# Multi-Objective Task Scheduling in Cloud Computing using Proximal Policy Optimization: Balancing Execution Time, Cost, and Energy

**[1]Iklaas Abdul-Jabbar Sultan***
[1]Software Department College of Computer Science and Mathematics, University of Mosul
Mosul, Iraq
*email: *Iklaas.sultan@uomosul.edu.iq*

(***received*:** 16 September 2026, ***revised*:** 20 October 2026, ***accepted*:** 28 October 2026)

## Abstract

Task scheduling in cloud computing is an NP-hard combinatorial problem in which incoming tasks must be assigned to heterogeneous servers under conflicting objectives. Classical heuristics such as First-Come-First-Served (FCFS), Round-Robin (RR), and priority-based scheduling optimize a single criterion and adapt poorly to dynamic workloads, while metaheuristics such as genetic algorithms (GA) require offline knowledge of the full workload. This paper formulates online multi-objective cloud task scheduling as a Markov decision process and proposes a scheduler based on Proximal Policy Optimization (PPO), a state-of-the-art deep reinforcement learning (DRL) algorithm. In experiments on synthetic workloads, the proposed scheduler reduces mean task response time by 59.1%, 51.5%, and 67.0% relative to FCFS, RR, and Priority, respectively, and by 20.7% relative to an offline GA, while improving the overall multi-objective cost by 11.6–22.0% over rule-based baselines; these differences are statistically significant with large effect sizes (Wilcoxon signed-rank, $p < 2 \times 10{-}4$). A Pareto analysis of the reward weights shows that the learned policies form a controllable trade-off front that no classical baseline Pareto-dominates, although no single policy is best on every individual metric, and a load-intensity study shows the advantage widens under heavy load. The complete implementation is released for reproducibility.

***Keywords:*** cloud computing, deep reinforcement learning, energy efficiency, multi-objective optimization, proximal policy optimization, task scheduling.

## 1- Introduction

Cloud computing delivers on-demand computing resources as a metered utility (Mell and Grance, 2011; Buyya et al., 2009). At the core of every Infrastructure-as-a-Service platform lies a task scheduler that decides, in real time, which physical or virtual server should execute each arriving task. The quality of these decisions directly determines the response time experienced by users, the monetary cost borne by the provider or customer, and the energy footprint of the data center, which is increasingly subject to environmental and economic pressure (Barroso and Hölzle, 2007; Beloglazov et al., 2012).

The underlying assignment problem is NP-hard even in strongly simplified settings (Ullman, 1975), so production systems traditionally rely on fast heuristics such as First-Come-First-Served (FCFS), Round-Robin (RR), or priority-based dispatching (Arunarani et al., 2019). These rules are simple and predictable, but they optimize at most one implicit criterion and ignore the joint effect of an assignment on time, cost, and energy. Metaheuristics such as genetic algorithms (GA) and particle swarm optimization explore the assignment space more thoroughly (Holland, 1992; Kennedy and Eberhart, 1995; Masdari et al., 2017; Houssein et al., 2021), but they are inherently *offline*: they require the complete batch of tasks in advance and re-run an expensive search whenever the workload changes, which is incompatible with the online, non-stationary nature of cloud traffic (Reiss et al., 2012).

Deep reinforcement learning (DRL) offers a fundamentally different approach. By interacting with (a simulation of) the system, an agent can *learn* a scheduling policy that maps the observed system state directly to an assignment decision, amortizing the cost of optimization into an offline training phase and providing constant-time inference online (Mao et al., 2016; Sutton and Barto,

 ISSN:2302-8149
e-ISSN:2540-9719

2018). Breakthroughs such as deep Q-networks (DQN) (Mnih et al., 2015) and AlphaGo (Silver et al., 2016) demonstrated that neural policies can master high-dimensional sequential decision problems, and a growing body of work applies DRL to cloud resource management (Liu et al., 2017; Cheng et al., 2018; Tong et al., 2020; Rjoub et al., 2021). However, much of the existing literature (i) optimizes a single objective, typically makespan or energy alone, (ii) compares against weak baselines without statistical validation, or (iii) does not examine how the reward design lets an operator steer the trade-off between conflicting objectives.

This paper addresses these gaps with a systematic, statistically validated study of multi-objective cloud task scheduling using Proximal Policy Optimization (PPO) (Schulman et al., 2017). To address the research problems, three key questions can be posed: How does the proposed scheduling approach based on the Proximal Policy Optimization (PPO) algorithm compare to traditional scheduling methods and genetic algorithms in terms of response time, financial cost, and energy consumption? What is the impact of varying reward weight settings on the trade-offs between response time, financial cost, and energy consumption? How does the performance of the proposed scheduling method vary according to workload intensity, particularly under heavy loads?

The main contributions are:

1. **Problem formulation.** Online scheduling of stochastically arriving tasks on a heterogeneous server pool is formulated as a Markov decision process (MDP) whose scalarized reward jointly penal-izes response time, monetary cost, and energy consumption through normalized, operator-tunable weights ($\alpha, \beta, \gamma$).
2. **PPO-based scheduler.** An actor–critic agent with a compact multilayer-perceptron architecture is trained with PPO for 500,000 environment steps on randomized workloads of mixed intensity, converging stably without reward shaping or curriculum.
3. **Rigorous evaluation.** The learned policy is compared against FCFS, RR, priority scheduling, and an offline GA on 30 unseen workloads under an identical-task-sequence protocol. Differences are tested with paired Wilcoxon signed-rank tests; every reported improvement is significant at $p < 2 \times 10{-}4$. Confidence intervals and effect sizes are reported alongside the *p*-values to separate statistical from practical significance.
4. **Trade-off and robustness analysis.** A Pareto study across five weight configurations shows that the family of learned policies forms a controllable trade-off front that no baseline Pareto-dominates, while making the per-metric trade-offs explicit, and a load-intensity study shows the advantage of the learned policy grows as the system approaches saturation.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 presents the system model and the MDP formulation. Section 4 details the PPO-based scheduling approach. Section 5 describes the experimental setup, and Section 6 reports and discusses the results. Section 7 concludes and outlines future work.

## 2- Literature Review

Static and dynamic heuristics have long dominated practical cloud scheduling. Braun et al. (2001) compared eleven static heuristics (including Min–Min, Max–Min, and GA) for mapping independent tasks onto heterogeneous machines, establishing a methodological template that later cloud studies adopted. List-scheduling heuristics such as HEFT (Topcuoglu et al., 2002) extend these ideas to dependent (work-flow) tasks. In commercial clouds, simple dispatching rules—FCFS, RR, and priority-based assignment—remain the default because of their (1) decision cost (Arunarani et al., 2019). Their key weakness, confirmed in our experiments, is the absence of any model of the consequences of an assignment: under heterogeneity, sending a large task to a slow server inflates response time, while always preferring fast servers inflates cost and energy.

