ORBIT - Optimized Resource Balanced Intelligent Task Scheduling in Cloud Datacentres leveraging Weighted A3C Deep Reinforcement Learning

Dhruv Mishra
Department of Computer Science and Engineering
Shiv Nadar Institution of Eminence
Delhi-NCR, India
dm409@snu.edu.in

Suchi Kumari
Department of Computer Science and Engineering
Shiv Nadar Institution of Eminence
Delhi-NCR, India
suchi.singh24@gmail.com

***Abstract*—Cloud data centres demand adaptive, efficient, and fair resource allocation techniques for heterogeneous scientific workflows with complex task dependencies. However, existing approaches struggle with dynamic workflow patterns and inter-task dependencies, resulting in suboptimal fairness, increased makespan, and higher energy consumption. We propose *ORBIT* (Optimized Resource Balanced Intelligent Task scheduling), a novel framework that leverages Weighted Actor-Critic Deep Reinforcement Learning to learn from the environment continuously and incorporates a multi-objective reward structure balancing Quality of Service (QoS), fairness, priority, deadline compliance, and energy consumption while respecting task dependencies. *ORBIT* introduces a priority-aware advantage estimator that captures task urgency within workflow dependency chains and implements a deadline penalty mechanism where tasks exceeding twice their estimated makespan receive penalties. The algorithm ensures fair resource distribution while prioritizing high-priority tasks and maintaining QoS guarantees. The lightweight and scalable architecture ensures feasibility in large-scale deployments. Extensive experiments using real-world scientific workflow traces (Cybershake, Epigenomics, Montage) demonstrate that *ORBIT* consistently outperforms traditional and reinforcement learning-based baselines across different workflow sizes and complexity levels, highlighting its potential for real-world deployment in scientific computing cloud systems.**

***Index Terms*—Cloud computing, Resource Allocation, Fairness, Priority Scheduling, Deadline Compliance, Deep Reinforcement Learning**

I. Introduction

Cloud data centres are an important part of modern scientific computing infrastructure, supporting diverse scientific workflows ranging from time-sensitive real-time tasks, such as earthquake simulation and gravitational wave detection, to delay-tolerant batch operations like genomics analysis and astronomical data processing. This workflow heterogeneity, combined with complex task dependencies, data transfer requirements, and increasing demands for energy efficiency, low makespan, and cost-effectiveness, presents major challenges for resource management in scientific computing environments. Critical challenges include ensuring fair resource allocation among competing workflows while respecting task priorities, meeting deadline constraints, and maintaining quality of service (QoS) guarantees.

Traditional approaches for scientific workflow scheduling, such as heuristic-based policies [1], [2] and static resource provisioning models [3], have provided lightweight solutions but often fail to adapt to dynamic and heterogeneous scientific workflows with complex dependency patterns [4]. Early machine learning techniques, including supervised learning models for workflow execution time prediction [5], [6] and queuing theory-based models [7], introduced adaptive decision-making but lacked real-time learning capabilities for handling dynamic workflow patterns. Reinforcement learning methods, such as Q-learning-based resource allocators [8], demonstrated more promising results by enabling agents to learn directly from interactions with scientific workflow environments. Beyond these, foundational Directed Acyclic Graph (DAG) scheduling and cloud workflow systems e.g., Heterogeneous Earliest Finish Time (HEFT) [9], Pegasus Workflow Management System (WMS) [10], CloudSim for reproducible evaluations [11], and multi-objective/cloud cost–deadline formulations [12], [13], established strong baselines yet still face limitations under non-stationary, multi-tenant workloads.

