high dependencies and data locality together, minimizing communication costs and improving overall workflow performance. This is achieved through the Improved Graph Partitioning (IGP) algorithm, which groups tasks with high dependencies and data locality together, minimizing communication costs and improving overall workflow performance.

Mathematically, the Workflow Segmentation Module is represented as G' = (S, E'), where S = (s1, s2, …, sk} denotes the set of segments, and E' represents the dependencies between these segments, indicating the order in which they should be executed. Each segment si in S contains a subset of tasks from the original workflow graph G = (T, E), and the dependencies between segments E' are derived from the original dependencies E, ensuring that the segmented workflow preserves the original workflow's execution logic. By partitioning the workflow into segments, the module enables the agent to schedule and allocate resources at the segment level, rather than at the individual task level, leading to improved execution efficiency and reduced scheduling complexity. The existing solutions for workflow scheduling, such as HEFT, A2C-based approaches, and MOPTSA3C, do task-level scheduling rather than optimize the workflow partitioning problem. The current workflow scheduler assumes a flat DAG structure and does not consider efficient partitioning. Hence, the overhead related to inter-task communication grows significantly when used in distributed cloud environments. The new method of IGP segmentation starts with splitting the graph based on task dependencies before executing the scheduling by the DRL agent. Algorithm 1 describes the proposed workflow segmentation of the IGP process.

Algorithm 1: IGP
Input: Workflow graph (G), Number of partitions (k), Maximum segment size (m)
Output: Partitioned workflow graph (S)
Process:
Step 1: Initialize k partitions
Step 2: Assign tasks to partitions using the graph partitioning technique
Step 3: For each partition s_i:
i. Calculate segment size $|s_i|$
ii. If $|si| > m$, split *si* into smaller sub-partitions
Step 4: Return partitioned workflow graph

Finally, a scheduling model in the MCWDRL approach is responsible for implementing the scheduling policy that governs task prioritization, resource allocation, and execution. This model plays a pivotal role in ensuring that tasks are executed efficiently, meeting the desired performance metrics such as minimizing makespan and resource cost consumption. Mathematically, the Scheduling Module is represented by a scheduling policy $\pi_s(a_t|s_t, \theta_s)$, where θ_s denotes the scheduling policy parameters that are learned and updated during the

training process. The objective function of the Scheduling Module is to minimize a combination of makespan, cost, and energy consumption, which are key performance metrics in cloud computing environments. The makespan refers to the total execution time of the workflow, cost encompasses the resource utilization costs, and energy consumption represents the energy expended during task execution. By optimizing these metrics, the Scheduling Module ensures that tasks are executed in a manner that maximizes resource utilization, reduces costs, and minimizes environmental impact, thereby achieving efficient workflow execution in the multi-cloud environment.

The reward function is designed to encourage the agent to minimize the makespan and resource cost. The reward at each time step *t* is calculated as:

$$rt = -(\alpha * \text{makespan} + \beta * \text{resource cost} + \lambda \times \text{EC}) + \phi \times \text{RU} \quad (8)$$

Where α, β, λ and ϕ are weighting coefficients controlling the importance of each objective. The MCWDRL method will iteratively update the scheduling policy parameters (θ) based on the observed state, selected action, and received reward, using policy gradient or value iteration, to search for the scheduling solution. Algorithm 2 outlines the workflow of the proposed MCWDRL model.

Algorithm 2: MCWDRL
Input: Workflow graph (G), Cloud resources (R), Objective function (F), Maximum iterations (MaxIteration)
Output: Optimal schedule
Process
Step 1: Initialize DRL model parameters (θ)
Step 2: Segment the workflow graph (G) into smaller partitions (P) using the improved graph partitioning segmentation technique
Step 3: DRL Training
For each iteration (iter < MaxIteration)
For each partition (*p* in *P*)
i. Observe state (s) of the environment
ii. Select action (a) using policy network (π)
iii. Schedule tasks in partition (p) using action (a)
iv. Receive reward (r) based on objective function (F)
v. Update DRL model parameters (θ) using policy gradient or value iteration
Step 4: Return the best solution (schedule) found during training

