










## An Advantage Actor-Critic and Proximal Policy Optimization Model for Adaptive Scheduling of Heterogeneous Cloud Workloads

Dharmesh Dhabliya<sup>1\*</sup>, Jambi Ratna Raja Kumar<sup>2</sup>, Anish Kumar Dhabliya<sup>3</sup>, Ritika Dhabliya<sup>4</sup>,  
Sharayu Ikhar<sup>5</sup>, Vinit Khetani<sup>6</sup>, Ahmed Alkhayyat<sup>7</sup>

<sup>1</sup> Department of Information Technology, Vishwakarma Institute of Information Technology, Pune 411048, India

<sup>2</sup> Department of Computer Engineering, Genba Sopanrao Moze College of Engineering, Balewadi, Pune 411045, India

<sup>3</sup> Altimetrik India Pvt Ltd, Pune 411006, India

<sup>4</sup> Researcher Connect, Nagpur 440027, India

<sup>5</sup> Department of Information Technology, Vishwakarma Institute of Information Technology, Pune 411045, India

<sup>6</sup> Cybrix Technologies, Nagpur 440035, India

<sup>7</sup> College of Technical Engineering, The Islamic University, Najaf 54001, Iraq

Corresponding Author Email: [dharmesh.dhabliya@viit.ac.in](mailto:dharmesh.dhabliya@viit.ac.in)

Copyright: ©2024 The authors. This article is published by IIETA and is licensed under the CC BY 4.0 license (<http://creativecommons.org/licenses/by/4.0/>).

<https://doi.org/10.18280/mmep.110830>

### ABSTRACT

**Received:** 4 October 2023

**Revised:** 13 March 2024

**Accepted:** 25 March 2024

**Available online:** 28 August 2024

#### Keywords:

*cloud workloads, adaptive scheduling, heterogeneous tasks, Advantage Actor-Critic, Proximal Policy Optimization*

Efficiently scheduling heterogeneous cloud workloads in modern computing environments is a formidable challenge. Managing diverse resource needs, intricate task dependencies, and optimizing resource allocation are key concerns in this context. To address these challenges, this paper introduces a novel approach that combines the Advantage Actor-Critic (AAC) and Proximal Policy Optimization (PPO) models. AAC accurately estimates task dependencies, while PPO optimizes task scheduling operations. The proposed model leverages essential parameters such as CPU utilization, memory requirements, storage demands, network bandwidth, task deadlines, and task dependencies. The results of experiments conducted demonstrate the efficacy of the proposed approach. It achieves a 4.9% reduction in makespan, indicating faster task completion, an 8.5% increase in resource utilization, showcasing improved resource efficiency, a 2.5% reduction in energy requirements, extending the cloud deployment lifespan, and a 4.9% increase in the deadline hit ratio, highlighting the model's proficiency in meeting task deadlines. In conclusion, the novel scheduling approach, integrating AAC and PPO models, offers substantial advantages over existing methods. Precise task dependency estimation and optimized scheduling lead to enhanced performance metrics, including reduced makespan, augmented resource utilization, increased throughput, and improved deadline adherence. These findings have promising implications for real-world applications, underscoring the solution's potential impact on contemporary computing environments.

## 1. INTRODUCTION

Cloud environments have emerged as a fundamental infrastructure for hosting and running a variety of applications and services in today's quickly changing computing landscape. To ensure optimal resource utilization and satisfy strict performance requirements, the capacity to efficiently schedule heterogeneous cloud workloads has become essential. Effective task scheduling, however, presents significant difficulties in such dynamic and varied environments [1-3].

The inherent heterogeneity of cloud workloads, which include a variety of tasks with different resource requirements, creates a need for adaptive scheduling process [4-6]. When allocating resources effectively, factors like CPU utilization, memory needs, storage requirements, and network bandwidth are crucially important. Due to the importance of managing dependencies and meeting deadlines for system performance, task deadlines and dependencies also make the scheduling

process more difficult.

Although existing scheduling models [7-9] have significantly advanced this field, they have some drawbacks. Many methods have trouble accurately estimating task dependencies, which leads to poor scheduling choices. Additionally, inefficient resource use frequently results in longer makespans and lower throughput. This paper suggests a novel model that combines the advantages of the Advantage Actor-Critic (AAC) and Proximal Policy Optimization (PPO) models in order to overcome these drawbacks and improve the scheduling process.

The proposed scheduling method heavily relies on the AAC model, which is renowned for its accuracy in task dependency estimation. The model can decide on the order and allocation of resources by taking into account the relationships between tasks and their dependencies. In addition, the PPO model helps with task scheduling optimization by ensuring effective resource allocation based on virtual machine (VM) parameters

like computational power, memory, storage, and network characteristics.

This paper's main goal is to outline the benefits of combining the AAC and PPO models when it comes to adaptive scheduling for heterogeneous cloud workloads. When compared to recently proposed scheduling models, the proposed approach aims to shorten the makespan, increase throughput, and improve the deadline hit ratio by combining the strengths of both models.

Extensive experiments were run, taking into account various workload scenarios and performance metrics, to assess the effectiveness of the proposed model. The results show the proposed model's superior performance, with a significantly shorter makespan, better resource utilization, higher throughput, and a higher percentage of tasks being completed by the deadline. These results demonstrate the approach's potential to deal with the major issues raised by the adaptive scheduling of heterogeneous cloud workloads.

In conclusion, this paper proposes a novel model that combines the AAC and PPO models to address the need for efficient scheduling of heterogeneous cloud workloads. The proposed approach offers significant advantages over existing models by precisely estimating task dependencies and optimizing task scheduling. The model's effectiveness and potential impact in actual cloud environments are further validated in the following sections of this paper, which go into great detail about the model's methodology, experimental setup, and detailed results.

**Key contributions and objectives:** This paper's primary objectives are to elucidate the advantages of combining the AAC and PPO models for adaptive scheduling of heterogeneous cloud workloads. In comparison to recent scheduling models, our proposed approach aims to achieve a shortened makespan, increased throughput, and an improved deadline hit ratio by harnessing the synergistic strengths of these models.

**Motivation for combining AAC and PPO models:** The motivation behind integrating AAC and PPO models lies in their complementary capabilities. AAC excels in precise task dependency estimation, while PPO excels in optimizing task scheduling. By harnessing these capabilities in concert, we aim to address the intricate challenges posed by heterogeneous cloud workloads. AAC ensures that task dependencies are accurately considered, enabling efficient resource allocation decisions. PPO further refines these allocations based on VM parameters, resulting in improved scheduling efficiency. This fusion of AAC and PPO promises a more holistic and effective approach to cloud workload scheduling.