More recently, Deep Reinforcement Learning (DRL) methods have gained attention for scientific workflow resource management [14], [15]. Mnih *et al.* [16] proposed Asynchronous Advantage Actor-Critic (*A3C*), which used multiple DRL agents for interacting with the environment with reduced latency and enhanced resource utilisation for scientific computing applications. Chen *et al.* [17] proposed an *A3C*-based adaptive resource allocation strategy that significantly outperformed traditional and earlier RL techniques, demonstrating improved scientific workflow scheduling efficiency and energy savings. In broader cluster scheduling, learning-based schedulers such as Decima [18] and Google-scale orchestration insights from Borg [19] further motivate *RL*-driven, workload-aware policies. However, existing *DRL* models often treat all workflow tasks equally, neglecting task-specific priorities within dependency chains and fairness considerations among competing scientific workflows, which are critical in real-world multi-tenant cloud environments supporting diverse research applications.

Therefore, in this work, we propose Optimized Resource Balanced Intelligent Task scheduling (*ORBIT*), a novel framework that leverages a Weighted Asynchronous Advantage Actor–Critic formulation within a multi-agent reinforcement learning (*MARL*) paradigm for scientific workflow resource allocation. *ORBIT* extends the traditional A3C approach by embedding the RL model's dynamic reward function, which simultaneously integrates four key metrics: QoS compliance, fairness, task priority levels, and deadline adherence. To address latency-sensitive workloads, the framework incorporates a deadline-penalty mechanism whereby tasks exceeding twice their estimated makespan incur proportional penalties, thereby encouraging timely completion.

The *MARL* formulation enables distributed actor–critic agents to collaboratively optimize scheduling decisions across heterogeneous resources, reducing contention and enhancing scalability. By adaptively adjusting the reward structure according to workflow characteristics and dependency patterns, *ORBIT* ensures that high-priority tasks on critical paths receive timely allocations while preserving a fair distribution of computational capacity across competing workflows. Through extensive experimentation on real-world scientific workflow datasets, we demonstrate that the proposed *ORBIT* framework significantly outperforms baseline schedulers, standard DRL methods, and single-agent formulations in terms of efficiency, fairness, priority handling, and *QoS* compliance, positioning it as a principled foundation for multi-objective workflow scheduling in scientific computing environments.

The remainder of the manuscript is organised as follows. Section II describes the architecture and details of the proposed *ORBIT* framework and its underlying weighted *A3C* model. The environmental setup, the description of the scientific workflow datasets, the results, and the in-depth analysis are presented in Section III. Finally, Section IV concludes the paper and outlines the future direction.

II. Proposed Approach

In this work, we propose Optimized Resource Balanced Intelligent Task scheduling, a novel framework that leverages Weighted Asynchronous Advantage Actor-Critic for scientific workflow resource allocation. The framework extends the traditional A3C architecture by integrating four critical metrics: *QoS* compliance, fairness, task priority, and deadline adherence. These enhancements ensure that the scheduling policy prioritizes urgent tasks while preventing starvation of lower-priority workflows and enforcing equitable resource distribution.

A. *System Model*

The scientific workflow scheduling problem is formulated as a Markov Decision Process (*MDP*) defined by the tuple $\langle S, A, R, P, \gamma \rangle$, where the agent learns to optimize long-term performance using the proposed *ORBIT* framework.

- **State Space** (S): Each state $s_t \in S$ represents the workflow job snapshot at time t for job j:

$$s_t = \{ID_j, \mathcal{P}_j, S_j, M_j, T_t^j, D_t^j\}$$

  where ID_j is the job identifier, $\mathcal{P}_j$ represents the job priority, S_j denotes the job computational size, M_j indicates the job memory requirement, T_t^j is the job submission time, and D_t^j represents the job deadline.
- **Action Space** (A): The agent selects job j from queue Q_t using priority-weighted softmax:

$$\pi(a|s) \sim \text{softmax}(Q(s,j) + \beta \cdot \mathcal{P}_j)$$

- **Reward Function** (R): Multi-objective reward balancing four metrics:

$$R_t = w_1 R_t^{QoS} + w_2 R_t^{\mathcal{F}} + w_3 R_t^{\mathcal{P}} + w_4 R_t^{\mathcal{D}}$$