Results and Discussion

The proposed MCWDRL method was utilized to assess the scheduling performance in multi-cloud computing environments, with a comprehensive performance evaluation conducted to compare its efficacy against existing state-of-the-art algorithms, including AGOSA (Gowri and Sumathi, 2025), HEFT (Wang and Vassileva, 2023), A2C (Lu et al., 2024), and MOPTSA3C (Mangalampalli et al., 2024a). The experimental setup involves simulating a multi-cloud environment using CloudSim 3.0, with three cloud data centers from Microsoft, Google, and Amazon, each with 100-500 VMs and varying resource capacities. Three workloads (100, 500, and 1000 tasks) are generated using a uniform distribution, with task execution times (10-100 seconds) and resource requirements (CPU, memory, and storage) randomly generated. The MCWDRL agent is trained using the Adam optimizer with a learning rate of 0.001 and a batch size of 32. The exploration-exploitation trade-off is managed using an ε-greedy policy with ε = 0.1. The agent is trained for a maximum of 100 episodes, with each episode consisting of 50-time steps. The training process is terminated when the agent's performance converges, which is determined by monitoring the average reward over a window of 100 episodes. If the average reward does not improve for 10 consecutive episodes, the training process is stopped.

The proposed MCWDRL differs from existing DRL-based workflow schedulers in its focus on multi-cloud environments and use of a novel Improved Graph Partitioning (IGP) segmentation technique, which considers data locality, task dependencies, and resource constraints to optimize both makespan and resource cost. Unlike other approaches, IGP's graph partitioning approach enables efficient handling of complex workflows, while its awareness of data locality and resource constraints ensures reduced data transfer costs and improved scheduling efficiency. This unique combination enables MCWDRL to outperform existing DRL-based schedulers in multi-cloud environments. The important hyperparameters used in the proposed MCWDRL framework are summarized in Table 1.

Table 2 and Figure 4 show the makespan comparisons using uniform distribution.

Table 1: Hyperparameter Setting

Hyperparameter	Value
Learning Rate	0.001
Discount Factor	0.99
Batch size	32
Number of Hidden Layers	2
Hidden Units per Layer	128
Activation Function	ReLU
Optimizer	Adam
Exploration Strategy	ϵ-greedy
Initial Exploration Rate (ϵ)	0.1
Minimum Exploration Rate	0.01

Table 2: Uniform distribution-based makespan comparison

Tasks	A2C	MOPTSA3C	HEFT	AGOSA	MCWDRL
100	712.08	688.18	642.5	600.21	589.51
500	809.26	709.27	650.3	618.22	601.58
1000	887.12	723.38	710.5	700.15	686.5

DOI: 10.3844/jcssp.2026.1949.1958

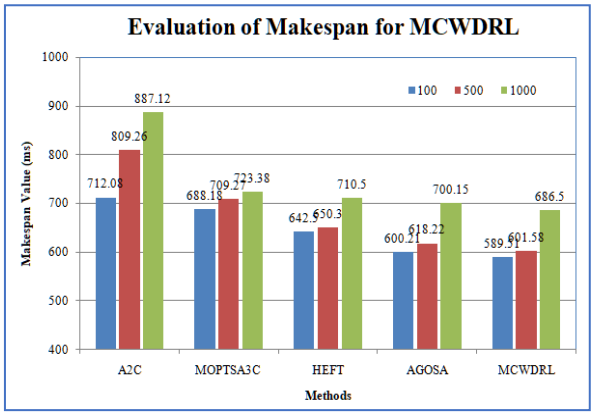

Fig. 4: Makespan chart

Table 3 presents a comprehensive comparison of the resource costs associated with the proposed MCWDRL method and existing algorithms, including A2C, MOPTSA3C, and AGOSA, for tasks of unstable dimensions, specifically 100, 500, and 1000, utilizing a uniform distribution. The results indicate that the MCWDRL method achieves considerably lower resource costs, with values of 4.15, 4.62, and 6.10 for 100, 500, and 1000 tasks, respectively, thereby outperforming the other algorithms. This suggests that MCWDRL is more efficient in terms of resource utilization, which is a critical factor in cloud computing environments where resource costs can have a substantial impact on overall operational expenses. The greater performance of MCWDRL can be attributed to its ability to optimize resource allocation and scheduling, making it a more effective solution for managing resources in cloud computing environments. The resource cost chart is described in Figure 5.

