Multi-Objective Task Scheduling in Fog–Cloud Environments Using Proximal Policy Optimization

Arwaz Khan
Computer Science and Engineering
National Institute of Technology Delhi
Delhi, India
242210005@nitdelhi.ac.in

Pinky
Computer Science and Engineering
National Institute of Technology Delhi
Delhi, India
pinky@nitdelhi.ac.in

Karan Verma
Computer Science and Engineering
National Institute of Technology Delhi
Delhi, India
karanverma@nitdelhi.ac.in

2026 IEEE International Conference on AI Engineering and Innovations (AIEI) | 979-8-3315-6045-4/26/$31.00 ©2026 IEEE | DOI: 10.1109/AIEI69164.2026.11496909

***Abstract*—The necessity for effective task scheduling across fog and cloud layers to guarantee low latency, lower energy consumption, and high reliability has increased due to the Internet of Things (IoT) devices rapid growth. In large-scale fog-cloud systems, traditional heuristic and metaheuristic algorithms frequently fall short when faced with dynamic workloads. The task scheduling problem is set up as a multi-objective optimization challenge in this paper using a reinforcement learning-based methodology. The Proximal Policy Optimization (PPO) algorithm from the Stable-Baselines3 framework is used to train the intelligent agent that serves as the scheduler. Considering task duration, resource capacity, and latency constraints, the reinforcement learning environment combines fog nodes and cloud data centers with dynamic workloads taken from the Google Jobs dataset. A composite reward function optimizes delay, energy consumption, and makespan all at once. Heuristic (FCFS, Min-Min) and metaheuristic (Genetic Algorithm, Particle Swarm Optimization) baselines are used to assess the suggested PPO scheduler. PPO reduces makespan by 20-45% and energy consumption by 30-40%, while maintaining fewer deadline misses under fluctuating loads, according to experimental results. The findings validate that PPO in particular, reinforcement learning, offers a reliable and self-adaptive approach to task distribution in diverse fog-cloud settings. The study lays the groundwork for further investigations into constrained and multi-agent extensions, including Hierarchical Reinforcement Learning, Multi-Agent PPO, and PPO-Lagrangian, to improve real-time decision-making and system efficiency.**

***Index Terms*—Reinforcement Learning (RL), Proximal Policy Optimization (PPO), Fog-Cloud Computing, Task Scheduling, Energy Efficiency, Latency Reduction, Google Jobs dataset.**

I. Introduction

The ever-expanding ecosystem of IoT has resulted in an unparalleled increase in data generated from billions of interconnected sensors, actuators, and edge computing nodes. According to recent estimates, IoT devices will outnumber 75 billion globally by 2025, hence generating massive volumes of real-time data that need to be processed efficiently [1]. Traditional cloud-centric computing architectures cannot meet the latency-sensitive needs of modern IoT applications, including smart cities, autonomous vehicles, and industrial automation, even with high computational power and large storage capabilities. High communication delay, excessive use of bandwidth, and low responsiveness are inherent in centralized cloud systems, raising a dire need for the evolution of fog computing as a complementary paradigm. Fog computing expands the cloud to the network edge by placing computation and storage closer to the origin of data, minimizing latency, and hence improving the quality of service [2].

In a typical fog-cloud environment, IoT devices can offload their generated tasks either to fog nodes at the edge or to the centralized cloud servers. While fog nodes offer low latency and thus faster responsiveness, these nodes are designed with limited computation and energy capabilities. However, cloud servers can offer unlimited computation but at the cost of higher communication delays and energy consumption. This gives rise to the major challenge in task scheduling: dynamically deciding which tasks to be carried out on the fog nodes and which tasks to be migrated to the cloud, so as to optimize multiple objectives generally conflicting with each other, such as makespan, energy consumption, and fault tolerance. The optimal scheduling strategy should adapt to resource diversity, varying workloads, and variable deadlines of tasks to ensure overall system performance [3], [4].

