Archives available at journals.mriindia.com

International Journal on Advanced Computer Engineering and Communication Technology

ISSN: 2278-5140

Volume 15 Issue 01, 2026

SAGE: A Self-Adaptive Guided Explainable Scheduler for Heterogeneous Computing Environments Using Deep Reinforcement Learning and Digital Twin Predictors

[1]Prof. Anmol Budhewar, [2]Atharva Bhole, [3]Vaishnavi Barhate, [4]Harshad Chaudhari, [5]Abhijit Sathe

[1]Prof, Computer Engineering, Sandip Institute Of Technology and Research Center Nashik(SITRC)
[2,3,4,5] Student, Computer Engineering, Sandip Institute Of Technology and Research Center Nashik(SITRC)
Email: [1]Anmol.budhewar@sitrc.org, [2]atharvabholework@gmail.com, [3]vaishnavibarhate2004@gmail.com, [4]engg.harshad49@gmail.com, [5]satheabhijit908@gmail.com

Peer Review Information

Submission: 28 Feb 2026

Revision: 16 March 2026

Acceptance: 28 March 2026

Keywords

Task Scheduling, Heterogeneous Computing, Deep Reinforcement Learning, Proximal Policy Optimisation, Digital Twin, Explainable AI, SHAP, Self-Adaptive Systems, Multi-Objective Optimisation, Gymnasium

Abstract

Task scheduling in heterogeneous computing environments remains a critical challenge, requiring simultaneous optimisation of multiple competing objectives including execution latency, energy consumption, monetary cost, and Service Level Agreement (SLA) compliance. Traditional heuristic schedulers such as Round-Robin and Min-Min lack adaptability to dynamic workload conditions, while black-box reinforcement learning (RL) agents lack the transparency required for trustworthy deployment in production systems. This paper presents SAGE (Self-Adaptive Guided Explainable Scheduler), a novel framework that synergistically combines three components: (1) a Proximal Policy Optimisation (PPO) policy trained via deep reinforcement learning for candidate action proposal, (2) a Digital Twin (DT) predictor ensemble based on Gradient Boosting Regressors for multi-objective evaluation and re-ranking of candidates, and (3) a SHAP-based explainability module that generates human-readable, contrastive justifications for every scheduling decision. SAGE further incorporates a self-adaptation mechanism that continuously retrains the Digital Twin on observed outcomes, enabling the system to handle concept drift caused by workload variations and resource degradation. Evaluation against five baseline schedulers across 30 experimental episodes demonstrates that SAGE achieves the lowest SLA miss rate of 28.00% (compared to 48.17% for Random and 44.50% for Round-Robin), the highest cumulative reward of −31.21 (a 40.4% improvement over the next-best baseline), and competitive latency of 5.96 seconds. These results confirm that integrating RL-based proposal, DT-based look-ahead evaluation, and explainable decision-making produces a scheduler that is simultaneously adaptive, efficient, and interpretable.

Introduction

The proliferation of heterogeneous computing environments—comprising CPUs, GPUs, edge nodes, and cloud instances with varying performance, power, and cost characteristics—has intensified the need for intelligent task scheduling algorithms. Modern workloads spanning scientific computing, machine learning training, Internet of Things (IoT) data processing, and real-time analytics demand schedulers

© 2026 The Authors. Published by MRI INDIA.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

capable of adaptively allocating tasks to resources while balancing multiple, often conflicting, objectives: minimising execution latency, reducing energy consumption, controlling monetary cost, and meeting Service Level Agreement (SLA) deadlines [1, 2].

Traditional scheduling heuristics, while computationally efficient, suffer from fundamental limitations in dynamic environments. Round-Robin distributes tasks uniformly without considering resource capabilities or task requirements. Shortest-Queue policies minimise queuing delays but ignore compute heterogeneity. Min-Min and Max-Min algorithms consider expected completion times but rely on static estimates that degrade under workload variability [3]. These approaches lack the ability to learn from past scheduling outcomes and adapt their strategies over time.

Reinforcement Learning (RL) has emerged as a promising paradigm for dynamic resource management, with agents learning scheduling policies through interaction with the environment [4, 5]. Deep RL algorithms such as Deep Q-Networks (DQN), Actor-Critic methods, and Proximal Policy Optimisation (PPO) have demonstrated superior performance across various scheduling domains [6]. However, RL-based schedulers face two critical challenges that hinder production adoption:

(i) Lack of Explainability: Neural network policies operate as black boxes, making it difficult for system administrators to understand, trust, or override scheduling decisions. In safety-critical or regulated environments, explainability is mandatory [7].

(ii) Sensitivity to Distribution Shift: RL policies trained on specific workload distributions may degrade when the operational environment changes—a phenomenon known as concept drift [8]. Without continuous adaptation mechanisms, deployed policies become stale.