The proposed method executes tasks on cloud VMs from Microsoft, Google, and Amazon, evaluating performance using makespan and cost metrics for 100, 500, and 1000 tasks. Results in Tables 4 to 5 and Figures 6-7 show its effectiveness in managing cloud resources and optimizing task execution, handling diverse workloads while minimizing costs.

Table 4 presents a comprehensive evaluation of the resource makespan for the proposed method across multiple cloud platforms, including Microsoft, Google, and Amazon, for tasks of unstable dimensions, particularly 100, 500, and 1000 tasks. The results indicate that the makespan values for Microsoft are 590.18, 621.28, and 698 for 100, 500, and 1000 tasks, respectively, while Google's makespan values are 608.15, 600, and 672, and Amazon's makespan values are 582, 614, and 701. These results demonstrate the method's performance in managing resources and executing tasks across different cloud platforms, highlighting its ability to efficiently handle diverse workload sizes and minimize makespan, a critical factor in cloud computing environments.

Table 3: Uniform distribution-based resource cost

Tasks	A2C	MOPTSA3C	HEFT	AGOSA	MCWDRL
100	4.98	4.41	4.38	4.32	4.15
500	5.87	5.25	4.98	4.86	4.62
1000	6.22	6.72	6.55	6.21	6.10

Table 4: Makespan Comparison across Cloud Platforms

Tasks	Microsoft	Google	Amazon
100	590.18	608.15	582
500	621.28	600	614
1000	698	672	701

Table 5: Resource cost comparison across Cloud Platforms

Tasks	Microsoft	Google	Amazon
100	4.11	5.25	6.23
500	4.75	4.16	5.89
1000	6.89	7.02	7.0

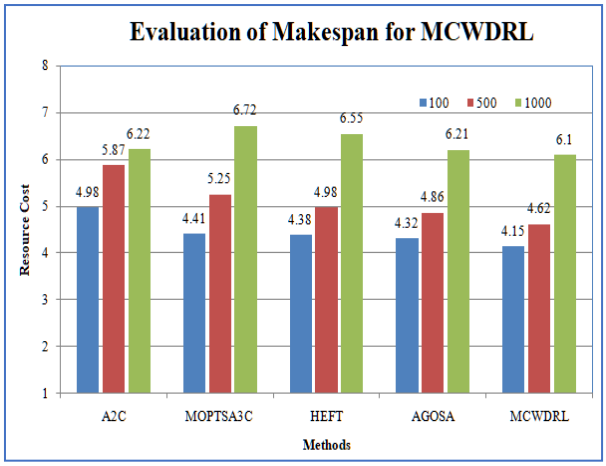

Fig. 5: Resource cost chart

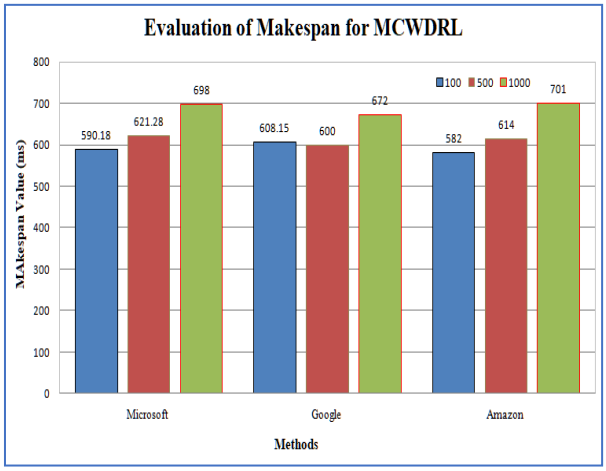

Fig. 6: Makespan performance across cloud providers (Microsoft, Google, Amazon)