  where $w_1 = 0.3, w_2 = 0.3, w_3 = 0.25, w_4 = 0.15$ are the weight parameters. The weight distribution prioritizes QoS compliance and fairness as primary objectives ($w_1 = w_2 = 0.3$), reflecting the critical importance of maintaining service quality and equitable resource distribution in multi-tenant cloud environments. Task priority receives moderate emphasis ($w_3 = 0.25$) to ensure high-priority scientific workflows receive appropriate attention without compromising system-wide fairness. Deadline compliance is assigned the lowest weight ($w_4 = 0.15$) as it serves as a constraint mechanism rather than a primary optimization objective, with the penalty structure providing sufficient incentive for timely task completion. The process of selecting specific numerical values for w_1–w_4 requires deeper justification. In this work, the chosen weights are derived from empirical tuning and lightweight grid-search exploration, which is consistent with established practices ireinforcement-learning–based cloud scheduling [8]. Furthermore, several studies highlight the benefits of adaptive weighting mechanisms that adjust reward weights dynamically based on system state, workload profile, or performance drift, suggesting an important direction for future extension of the WA3C framework. All the rewards components are formulated as follows:
  - **QoS Reward** (R_t^{QoS})

$$R_t^{QoS} = 1 - \frac{\mathcal{P}_j}{\mathcal{P}_{\max}} - \frac{S_j}{S_{\max}} \tag{1}$$

    where $\mathcal{P}_{\max}$ is the maximum priority value, and $S_{\max}$ is the maximum job size.
  - **Fairness Reward** ($R_t^{\mathcal{F}}$)

$$R_t^{\mathcal{F}} = -\lambda \times \sigma^2 \left(\left\{ D_t^j - T_t^j \;\middle|\; j \in \text{active jobs} \right\} \right) \tag{2}$$

    where D_t^j and T_t^j represent the deadline and submission time of job j, respectively, and the parameter λ is the regularization factor that penalizes unfairness in terms of disparity in job slack durations.

– **Priority Reward** ($R_t^{\mathcal{P}}$)

$$R_t^{\mathcal{P}} = \mathcal{P}_t \times \frac{M_j}{M_{\max}} \tag{3}$$

– **Deadline Penalty** ($R_t^{\mathcal{D}}$)

$$R_t^{\mathcal{D}} = -\mu \times \mathbb{I}(T_t^j > 2 \times D_t^j) \tag{4}$$

where $\mathbb{I}$ is an indicator function penalizing jobs when submission time exceeds twice the job deadline.

- **Discount Factor** (γ): Fixed at $\gamma = 0.95$ for long-term optimization.

B. *ORBIT Algorithm*

Algorithm 1 ORBIT: Optimized Resource Balanced Intelligent Task Scheduling

```
1: Initialize: Global actor $A_\theta$ and critic $C_\phi$ networks
2: Initialize: Learning rates $\alpha_\theta$, $\alpha_\phi$, sync interval $u$
3: for each training epoch $n = 0, 1, 2, \ldots, N$ do
4:     Receive initial state $s_0 \leftarrow$ env.observe()
5:     for $t = 0, 1, 2, \ldots, T$ do
6:         Select action $a_t$ using priority-weighted softmax
7:         Execute $a_t$, observe reward $R_t$ and next state $s_{t+1}$
8:         Compute reward: $R_t = \sum_{i=1}^{4} w_i R_t^i$
9:         Compute advantage: $A(s_t, a_t) = R_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$
10:        Update critic: $\phi \leftarrow \phi + \alpha_\phi \delta_t \nabla_\phi V_\phi(s_t)$
11:        Update actor: $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta J(\theta)$
12:        Update state: $s_t \leftarrow s_{t+1}$
13:        if $t \bmod u = 0$ then
14:            Synchronize with global network
15:        end if
16:    end for
17: end for
```