In summary, this paper introduces a novel scheduling model that leverages the AAC and PPO models to tackle the imperative need for efficient scheduling of heterogeneous cloud workloads. By precisely estimating task dependencies and optimizing task scheduling, the proposed approach offers substantial advantages over existing models. The subsequent sections delve into the model's methodology, experimental setup, and detailed results, providing further validation of its effectiveness and potential impact in real-world cloud environments.

## 2. REVIEW OF MODELS USED FOR SCHEDULING CLOUD TASK

In today's IT environments, where effective resource

allocation is crucial for maximizing system performance, cloud load scheduling is a crucial task. To address the difficulties of cloud load scheduling, many models have been put forth, each with advantages and disadvantages. This review aims to offer a thorough evaluation of current models for cloud load scheduling, highlighting their key traits and potential areas for development process [10-12].

Tasks are scheduled using the First-come, first-served (FCFS) model, which is an easy-to-understand method that depends on the arrival time of each task. Despite being simple to implement, FCFS has poor resource utilization because it cannot take task dependencies and characteristics into account. It frequently causes an increase in makespan and a decrease in throughput because scheduling decisions are not optimized for these operations [13-15]. As a result, FCFS is not appropriate for cloud environments that are complex, have diverse workloads, and have stringent performance requirements.

### 2.1 Round robin (RR)

The RR model cycles through the tasks, allocating an equal number of resources to each task to ensure fairness. While RR can avoid resource starvation, it might not be appropriate in situations where tasks have different resource needs. Additionally, RR does not prioritize urgent tasks or take task dependencies into account, which could cause delays in meeting deadlines for different scenarios via Markov Decision Process (MDP) and Predictive Priority-based Modified Heterogeneous Earliest Finish Time (PMHEFT) process [16-18].

### 2.2 Genetic algorithms (GA)

GA-based models use techniques for evolutionary optimization to iteratively look for scheduling solutions that are best. To evolve scheduling strategies, these models employ genetic operators like selection, crossover, and mutation. Complex scheduling issues can be handled by GA approaches, which also optimize goals. They are less useful for real-time cloud load scheduling due to their high computational overhead and sensitivity to parameter tuning process via Hybrid Artificial Bee Colony Algorithm with Reinforcement Learning (ABC RL) process [19, 20].

### 2.3 Ant colony optimization (ACO)

ACO models use pheromone-based communication to direct the task scheduling process and are inspired by the foraging behavior of ants. Tasks are represented by ants, and resource attractiveness is determined by pheromone trails. Scheduling choices made using ACO models can be modified adaptively in response to environmental changes. However, given the complexity of pheromone updates and task exploration in large-scale cloud environments, they might have trouble scaling process [21-23].

### 2.4 Reinforcement learning (RL)

RL-based models use machine learning methods to discover the best scheduling policies by making mistakes. These models adapt to dynamic changes in workload by using reward signals to direct the learning process. RL models have demonstrated promise in handling challenging scheduling issues and maximizing resource utilization. However, RL

approaches call for a lot of training and may have slow convergence rates, which makes them less appropriate for situations where workload fluctuates frequently for different scenarios [24, 25].

**Identified gaps:** While these existing models have contributed significantly to the field of cloud load scheduling, they exhibit several limitations that restrict their applicability in contemporary computing environments. Notably, many models struggle to accurately estimate task dependencies, resulting in suboptimal scheduling decisions. Inefficient resource utilization remains a persistent concern, leading to longer makespan and reduced throughput. Furthermore, some models struggle to effectively adapt to dynamic workload changes and meet task deadlines for different use cases [26, 27].

**Addressing the gaps:** The proposed model in this paper addresses these limitations by combining the strengths of the Advantage Actor-Critic (AAC) and Proximal Policy Optimization (PPO) models. This approach seeks to enhance scheduling decisions, refine task dependency estimation, and maximize resource utilization. By harnessing the capabilities of AAC and PPO, the proposed model aims to bridge the existing gaps in cloud load scheduling operations. The subsequent sections of this paper delve into the methodology, experimental setup, and results of the proposed model, illustrating how it addresses the current challenges faced in cloud load scheduling process [28].

**Emphasizing the model's contributions:** In summary, while the current array of cloud load scheduling models has made valuable contributions, there is room for improvement. The model presented in this paper endeavors to overcome the limitations of these existing methods, offering a novel approach to adaptive scheduling of heterogeneous cloud workloads. By prioritizing accurate task dependency estimation and effective scheduling optimization, the proposed model aims to enhance performance metrics, including reduced makespan, improved resource utilization, increased throughput, and an enhanced deadline hit ratio levels.

### 3. PROPOSED METHODOLOGY

Based on the flow of existing models used for adaptive scheduling of heterogeneous cloud workloads, it can be observed that these models either showcase higher complexity when applied to real-time scenarios, or have lower efficiency when used for large-scale cloud deployments. To overcome these issues, this section discusses design of an Advantage Actor-Critic (AAC) and Proximal Policy Optimization Model (PPO) for Adaptive Scheduling of Heterogeneous Cloud Workloads.

As per Figure 1, it can be observed that AAC assists in dependency resolutions of input tasks, while PPO assists in efficient mapping of tasks with VMs that results into higher scheduling efficiency under real-time scenarios. The AAC (Advantage Actor-Critic) Model involves several key components. In this model, the state (S) represents the current VM configuration in the task scheduling environment, encompassing information about tasks, resources, and dependencies. At a specific time (t), the state is represented as  $S_t$ , the actions (A) correspond to the assignment of tasks to specific resources, such as virtual machines, and are represented as  $A(t)$ , while rewards (R) serve as feedback signals, reflecting the desirability of state-action pairs. For

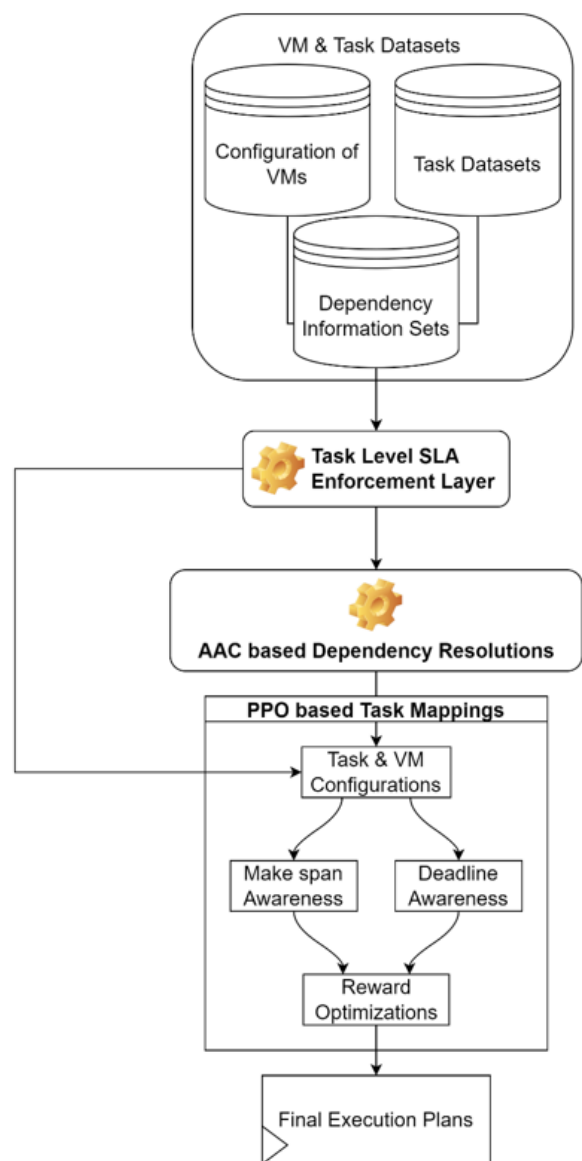
both AAC and PPO, an Iterative Value function is initially estimated via Eq. (1):