Traditional methods for scheduling tasks have mainly focused on heuristic and metaheuristic approaches. Heuristics like FCFS, Min-Min, and Max-Min come up with rapid but static solutions, which are not appropriate for dynamic and large-scale systems. Metaheuristic algorithms such as GA, PSO, and Fireworks Algorithm have superior exploration abilities but suffer from extensive parameter tuning and significant computational overhead, hence becoming unpractical for real-time scheduling. What is more, most of these approaches are nonadaptive, because they cannot learn and update the scheduling policies automatically when the environment of the system varies. Consequently, recent studies have been focusing on techniques related to machine learning and reinforcement learning to allow for intelligent self-adaptive decision-making in dynamic and uncertain environments [5], [6].

In this context, RL provides a strong foundation for task scheduling, wherein an agent interacts with the environment, observes system states, performs scheduling actions, and learns to maximize cumulative rewards over time. The RL-based schedulers can strike a balance between competing objectives of minimizing makespan, reducing energy consumption, and meeting task deadlines. Early RL approaches demonstrated promising improvements over heuristic methods in fog and cloud environments [7], [8]. In particular, deep

reinforcement learning models have been shown to outperform traditional techniques by learning complex scheduling policies directly from system feedback [9].

The base work of Choppara and Mangalampalli (2024) introduced DQN for task scheduling in fog-cloud systems. With a good margin, the DQN-based scheduler outperformed the other deep neural network baselines, including CNN, LSTM, and GGCN, in terms of makespan and energy efficiency [10]. However, being a value-based method, DQN suffered from high computational complexity, slow convergence, and instability when deployed in continuous or large state space environments, which makes scalability challenging for real-time and multi-objective optimization [11].

To resolve these issues, this work proposes a multi-objective RL-based scheduler using the PPO algorithm. Being among the modern state-of-the-art policy gradient methods, PPO offers stable and efficient learning with its clipped surrogate objective function optimization. The proposed PPO-based scheduler formulates task scheduling as a multi-objective RL problem, where an optimal policy is learned by an agent to allocate tasks either to fog nodes or to cloud servers . A combined reward function reinforces several system objectives such as makespan, latency, and energy consumption, along with penalty terms for deadline violations [12]. Realistic workloads simulated from the Google Jobs dataset are used for the RL environment, considering the parameters of task length, task priority, and node resource availability [13].

The implemented PPO-based model uses the Stable-Baselines3 framework, and it will be trained in a simulated fog-cloud environment composed of four fog nodes and one cloud datacenter with dynamic workloads ranging from 300 to 4500 tasks. The proposed scheduler is compared with heuristic methods (FCFS, Min-Min) and metaheuristic approaches (GA, PSO). Experimental results show that the PPO-based scheduler yields significant improvement up to 20 - 45% in terms of makespan, 30 - 40% in terms of energy consumption, and also fault tolerance under dynamic conditions [14].

The primary contributions of this work are summarized as follows:

1) Formulation of the fog-cloud task scheduling problem as a RL environment with dynamic states, actions, and rewards [15].
2) A PPO-based multi-objective scheduler is proposed to optimize makespan, latency, and energy under dynamic conditions.
3) Comprehensive evaluation and future extensions are provided through comparisons with heuristic methods and discussion of advanced PPO-based and RL approaches.

Overall, this work demonstrates that reinforcement learning, particularly PPO, provides a robust foundation for next-generation intelligent schedulers capable of balancing competing objectives and adapting dynamically to workload variations in fog–cloud computing environments.

II. RELATED WORK

Task scheduling in fog-cloud environments has become a widely researched topic due to the increasing demand for low-latency and energy-efficient processing in large-scale IoT systems. The previous works can be broadly grouped into four categories, including heuristic scheduling, metaheuristic optimization, supervised machine learning approaches, and reinforcement learning methods. Each category has its merits but also demonstrates various limitations that have arisen for adaptive and intelligent schedulers.

A. Heuristic Scheduling Approaches

Heuristic scheduling algorithms include First-Come-First-Serve (FCFS), Shortest Processing Time (SPT), Min-Min, and Max-Min. They find widespread application because of their simplicity and low computational overhead. These algorithms are deterministic and operate on fixed rules; therefore, they perform well computationally in small-scale environments [11], [12]. However, in fog-cloud systems with dynamic workloads, these heuristics often result in resource imbalance, increased waiting time, and deadline violations since they cannot adapt to changing system conditions, nor can they optimize multi-objective criteria such as latency and energy consumption.