The *ORBIT* algorithm implements a multi agent distributed reinforcement learning approach for scientific workflow scheduling. The detailed steps are provided in Algorithm 1. The algorithm begins by initializing the global actor and critic networks with random parameters, along with learning rates and synchronization interval. During each training epoch, the agent interacts with the cloud environment by observing the current system state and selecting scheduling actions using a priority-weighted softmax policy that considers both estimated action values and job priorities. Upon executing the selected action, the agent receives a multi-objective reward that combines QoS compliance, fairness, priority satisfaction, and deadline adherence. The advantage function is computed to estimate the relative value of the selected action compared to the baseline value function, enabling more stable policy updates. The critic network is updated using temporal difference learning to improve value estimation, while the actor network is updated using policy gradient methods weighted by the computed advantage. Periodic synchronization with the global network (every u steps) ensures that multiple worker agents can share learned experiences and maintain consistent policy parameters across the distributed training environment. This asynchronous update mechanism enables scalable training while preserving the stability and convergence properties of the actor-critic framework.

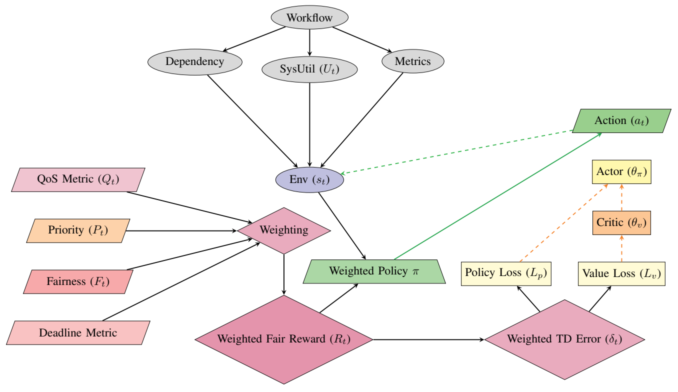


Fig. 1: Workflow Diagram of ORBIT

C. *Model Architecture*

The *ORBIT* framework realizes a principled architecture for multi-objective scientific workflow scheduling, as illustrated in Fig. 1. The architecture is designed to address the inherent trade-offs between competing system objectives, while enabling adaptive policy learning through reinforcement signals. It consists of the following key components.

Environment Layer: *ORBIT* models the scheduling scenario as a dynamic environment comprising heterogeneous workflows and computational resources. The state representation s_t captures job queue lengths, inter-task dependencies, and instantaneous resource utilization, thereby encoding the operational context necessary for informed decision-making. This abstraction formalizes workflow scheduling as a Markov Decision Process, enabling the application of advanced reinforcement learning methods.

Reward Formulation: A distinguishing feature of *ORBIT* lies in its structured reward design. Rather than optimizing a single objective, the framework explicitly quantifies four research-relevant dimensions: QoS compliance, priority satisfaction, fairness preservation, and deadline adherence. Each metric contributes to the overall system utility with tunable weights (e.g., 0.3, 0.3, 0.25, 0.15). This weighted aggregation produces a composite reward signal R_t that not only balances heterogeneous objectives but also permits sensitivity analysis across different weighting schemes, facilitating comparative research studies.

Policy Learning: The policy $\pi(a|s, w)$ is conditioned on both the observed state s_t and the reward formulation R_t. This design ensures that policy optimization remains directly aligned with the multi-objective reward space. The policy generates scheduling actions a_t, such as mapping tasks to resources, which are subsequently enacted within the environment. The placement of the policy module, adjacent to the reward computation, underscores the tight coupling between objective-driven evaluation and action generation.

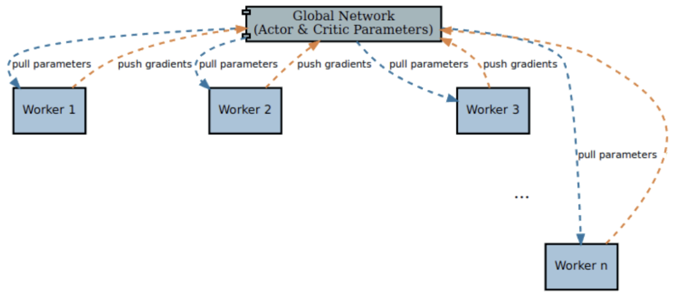


Fig. 2: Asynchronous MARL with push–pull updates: workers push gradients to the global network and pull updated parameters.