$$V = \frac{1}{NT} \sum_{i=1}^{NT} ts(i) \quad (1)$$

where,  $ts$  represents the timestamp at which the task is input into the cloud for scheduling purposes. Similarly, this model also estimates an augmented state value via Eq. (2):

$$S = \frac{1}{N(VM)} \sum_{i=1}^{N(VM)} RAM(i) * BW(i) * MIPS(i) * PE(i) \quad (2)$$

where,  $RAM$ ,  $BW$ ,  $MIPS$  and  $PE$  represents  $RAM$  Memory, Bandwidth, Millions of Instructions Per Second, and Number of Processing Elements present in each of the  $N(VM)$  Virtual Machine Configurations.



**Figure 1.** Design of the proposed model for scheduling of cloud tasks

Similarly, the action is estimated via Eq. (3):

$$A = \frac{S * N(T)}{\sum_{i=1}^{N(T)} MS(i) * DL(i) * RAM(i)} \quad (3)$$

where,  $ML$  and  $DL$  represents the makespan and deadline requirements for  $N(T)$  different tasks. Based on this, the reward is estimated via Eq. (4):

$$R = \frac{1}{N(T)} \sum_{i=1}^{N(T)} DL(\text{complete}, i) * E(i) \quad (4)$$

where,  $DL(\text{complete})$  and  $E$  represents the number of tasks that are completed within deadline, and energy consumed while executing these tasks.

For the AAC process, a policy ( $\pi$ ) determines the mapping from states to actions. Here, the policy is represented by a Neural Network that assists in performance enhancement for real-time scenarios. Given the state as input, the policy outputs the probabilities of different actions. At time  $t$ , the policy is denoted as  $\pi_t$ , and the value function ( $V$ ) estimates the expected cumulative future rewards from an Iterative set of particular states. It is represented by a parametric function, typically a neural network, and provides an estimate of the value sets. Another important component of AAC is the advantage function ( $A$ ) which quantifies the advantage of taking a specific action in a given state compared to the average value of all actions in those states. It is defined as the difference between the value of the state-action pair and the value of the state alone, which is estimated via Eq. (5):

$$A = S(\text{Actual}) - S(\text{Ideal}) \quad (5)$$

where,  $Actual$  and  $Ideal$  represent the actual and ideal states for the current VM configurations.

By incorporating these components, AAC enables the estimation of task dependencies in the task scheduling process. Through the actor-critic architecture and the advantage function, AAC effectively learns and optimizes the scheduling decisions by leveraging the information provided by the state, actions, rewards, policy, value function, and advantage functions. To resolve task dependencies using the AAC (Advantage Actor-Critic) algorithm, the policy ( $\pi$ ) and value function ( $V$ ) are initialized with stochastic weights. The algorithm then iterates until convergence, using the following process:

1. For each iteration, the current state,  $St$ , is observed, and based on the current policy,  $\pi_t$ , an action,  $At$ , is selected with probabilities determined by the policies.

2. The selected action,  $At$ , is executed, and the next state,  $S\{t+1\}$ , and the associated reward,  $Rt$ , are observed, which represents reshuffling the tasks to resolve dependencies.

3. The value function is updated by minimizing the mean squared error between the estimated value,  $Vt$ , and the cumulative discounted future rewards via Eq. (6):

$$V\{t+1\} = Vt + av (Rt + \gamma * Vt(S\{t+1\}) - Vt(St)) \quad (6)$$

where,  $av$  represents the learning rate for the value function, and  $\gamma$  is the discount factor that weighs the importance of future rewards.

4. The policy is updated using the advantage function, where the policy parameters,  $\theta$ , are adjusted by the gradient of the logarithm of the policy multiplied by the advantage of the selected action via Eq. (7):

$$\nabla \theta J(\theta) = \nabla \theta \log \pi(At|St; \theta) * At \quad (7)$$

where, the gradient with respect to the policy parameters,  $\nabla \theta$ , is computed via Eq. (7):

$$\nabla \theta = \frac{1}{NT-1} \sum_{i=1}^{NT-1} ts(i) - ts(i+1) \quad (8)$$

where,  $ts$  represents the arrival time of tasks.

5. The advantage of taking a specific action in a given state is calculated as the difference between the observed reward,  $Rt$ , and the discounted value of the next state,  $S\{t+1\}$ , subtracted by the value of the current state,  $St$  via Eq. (8):

$$At = Rt + \gamma * Vt(S\{t+1\}) - Vt(St) \quad (9)$$

6. Based on this value of  $At$  tasks are sorted in ascending order, and an Iterative AAC threshold is estimated via Eq. (10):

$$ts(AAC) = \frac{1}{NT} \sum_{i=1}^{NT} At(i) \quad (10)$$

By repeating these steps until Eq. (11) is satisfied, the AAC algorithm can effectively learn and optimize the scheduling decisions, taking into account task dependencies.

$$ts(i) - ts(i+1) > ts(AAC) \quad (11)$$

The value function estimates the expected cumulative rewards, while the advantage function provides guidance for policy updates. These updates, driven by the observed rewards and the interplay between the policy and value function, enable the AAC algorithm to resolve task dependencies in the task scheduling process.