DOI: 10.3844/jcssp.2026.1949.1958

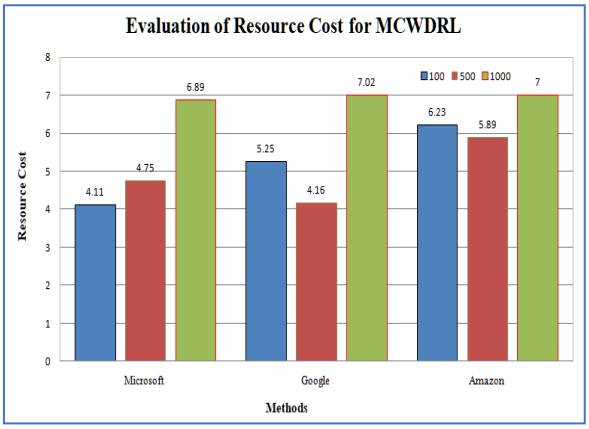

Fig. 7: Cost evaluation performance across cloud providers (Microsoft, Google, Amazon)

Table 5 presents a detailed evaluation of the resource costs associated with the proposed method across multiple cloud platforms, including Microsoft, Google, and Amazon, for tasks of unstable dimensions, particularly 100, 500, and 1000 tasks, using a uniform distribution. The results indicate that the resource costs for Microsoft are 4.11, 4.75, and 6.89 for 100, 500, and 1000 tasks, respectively, while Google's resource costs are 5.25, 4.16, and 7.02, and Amazon's resource costs are 6.23, 5.89, and 7.0. These results demonstrate the method's ability to manage resource costs effectively across different cloud platforms, highlighting its potential to minimize costs and optimize resource utilization in cloud computing environments.

To statistically validate the performance improvements achieved by MCWDRL, confidence interval analysis and hypothesis testing were performed. The 95% confidence interval for each performance metric was calculated using:

$$ConfInrvl = sm \pm 1.96 \times \frac{\sigma}{\sqrt{n}} \quad (9)$$

Where *sm* represents the sample mean, σ denotes the standard deviation, *and n* represents the number of experimental runs. The obtained *p*-values for makespan and resource cost were consistently lower than 0.05, confirming that the improvements achieved by MCWDRL are statistically significant at a 95% confidence level. Table 6 and Figure 8 show the confidence chart.

Table 6: Comparison of *p*-value

Methods	P = value
A2C	0.031
MOPTSA3C	0.024
HEFT	0.018
AGOSA	0.011
MCWDRL	0.009

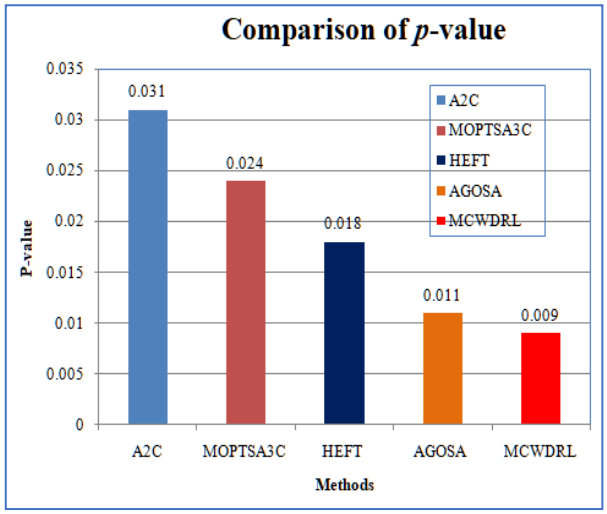

Fig. 8: p-value chart

Conclusion

The paper proposed a Multi-Cloud Workflow Scheduling using Deep Reinforcement Learning and Improved Workflow Segmentation (MCWDRL) approach, which presents a groundbreaking solution for optimizing workflow execution in multi-cloud environments. By synergistically combining deep reinforcement learning, workflow segmentation, and scheduling techniques, MCWDRL minimizes makespan and resource cost consumption while ensuring efficient resource utilization. The approach leverages CloudSim 3.0 to simulate a cloud environment, creating a cloud model with multiple data centers and virtual machines, and defining their characteristics. The workflow is represented as a graph with tasks and dependencies. The Workflow Segmentation Module partitions the workflow into smaller segments, reducing data transmission overhead. The Scheduling Module implements the scheduling policy, governing task prioritization, resource allocation, and execution. MCWDRL offers a robust and adaptive solution for workflow scheduling in multi-cloud environments, outperforming existing algorithms and demonstrating significant improvements in workflow execution efficiency and resource utilization.