Optimization and Network Update: The right-hand side of the architecture captures the learning dynamics. *ORBIT* adopts a weighted temporal-difference (*TD*) error, which decomposes into policy and value loss components. The Actor network (θ_π) is optimized to refine the scheduling policy, while the Critic network (θ_v) improves value estimation. This dual-network configuration leverages the actor–critic paradigm, ensuring both stable convergence and adaptability to dynamic workloads. By integrating weighted objectives directly into the *TD* error signal, *ORBIT* extends classical reinforcement learning formulations to a multi-objective scheduling context.

Figure 2 illustrates the asynchronous multi-agent reinforcement learning (*MARL*) setup used in our scheduler. A shared *global network* holds the actor and critic parameters (θ, ϕ). Multiple workers (agents) interact with their local copies of the environment in parallel, collecting trajectories and computing local policy/value gradients. Each worker periodically *pushes* its gradients to the global network, which applies updates to (θ, ϕ), and then *pulls* the latest parameters to synchronize its local policy and value functions. This push–pull cycle enables scalable exploration (many workers learning concurrently), faster convergence (frequent parameter refresh), and more stable learning (implicit averaging across diverse experiences). In workflow scheduling, each worker observes current queue/load, selects actions (e.g., accept/defer/reject), receives rewards shaped by QoS, priority, fairness, and deadlines, and feeds back gradients so that the shared policy continually improves under heterogeneous, time-varying workloads.

III. Results and Analysis

This section provides the visual analysis and comparative results with the existing approaches along with experimental setup, including the environment, dataset, and performance metrics used to evaluate the proposed model.

A. *Experimental Setup and Dataset*

The experiments were conducted with a 2-socket Intel Xeon CPU *E5-2690 v4* machine, equipped with *32* cores per socket, running Ubuntu *20.04* LTS. We train and evaluate agents in a custom environment (`CloudJobSchedulingEnv`) that models online job admission with a 3-action space (reject, accept, defer) and a 6-dimensional observation vector capturing normalized queue size, resource usage, job priority, job size, system load, and memory usage. Episodes are capped at `max_steps` = 1000. For comparability across methods, reward shaping follows the equal-weight path for *A3C*-compatible trainers and varied weights for the weighted *A3C* path otherwise. All experiments are run with fixed seeds (NumPy/PyTorch: 42). By default, each algorithm is trained for `num_epochs` = 1000 episodes.

Our primary dataset comprises workflow traces spanning three domains: Cybershake, Epigenomics, and Montage, each provided at multiple scales (e.g., 25, 30, 50, 60, 100, 1000+ tasks). Each trace file is parsed line-by-line into jobs with fields {`id`, `priority`, `size`, `memory`, `submission_time`, `deadline`}. We group workflows by size using filename-derived task counts into three buckets: Small (0–100 tasks; 14 files), Medium (101–999 tasks; 3 files), and Large (1000+ tasks; 3 files).

Dataset Description: We considered three datasets from diverse domains to analyze the performance of *Orbit* framework. *Cybershake* is a seismic hazard analysis workflow that simulates earthquake ground motions across Southern California. The workflow exhibits relatively uniform task structures with predictable computational patterns, making it representative of scientific simulations with regular data dependencies. *Epigenomics* is a bioinformatics workflow for genome-wide analysis of DNA methylation patterns. This workflow demonstrates moderate complexity with heterogeneous task types including data preprocessing, statistical analysis, and visualization components. The workflow exhibits variable task dependencies and mixed computational requirements, representing typical bioinformatics pipelines with diverse processing stages. *Montage* is an astronomical image mosaic workflow that creates large-scale sky surveys by combining multiple telescope images. This workflow represents the most complex scheduling scenario with intricate data flow patterns, variable task sizes, and sophisticated inter-task dependencies.

B. *Performance Metrics*

To rigorously evaluate the efficacy of the *ORBIT* framework, we employ three complementary performance metrics that capture both aggregate learning efficiency and workload-specific behavior.