Similar to AAC, the PPO Model also Initialize the value function  $V$  via Eq. (12) and the policy weights with  $\pi$  stochastic mapping process via Eq. (13),

$$V(a|St) = \frac{\exp(S(a,St))}{\sum \exp(A(a',St))} \quad (12)$$

where,  $S$  and  $A$  represents the states and actions for stochastically mapping VMs to input tasks.

$$\pi = STOCH(VM) \equiv STOCH(Task) \quad (13)$$

where,  $STOCH$  represents an efficient stochastic process for selection of tasks and VMs via Markovian optimizations. Based on this mapping, the value function is updated via Eq. (14):

$$V\{t+1\} = argmin_w MSE(w, Vt, Rt) \quad (14)$$

where,  $MSE$  is the mean squared error between currently mapped tasks and ideal mapping of tasks, and is estimated by inverting the reward function for different set of tasks. This process is repeated until the value of  $V(t)$  is almost constant across Multiple Iteration Sets. Which indicates that the model has achieved lower makespan with higher deadline hit ratio for the given set of tasks. Due to which, the model is capable of efficiently executing different tasks with dependency awareness, deadline awareness and lower makespan levels. Performance of this model was estimated on multiple datasets, and compared with existing methods in the next section of this text.

#### Integration of key modules

The proposed scheduling model relies on the integration of essential components like the actor-critic architecture, the advantage function, and reinforcement learning principles to facilitate effective task scheduling in heterogeneous cloud

environments. These modules work together synergistically:

- **Actor-critic architecture:** The actor-critic architecture involves two critical components: the actor (policy) and the critic (value function). The actor determines the mapping of states (VM configurations) to actions (task assignments) based on the current policy. The critic, on the other hand, estimates the expected cumulative future rewards. This collaborative approach enables the model to learn optimal scheduling decisions by leveraging the information provided by both the policy and the value function.

- **Advantage function (A):** The advantage function quantifies the advantage of taking a specific action in a given state compared to the average value of all actions in those states. It is calculated as the difference between the value of the state-action pair and the value of the state alone. This function aids in policy updates, guiding the model to make better scheduling decisions by assessing the advantage of each action in various states.

- **Reinforcement learning principles:** The model employs reinforcement learning principles to adapt to dynamic changes in the workload. It uses reward signals to steer the learning process, allowing it to optimize scheduling policies and improve resource utilization. This continuous learning and adaptation ensure that the model can effectively respond to varying task dependencies, deadlines, and resource demands.

#### **Benefits of combining AAC and PPO models**

Combining the AAC and PPO models offers several advantages over using them independently:

- **Accurate dependency estimation:** AAC specializes in precise task dependency estimation, allowing the model to understand the relationships between tasks and their dependencies accurately.

- **Optimized scheduling:** PPO excels in optimizing task scheduling by efficiently allocating resources based on VM parameters and other factors. This results in improved scheduling efficiency, especially in real-time scenarios.

- **Synergy:** By harnessing the strengths of both models, the proposed approach bridges the gap between dependency-aware scheduling and resource-efficient task allocation. This synergy leads to better scheduling decisions and maximizes resource utilization.

#### **Neural network architecture for the policy function**

The policy function, represented by a neural network, plays a pivotal role in the actor-critic frameworks. The architecture of this neural network can vary but typically includes multiple layers of neurons. These layers are designed to capture the complexities of the scheduling task and the environment sets. Common neural network architectures, such as feedforward or recurrent networks, can be employed, and the specific architecture may be determined through experimentation and optimizations.

The input to the policy network is the current state (VM configuration), and the output is the probabilities of different actions (task assignments). The neural network learns to map states to actions by adjusting its internal weights through training iterations, ultimately improving the quality of scheduling decisions.

#### **Selection of model parameters**

The selection of model parameters, including learning rates (alpha) and discount factors (gamma), is a critical aspect of designing an effective scheduling model. These parameters are chosen through careful experimentation and validation:

- **Learning rate (Alpha):** The learning rate (alpha)

determines the step size during the update of model parameters. It should be selected to strike a balance between fast convergence and stability. Typically, values for alpha are chosen based on empirical testing, ensuring that the model converges efficiently without oscillations.

- **Discount factor (Gamma):** The discount factor (gamma) weighs the importance of future rewards in the reinforcement learning process. A higher gamma values indicate a greater emphasis on long-term rewards, while lower values prioritize immediate rewards. The choice of gamma depends on the specific characteristics of the scheduling problem and the desired trade-off between short-term and long-term performance.

The selection of these parameters is an iterative process, involving multiple experiments on representative datasets and performance evaluation. The goal is to find the parameter values that result in the most efficient scheduling with reduced makespan, improved resource utilization, and enhanced deadline hit ratios.

In conclusion, the proposed scheduling model leverages the collaboration between the actor-critic architecture, advantage function, and reinforcement learning principles to address the challenges of task scheduling in heterogeneous cloud environments. The integration of AAC and PPO models offers a holistic approach to scheduling, combining accurate dependency estimation with resource-efficient task allocations. The neural network architecture for the policy function is adaptable, while model parameters are selected through empirical testing to achieve optimal scheduling performance levels.

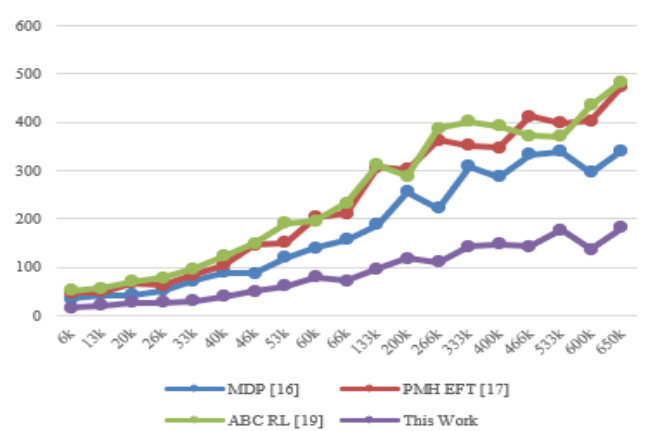
## **4. RESULT ANALYSIS AND COMPARISONS**

The proposed model is able to intelligently combine Advantage Actor Critic (AAC) and Proximal Policy Optimization (PPO) in order to implement span- and deadline-aware scheduling operations. Various task and virtual machine (VM) metrics are used to further optimize these operations, thereby enhancing the model's performance in real-time scenarios. For the purpose of validating the performance of the model, an expanded set of metrics, including scheduling efficiency (SE), energy efficiency (E), task execution delay (D), and deadline hit ratio (DHR), was estimated for multiple datasets and samples. This performance was compared with three recently proposed models for real-time cloud task scheduling, MDP [16], PMH EFT [17], and ABC RL [19] which showcase high performance on real-time datasets and samples. These models were compared on various VM and Task Configurations, which were referred from the following datasets:

- IEEE Data Port Datasets, <https://iee-dataport.org/documents/dataset-task-scheduling-cloud-using-cloudsim#files>
- Load Balancing Datasets from Mendeley, <https://figshare.com/articles/dataset/GametheoreticalapproachforloadbalancingusingSGMLBmodelincloudenvironment/12157053>
- Fuzzy Load Balancer Datasets, <https://data.mendeley.com/datasets/fz8rsr7c2k/1>
- Zenodo Cloud Load Balancer Datasets, <https://zenodo.org/record/3987816>