Metaheuristics improve solution quality by global search. Masdari et al. (2017) survey PSO-based cloud schedulers, and Houssein et al. (2021) provide a comprehensive taxonomy of metaheuristic cloud scheduling, observing that most approaches optimize makespan and cost while energy is comparatively neglected. Energy-aware heuristics were studied by Beloglazov et al. (2012) in the context of VM consolidation. The fundamental limitation shared by this family is the offline assumption: the optimizer must see the whole batch and must be re-executed when conditions change.

 ISSN:2302-8149
e-ISSN:2540-9719

### 2.1 Reinforcement Learning for Resource Management

Mao et al. (2016) introduced DeepRM, showing that a policy-gradient agent can learn multi-resource packing policies that outperform Shortest-Job -First in simulation. Liu et al. (2017) proposed a hierarchical DRL framework combining global VM allocation with local power management. DRL-Cloud (Cheng et al., 2018) applied deep Q-learning to joint resource provisioning and task scheduling, report-ing energy-cost reductions on large task sets. Tong et al. (2020) combined deep Q-learning with heuristic queue management for cloud scheduling, and Rjoub et al. (2021) compared several DRL variants (DQN, RNN-LSTM, DRL-LSTM) for large-scale task scheduling, finding DRL-LSTM most effective at min-imizing waiting time. Multi-agent extensions have also been explored for workflow scheduling (Wang et al., 2019).

Compared with value-based methods (DQN family) used in most of the above, policy-gradient methods directly parameterize the scheduling policy. PPO (Schulman et al., 2017) stabilizes policy-gradient training with a clipped surrogate objective and is widely regarded as a robust default for discrete and continuous control. Despite this, PPO remains under-explored for multi-objective cloud scheduling, and few studies couple it with (i) an explicit, normalized multi-objective reward, (ii) statistical significance testing against both rule-based *and* metaheuristic baselines, and (iii) an analysis of how reward weights shape the attainable trade-off front. These are the gaps addressed here.

**Table 1 Heterogeneous server profiles used in the simulation**

| Profile | Speed (MIPS) | Memory | Price ($/s) | $P^{act}$ (W) | $P^{idle}$ (W) |
|---|---|---|---|---|---|
| Small | 1000 | 4 | 0.0020 | 120 | 60 |
| Medium | 2000 | 8 | 0.0045 | 200 | 90 |
| Large | 4000 | 16 | 0.0100 | 350 | 140 |
| Compute-optimized | 3000 | 8 | 0.0070 | 260 | 110 |

## 3 Research method

### a. Cloud System Model

We consider a cloud site with a pool of $M$ heterogeneous servers $\mathcal{S} = \{s_1, \dots, s_M\}$. Server $s_j$ is characterized by its processing speed $C_j$ (million instructions per second, MIPS), memory capacity $R_j$, monetary price $p_j$ ($ per busy second), and a linear power model with idle power $P_j^{\text{idle}}$ and peak power $P_j^{\text{act}}$ (Barroso and Hölzle, 2007). Four server profiles are instantiated in rotation to form a pool of $M = 8$ servers (Table 1), reproducing the speed–price–power tension found in real instance catalogs: fast servers are expensive and power-hungry, slow servers are cheap and frugal.

Each server executes tasks concurrently in a processor-sharing fashion as long as the sum of the fractional CPU and memory demands of its resident tasks does not exceed unity; otherwise, the incoming task waits in the server queue until enough resident tasks have completed.

### b. Workload Model

Tasks $\mathcal{T} = \{t_1, \dots, t_N\}$ arrive according to a Poisson process with mean inter-arrival time $1/\lambda \in \{0.5, 0.25, 0.1\}$ s, denoted *low*, *medium*, and *high* load, respectively. Task $t_i$ is described by the tuple $\left(d_i^{\text{cpu}}, d_i^{\text{mem}}, \ell_i, \pi_i, a_i\right)$: fractional CPU demand $d_i^{\text{cpu}} \sim \text{clip}(\text{LogNormal}(-1.5, 0.5), 0.05, 1)$, fractional memory demand $d_i^{\text{mem}}$ drawn from the same distribution, length $\ell_i \sim \text{clip}(\text{LogNormal}(6.5, 0.8), 10^2, 2 \times 10^4)$ million instructions (MI), priority $\pi_i \in \{1, 2, 3\}$ drawn uniformly, and arrival time $a_i$. Here $\text{LogNormal}(\mu, \sigma)$ denotes the distribution of $e^Z$ with $Z \sim \mathcal{N}(\mu, \sigma^2)$, and $\text{clip}(X, x_{\min}, x_{\max}) = \min(\max(X, x_{\min}), x_{\max})$ truncates a sample to the given interval. Log-normal size distributions mimic the heavy-tailed behavior observed in production traces, where most tasks are small but a minority is very large (Reiss et al., 2012). All workloads are generated from fixed random seeds for full reproducibility.

**Workload selection.** Synthetic workloads were chosen deliberately for this study because they allow (i) exact control over arrival intensity, so that the effect of load can be isolated in the load-intensity study; (ii) an unlimited supply of independent episodes, which on-policy DRL requires for training; and (iii) paired comparisons in which every scheduler processes exactly the same task

 ISSN:2302-8149
e-ISSN:2540-9719

sequence. The distribution parameters were selected to reproduce qualitative properties reported for production clusters rather than fitted to a specific trace: most tasks are short (median length $e^{6.5} \approx$ 665 MI) with a heavy right tail up to $2 \times 10^4$ MI, and fractional resource demands concentrate around one fifth of a server (median $e^{-1.5} \approx 0.22$). The three inter-arrival times span a lightly loaded to a heavily contended regime for the $M = 8$ server pool, and $N = 200$ tasks per workload is long enough for queues to build up and drain while keeping episodes short enough for 2,500 training episodes. Such workloads do not reproduce the diurnal cycles, correlated bursts, and empirical size and resource-request distributions of real clusters (Reiss et al., 2012); the external validity of the results is therefore limited, and replaying public production traces such as the Google and Alibaba cluster datasets in the same simulator is the first item of future work.