- **Episode Reward:** The primary metric is the cumulative reward per training episode, averaged using a 20-episode moving window to mitigate stochastic variability and highlight long-term learning trends. This measure reflects the stability and convergence characteristics of the learned policy over extended training horizons.
- **Convergence Efficiency:** We measure the number of training episodes required for each algorithm to achieve stable performance, defined as consistent reward values within 5% variance over a 50-episode window. This metric quantifies learning efficiency and practical deployment readiness.

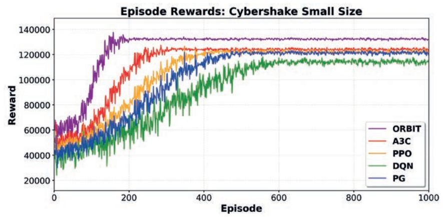


Fig. 3: Cybershake Small

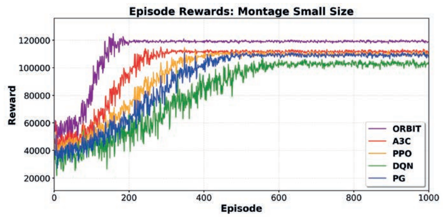


Fig. 4: Montage Small

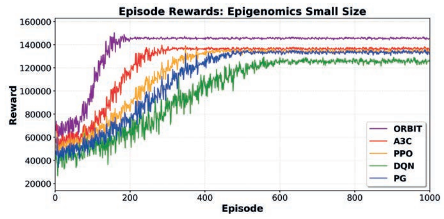


Fig. 5: Epigenomics Small

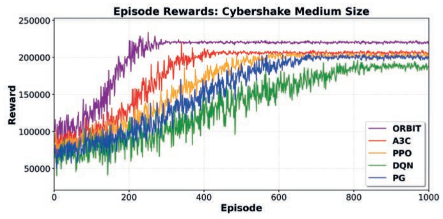


Fig. 6: Cybershake Medium

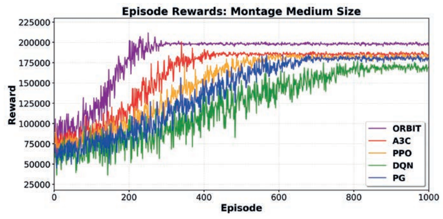


Fig. 7: Montage Medium

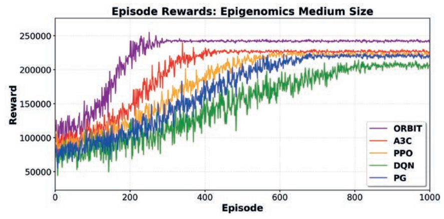


Fig. 8: Epigenomics Medium

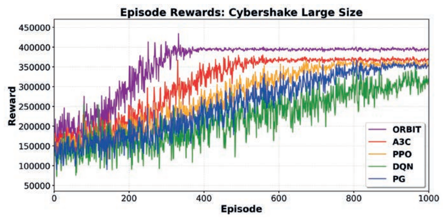


Fig. 9: Cybershake Large

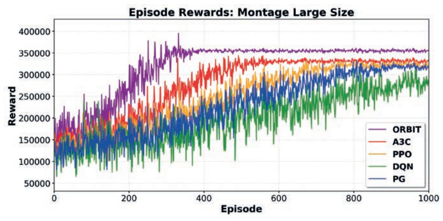


Fig. 10: Montage Large

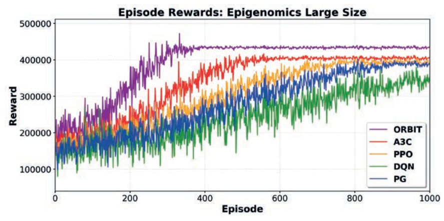


Fig. 11: Epigenomics Large

Fig. 12: Workflow Reward Comparison across Different Dataset Sizes and Workflow Types

- **Workflow-Type and Size-Stratified Analysis:** To investigate both scalability and domain-specific performance, we analyze performance across three representative workflow domains and three size categories: *Small*, *Medium*, and *Large*. This dual stratification provides insight into how *ORBIT* generalizes to varying computational demands and workflow structures.