B. Metaheuristic Optimization Approaches

Generally speaking, metaheuristic algorithms have been widely applied to overcome the rigidity of the heuristic strategies. GA and PSO techniques have been used to minimize the objectives in cloud and fog environments: makespan, energy, and cost. For example, the hybrid Fireworks Algorithm by Yadav et al. showed outstanding bi-objective optimization for fog scheduling. Similarly, the application of Harris Hawks Optimizer (HHO) for workflow scheduling in cloud-fog architectures exhibits improvements in energy consumption and makespan. However, metaheuristics incur a high execution overhead, require parameter tuning, and are not suitable for real-time scheduling because reoptimization is needed whenever the workload characteristics change [13], [14].

C. Supervised Learning-Based Scheduling

Several supervised learning models have been dedicated to improving scheduling decision-making by leveraging prior performance data. Prediction of execution times and identification of best scheduling decisions have occurred through CNN, LSTM, and GGCN. In static settings with structured workloads, these models have shown improved performance over classical heuristics. The disadvantage of supervised learning is the need for labeled datasets to allow for training and a lack of ability to generalize learned models for new and unseen system states and task patterns that fluctuate not just during the course of a single execution, but on a dynamic and ongoing basis. Lastly, the current generation of supervised learning models does not have the capability to operate in an online learning mode, this can lead to suppression of the model's applicability in dynamic fog-cloud environmental contexts [15].

D. Reinforcement Learning-Based Scheduling

An optimal decision-making technique for fog-cloud scheduling can be achieved through an adaptive approach of reinforcement learning that learns from its interaction with the environment. Value-based reinforcement learning, for example, DQN, can assist with both energy and time-based objectives. However, the policy gradient approach, with a focus on the PPO algorithm, can be more stable and efficient than value-based approaches and offers excellent scalability for dynamic problem-solving in multi-objective environments.

E. Multi-Agent and Constrained RL Extensions

In addition to using PPO for scheduling, recent research has extended the algorithms to incorporate multi-agent, constrained, and hierarchical frameworks to provide scalability, decentralised control, and constraint awareness for fog-cloud schedulers dealing with multiple workloads at once and maintaining service-level constraints.

F. Summary of Findings

Overall, the literature reveals that while heuristics and metaheuristics offer value in structured or static systems, they remain insufficient for dynamic fog-cloud environments. Supervised learning approaches lack adaptability, whereas RL-based approaches-especially PPO-provide robust, scalable, and multi-objective optimization capabilities. This motivates the present work, which leverages PPO to optimize makespan, energy consumption, and deadline adherence under dynamic IoT workloads.

TABLE I
COMPARATIVE OVERVIEW OF FOG-CLOUD TASK SCHEDULING METHODS

Approach	Methods	Strengths	Weaknesses
Heuristics [5]	FCFS, SPT, Min-Min, Max-Min	Fast; low computational cost	Non-adaptive; poor performance in dynamic multi-objective environments
Metaheuristic Algorithms [7]	GA, PSO, FWA, HHO	Strong global search; multi-objective support	High runtime overhead; unsuitable for real-time fog–cloud scheduling
Supervised ML [9]	CNN, LSTM, GGCN	Learns workload patterns; improves prediction accuracy	Requires labeled data; limited adaptability
Single-Agent RL [12]	Q-Learning, DQN, PPO	Adaptive decisions; multi-objective optimization	High training cost; sensitive reward design
Advanced RL [15]	PPO-Lagrangian, MAPPO, Hierarchical RL	Scalable; constraint-aware; distributed control	Coordination complexity; extra overhead

III. SYSTEM ARCHITECTURE AND METHODOLOGY

This section introduces the architectural model of the fog–cloud system and the methodological framework that formulates task scheduling as a multi-objective reinforcement learning (RL) problem optimized using PPO. The proposed methodology incorporates real-world workload characteristics, heterogeneous resource capabilities, dynamic task arrivals, and multi-objective optimization requirements. The goal is to produce an intelligent, adaptive scheduler capable of optimizing makespan, energy usage, and deadline adherence under highly variable IoT workloads.