### c. Cost Model

If task $t_i$ is assigned to server $s_j$, its execution time $e_{ij}$, waiting time $w_{ij}$, monetary cost $c_{ij}$, and dynamic energy $\varepsilon_{ij}$ are

$$e_{ij} = \frac{\ell_i}{C_j} \tag{1}$$

$$w_{ij} = \max\left(0, f_j\left(a_i, d_i^{\text{cpu}}, d_i^{\text{mem}}\right) - a_i\right) \tag{2}$$

$$c_{ij} = e_{ij} p_j \tag{3}$$

$$\varepsilon_{ij} = e_{ij}\left(P_j^{\text{act}} - P_j^{\text{idle}}\right) d_i^{\text{cpu}} \tag{4}$$

where $f_j(\cdot)$ returns the earliest time at which the task fits on server $s_j$ given its resident tasks. The response time of the task is $\tau_{ij} = w_{ij} + e_{ij}$. Equation (4) attributes to each task the dynamic part of the server power in proportion to its CPU demand, following the energy-proportionality model of Barroso and Hölzle (2007). Writing $\tau_i$ for the response time of $t_i$ on the server selected by the policy, we additionally measure the makespan and the throughput

$$\text{makespan} = \max_{1 \le i \le N} (a_i + \tau_i), \quad \text{throughput} = \frac{N}{\text{makespan}}.$$

**D MDP Formulation**

Scheduling decisions are made online, one task at a time, which naturally defines a finite-horizon MDP $(\mathcal{X}, \mathcal{A}, P, R)$, where $\mathcal{X}$ is the state space, $\mathcal{A}$ the action space, $P$ the transition kernel induced by task arrivals and completions, and $R$ the reward function.

**State.** When task $t_i$ arrives, the agent observes the state vector

$$x_i = \left[u_1^{\text{cpu}}, u_1^{\text{mem}}, q_1, \ldots, u_M^{\text{cpu}}, u_M^{\text{mem}}, q_M, d_i^{\text{cpu}}, d_i^{\text{mem}}, \tilde{\ell}_i, \tilde{\pi}_i\right]^{\top} \in \mathbb{R}^{3M+4} \tag{5}$$

where $u_j^{\text{cpu}}, u_j^{\text{mem}} \in [0,1]$ are the current CPU and memory utilizations of server $s_j$, $q_j$ is its normalized queue length, and $\tilde{\ell}_i$ and $\tilde{\pi}_i$ are the normalized length and priority of the task. The first $3M$ components describe the servers and the last four describe the arriving task; for $M = 8$ the state has 28 dimensions.

**Action.** The action is the index $j \in \mathcal{A} = \{1, \ldots, M\}$ of the server that will execute the current task.

**Reward.** Assigning task $t_i$ to server $s_j$ yields the negative scalarized multi-objective cost

$$R(x_i, j) = -\left(\alpha \frac{\tau_{ij}}{T_{\text{ref}}} + \beta \frac{c_{ij}}{C_{\text{ref}}} + \gamma \frac{\varepsilon_{ij}}{E_{\text{ref}}}\right) \tag{6}$$

with reference scales $T_{\text{ref}} = 2$ s, $C_{\text{ref}} = \$0.005$, and $E_{\text{ref}} = 200$ J, which are fixed normalization constants that make the three terms dimensionless. The weights $(\alpha, \beta, \gamma) \ge 0$ encode operator preferences; the balanced setting $(\alpha, \beta, \gamma) = (1,1,1)$ is used unless stated otherwise, and Section 6.3 analyzes alternative settings. An episode terminates after all $N = 200$ tasks of a workload have been scheduled, and the learning objective is to find policy parameters $\theta$ that maximize the expected discounted return

 ISSN:2302-8149
e-ISSN:2540-9719

$$\max_{\theta} J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{i=1}^{N} \gamma_d^{i-1} R_i\right]$$

where $R_i$ is the reward of the $i$-th scheduling decision and $\gamma_d = 0.99$ is the discount factor (distinct from the energy weight $\gamma$).

**Choice of reference values.** The reference scales act as exchange rates between the objectives. With balanced weights, a reduction of 1 s in response time is valued the same as a saving of \$0.0025 or of 100 J of dynamic energy, because only the ratios $\alpha/T_{\text{ref}}$, $\beta/C_{\text{ref}}$ and $\gamma/E_{\text{ref}}$ enter Eq. (6); changing the weights is therefore equivalent to changing the reference scales. The values are round-number scales of the same order as the per-task quantities produced by the simulator: response times of a fraction of a second to a few seconds under load, per-task costs of the order of $\$10^{-3}$, and per-task dynamic energies from a few joules to several hundred joules for long tasks. They are not equal to the empirical means, so the three terms do not contribute equally to the return: for the balanced PPO policy on the test workloads (Table 3), the mean per-task contributions are approximately 0.12 (time), 0.46 (cost) and 0.06 (energy). "Balanced" thus denotes equal weights rather than equal contributions. What drives the learned behavior is the marginal exchange rate between objectives; Table 6 decomposes the return of every scheduler into the three terms, and the Pareto analysis shows how the policy responds when the exchange rates change.

### 3.1 Proposed PPO-Based Scheduling Approach

**a.Proximal Policy Optimization**

PPO (Schulman et al., 2017) is an on-policy actor–critic algorithm that maximizes the clipped surrogate objective

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \quad (7)$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|x_t)}{\pi_{\theta_{\text{old}}}(a_t|x_t)} \quad (8)$$

where $x_t$ and $a_t$ are the state and action at decision step $t$, $\hat{\mathbb{E}}_t$ is the empirical average over the transitions of a rollout, $r_t(\theta)$ is the probability ratio between the updated and the data-collecting policy, and $\epsilon$ is the clipping range. The advantage $\hat{A}_t$ is computed with generalized advantage estimation (GAE) (Schulman et al., 2016):

$$\hat{A}_t = \sum_{l=0}^{L-t-1} (\gamma_d \lambda)^l \delta_{t+l}, \quad \delta_t = R_t + \gamma_d V_\varphi(x_{t+1}) - V_\varphi(x_t) \quad (9)$$

where $V_\varphi$ is the critic, $L$ is the rollout length, $\lambda = 0.95$ is the GAE parameter (Table 2), and $V_\varphi(x_{t+1}) = 0$ at episode boundaries. The clipping prevents destructively large policy updates, making PPO markedly more stable than vanilla policy gradients and less hyperparameter-sensitive than A3C (Mnih et al., 2016), which motivated its choice over the DQN-style value-based methods that dominate prior cloud scheduling work (Cheng et al., 2018; Tong et al., 2020). An entropy bonus weighted by $c_e$ encourages exploration and delays premature convergence to deterministic suboptimal rules.