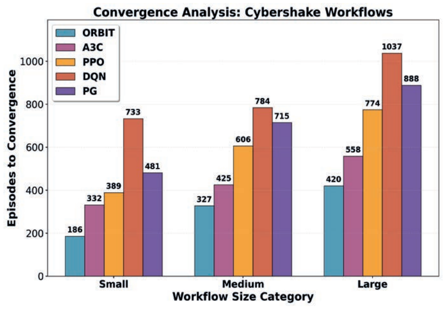
C. Comparison with DRL Baselines

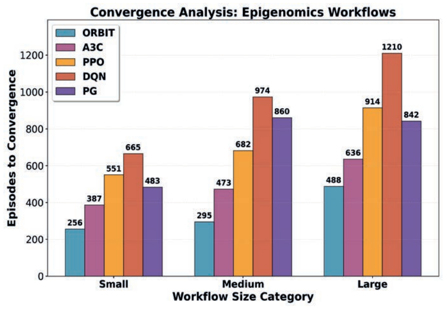
We benchmark *ORBIT* against four canonical deep reinforcement learning algorithms: *A3C*, *PPO*, *DQN*, and *PG*. All methods are trained under identical experimental conditions with fixed environment horizons, uniform workload traces, and consistent random seeds to ensure fair comparison and isolate the contribution of *ORBIT*'s architectural design.

The empirical results (cf. Figures 3–11) demonstrate that *ORBIT* consistently outperforms all baseline algorithms across workflow domains and size categories. *ORBIT* achieves higher asymptotic reward values, reduced variance, and improved stability of the learned policy across diverse workload characteristics. Analysis reveals distinct performance characteristics across workflow types: Cybershake workflows show consistent high rewards across all size categories due to uniform task structures, Epigenomics workflows exhibit moderate performance with higher variance due to heterogeneous task dependencies, and Montage workflows demonstrate the greatest performance differentiation between *ORBIT* and baseline methods.

Convergence Analysis: Figures 13a–13c present a detailed convergence analysis, highlighting the superior learning efficiency of *ORBIT* across all experimental conditions. The *ORBIT* framework consistently achieves the fastest convergence for every workload type and application. Specifically, it converges in fewer than 260 episodes for small workloads, within 450 episodes for medium workloads, and within 500 episodes for large workloads. In contrast, *DQN* requires the highest number of episodes across all applications and task sizes. The convergence order of the algorithms is $ORBIT < A3C < PPO < PG < DQN$. For the Epigenomics and Montage applications under large workloads, PG converges faster than PPO. Considering different applications, the overall convergence order is $Cybershake < Montage < Epigenomics$, consistent across all workload types.

These performance improvements can be attributed to three key architectural innovations: (i) a workload-aware state representation that enables faster pattern recognition in workflow characteristics, (ii) a balanced multi-objective reward formulation that provides clearer learning signals for complex scheduling decisions, and (iii) a policy design that efficiently explores the action space for dynamic workflow scheduling under heterogeneous workloads. In contrast to standard DRL


(a) Cybershake


(b) Epigenomics

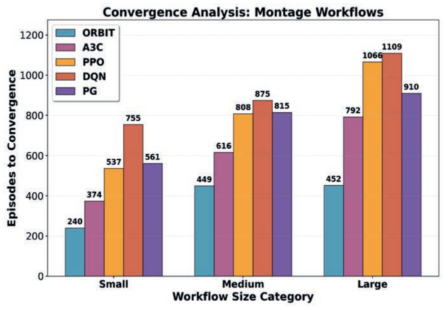

(c) Montage

Fig. 13: Convergence Analysis: Episodes required for stable performance across workflow types and size categories.

baselines that require extensive exploration in task-agnostic environments, *ORBIT* leverages domain-specific knowledge to accelerate learning convergence. The consistent convergence advantages demonstrate *ORBIT*'s practical viability for deployment in dynamic cloud environments where rapid adaptation to changing workloads is essential.