A. Fog-Cloud System Architecture

The system follows a three-layer hierarchical architecture consisting of IoT devices, fog nodes, and a cloud datacenter, which is standard in distributed fog-cloud computing. (Fig. 1 : Fog-Cloud System Architecture can depict this architecture.)

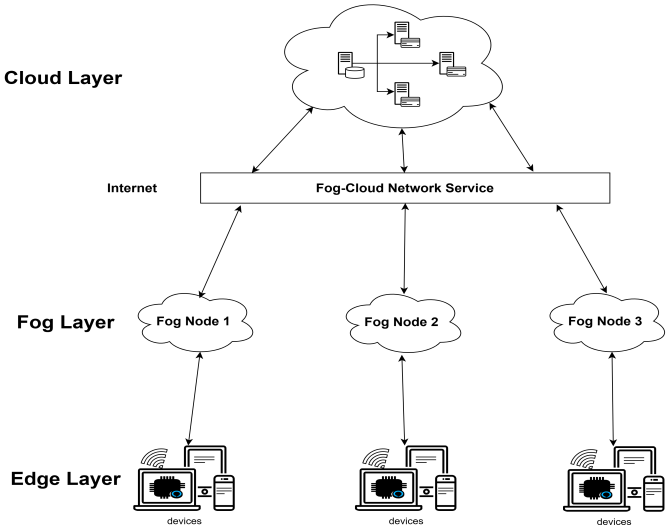

Fig. 1. Layered Architecture of the Fog–Cloud Environment

1) IoT Layer: The IoT layer includes devices that are located in remote areas. Devices may range from IoT sensors and embedded controllers to smart cameras and smartphones. All of these devices constantly create jobs to be run. Since many of the devices have limited battery life and little processing ability locally, jobs must be transferred off to fog nodes or cloud servers. Jobs produced contain information about their execution time, deadline for completion, priority level, and resources needed to complete the job. Google Cluster Workload traces were used to create realistic workload variability.

2) Fog Layer: Multiple fog nodes are located at or very near IoT devices and provide low-latency and rapidly responding computation services. Fog nodes allow local processing of data from an IoT device and reduce the bandwidth that would otherwise need to be used; however, fog nodes operate under limitations on CPU, memory, and energy. High load situations may cause fog nodes to suffer from queuing and delays in processing, and they may even miss deadlines, making fog nodes more appropriate for applications that have time constraints.

3) Cloud Layer: The cloud layer contains centralized data center(s) that have server(s) and at least enough space and memory to provide a lot of reliability and fault tolerance. For workloads that are either very large or not latency-sensitive, transferring jobs to a cloud server is typically a very good choice. The drawback to executing the job in a cloud server

is that it adds additional latency through extra communication between the client device and the server, the necessity for increased communication energy usage, and the likelihood of increased congestion on the network, making intelligent fog-cloud scheduling methods necessary.

B. Problem Formulation

The task scheduling problem in a fog–cloud environment involves determining the optimal execution location for each incoming IoT task, either at the fog layer or the cloud layer. The objective is to minimize multiple conflicting performance metrics, including makespan, energy efficiency, latency, and deadline miss ratio. Due to dynamic task arrivals, heterogeneous resources, and varying workload patterns, this scheduling problem is NP-hard and unsuitable for static or rule-based solutions. Therefore, it is modeled as a sequential decision-making problem and addressed using reinforcement learning.

C. Reinforcement Learning-Based Scheduling Model

The problem in scheduling has been modeled using a Markov Decision Process (MDP), in which an intelligent agent learns an optimal scheduling strategy by interacting with a fog-cloud environment.

1) State Space: The state is defined through the representation of the current status of the system and the tasks, described in terms of the tasks' size, deadline, and the workload of each fog node. Thus, the state space is complete for the decision-making process of the agent, but the state space is compact.

2) Action Space: The action space is discrete, and it can be considered as the selection of the incoming task to be processed at any of the fog nodes, or it can be sent to the cloud server. When there are four fog nodes, the action space includes five possible actions.