**Algorithm 1** PPO training for the cloud task scheduler

1: Initialize actor $\pi_\theta$ and critic $V_\phi$
2: **for** iteration = 1, 2, . . . until $5 \times 10^5$ steps **do**
3: Sample load level ∈ {low, medium, high}, generate workload
4: **for** $t$ = 1 **to** 2048 **do** ◁ rollout collection
5: Observe state $x_t$ ◁ Eq. (5)
6: Sample server $a_t \sim \pi_\theta(\cdot \mid )$
7: Assign task, obtain reward $R_t$ ◁ Eq. (6)
8: **end for**
9: Compute advantages $\hat{A}_t$ with GAE($\gamma_d$, $\lambda$)

 ISSN:2302-8149
e-ISSN:2540-9719

10: **for** epoch = 1 **to** 10 **do**
11: Update $\theta$, $\phi$ on minibatches of 64 via Eq. (7) plus value loss and entropy bonus
12: **end for**
13: **if** evaluation reward improved **then**
14: checkpoint policy
15: **end if**
16: **end for**

**b.Network Architecture**

Both the policy (actor) and value function (critic) are multilayer perceptron that share the input layer of dimension $3M + 4 = 28$. Each head has two hidden layers with 128 and 64 ReLU units, respectively; the actor terminates in an $M$-way softmax over servers, while the critic outputs a scalar state value. The compact architecture (about $2 \times 10^4$ parameters per head) is deliberate: scheduling decisions must be cheap, and inference requires a single forward pass per task.

**c. Training Procedure**

Algorithm 1 summarizes training. Each episode draws a fresh workload (new seed) with a load level sampled uniformly from {low, medium, high}, so the policy cannot memorize a particular task sequence and must generalize across arrival intensities. Optimization uses Adam (Kingma and Ba, 2015) with the hyperparameters of Table 2. Training runs for 500,000 environment steps (2,500 episodes); a periodic evaluation callback on a held-out validation workload checkpoints the best policy. Training completes in approximately eight minutes on a commodity CPU, underscoring the practicality of the approach.

**Random seeds.** All sources of randomness are seeded. Workload generation uses separate, non-overlapping seed pools for the training episodes, the validation workload used for checkpoint selection, the 30 test workloads of Table 3, and the workloads of the Pareto and load-intensity studies, so that no evaluation workload is ever seen during training. A single global seed initializes the network weights, the action sampling during training, and the random operators of the GA; the exact seed values are part of the configuration files of the released implementation. Because a workload is fully determined by its seed, all schedulers are evaluated on identical task sequences. Each weight configuration was trained once; the reported standard deviations therefore reflect workload-to-workload variability, not variability across training runs (see Discussion and Limitations).

## 4-Results and Analysis

The simulator implements the model of Section 3 as a Gymnasium environment (Towers et al., 2024); the agent uses the PPO implementation of Stable-Baselines3 (Raffin et al., 2021) on PyTorch. The discrete-event core follows the spirit of established cloud simulators such as CloudSim (Calheiros et al., 2011) while remaining lightweight enough for millions of training interactions. All experiments use $M = 8$ servers and workloads of $N = 200$ tasks.

**Table 2 PPO hyperparameters.**

| Parameter | Value |
|---|---|
| Learning rate | $3 \times 10^{-4}$ |
| Discount factor $\gamma_d$ | 0.99 |
| GAE parameter $\lambda$ | 0.95 |
| Rollout length | 2048 |
| Minibatch size | 64 |
| Optimization epochs per rollout | 10 |
| Clipping range $\epsilon$ | 0.2 |
| Entropy coefficient $c_e$ | 0.01 |
| Total environment steps | 500,000 |
| Hidden layers (actor and critic) | [128, 64], ReLU |

 ISSN:2302-8149
e-ISSN:2540-9719

### a. Baseline Schedulers

Four baselines spanning the main families of Section 2 are evaluated on the *identical* task sequences as the DRL agent:

i. **FCFS** assigns each arriving task to the server that becomes free earliest, in arrival order.
ii. **Round-Robin (RR)** cycles deterministically through the server pool.
iii. **Priority** sends high-priority tasks to the fastest least-loaded server and low-priority tasks to the cheapest server.
iv. **Genetic Algorithm (GA)** performs an *offline* search over the complete assignment vector of the workload (population 30, 60 generations during tuning runs and 40 during evaluation, uniform crossover, tournament selection, mutation rate 0.02, elitism 2), with the episode return of Eq. (6) as fitness—i.e., the GA optimizes exactly the same objective as the agent but with full knowledge of the future.

**Baseline configuration.** FCFS, RR and Priority are deterministic and have no tunable parameters; they observe the same server state as the agent but do not use the reward weights. In FCFS, ties between servers that become free at the same time are broken in favor of the lowest server index; in Priority, tasks of the highest class ($\pi_i$ = 3) are treated as high priority and tasks of the two lower classes as low priority. The GA hyperparameters were fixed in preliminary tuning runs on training workloads, never on the test workloads; for evaluation the number of generations was reduced from 60 to 40 to bound its run time on the 30 test workloads (1,200 fitness evaluations per workload), which may slightly favor the proposed method. The GA assignment vector is replayed in the same simulator as the agent, so that all five schedulers are measured by identical code. Because the GA is given the complete task sequence in advance, it serves as a strong offline reference rather than as a deployable online competitor.

### b. Evaluation Protocol and Metrics

The trained policy is evaluated with deterministic (greedy) action selection on 30 test workloads whose seeds are disjoint from all training seeds. Six metrics are reported: mean task response time, total monetary cost, total dynamic energy, makespan, throughput, and the episode return (Eq. (6), balanced weights). Because every scheduler processes the same 30 task sequences, paired two-sided Wilcoxon signed-rank tests (Wilcoxon, 1945) are used to assess the significance of each pairwise difference, avoiding normality assumptions. In addition to *p*-values, we report 95% confidence intervals of each mean (*t*-distribution, $n$ = 30), Hedges' $g$ as a standardized mean difference, and the matched-pairs rank-biserial correlation $r_{rb}$ as the effect size of the Wilcoxon test ($r_{rb}$ = 1 when PPO is better on all 30 workloads). Values of $|g|$ of about 0.2, 0.5 and 0.8 are conventionally read as small, medium and large effects.