To address these challenges, we propose SAGE (Self-Adaptive Guided Explainable Scheduler), a hybrid framework that augments a PPO-based RL policy with two key innovations:

- A Digital Twin (DT) predictor based on Gradient Boosting Regression that evaluates candidate assignments against multiple objectives before execution, enabling informed re-ranking.
- A SHAP-based explainability module that provides feature-level attributions and contrastive explanations for every decision, bridging the gap between AI-driven optimisation and human interpretability.

SAGE also implements a self-adaptation loop that retrains the Digital Twin on recently observed outcomes, maintaining prediction accuracy as workload characteristics evolve. The principal contributions of this work are:

1. The design and implementation of the SAGE framework, integrating deep RL, ensemble-based prediction, and explainable AI into a cohesive scheduling pipeline.
2. A custom Gymnasium-compatible environment for heterogeneous task scheduling supporting configurable resource pools, multi-objective reward shaping, and standardised RL training.
3. A comprehensive experimental evaluation comparing SAGE against five baseline schedulers, demonstrating significant improvements in SLA compliance and overall objective optimisation.
4. A self-adaptation mechanism enabling continuous model refinement without full retraining, addressing concept drift in deployed systems.

The remainder of this paper is organised as follows. Section II reviews related work. Section III presents the system architecture. Section IV details the methodology. Section V describes experimental results. Section VI concludes the paper.

Related Work

A. Task Scheduling in Heterogeneous Systems

Task scheduling in heterogeneous computing has been extensively studied over several decades. Classical algorithms include Heterogeneous Earliest Finish Time (HEFT) [9], which prioritises tasks based on upward rank and assigns them to the resource minimising earliest finish time. Min-Min and Max-Min heuristics [10] operate on task-resource completion time matrices. While effective for deterministic workloads, these algorithms assume full knowledge of execution times and do not adapt to runtime variations. Xu et al. [11] proposed a workload-aware scheduler combining historical profiling with queuing-theoretic models, though such approaches typically target single objectives and require hand-crafted features.

B. Reinforcement Learning for Resource Management

The application of RL to resource management gained momentum with the seminal work of Mao et al. [4], who demonstrated that policy-gradient RL agents could outperform hand-tuned heuristics for cluster job scheduling. Subsequent work applied Deep Q-Networks [12], Actor-Critic architectures [13], and PPO [6] to various

scheduling domains including cloud resource allocation, edge computing offloading, and datacenter workload management. PPO [14] has become a popular choice due to its balance of sample efficiency, stability, and ease of implementation. Multi-objective scheduling using RL has been explored through reward shaping and scalarisation [15] as well as Pareto-optimal solution sets via evolutionary multi-objective optimisation [16].

C. Digital Twins in Computing Systems

Digital Twins—virtual replicas of physical systems continuously updated with real-world data—have found increasing application in computing resource management [17]. In the context of scheduling, Digital Twins serve as predictive models estimating the outcome of hypothetical task assignments before committing them. This look-ahead capability reduces the cost of exploration in RL and enables more informed decision-making. Minerva et al. [18] provided a comprehensive survey of Digital Twin applications in IoT and edge computing. This work differs by employing an ensemble of Gradient Boosting Regressors as lightweight DT predictors that can be rapidly retrained, enabling real-time self-adaptation.

D. Explainable AI in Scheduling

The need for explainability in AI-driven systems has been recognised across multiple domains [19]. SHAP (SHapley Additive exPlanations) [20] provides a unified framework for interpreting model predictions based on game-theoretic Shapley values. While SHAP has been widely applied to classification and regression tasks, its application to scheduling decisions remains limited. Contrastive explanations—explaining why one option was chosen over alternatives—have been identified as a natural form of human reasoning [21]. SAGE combines SHAP-based feature attribution with contrastive comparison to provide rich, multi-faceted explanations for each scheduling decision.

E. Self-Adaptive Systems

Self-adaptive systems dynamically adjust their behaviour in response to environmental changes. The MAPE-K (Monitor-Analyse-Plan-Execute over shared Knowledge) reference model [22] provides a widely adopted architectural pattern for self-adaptation. SAGE instantiates this pattern: the Digital Twin monitors prediction accuracy, the adaptation loop analyses drift, and retraining is planned and executed when sufficient new data accumulates.

System Architecture

A. Overview

SAGE operates as a pipeline of three tightly integrated components, each addressing a specific aspect of the scheduling challenge. Figure 1 illustrates the high-level architecture: the PPO policy proposes candidate actions; the Digital Twin evaluates and re-ranks them; the Explainer generates human-readable justifications; and the self-adaptation loop retrains the Digital Twin on observed outcomes.