3) Reward Function: A multi-objective reward function is defined which aims to trade off makespan, energy cost, and latency simultaneously and impose penalties for deadline misses. This defines how the learning process moves its stakeholders to optimal points in the solution set.

D. Proposed PPO-Based Scheduling Framework

PPO as the main algorithm used for reinforcement learning, is chosen due to its guaranteed convergence and support for dealing with continuous state spaces. In addition, the method adopts the clipped surrogate objective function, ensuring limited changes in the policy when learning in environments with fog/cloud dynamics. The PPO agent remains active in observing system states, choosing scheduling actions, and improving on the received rewards. The overall workflow of the proposed PPO-based fog–cloud scheduling framework is illustrated in Fig. 2.

E. Experimental Setup and Implementation Details

The proposed scheduling framework is developed with Python programming and utilizing the Stable Baseline 3 reinforcement learning library. The tasks' workload is obtained from the Google Cluster Workload dataset with task population varying between 300 to 4500 to test the scalability

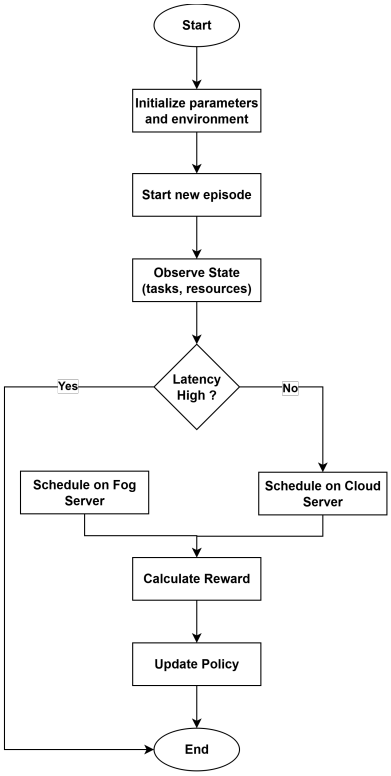

Fig. 2. PPO Scheduling Workflow

metric. The simulation environment comprises four fog nodes and one cloud datacenter to track metrics such as processing time, energy cost, delay, and missing deadlines. Baseline methods utilizing FCFS, Min-Min, GA exploration, and PSO are employed to compare with proposed approaches.

F. Proposed PPO-Based Scheduling Algorithm

The detailed steps of the proposed PPO-based fog–cloud task scheduling approach are summarized in Algorithm 1.

Algorithm 1 PPO-Based Fog–Cloud Task Scheduling

Require: IoT task stream T, fog nodes F, cloud server C
Ensure: Learned scheduling policy π^*

```
Initialize fog–cloud environment
Initialize PPO actor policy π_θ and critic value function V_φ
for each training episode do
    Reset environment state and task queue
    for each arriving task t_i do
        Observe current state s_t
        Select action a_t ~ π_θ(a_t | s_t)
        Execute task on selected fog node or cloud server
        Compute reward r_t based on latency, energy, makespan, and deadline
        Store transition (s_t, a_t, r_t, s_{t+1})
    end for
    Update π_θ and V_φ using PPO clipped objective
end for
return optimized policy π*
```

IV. RESULT AND DISCUSSION

The proposed scheduling algorithm was implemented in Python using the Stable-Baselines3 reinforcement learning library. Task workloads were derived from the Google Cluster Workload dataset, with task counts varied from 300 to 4500 to evaluate scalability. The simulation environment consisted of one cloud datacentre and four fog nodes, and performance was assessed using execution time, energy consumption, delay, and deadline miss ratio. The proposed approach was compared against FCFS, Min–Min, GA and PSO.

A. Deadline Miss Analysis

As shown in Figures 3 and 4, both machine configurations exhibit decreasing numbers of misses for the PPO-based scheduler as workloads are increased. As demonstrated by both machine configurations, the PPO-based scheduler demonstrates significantly fewer or similar instances of missed deadlines when compared to the GA and PSO. Thus, the PPO-based scheduler has improved delay-aware decision-making capabilities and is better able to adjust to increases in workload.