### c. Training Convergence

Figure 1 shows the episode return over 500,000 training steps. The 20-episode moving average improves from −145.3 (random-like initial policy) to −126.2, with most of the gain realized in the first 300,000 steps and a stable plateau thereafter; no divergence or catastrophic forgetting is observed. The residual band of

 ISSN:2302-8149
e-ISSN:2540-9719

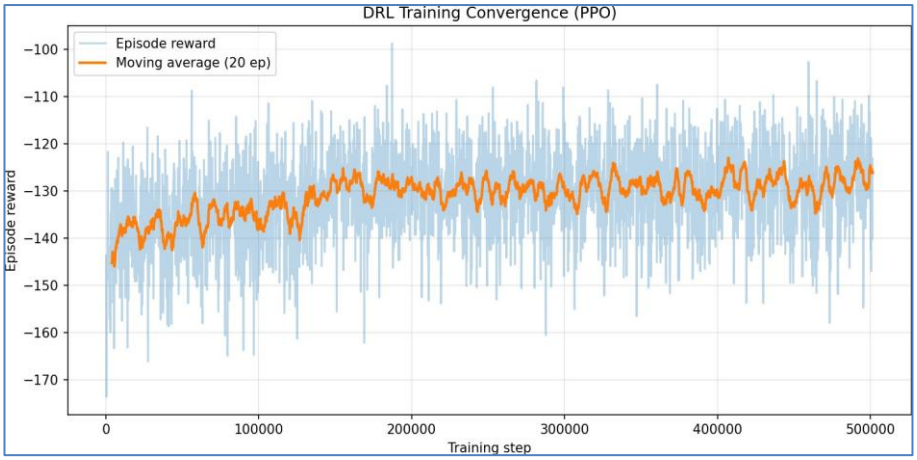



**Figure 1 Training convergence of the PPO scheduler over $5 \times 10^5$ environment steps on mixed-intensity workloads. Light: episode return; dark: 20-episode moving average.**

**Table 3 Performance on 30 unseen workloads (200 tasks, 8 servers, low load); mean ± standard deviation. Best value per column in bold.**

| Scheduler | response time (s) | Cost($) | Energy (KJ) | Makespan(s) throughput | (tasks/s) | Episode return |
|---|---|---|---|---|---|---|
| DRL (PPO) | **0.231 ± 0.017** | 0.457 ± 0.032 | 2.438 ± 0.182 | **99.55 ± 6.31** | **2.017 ± 0.126** | **−126.7 ± 8.7** |
| FCFS | 0.565 ± 0.030 | 0.405 ± 0.030 | 2.605 ± 0.190 | 100.23 ± 6.34 | 2.003 ± 0.126 | −150.6 ± 9.5 |
| Round-Robin | 0.476 ± 0.038 | 0.415 ± 0.029 | 2.523 ± 0.189 | 99.88 ± 6.23 | 2.010 ± 0.124 | −143.2 ± 10.1 |
| Priority | 0.700 ± 0.049 | **0.395 ± 0.030** | 2.675 ± 0.198 | 100.46 ± 6.47 | 1.999 ± 0.128 | −162.4 ± 10.8 |
| GA (offline) | 0.291 ± 0.022 | 0.439 ± 0.031 | **2.405 ± 0.178** | 99.61 ± 6.31 | 2.015 ± 0.126 | −128.8 ± 8.9 |

Per-episode variation reflects genuine workload stochasticity (each episode is a new random workload at a random load level) rather than policy instability.

### d. Comparison with Baselines

Table 3 reports the six metrics (mean ± standard deviation over the 30 unseen workloads) and Table 4 the relative improvement of the proposed scheduler together with the Wilcoxon $p$-values. Figure 2 visualizes response time and episode return.

Three observations stand out. First, on the primary user-facing metric, the learned policy cuts mean response time by half or more relative to every rule-based baseline (0.231 s vs. 0.476–0.700 s) and by 20.7% relative to the GA, and it achieves the best overall multi-objective return. Every difference is significant at $p < 2 \times 10^{-4}$, with most at $p \approx 1.9 \times 10^{-9}$, the smallest value attainable with $n = 30$.

Second, the comparison with the GA deserves emphasis: the GA optimizes the *identical* objective with full offline knowledge of all 200 task arrivals and thousands of fitness evaluations per workload, whereas the agent decides online, task by task, in a single forward pass. That the online policy nevertheless attains a 1.7% better return and 20.7% lower response time indicates that the learned state-conditioned policy captures load-balancing structure that a static assignment vector cannot express, since the GA cannot react to the queue states realized during execution.

Third, the policy is honest about its trade-off: its monetary cost is 4–16% higher than the cost-

**Table 4 Improvement of the PPO scheduler over each baseline (positive = better) and paired Wilcoxon signed-rank $p$-values ($n = 30$).**

| | FCFS | Round-Robin | Priority | GA (offline) |
|---|---|---|---|---|
| Response time | +59.1% | +51.5% | +67.0% | +20.7% |
| Cost | −12.7% | −10.1% | −15.6% | −4.2% |
| Energy | +6.4% | +3.4% | +8.8% | −1.4% |

*ISSN:2302-8149*
*e-ISSN:2540-9719*

| | | | | |
|---|---|---|---|---|
| Makespan | +0.7% | +0.3% | +0.9% | +0.1% |
| Episode return | +15.9% | +11.6% | +22.0% | +1.7% |
| $p$ (response time) | $1.9 \times 10^{-9}$ | $1.9 \times 10^{-9}$ | $1.9 \times 10^{-9}$ | $1.9 \times 10^{-9}$ |
| $p$ (return) | $1.9 \times 10^{-9}$ | $1.9 \times 10^{-9}$ | $1.9 \times 10^{-9}$ | $1.9 \times 10^{-9}$ |
| $p$ (makespan) | $1.9 \times 10^{-9}$ | $3.8 \times 10^{-6}$ | $3.8 \times 10^{-6}$ | $2.0 \times 10^{-4}$ |

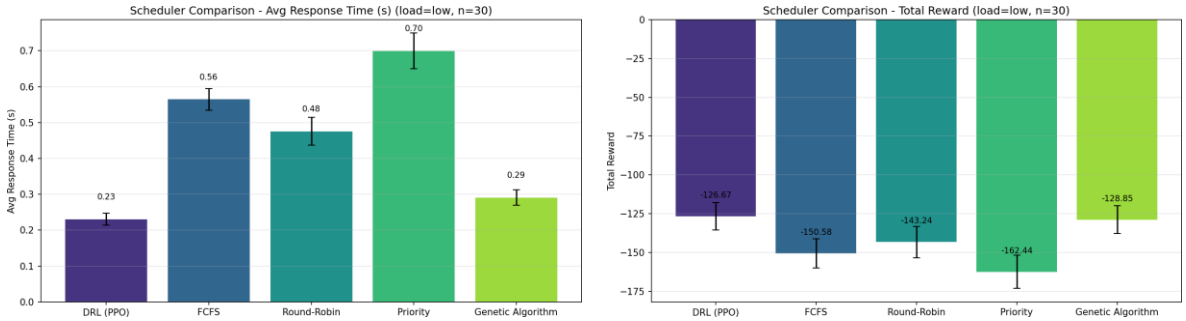



**Figure 2 Mean task response time (left) and episode return (right) on the 30 unseen test workloads; error bars show one standard deviation.**

minimizing baselines. Inspection of the learned behavior shows the agent preferentially routes large or urgent tasks to fast (expensive) servers, deliberately paying more to reduce time and energy, which is exactly what the balanced reward asks for. The next subsection shows this balance is a tunable choice, not a fixed property.