Acknowledgment

The authors acknowledge the support and cooperation rendered by all the members, directly and indirectly.

Funding Information

The authors received no specific funding for this study.

Author's Contributions

S. Gowri: Responsible for content creation, design, data collection, and ensuring the novelty of the work.

A. Sumathi's: Refined the content, verified the novelty, and ensured the logical flow of the work.

Ethics

No ethics approval is required.

References

Addis, B., Ardagna, D., Panicucci, B., Squillante, M. S., & Zhang, L. (2013). A Hierarchical Approach for the Resource Management of Very Large Cloud Platforms. *IEEE Transactions on Dependable and Secure Computing*, *10*(5), 253–272. https://doi.org/10.1109/tdsc.2013.4

Ajmera, K., & Tewari, T. K. (2021). VMS-MCSA: virtual machine scheduling using modified clonal selection algorithm. *Cluster Computing*, *24*(4), 3531–3549. https://doi.org/10.1007/s10586-021-03320-5

Ardagna, D., Gibilisco, G. P., Ciavotta, M., & Lavrentev, A. (2014). A Multi-model Optimization Framework for the Model Driven Design of Cloud Applications. *Search-Based Software Engineering*, *8636*, 61–76. https://doi.org/10.1007/978-3-319-09940-8_5

Ferrer, A. J., Pérez, D. G., & González, R. S. (2016). Multi-cloud Platform-as-a-service Model, Functionalities and Approaches. *Procedia Computer Science*, *97*, 63–72. https://doi.org/10.1016/j.procs.2016.08.281

Gowri, S., & Sumathi, A. (2025). Advanced Grasshopper Optimization Scheduling Algorithm (AGOSA) for Multi-Cloud Environments. *Journal of Computer Science*, *21*(10), 2265–2272. https://doi.org/10.3844/jcssp.2025.2265.2272

Imran, H. A., Latif, U., Ikram, A. A., Ehsan, M., Ikram, A. J., Khan, W. A., & Wazir, S. (2020). Multi-Cloud: A Comprehensive Review. *Proceedings of the 2020 IEEE 23rd International Multitopic Conference*, 1–5. https://doi.org/10.1109/inmic50486.2020.9318176

Lee, Y. C., Han, H., Zomaya, A. Y., & Yousif, M. (2015). Resource-efficient workflow scheduling in clouds. *Knowledge-Based Systems*, *80*, 153–162. https://doi.org/10.1016/j.knosys.2015.02.012

Lu, J., Yang, J., Li, S., Li, Y., Jiang, W., Dai, J., & Hu, J. (2024). A2C-DRL: Dynamic Scheduling for Stochastic Edge–Cloud Environments Using A2C and Deep Reinforcement Learning. *IEEE Internet of Things Journal*, *11*(9), 16915–16927. https://doi.org/10.1109/jiot.2024.3366252

Mangalampalli, S. S., Karri, G. R., Mohanty, S. N., Ali, S., Ijaz Khan, M., Abdullaev, S., & AlQahtani, S. A. (2024a). Multi-Objective Prioritized Task Scheduler Using Improved Asynchronous Advantage Actor Critic (a3c) Algorithm in Multi Cloud Environment. *IEEE Access*, *12*, 11354–11377. https://doi.org/10.1109/access.2024.3355092

Mangalampalli, S., Hashmi, S. S., Gupta, A., Karri, G. R., Rajkumar, K. V., Chakrabarti, T., Chakrabarti, P., & Margala, M. (2024b). Multi Objective Prioritized Workflow Scheduling Using Deep Reinforcement Based Learning in Cloud Computing. *IEEE Access*, *12*, 5373–5392. https://doi.org/10.1109/access.2024.3350741