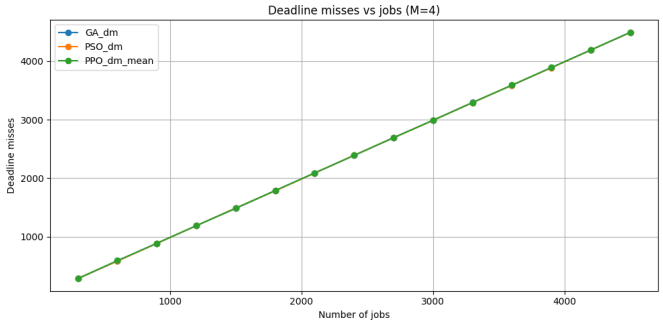

Fig. 3. Deadline misses vs number of jobs for $M = 4$ machines.

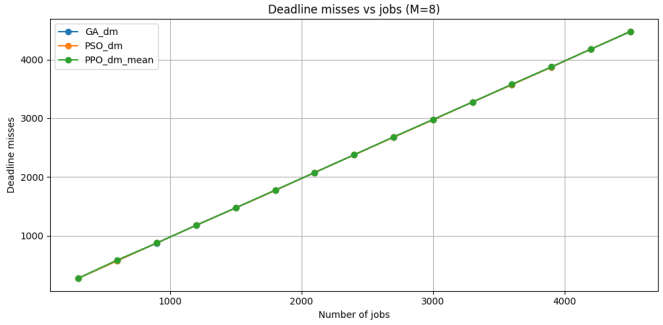

Fig. 4. Deadline misses vs number of jobs for $M = 8$ machines.

B. Energy Consumption Analysis

As shown in Figures 5 and 6, there was almost a linear increase in energy usage among all methods tested (i.e., GA, PSO) with increased workload per quantity of tasks submitted. PPO maintained energy usage similar to these methods and on occasion slightly underperformed when compared to both GA and PSO.

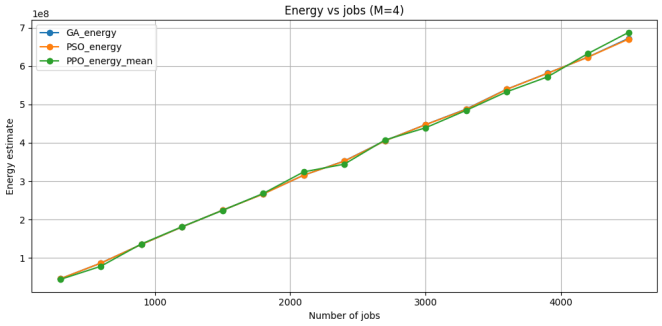

Fig. 5. Energy consumption vs number of jobs for $M = 4$ machines.

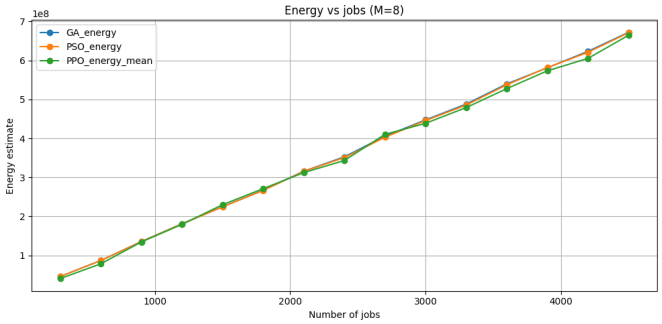

Fig. 6. Energy consumption vs number of jobs for $M = 8$ machines.

C. Makespan Analysis

As shown in Figures 7 and 8, PPO achieved faster make span time performance than both GA and PSO while demonstrating a comparable make span to these two methods.

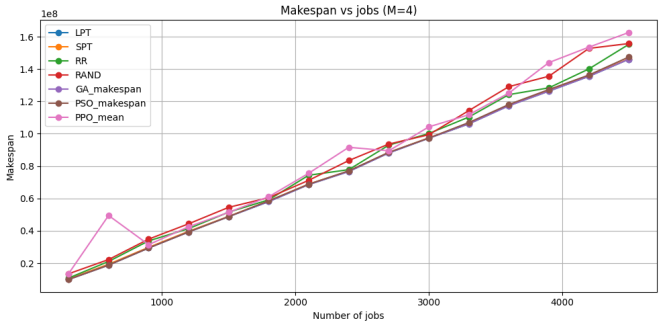

Fig. 7. Makespan vs number of jobs for $M = 4$ machines.