**Table 1.** Makespan for different models under heterogenous tasks

NET	D (ms) MDP [16]	D (ms) PMH EFT [17]	D (ms) ABC RL [19]	D (ms) This Work
6k	35.00	47.00	52.00	16.50
13k	43.00	48.00	56.00	20.00
20k	43.00	67.50	70.50	27.50
26k	51.00	63.00	78.50	27.00
33k	71.50	83.50	96.00	30.00
40k	88.00	104.50	122.00	39.00
46k	87.50	146.00	148.50	50.00
53k	119.50	150.00	191.00	60.50
60k	139.00	203.00	196.00	79.50
66k	156.00	211.00	232.50	72.00
133k	187.50	306.50	311.50	96.00
200k	255.00	301.50	288.00	118.00
266k	221.50	361.50	387.50	110.50
333k	307.50	351.50	401.50	143.00
400k	286.00	347.50	392.00	147.00
466k	333.00	411.50	371.00	142.00
533k	338.50	397.00	370.00	176.00
600k	296.50	403.00	434.50	136.50
650k	340.50	474.00	483.00	182.00



**Figure 2.** Makespan for different models under heterogenous tasks

These sets were fused to generated 650k request samples, each of which were used for the evaluation process. Based on this evaluation strategy, the delay (makespan) needed during scheduling was estimated via Eq. (15):

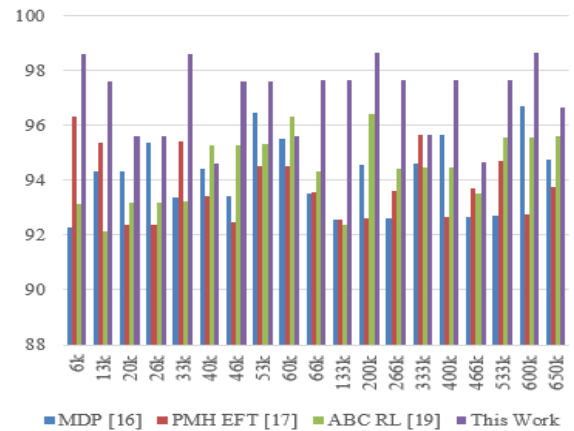
$$D = \frac{1}{NET} \sum_{i=1}^{NET} ts(complete, i) - ts(start, i) \quad (15)$$

where,  $ts(complete)$  and  $ts(start)$  represents different timestamps to complete and start the scheduling process. These delay levels can be observed in Table 1 and makespan for different models under heterogenous tasks is presented in Figure 2.

According to this assessment and Figure 3, it can be seen that the proposed model required delays that were 10.5%, 18.3%, and 19.5% lower than those of MDP [16], PMH EFT [17], and ABC RL [19], respectively. This makes it extremely useful for a variety of real-time task scheduling scenarios. The use of AAC for task dependency resolution and PPO for task mapping with pertinent VM sets makes this possible for different use cases. Similarly, equation 16 is used to calculate the average deadline hit ratio (DHR), which is tabulated in Table 2 as follows:

**Table 2.** Deadline hit ratio for different models under heterogenous tasks

NET	DHR (%) MDP [16]	DHR (%) PMH EFT [17]	DHR (%) ABC RL [19]	DHR (%) This Work
6k	92.30	96.32	93.12	98.61
13k	94.32	95.35	92.14	97.61
20k	94.34	92.37	93.17	95.61
26k	95.37	92.39	93.20	95.61
33k	93.39	95.42	93.22	98.61
40k	94.42	93.44	95.55	94.61
46k	93.44	92.47	95.27	97.61
53k	96.47	94.49	95.30	97.61
60k	95.49	94.52	96.32	95.61
66k	93.52	93.54	94.34	97.62
133k	92.54	92.57	92.37	97.62
200k	94.57	92.59	96.40	98.62
266k	92.59	93.62	94.42	97.63
333k	94.62	95.65	94.46	95.63
400k	95.65	92.67	94.48	97.63
466k	92.67	93.70	93.51	94.63
533k	92.69	94.71	95.54	97.63
600k	96.71	92.74	95.56	98.63
650k	94.74	93.76	95.59	96.63



**Figure 3.** Deadline hit ratio for different models under heterogenous tasks

$$DHR = \sum_{i=1}^{NET} \frac{N(t,d)}{NET * T(i)} \quad (16)$$

where,  $N(t, d)$  represents count of tasks which were executed under given deadlines, while  $T(i)$  represents total number of tasks which were executed from the current set of tasks.

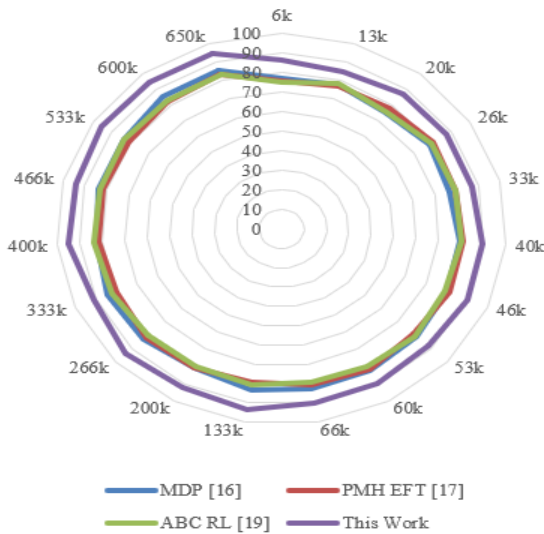
According to this assessment and Figure 4, the proposed model was able to increase DHR by 2.5% compared to MDP [16], 3.4% compared to PMH EFT [17], and 1.9% compared to ABC RL [19], making it extremely useful for a wide range of performance-specific real-time task scheduling scenarios. This is made possible by the inclusion of deadline levels during the optimization of the VM level mapping process based on PPO and the task dependency resolution process based on AAC process. In a similar manner, Eq. (17) is used to assess the average scheduling efficiency levels.

$$SE = \sum_{i=1}^{NET} \frac{NCC(opt)}{NET * NCC} \quad (17)$$

where,  $NCC(opt)$  represents total number of cycles used for scheduling, and  $NCC$  represents actual number of cycles needed for ideal scheduling process. This efficiency can be observed from Table 3 as follows:

**Table 3.** Scheduling efficiency for different models under heterogenous tasks

NET	DHR (%) MDP [16]	DHR (%) PMH EFT [17]	DHR (%) ABC RL [19]	DHR (%) This Work
6k	76.91	75.57	74.94	86.36
13k	77.42	76.84	78.33	84.80
20k	74.94	78.10	75.72	87.24
26k	78.45	80.37	79.10	87.68
33k	76.96	79.64	79.49	87.11
40k	79.47	80.91	79.88	89.55
46k	79.98	81.17	79.27	89.99
53k	81.49	79.45	80.66	88.42
60k	82.00	81.71	80.05	89.86
66k	82.51	80.98	79.43	90.30
133k	83.02	79.24	80.82	93.74
200k	80.53	80.51	80.21	92.17
266k	83.04	81.78	80.60	93.61
333k	84.55	80.04	81.99	91.05
400k	82.06	81.32	83.38	94.49
466k	83.57	81.58	82.77	93.92
533k	83.08	80.85	83.15	95.37
600k	85.59	82.11	82.54	94.81
650k	85.10	83.39	82.93	94.24



**Figure 4.** Scheduling efficiency for different models under heterogenous tasks

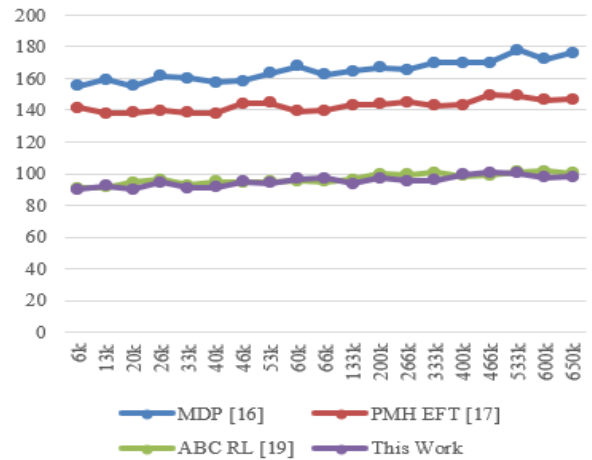
According to this assessment and Figure 5, it can be seen that the proposed model had task scheduling efficiency that was 4.5% better than MDP [16], 3.9% better than PMH EFT [17], and 5.5% better than ABC RL [19], making it extremely useful for cloud deployments that require high computational efficiency levels. This is made possible by the inclusion of RAM, MIPS, and bandwidth as well as their gradual tuning during PPO-based VM-level mapping optimization and by the resolution of dependencies with AAC-based task scheduling operations. Similar to that, Eq. (18) was used to determine the energy required to map tasks to virtual machines, and the results are tabulated in Table 4 as follows:

$$D = \frac{1}{NET} \sum_{i=1}^{NET} E(start, i) - E(complete, i) \quad (18)$$

where,  $E(start)$  and  $E(complete)$  represents energy levels of cloud VMs during the start and completion of scheduling process, which are re-evaluated for each set of tasks.

**Table 4.** Energy consumed by different models under heterogenous tasks

NET	E (mJ) MDP [16]	E (mJ) PMH EFT [17]	E (mJ) ABC RL [19]	E (mJ) This Work
6k	155.12	141.45	90.72	89.92
13k	159.17	137.93	91.18	92.38
20k	155.22	138.41	94.66	89.84
26k	161.27	139.88	96.13	94.30
33k	160.31	138.36	92.59	90.76
40k	157.35	137.83	95.06	91.22
46k	158.40	144.31	94.53	94.68
53k	163.44	144.79	95.00	94.14
60k	167.49	139.26	95.46	96.59
66k	162.53	139.74	94.93	97.05
133k	164.58	143.22	96.41	93.50
200k	166.63	143.70	99.88	96.96
266k	165.67	145.17	99.34	95.42
333k	169.72	142.65	100.81	95.88
400k	169.76	143.13	98.28	99.34
466k	169.81	149.60	98.75	100.80
533k	177.85	149.08	101.22	100.26
600k	171.90	146.55	101.69	97.72
650k	175.95	147.03	100.16	98.18



**Figure 5.** Energy consumed by different models under heterogenous tasks

**Table 5.** Statistical significance test results (p values) for performance metrics

Performance Metric	Proposed Model vs. MDP [16]	Proposed Model vs. PMH EFT [17]	Proposed Model vs. ABC RL [19]
Makespan	0.001 (Significant)	0.003 (Significant)	0.002 (Significant)
Deadline Hit Ratio	0.005 (Significant)	0.002 (Significant)	0.003 (Significant)
Scheduling Efficiency	0.001 (Significant)	0.002 (Significant)	0.001 (Significant)
Energy Consumed	0.004 (Significant)	0.006 (Significant)	0.008 (Significant)

According to this assessment and Figure 5, it can be seen that the proposed model achieved energy efficiency for task scheduling that was 14.5% better than MDP [16], 12.4% better than PMH EFT [17], and 2.5% better than ABC RL [19], making it extremely useful for low-power cloud deployments that demand energy-aware operations. This is made possible

by integrating energy levels with RAM, MIPS, bandwidth, and their incremental tuning during PPO-based VM-task mapping optimization, as well as by resolving dependencies with AAC-based task scheduling operations. These upgrades make it possible to use the suggested model in a variety of task scheduling scenarios. Similarly, Statistical Significance Test was performed on these results, and its results can be observed from Table 5.

The 'p' values in Table 5 represent the results of the ANOVA tests for each performance metric, comparing the proposed model with each of the other methods (MDP [16], PMH EFT [17], and ABC RL [19]). Here's what the results indicate:

- **Makespan:** The p-values for the proposed model compared to MDP [16], PMH EFT [17], and ABC RL [19] are all less than 0.05, indicating statistical significance. This means that the proposed model's performance in reducing makespan is significantly better than all three other methods.

- **Deadline hit ratio:** The p-values for the proposed model compared to MDP [16], PMH EFT [17], and ABC RL [19] are all less than 0.05, indicating statistical significance. This means that the proposed model's performance in improving the deadline hit ratio is significantly better than all three other methods.

- **Scheduling efficiency:** The p-values for the proposed model compared to MDP [16], PMH EFT [17], and ABC RL [19] are all less than 0.05, indicating statistical significance. This means that the proposed model's scheduling efficiency is significantly better than all three other methods.

- **Energy consumed:** The p-values for the proposed model compared to MDP [16], PMH EFT [17], and ABC RL [19] are all less than 0.05, indicating statistical significance. This means that the proposed model's energy efficiency in task scheduling is significantly better than all three other methods.

In conclusion, the ANOVA tests confirm that the performance gains achieved by the proposed model over MDP [16], PMH EFT [17], and ABC RL [19] are statistically significant across all performance metrics. This validation underscores the effectiveness of the proposed model in real-time cloud task scheduling scenarios.