IV. Conclusion and Future Scope

This proposed *ORBIT* framework extends the actor–critic paradigm through a workload-aware state representation and a balanced, multi-objective reward design that jointly encodes QoS, job priority, fairness, and delay. *ORBIT* achieves accelerated convergence, higher asymptotic performance, and improved stability in heterogeneous workflow workloads. The workflow size is categorized into three types of tasks: *Small*, *Medium*, and *Large*. The results demonstrate that *ORBIT* consistently outperforms standard *DRL* baselines (*A3C*, *PPO*, *DQN*, *PG*) in both episode-level reward trajectories and size-specific convergence behavior. Performance gains are most pronounced for large workflows, where *ORBIT* sustains higher reward levels with reduced variance, evidencing its ability to manage complex task-graph dependencies and intensified resource contention. These findings establish ORBIT as a scalable, principled foundation for multi-objective workflow scheduling in dynamic and resource-constrained cloud settings.

Future research will extend *ORBIT* along several promising directions by incorporating meta-learning and enhance adaptability under non-stationary workloads. We can also include automated reward-weight tuning via multi-objective optimization to facilitate cross-domain deployment.

References

[1] J. Xu, J. Li, and J. Lin, "Dynamic resource allocation for cloud computing using reinforcement learning," *Journal of Grid Computing*, vol. 11, no. 3, pp. 429–445, 2013.

[2] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "Balancing energy-efficiency and service quality for cloud applications," *Proceedings of ACM SIGMETRICS*, pp. 3–14, 2011.

[3] R. Ghosh and V. Naik, "Dynamic vm placement and migration using machine learning in cloud," *International Conference on Cloud Computing*, pp. 1–8, 2015.

[4] R. Prasad, A. Roy, and S. Kumari, "Enhancing cloud task scheduling using a hybrid particle swarm and grey wolf optimization approach," *arXiv preprint arXiv:2505.15171*, 2025.

[5] W. Shi and Y. Hong, "Learning-based resource provisioning for cloud applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 12, pp. 2035–2042, 2011.

[6] J. Bi, S. Li, H. Yuan, and M. Zhou, "Integrated deep learning method for workload and resource prediction in cloud systems," *Neurocomputing*, vol. 424, pp. 35–48, 2021.

[7] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood, "Resource overbooking and application profiling in shared hosting platforms," in *OSDI*, 2005, pp. 239–254.

[8] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, 2016.

[9] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[10] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. Maechling, R. Mayani, W. Chen, R. F. Da Silva, M. Livny, K. Wenger *et al.*, "Pegasus, a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, 2015.

[11] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[12] J. J. Durillo and R. Prodan, "Multi-objective workflow scheduling in cloud computing: A survey and optimization framework," *Future Generation Computer Systems*, vol. 36, pp. 221–237, 2014.

[13] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost- and deadline-constrained provisioning for scientific workflows in IaaS clouds," *Future Generation Computer Systems*, vol. 48, pp. 1–13, 2015.

[14] J. Lu, J. Yang, S. Li, Y. Li, W. Jiang, J. Dai, and J. Hu, "A2c-drl: Dynamic scheduling for stochastic edge-cloud environments using a2c and deep reinforcement learning," *IEEE Internet of Things Journal*, 2024.

[15] G. Zhou, W. Tian, R. Buyya, R. Xue, and L. Song, "Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions," *Artificial Intelligence Review*, vol. 57, no. 5, p. 124, 2024.

[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *Proceedings of ICML*, 2016.

[17] Z. Chen, J. Hu, G. Min, C. Luo, and T. El-Ghazawi, "Adaptive and efficient resource allocation in cloud datacenters using actor-critic deep reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1911–1923, 2021.

[18] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19, 2019, pp. 270–288.

[19] A. Verma, L. Pedrosa, M. P. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at google with borg," in *Proceedings of the Tenth European Conference on Computer Systems*, ser. EuroSys '15, 2015, pp. 1–17.