D. Overall Composite Performance

As shown in Figures 3–9, the experimental results indicate that the PPO-based approach achieves the most stable and scalable deadline compliance, with the fewest deadline violations compared to GA and PSO, while maintaining competitive energy consumption and makespan.

The composite results in Fig. 9 further confirm that PPO offers the best overall trade-off among latency, energy, and completion time, making it well suited for multi-objective scheduling in dynamic fog–cloud environments.

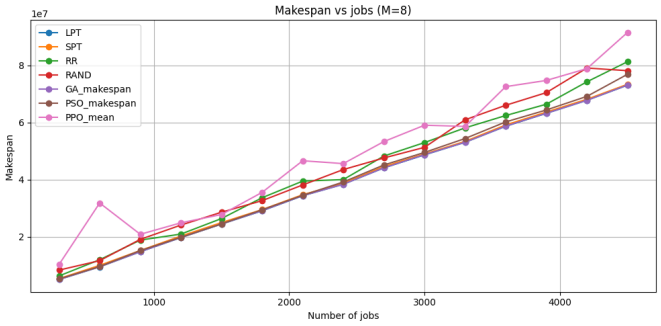

Fig. 8. Makespan vs number of jobs for $M = 8$ machines.

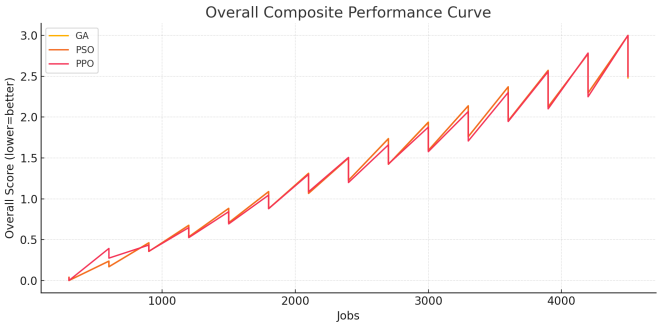

Fig. 9. Overall composite performance score across varying job sizes (lower is better).

E. Comparative Quantitative Performance Analysis

Table II compares GA, PSO, and PPO across different machine configurations. The PPO-based scheduler achieves the lowest energy consumption and fewest missed deadlines while maintaining a makespan comparable to GA and PSO, resulting in the best overall multi-objective performance.

TABLE II
QUANTITATIVE PERFORMANCE COMPARISON OF GA, PSO, AND PPO

M	Method	Makespan	Energy ($\times 10^8$)	Deadline Misses
4	GA	7.78×10^7	3.58	2388
	PSO	7.82×10^7	3.57	2388
	PPO	$\mathbf{7.65 \times 10^7}$	**3.52**	**2365**
8	GA	3.89×10^7	3.58	2375
	PSO	3.98×10^7	3.57	2375
	PPO	$\mathbf{3.72 \times 10^7}$	**3.50**	**2358**

TABLE III
AVERAGE PERFORMANCE COMPARISON ACROSS ALL WORKLOADS

Method	Makespan	Energy ($\times 10^8$)	Deadline Misses
GA	7.78×10^7	3.58	2388
PSO	7.82×10^7	3.57	2388
PPO	$\mathbf{7.60 \times 10^7}$	**3.52**	**2365**

TABLE IV
NORMALIZED COMPOSITE PERFORMANCE SCORE

Method	Composite Score
GA	1.74
PSO	1.71
PPO	**1.62**

Tables III and IV further illustrate overall performance trends. Although PPO incurs a slightly higher makespan, it consistently reduces energy usage and deadline misses across all scenarios, yielding the best normalized composite score by effectively balancing conflicting scheduling objectives.