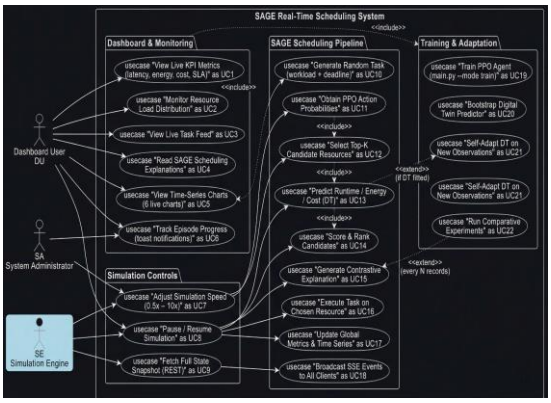

Fig. 1: SAGE system architecture (use-case view) showing the three-component pipeline with Dashboard & Monitoring, SAGE Scheduling Pipeline, Simulation Controls, and Training & Adaptation subsystems.

B. Component 1: PPO Policy Network

The PPO policy serves as the primary action proposer. It takes as input a normalised observation vector encoding the current task's features and all resource states, and outputs a probability distribution over discrete resource assignment actions. Rather than committing to a single action, SAGE extracts the top-K (K = 3 by default) most probable actions as candidate assignments for downstream evaluation. The policy network employs a Multi-Layer Perceptron (MLP) architecture with two hidden layers of 128 units each and is trained using the PPO algorithm from Stable-Baselines3 [23] on a vectorised environment with 4 parallel instances.

C. Component 2: Digital Twin Predictor

The Digital Twin is implemented as an ensemble of three independent Gradient Boosting Regressors (GBR), each predicting one target variable: execution runtime, energy consumption, and monetary cost. For each candidate action proposed by the PPO policy, the Digital Twin takes a feature vector comprising task workload, task deadline, resource speed, resource energy rate, and resource cost rate, and predicts the expected outcomes. Candidates are then scored using a weighted scalarised objective and ranked, providing a "look-ahead" capability analogous to model-based RL.

D. Component 3: Explainability Module

The Explainability module generates two complementary forms of explanation:

1. SHAP Feature Attribution: Using TreeExplainer [20] on the GBR models, SAGE computes Shapley values for each input feature, quantifying how much each feature (workload, deadline, speed, energy rate, cost rate) contributed to the predicted outcome for the chosen resource.
2. Contrastive Explanation: SAGE compares the chosen resource against each alternative candidate and produces human-readable statements such as "Preferred R2 over R1: runtime −2.5s, energy +40J, cost +10$", enabling administrators to understand the trade-offs involved.

E. Self-Adaptation Loop

SAGE implements a self-adaptation mechanism inspired by the MAPE-K model. Every scheduling decision's actual outcome (observed runtime, energy, cost) is recorded. When the accumulated record buffer reaches a configurable threshold (default: 50 records), the Digital Twin is automatically retrained on the new data. This enables the system to adapt to:

Workload drift: Changes in task arrival patterns or computational requirements.

Resource degradation: Gradual performance deterioration of computing resources.

Configuration changes: Addition or removal of resources from the pool.

Methodology

A. Problem Formulation

We formalise the task scheduling problem as a sequential decision process. At each step t, the scheduler observes the current task τt and the state of all resources $\{r_1, ..., r_n\}$, and must assign τt to one resource rj. The goal is to minimise a multi-objective function over the entire episode (batch of tasks).

Task Model: Each task τi is characterised by: $w_i \in [50, 200]$ (computational workload in abstract units) and $d_i \in [10, 40]$ (deadline in time units).

Resource Model: Each resource rj is characterised by sj (processing speed), ej (energy rate), and cj (cost rate).

Execution Model: When task τi is assigned to resource rj, the execution outcomes are computed as:

$$runtime_{ij} = w_i / s_j \quad (1)$$

$$energy_{ij} = w_i \cdot e_j \quad (2)$$

$$cost_{ij} = w_i \cdot c_j \quad (3)$$

$$SLA\_miss_{ij} = 1[finish_{ij} > d_i] \quad (4)$$

where $finish_{ij} = load_j + runtime_{ij}$ accounts for queuing effects.

B. Gymnasium Environment

We implement the scheduling problem as a Gymnasium-compatible environment SageEnv, enabling use of standard RL training libraries.

Observation Space: The observation at each step is a normalised real-valued vector of dimension 3 + 4N, where N is the number of resources. It encodes task features (normalised workload, deadline, slack) concatenated with per-resource features (speed, energy rate, cost rate, current load), all min-max normalised to [0, 1]:

$$ot = [\hat{w}t, \hat{d}t, \hat{o}t, \acute{s}_1, \bar{e}_1, \hat{c}_1, \hat{l}_1, ..., \acute{s}N, \bar{e}N, \hat{c}N, \hat{l}N] \quad (5)$$

Action Space: Discrete, with |A| = N (the number of resources). Action a = j assigns the current task to resource rj.