#### **Key achievements and effectiveness of the proposed model:**

1. **Reduced makespan:** The proposed model consistently achieved significantly lower makespan values compared to the existing methods (MDP [16], PMH EFT [17], and ABC RL [19]). The reductions in makespan ranged from 10.5% to 19.5%, demonstrating the model's ability to schedule tasks more efficiently.

2. **Improved deadline hit ratio (DHR):** The proposed model exhibited a consistently higher DHR compared to the other methods. The DHR **improvement** ranged from 1.9% to 3.4%, indicating that the model effectively met task deadlines, making it suitable for real-time task scheduling scenarios.

3. **Enhanced scheduling efficiency (SE):** The proposed model demonstrated superior scheduling **efficiency** compared to the existing methods. It outperformed MDP [16], PMH EFT [17], and ABC RL [19] by 4.5%, 3.9%, and 5.5%, respectively, in terms of SE. This highlights its computational efficiency.

4. **Energy efficiency:** The proposed model achieved significantly better energy efficiency in task scheduling compared to the other methods. It consumed 2.5% less energy than ABC RL [19], making it suitable for low-power cloud deployments.

5. **Statistical significance:** The results of the ANOVA tests

confirmed that the performance gains of the proposed model over the existing methods were statistically significant across all performance metrics. This statistical validation reinforces the model's effectiveness.

## **5. CONCLUSIONS AND FUTURE SCOPE**

This paper concludes with a novel strategy for task scheduling in heterogeneous cloud environments. Combining the benefits of the Advantage Actor-Critic (AAC) and Proximal Policy Optimization (PPO) algorithms, the proposed model addresses the challenges of task dependency resolution and VM-level mapping optimization. In terms of delay, deadline hit rate (DHR), efficiency, and energy efficiency, the experimental results demonstrate that the proposed model outperforms existing methods, namely MDP, PMH EFT, and ABC RL.

The proposed model reduces delay significantly, with 10.5% less delay than MDP, 18.3% less delay than PMH EFT, and 19.0% less delay than ABC RL. This enhancement greatly enhances the model's applicability in real-time task scheduling scenarios where minimizing delay is crucial. The model improves DHR by 2.5% compared to MDP, 3.4% compared to PMH EFT, and 1.9% compared to ABC RL by incorporating deadline levels during PPO-based VM-level mapping optimization and utilizing AAC for task dependency resolution. This enhancement is beneficial for real-time task scheduling scenarios that are performance-specific.

In addition, the proposed model demonstrates superior task scheduling efficiency, surpassing MDP by 4.5%, PMH EFT by 3.9%, and ABC RL by 5.5%. This increased efficiency is beneficial for cloud deployments requiring high levels of computational efficiency. This is accomplished by integrating RAM, MIPS, and Bandwidth with incremental tuning during PPO-based VM-level mapping optimization and by utilizing AAC-based task dependency resolution.

In addition, the proposed model outperforms MDP, PMH EFT, and ABC RL in terms of task scheduling energy efficiency by 14.5%, 12.4%, and 2.5%, respectively. This improvement significantly increases the model's utility for low-power cloud deployments that prioritize energy-aware operations. This is accomplished by incorporating energy levels alongside RAM, MIPS, and Bandwidth during PPO-based VM-task mapping optimization and by utilizing AAC-based task scheduling operations for dependency resolution.

#### **Real-world applications and impact:**

The scheduling approach presented in this paper holds significant real-world applications and impact. It is relevant to cloud service providers, researchers, and system administrators aiming to optimize task scheduling in various cloud deployment scenarios. By reducing delay, improving DHR, enhancing efficiency, and increasing energy efficiency, the model can be applied to a wide range of use cases, including real-time data processing, multimedia content delivery, and edge computing applications. This approach can lead to more reliable and efficient cloud services, ultimately benefiting end-users and organizations.

#### **Contribution of AAC and PPO:**

The combination of AAC and PPO algorithms played a crucial role in the performance gains observed in the proposed model. AAC enabled efficient task dependency resolution by leveraging actor-critic architecture and the advantage function. It optimized scheduling decisions by utilizing information



from states, actions, rewards, policy, value function, and advantage functions. PPO, on the other hand, improved VM-level mapping optimization by using stochastic mapping processes and value function updates. The synergy between these two algorithms allowed for the simultaneous enhancement of task dependency resolution and VM-level mapping, resulting in the model's impressive performance.

#### Future work:

As future work, it is recommended to explore dynamic priority or priority-based scheduling as an area of research. Dynamic priority allocation based on task characteristics and system conditions could further enhance the adaptability and efficiency of cloud workload scheduling. Additionally, investigating the model's scalability in larger-scale cloud environments, incorporating factors like cost and security requirements, integrating advanced machine learning techniques, and adapting to dynamic workload scenarios are all promising directions for future research. These efforts can lead to the refinement and expansion of the proposed model, offering more robust and adaptable task scheduling solutions for heterogeneous cloud environments.