**Per-metric trade-offs.** The best overall return should not be read as uniform superiority. As Table 3 shows, no scheduler is best on every metric: PPO is best on response time, makespan, throughput and episode return, Priority is the cheapest scheduler (PPO costs 15.7% more), and the offline GA consumes the least energy (1.4% less than PPO) and is 4.2% cheaper. Table 5 gives 95% confidence intervals for these means. The response-time intervals of PPO and of every baseline are far apart, whereas the makespan intervals of all five schedulers overlap almost completely: makespan is governed mainly by the arrival process ($N/\lambda \approx 100$ s at low load), so the 0.1–0.9% makespan and throughput gains, although statistically significant in the paired test, are of no practical importance. The return intervals of PPO and the GA also overlap.

**Table 5 95% confidence intervals of the mean (*t*-distribution, $n = 30$) for the results of Table 3.**

| Scheduler | Response time (s) | Cost ($) | Energy (kJ) | Makespan (s) | Episode return |
|---|---|---|---|---|---|
| DRL (PPO) | [0.225, 0.237] | [0.445, 0.469] | [2.370, 2.506] | [97.2, 101.9] | [−129.9, −123.5] |
| FCFS | [0.554, 0.576] | [0.394, 0.416] | [2.534, 2.676] | [97.9, 102.6] | [−154.1, −147.1] |
| Round-Robin | [0.462, 0.490] | [0.404, 0.426] | [2.452, 2.594] | [97.6, 102.2] | [−147.0, −139.4] |
| Priority | [0.682, 0.718] | [0.384, 0.406] | [2.601, 2.749] | [98.0, 102.9] | [−166.4, −158.4] |
| GA (offline) | [0.283, 0.299] | [0.427, 0.451] | [2.339, 2.471] | [97.3, 102.0] | [−132.1, −125.5] |

Table 6 decomposes the mean episode return into its three weighted terms. Under balanced weights the cost term accounts for about 72% of PPO's total penalty, yet PPO's advantage over the rule-based baselines comes almost entirely from the time term. Relative to FCFS, for example, PPO gains 33.4 units in the time term and 0.8 in the energy term but loses 10.4 in the cost term, a net gain of about 23.9 units; relative to the GA the net gain is only about 2 units (6.0 gained in time, 3.6 lost in cost, 0.2 lost in energy). The ranking of the schedulers therefore depends on the weights: an operator who values cost more strongly than the balanced setting implies would prefer Priority or

 ISSN:2302-8149
e-ISSN:2540-9719

the cost-focused PPO policy of the Pareto analysis, and the GA remains a competitive choice when the complete workload is known in advance and energy is the main concern.

**Table 6 Decomposition of the mean episode penalty (−return) into its weighted terms under balanced weights, computed from the means in Table 3 small differences from the reported return are due to rounding.**

| Scheduler | Time term $\Sigma\tau/T$ref | Cost term $\Sigma c/C$ref | Energy term $\Sigma\varepsilon/E$ref | Total penalty |
|---|---|---|---|---|
| DRL (PPO) | 23.1 | 91.4 | 12.2 | 126.7 |
| FCFS | 56.5 | 81.0 | 13.0 | 150.5 |
| Round-Robin | 47.6 | 83.0 | 12.6 | 143.2 |
| Priority | 70.0 | 79.0 | 13.4 | 162.4 |
| GA (offline) | 29.1 | 87.8 | 12.0 | 128.9 |

Table 7 reports effect sizes. For response time and episode return PPO is better than every baseline on all 30 workloads ($r_{rb}$ = 1.00), and the standardized differences are very large against the rule-based schedulers ($g$ = 8.2–13.5 for response time and 1.7–3.6 for return) and large against the GA for response time ($g$ = 3.0). Against the GA, however, the return effect is small ($g$ = 0.24): the paired test detects a consistent advantage, but its size (1.7%) is modest compared with workload-to-workload variability. The cost effects are large in favor of the rule-based baselines ($g$ = −1.4 to −2.0) and medium in favor of the GA ($g$ = −0.56), confirming that the lower response time is bought with higher cost. For makespan, $r_{rb}$ remains high (0.74–1.00) while $g \leq 0.14$, illustrating that a consistent but very small difference can be highly significant.

**Table 7 Effect sizes of PPO versus each baseline. Hedges' $g$ is computed from the means and standard deviations of Table 3 (positive = favors PPO); $r_{rb}$ is the matched-pairs rank-biserial correlation implied by the exact Wilcoxon $p$-values of Table 4 ($n$ = 30).**

| Effect size | FCFS | Round-Robin | Priority | GA (offline) |
|---|---|---|---|---|
| $g$ response time | +13.52 | +8.21 | +12.62 | +3.01 |
| $g$ cost | −1.65 | −1.36 | −1.97 | −0.56 |
| $g$ energy | +0.89 | +0.45 | +1.23 | −0.18 |
| $g$ makespan | +0.11 | +0.05 | +0.14 | +0.01 |
| $g$ episode return | +2.59 | +1.73 | +3.59 | +0.24 |
| $r$rb response time | 1.00 | 1.00 | 1.00 | 1.00 |
| $r$rb episode return | 1.00 | 1.00 | 1.00 | 1.00 |
| $r$rb makespan | 1.00 | 0.87 | 0.87 | 0.74 |

### e. Pareto Analysis of Reward Weights

To quantify the operator's control over the time–cost–energy balance, four additional policies were trained (200,000 steps each) with weight configurations emphasizing different objectives: time-focused (2, 0.5, 0.5), cost-focused (0.5, 2, 0.5), energy-focused (0.5, 0.5, 2), and time+cost (1.5, 1.5, 0.5). All policies were evaluated on 15 common unseen workloads at medium load. Figure 3 plots the resulting front against the baselines. Because these four policies were trained with a smaller budget (200,000 steps) than the balanced policy, the front reported here may understate what is attainable with full training.