Mangalampalli, S., Sree, Pokkuluri Kiran, Swain, S. K., & Karri, G. R. (2023). Cloud computing and virtualization. *Convergence of Cloud with AI for Big Data Analytics: Foundations and Innovation*, 13–40. https://doi.org/10.1002/9781119905233.ch2

Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource Management with Deep Reinforcement Learning. *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, 50–56. https://doi.org/10.1145/3005745.3005750

Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., & Alizadeh, M. (2019). Learning scheduling algorithms for data processing clusters. *Proceedings of the ACM Special Interest Group on Data Communication*, 270–288. https://doi.org/10.1145/3341302.3342080

Maryam, K., Sardaraz, M., & Tahir, M. (2018). Evolutionary Algorithms in Cloud Computing from the Perspective of Energy Consumption: A Review. *Proceedings of the 2018 14th International Conference on Emerging Technologies*, 1–6. https://doi.org/10.1109/icet.2018.8603582

Sami, I., Ali, S. M., Nazir, S., Khan, I., Asghar, R., Abid, M. A., Ullah, Z., Khan, B., & Mehmood, C. A. (2019). Cloud computing (CC) centers - a fast-processing engine in smart grid. *Proceedings of the 2019 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, 1–6. https://doi.org/10.1109/icecce47252.2019.8940710

Tong, Z., Chen, H., Deng, X., Li, K., & Li, K. (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

Vullam, N. R., Geetha, G., Rao, N., Vellela, S. S., Syamsundara Rao, T., Thommandru, R., & Rao, K. N. S. (2025). Optimized Multitask Scheduling in Cloud Computing Using Advanced Machine Learning Techniques. *Proceedings of the 2025 International Conference on Intelligent Control, Computing and Communications (IC3)*, 410–415. https://doi.org/10.1109/ic363308.2025.10956844

Wang, H., & Deng, J. (2023). A Multi-Objective Cloud Workflow Scheduling Strategy Based on the Modified HEFT Algorithm. *Journal of Physics: Conference Series*, *2504*(1), 012060. https://doi.org/10.1088/1742-6596/2504/1/012060

Wang, Y., & Vassileva, J. (2023). Auto-scaling and resource management in cloud computing: A review. *Future Generation Computer Systems*, *123*(pp), 13–26. https://doi.org/10.3390/s24175551

Xia, X., Qiu, H., Xu, X., & Zhang, Y. (2022). Multi-objective workflow scheduling based on genetic algorithm in cloud environment. *Information Sciences*, *606*, 38–59. https://doi.org/10.1016/j.ins.2022.05.053

Younge, A. J., von Laszewski, G., Wang, L., Lopez-Alarcon, S., & Carithers, W. (2010). Efficient resource management for Cloud computing environments. *Proceedings of the 2010 International Conference on Green Computing*, 357–364. https://doi.org/10.1109/greencomp.2010.5598294

Younis, R., Iqbal, M., Munir, K., Javed, M. A., Haris, M., & Alahmari, S. (2024). A Comprehensive Analysis of Cloud Service Models: IaaS, PaaS, and SaaS in the Context of Emerging Technologies and Trend. *Proceedings of the 2024 International Conference on Electrical, Communication and Computer Engineering (ICECCE)*, 1–6. https://doi.org/10.1109/icecce63537.2024.10823401

Yu, X., Mi, J., Tang, L., Long, L., & Qin, X. (2025). Dynamic multi objective task scheduling in cloud computing using reinforcement learning for energy and cost optimization. *Scientific Reports*, *15*(1), 45387. https://doi.org/10.1038/s41598-025-29280-z

Zhao, J., Rodriguez, M. A., & Buyya, R. (2021). A Deep Reinforcement Learning Approach to Resource Management in Hybrid Clouds Harnessing Renewable Energy and Task Scheduling. *Proceedings of the 2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, 240–249. https://doi.org/10.1109/cloud53861.2021.00037