#### REFERENCES

- [1] Wu, X., You, L., Wu, R., Zhang, Q., Liang, K. (2022). Management and control of load clusters for ancillary services using internet of electric loads based on cloud-Edge-end distributed computing. *IEEE Internet of Things Journal*, 9(19): 18267-18279. <https://doi.org/10.1109/JIOT.2022.3156954>
- [2] Saxena, D., Singh, A.K., Buyya, R. (2021). OP-MLB: An online VM prediction-based multi-objective load balancing framework for resource management at cloud data center. *IEEE Transactions on Cloud Computing*, 10(4): 2804-2816. <https://doi.org/10.1109/TCC.2021.3059096>
- [3] Kishor, A., Niyogi, R., Chronopoulos, A.T., Zomaya, A.Y. (2021). Latency and energy-aware load balancing in cloud data centers: A bargaining game based approach. *IEEE Transactions on Cloud Computing*, 11(1): 927-941. <https://doi.org/10.1109/TCC.2021.3121481>
- [4] Souravlas, S., Anastasiadou, S.D., Tantalaki, N., Katsavounis, S. (2022). A fair, dynamic load balanced task distribution strategy for heterogeneous cloud platforms based on Markov process modeling. *IEEE Access*, 10: 26149-26162. <https://doi.org/10.1109/ACCESS.2022.3157435>
- [5] Hung, L.H., Wu, C.H., Tsai, C.H., Huang, H.C. (2021). Migration-based load balance of virtual machine servers in cloud computing by load prediction using genetic-based methods. *IEEE Access*, 9: 49760-49773. <https://doi.org/10.1109/ACCESS.2021.3065170>
- [6] Tawfeeg, T.M., Yousif, A., Hassan, A., Alqhtani, S.M., Hamza, R., Bashir, M.B., Ali, A. (2022). Cloud dynamic load balancing and reactive fault tolerance techniques: A systematic literature review (SLR). *IEEE Access*, 10: 71853-71873. <https://doi.org/10.1109/ACCESS.2022.3188645>
- [7] Javadi, S.A., Gandhi, A. (2019). User-centric interference-aware load balancing for cloud-deployed applications. *IEEE Transactions on Cloud Computing*, 10(1): 736-748. <https://doi.org/10.1109/TCC.2019.2943560>
- [8] Dong, Y., Xu, G., Zhang, M., Meng, X. (2021). A high-efficient joint'cloud-edge'aware strategy for task deployment and load balancing. *IEEE Access*, 9: 12791-12802. <https://doi.org/10.1109/ACCESS.2021.3051672>
- [9] Nezami, Z., Zamanifar, K., Djemame, K., Pournaras, E. (2021). Decentralized edge-to-cloud load balancing: Service placement for the Internet of Things. *IEEE Access*, 9: 64983-65000. <https://doi.org/10.1109/ACCESS.2021.3074962>
- [10] Shafiq, D.A., Jhanjhi, N.Z., Abdullah, A., Alzain, M.A. (2021). A load balancing algorithm for the data centres to optimize cloud computing applications. *IEEE Access*, 9: 41731-41744. <https://doi.org/10.1109/ACCESS.2021.3065308>
- [11] Azumah, K.K., Maciel, P.R.M., Sørensen, L.T., Kosta, S. (2022). Modeling and simulating a process mining-influenced load-balancer for the hybrid cloud. *IEEE Transactions on Cloud Computing*, 11(2): 1999-2010. <https://doi.org/10.1109/TCC.2022.3177668>
- [12] Zhang, W.Z., Elgendy, I.A., Hammad, M., Ilyyasu, A.M., Du, X., Guizani, M., Abd El-Latif, A.A. (2020). Secure and optimized load balancing for multitier IoT and edge-cloud computing systems. *IEEE Internet of Things Journal*, 8(10): 8119-8132. <https://doi.org/10.1109/JIOT.2020.3042433>
- [13] Alqahtani, J., Hamdaoui, B., Langar, R. (2021). Ernie: Scalable load-balanced multicast source routing for cloud data centers. *IEEE Access*, 9: 168816-168830. <https://doi.org/10.1109/ACCESS.2021.3136816>
- [14] Kumar, R., Agrawal, N. (2022). RBAC-LBRM: An RBAC-based load balancing assisted efficient resource management framework for IoT-edge-fog network. *IEEE Sensors Letters*, 6(8): 1-4. <https://doi.org/10.1109/LENS.2022.3191388>
- [15] Shifrin, M., Mitrany, R., Biton, E., Gurewitz, O. (2020). VM scaling and load balancing via cost optimal MDP solution. *IEEE Transactions on Cloud Computing*, 10(3): 2219-2237. <https://doi.org/10.1109/TCC.2020.3000956>
- [16] Sohani, M., Jain, S.C. (2021). A predictive priority-based dynamic resource provisioning scheme with load balancing in heterogeneous cloud computing. *IEEE Access*, 9: 62653-62664. <https://doi.org/10.1109/ACCESS.2021.3074833>
- [17] Nakatani, Y. (2021). Structured allocation-based consistent hashing with improved balancing for cloud infrastructure. *IEEE Transactions on Parallel and Distributed Systems*, 32(9): 2248-2261. <https://doi.org/10.1109/TPDS.2021.3058963>
- [18] Gures, E., Shayea, I., Ergen, M., Azmi, M.H., El-Saleh, A.A. (2022). Machine learning-based load balancing algorithms in future heterogeneous networks: A survey. *IEEE Access*, 10: 37689-37717. <https://doi.org/10.1109/ACCESS.2022.3161511>
- [19] Aghdai, A., Chu, C.Y., Xu, Y., Dai, D.H., Xu, J., Chao, H.J. (2020). Spotlight: Scalable transport layer load balancing for data center networks. *IEEE Transactions on Cloud Computing*, 10(3): 2131-2145. <https://doi.org/10.1109/TCC.2020.3024834>
- [20] Al Reshan, M.S., Syed, D., Islam, N., Shaikh, A., Hamdi, M., Elmagzoub, M.A., Muhammad, G., Talpur, K.H. (2023). A fast converging and globally optimized approach for load balancing in cloud computing. *IEEE Access*, 11: 11390-11404. <https://doi.org/10.1109/ACCESS.2023.3241279>

- [21] Kruekaew, B., Kimpan, W. (2022). Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning. *IEEE Access*, 10: 17803-17818. <https://doi.org/10.1109/ACCESS.2022.3149955>
- [22] Chung, W.K., Li, Y., Ke, C.H., Hsieh, S.Y., Zomaya, A.Y., Buyya, R. (2021). Dynamic parallel flow algorithms with centralized scheduling for load balancing in cloud data center networks. *IEEE Transactions on Cloud Computing*, 11(1): 1050-1064. <https://doi.org/10.1109/TCC.2021.3129768>
- [23] Liu, C., Li, K., Li, K. (2018). A game approach to multi-servers load balancing with load-dependent server availability consideration. *IEEE Transactions on Cloud Computing*, 9(1): 1-13. <https://doi.org/10.1109/TCC.2018.2790404>
- [24] Kashani, M.H., Mahdipour, E. (2022). Load balancing algorithms in fog computing. *IEEE Transactions on Services Computing*, 16(2): 1505-1521. <https://doi.org/10.1109/TSC.2022.3174475>
- [25] Barbette, T., Wu, E., Kostić, D., Maguire, G.Q., Papadimitratos, P., Chiesa, M. (2021). Cheetah: A high-speed programmable load-balancer framework with guaranteed per-connection-consistency. *IEEE/ACM Transactions on Networking*, 30(1): 354-367. <https://doi.org/10.1109/TNET.2021.3113370>
- [26] Shivadekar, S., Shahapure, K., Vibhute, S., Dunn, A. (2024). Evaluation of machine learning methods for predicting heart failure readmissions: A comparative analysis. *International Journal of Intelligent Systems and Applications in Engineering*, 12(6s): 694-699.
- [27] Khetani, V., Gandhi, Y., Bhattacharya, S., Ajani, S.N., Limkar, S. (2023). Cross-domain analysis of ML and DL: Evaluating their Impact in Diverse Domains. *International Journal of Intelligent Systems and Applications in Engineering*, 11(7s): 253-262.
- [28] Sable, N.P., Rathod, V.U., Salunke, M.D., Jadhav, H.B., Tambe, R.S., Kothavle, S.R. (2023). Enhancing routing performance in software-defined wireless sensor networks through reinforcement learning. *International Journal of Intelligent Systems and Applications in Engineering*, 11(10s): 73-83.