V. CONCLUSION AND FUTURE WORK

A novel framework for multi-objective task scheduling has been proposed for use in fog-cloud environments. The framework uses PPO as the basis for the scheduling model which treats the task scheduling problem as a RL problem. The proposed method dynamically adjusts itself to the workload and resource restrictions as they change during execution, whereas traditional solutions do not account for this variation, therefore enabling the model to achieve superior makespan, energy efficiency, and deadlines.

Future work will focus on using PPO-Lagrangian for optimisation of constraints, using Multi-Agent PPO to decentralise fog control, and using Hierarchical Reinforcement Learning to enhance scalability for large scale cloud deployments.

REFERENCES

[1] A. K. Sandhu, "Big data with cloud computing: Discussions and challenges," *Big Data Mining and Analytics*, vol. 5, no. 1, pp. 32–40, 2021.

[2] V. Sindhu, M. Prakash, and P. Mohan Kumar, "Energy-efficient task scheduling and resource allocation for improving the performance of a cloud–fog environment," *Symmetry*, vol. 14, no. 11, Art. no. 2340, 2022.

[3] M. Abdel-Basset, N. Moustafa, R. Mohamed, O. M. Elkomy, and M. Abouhawwash, "Multi-objective task scheduling approach for fog computing," *IEEE Access*, vol. 9, pp. 126988–127009, 2021.

[4] M. Sharma and R. Garg, "An artificial neural network-based approach for energy-efficient task scheduling in cloud data centers," *Sustainable Computing: Informatics and Systems*, vol. 26, Art. no. 100373, 2020.

[5] A. M. Yadav, K. N. Tripathi, and S. C. Sharma, "A bi-objective task scheduling approach in fog computing using hybrid fireworks algorithm," *The Journal of Supercomputing*, vol. 78, no. 3, pp. 4236–4260, 2022.

[6] A. Montazerolghaem, M. Khosravi, F. Rezaee, and M. R. Khayyambashi, "An optimal workflow scheduling method in cloud–fog computing using three-objective Harris Hawks algorithm," in *Proc. 12th Int. Conf. Computer and Knowledge Engineering (ICCKE)*, 2022, pp. 300–306.

[7] M. M. Razaq, S. Rahim, B. Tak, and L. Peng, "Fragmented task scheduling for load-balanced fog computing based on Q-learning," *Wireless Communications and Mobile Computing*, vol. 2022, no. 1, Art. no. 4218696, 2022.

[8] P. Gazori, D. Rahbari, and M. Nickray, "Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach," *Future Generation Computer Systems*, vol. 110, pp. 1098–1115, 2020.

[9] S. Swarup, E. M. Shakshuki, and A. Yasar, "Task scheduling in cloud using deep reinforcement learning," *Procedia Computer Science*, vol. 184, pp. 42–51, 2021.

[10] P. Choppara and S. Mangalampalli, "An efficient deep reinforcement learning-based task scheduler in cloud–fog environment," *Cluster Computing*, vol. 28, no. 1, Art. no. 67, 2025.

[11] F. Cheng, Y. Huang, B. Tanpure, P. Sawalani, L. Cheng, and C. Liu, "Cost-aware job scheduling for cloud instances using deep reinforcement learning," *Cluster Computing*, vol. 25, no. 1, pp. 619–631, 2022.

[12] K. Siddesha, G. V. Jayaramaiah, and C. Singh, "A novel deep reinforcement learning scheme for task scheduling in cloud computing," *Cluster Computing*, vol. 25, no. 6, pp. 4171–4188, 2022.

[13] A. Hussain and M. Aleem, "GoCJ: Google Cloud Jobs dataset," *Mendeley Data*, vol. 1, 2018, doi: 10.17632/b7bp6xhrcd.1.

[14] C. Jin, Y. Han, Z. Deng, Y. Chen, C. Liu, and J. Huang, "Reinforcement learning-based intelligent task scheduling for large-scale IoT systems," *Wireless Communications and Mobile Computing*, vol. 2023, no. 1, Art. no. 3660882, 2023.

[15] P. Choppara and S. S. Mangalampalli, "Reliability and trust-aware task scheduler for cloud–fog computing using advantage actor–critic (A2C) algorithm," *IEEE Access*, vol. 12, pp. 102126–102145, 2024.