The learned policies form a coherent trade-off front. The cost-focused policy drives total cost down to \$0.395—cheaper than FCFS (\$0.408) and RR (\$0.409) and statistically indistinguishable from the cheapest baseline (Priority, \$0.389)—at the price of higher response time (0.572 s), while the balanced and time-focused policies occupy the fast corner (0.230–0.242 s) at \$0.445–0.450. No baseline Pareto-dominates

 *ISSN:2302-8149*
*e-ISSN:2540-9719*

any of the learned policies, but the front does not dominate every baseline either: FCFS and RR sit essentially on the line segment between the cost-focused and balanced policies, Priority is dominated in the time–energy plane, and the offline GA sits marginally below the front in cost but with 25% higher response time than the balanced policy. Priority remains the cheapest scheduler overall, so the appropriate claim is non-dominance and controllability rather than superiority on every objective. The practical implication is that ($\alpha$, $\beta$, $\gamma$) acts as a reliable steering wheel: an operator can re-train (or maintain a small portfolio of) policies to match commercial priorities without modifying the system.

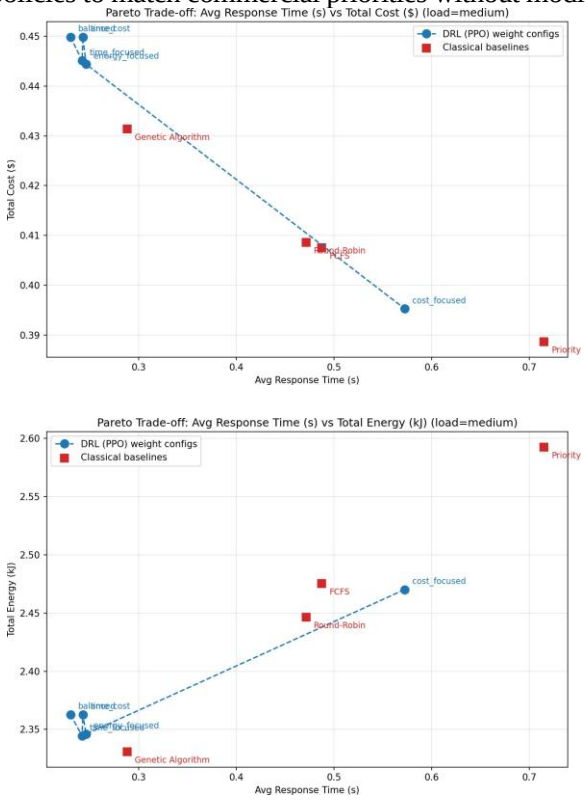



**Figure 3 Trade-off fronts spanned by PPO policies trained with different reward weights (circles, dashed front) versus classical baselines (squares): response time vs. cost (left) and response time vs. energy (right), medium load, 15 unseen workloads.**

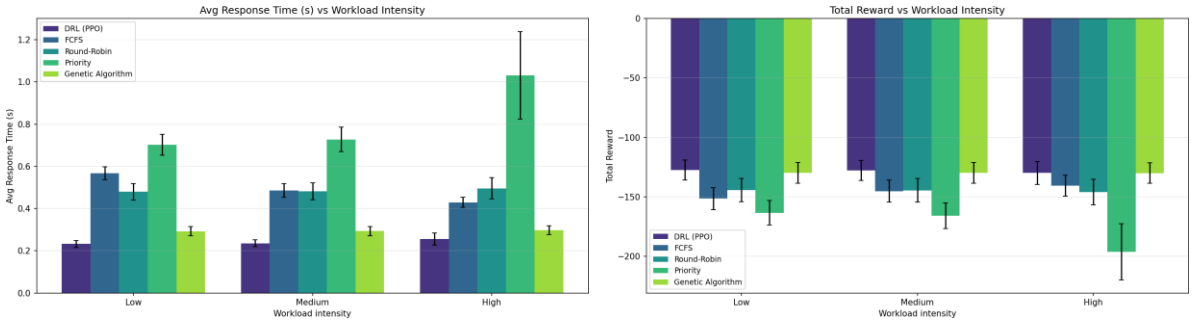



**Figure 4 Mean response time (left) and episode return (right) under low, medium, and high arrival intensity (15 unseen workloads per level; error bars show one standard deviation).**

### f. Robustness to Load Intensity

Figure 4 compares all schedulers at the three arrival intensities of Section 3.2 (15 workloads each). Under low load, queues are short and even naive policies eventually find capacity; under high load (mean inter-arrival 0.1 s), poor placement decisions compound. The learned policy degrades gracefully—response time rises only from 0.232 s to 0.256 s (+10%)—whereas Priority degrades by 47% to 1.031 s (four times the DRL value), and the rule-based baselines remain 1.7–4.0× slower. The episode-return gap widens

 ISSN:2302-8149
e-ISSN:2540-9719

correspondingly (DRL −129.9 vs. Priority −196.2 at high load), confirming that the value of state-aware scheduling grows precisely when the system is under pressure, the regime in which scheduling matters most.

### g. Discussion and Limitations

The results support three claims: (i) a compact PPO policy trained purely in simulation provides large, statistically significant response-time and multi-objective gains over rule-based scheduling; (ii) it outperforms an offline metaheuristic that enjoys strictly more information in response time and, by a small margin, in overall return, although the GA remains slightly cheaper and more energy-efficient; and (iii) the reward weights expose a controllable Pareto front, addressing the common criticism that scalarized DRL hides the multi-objective trade-off. Several limitations qualify these findings. First, workloads are synthetic; although log-normal sizes and Poisson arrivals are standard modeling choices, production traces exhibit diurnal patterns and correlated bursts (Reiss et al., 2012), and validation on the Google or Alibaba cluster traces is an immediate next step. Second, the energy model captures only CPU-proportional dynamic power and ignores DVFS, cooling, and network energy. Third, the simulator abstracts away task preemption, migration, and failures. Fourth, each weight configuration currently requires retraining; preference-conditioned or multi-objective RL could produce a single policy spanning the front. Fifth, each configuration was trained with a single seed, so the reported variability reflects workloads rather than training runs. Sixth, the reward depends on fixed reference scales, so the balanced setting embodies one particular exchange rate between time, cost and energy (Table 6). Because of the synthetic workloads in particular, the reported improvements should be regarded as evidence obtained under controlled conditions rather than as predictions of the gains in a production cloud. None of these limitations is specific to the proposed method, and the evaluation protocol (identical task sequences, disjoint seeds, paired non-parametric tests) is designed to transfer directly to trace-driven studies.

## 5. Conclusion

This paper formulated online multi-objective task scheduling on heterogeneous cloud servers as a Markov decision process with a weighted, normalized reward over response time, monetary cost and energy, and solved it with a compact PPO actor–critic scheduler. Its main contributions are the formulation itself; an evaluation protocol based on identical task sequences, disjoint seeds, and paired non-parametric tests with effect sizes; and an analysis of how the reward weights shape the attainable trade-offs. On synthetic workloads, the learned policy substantially reduced response time relative to rule-based schedulers and to an offline genetic algorithm and obtained the best overall return, with an advantage that grew with load. These gains were not uniform across objectives: the policy incurred higher monetary cost than the cost-oriented baselines, the genetic algorithm remained slightly more energy-efficient, and makespan and throughput were practically unchanged. Varying the reward weights produced a family of policies that no baseline Pareto-dominates, giving operators a direct way to choose a time–cost–energy compromise instead of accepting the one implied by a fixed heuristic.

These conclusions are bounded by the experimental setting. The workloads were generated from parametric distributions rather than taken from production systems, the simulator omits preemption, migration, failures and non-CPU energy, and each policy was trained with a single seed. The results should therefore be read as evidence that state-aware learned scheduling is promising under controlled conditions, not as a forecast of the gains in a real cloud. For practitioners, the main implication is that the reward weights and reference scales are design decisions that determine which objective is favored, and they should be set explicitly from the provider's priorities.

Future work proceeds along four lines: (i) trace-driven evaluation on public production traces such as the Google and Alibaba cluster datasets, with multiple training seeds per configuration; (ii) richer infrastructure models including DVFS-aware energy, preemption, and VM migration;
(iii) preference-conditioned and multi-agent DRL to cover the Pareto front with a single policy and to scale to multi-datacenter settings; and (iv) online continual learning to track non-stationary workloads without full retraining.

 ISSN:2302-8149
e-ISSN:2540-9719

**References**

[1] Arunarani, A., Manjula, D., and Sugumaran, V. (2019). *Task Scheduling Techniques in Cloud Computing: A Literature Survey*. *Future Generation Computer Systems*, 91:407–415.

[2] Barroso, L. A. and Hölzle, U. (2007). *The Case for Energy-Proportional Computing*. *Computer*, 40(12):33–37.

[3] Beloglazov, A., Abawajy, J., and Buyya, R. (2012). *Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing*. *Future Generation Computer Systems*, 28(5):755–768.

[4] Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L. L., Maheswaran, M., Reuther, A. I., Robertson, J. P., Theys, M. D., Yao, B., Hensgen, D., and Freund, R. F. (2001). *A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks Onto Heterogeneous Distributed Computing Systems*. *Journal of Parallel and Distributed Computing*, 61(6):810–837.

[5] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). *Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility*. *Future Generation Computer Systems*, 25(6):599–616.

[6] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., and Buyya, R. (2011). *CloudSim: a Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms*. *Software: Practice and Experience*, 41(1):23–50.

[7] Cheng, M., Li, J., and Nazarian, S. (2018). *DRL-Cloud: Deep Reinforcement Learning-based Resource Provisioning and Task Scheduling for Cloud Service Providers*. In *Proceedings of the 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 129–134.

[8] Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, Cambridge, MA.

[9] Houssein, E. H., Gad, A. G., Wazery, Y. M., and Suganthan, P. N. (2021). *Task Scheduling in Cloud Computing based on Meta-Heuristics: Review, Taxonomy, Open Challenges, and Future Trends*. *Swarm and Evolutionary Computation*, 62:100841.

[10] Kennedy, J. and Eberhart, R. (1995). *Particle Swarm Optimization*. In *Proceedings of the IEEE Interna-tional Conference on Neural Networks*, Volume 4, pages 1942–1948.

[11] Kingma, D. P. and Ba, J. (2015). *Adam: A Method for Stochastic Optimization*. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. arXiv:1412.6980.

[12] Liu, N., Li, Z., Xu, J., Xu, Z., Lin, S., Qiu, Q., Tang, J., and Wang, Y. (2017). *A Hierarchical Framework of Cloud Resource Allocation and Power Management using Deep Reinforcement Learning*. In *Proceedings of the 37th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 372–382.

[13] Mao, H., Alizadeh, M., Menache, I., and Kandula, S. (2016). *Resource Management with Deep Reinforce-Ment Learning*. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets)*, pages 50–56.

[14] Masdari, M., Salehi, F., Jalali, M., and Bidaki, M. (2017). *A Survey of PSO-based Scheduling Algorithms in Cloud Computing*. *Journal of Network and Systems Management*, 25(1):122–158.

[15] Mell, P. and Grance, T. (2011). *The NIST Definition of Cloud Computing*. *NIST Special Publication*, 800-145.

*[16]* Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). *Asynchronous Methods for Deep Reinforcement Learning*. In *Proceedings of the 33rd Interna-tional Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937.

[17] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). *Human-Level Control through Deep Reinforcement Learning*. *Nature*, 518(7540):529–533.

[18] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). *Stable-Baselines3: Reliable Reinforcement Learning Implementations*. *Journal of Machine Learning Research*, 22(268):1–8.

[19] Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., and Kozuch, M. A. (2012).

 *ISSN:2302-8149*
*e-ISSN:2540-9719*

*Heterogeneity and Dynamicity of Clouds at Scale: Google trace analysis.* In *Proceedings of the Third ACM Symposium on Cloud Computing (SoCC)*, pages 1–13.

[20] Rjoub, G., Bentahar, J., Abdel Wahab, O., and Saleh Bataineh, A. (2021). *Deep and Reinforcement Learning for Automated Task Scheduling in Large-Scale Cloud Computing Systems*. *Concurrency and Computation: Practice and Experience*, 33(23):e5919.

[21] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2016). *High-Dimensional Continuous Control using Generalized Advantage Estimation.* In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*. arXiv:1506.02438.

[22] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. arXiv preprint. arXiv:1707.06347.

[23] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). *Mastering the Game of Go with Deep Neural Networks and Tree Search*. *Nature*, 529(7587):484–489.

[24] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition.

[25] Tong, Z., Chen, H., Deng, X., Li, K., and Li, K. (2020). *A Scheduling Scheme in the Cloud Computing Environment using Deep Q-learning*. *Information Sciences*, 512:1170–1191.

[26] Topcuoglu, H., Hariri, S., and Wu, M.-Y. (2002). *Performance-Effective and Low-Complexity Task Schedul-ing for Heterogeneous Computing*. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274.

[27] Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., et al. (2024). *Gymnasium: A Standard Interface for Reinforcement Learning Environments*. arXiv preprint. arXiv:2407.17032.

[28] Ullman, J. D. (1975). *NP-Complete Scheduling Problems*. *Journal of Computer and System Sciences*, 10(3):384–393.

[29] Wang, Y., Liu, H., Zheng, W., Xia, Y., Li, Y., Chen, P., Guo, K., and Xie, H. (2019). *Multi-Objective Workflow Scheduling with Deep-Q-Network-based Multi-Agent Reinforcement Learning*. *IEEE Access*, 7:39974–39982.

[30] Wilcoxon, F. (1945). *Individual Comparisons by Ranking Methods*. *Biometrics Bulletin*, 1(6):80–83.