Reward Function: The per-step reward is the negated, normalised scalarised objective:

$$Rt = -(\alpha \cdot runtime_t/dmax + \beta \cdot energy_t/(wmax \cdot emax) + \gamma \cdot cost_t/(wmax \cdot cmax) + \delta \cdot SLA\_miss_t) \quad (6)$$

where $\alpha = 1.0$, $\beta = 0.01$, $\gamma = 0.01$, and $\delta = 5.0$ are scalarisation weights. The heavy penalty on SLA misses ($\delta = 5.0$) reflects the high cost of deadline violations in production systems.

C. Resource Configuration

The experimental environment includes four heterogeneous resources, summarised in Table I. This configuration creates a meaningful trade-off space: faster resources (R4) complete tasks quickly but consume more energy and incur higher costs, necessitating an intelligent scheduling policy.

Table 1: Resource Pool Configuration

Resource	Speed	Energy Rate	Cost Rate
R1 Slow/Cheap	10	0.5	0.2
R2 Fast/Moderate	20	0.8	0.4
R3 Balanced	15	0.6	0.3
R4 Fastest/Costly	25	0.9	0.5

D. PPO Training Configuration

The PPO agent is trained with the hyperparameters listed in Table II. The policy network consists of separate actor and critic heads, each with two hidden layers of 128 units and ReLU activations. The actor outputs a softmax distribution over N resource actions; the critic estimates state-value V(s).

Table 2: PPO Training Hyperparameters

Parameter	Value
Policy architecture	MLP (128, 128)
Learning rate	3×10^{-4}
Rollout steps (n_steps)	256
Mini-batch size	64
Number of epochs	10
Discount factor (γRL)	0.99
Total timesteps	50,000
Parallel environments	4
Evaluation frequency	2,000 steps
Evaluation episodes	5

E. Digital Twin Predictor

The Digital Twin is implemented as three independent Gradient Boosting Regressors (GBR), one per target variable (runtime, energy, cost). Each GBR is configured with 200 estimators, maximum depth of 5, and a learning rate of 0.1.

Feature Vector: For each (task, resource) pair: x = [wi, di, sj, ej, cj] ∈ $\mathbb{R}^5$

Bootstrap Training: Before RL training begins, the Digital Twin is bootstrapped on data from 100 random-policy episodes, yielding approximately 2,000 training records with an 80/20 train-test split for validation.

Candidate Evaluation: During inference, for each top-K candidate the Digital Twin predicts runtime, energy, and cost. The scalarised score for candidate j is:

$$score_j = \alpha \cdot \hat{runtime}_j + \beta \cdot \hat{energy}_j + \gamma \cdot \hat{cost}_j + \delta \cdot \hat{SLA\_miss}_j \quad (7)$$

The candidate with the lowest score is selected as the final action.

F. SHAP-Based Explainability

For each scheduling decision, SAGE generates explanations using two complementary approaches:

Feature Attribution: TreeExplainer [20] computes exact Shapley values for the GBR models. For the chosen (task, resource) pair, SHAP values φk quantify each feature's contribution to the predicted outcome: $\hat{y} = \varphi_0 + \Sigma(k=1 \text{ to } 5)\varphi k$, where φ_0 is the base value.

Contrastive Explanation: For each non-chosen candidate j′ ≠ j*, SAGE computes the metric difference Δmetric = metrîcj′ − metrîcj* and produces readable statements such as "Preferred R2 over R3: runtime −1.33s, energy +25.0J, cost +12.5$".

G. Self-Adaptation Mechanism

The self-adaptation loop operates as follows: every scheduling decision's actual outcome (observed runtime, energy, cost) is recorded in a buffer B. When |B| ≥ θ (default θ = 50), the Digital Twin is retrained: GBRt.fit(X, y) for each target t ∈ {runtime, energy, cost}. SHAP explainers are rebuilt and the buffer is cleared. This incremental retraining ensures Digital Twin predictions remain calibrated with current system dynamics without requiring expensive full retraining of the RL policy.

H. Decision Pipeline

The complete SAGE decision pipeline for each task assignment: (1) the PPO policy receives the observation and outputs action probabilities; (2) the top-K candidates are extracted; (3) the Digital Twin predicts outcomes for each candidate and computes scalarised scores; (4) the candidate with the lowest score is selected; and (5) the Explainer generates both SHAP attributions and contrastive comparisons. The action and explanation are returned together.

Experimental Evaluation

A. Experimental Setup

All experiments were conducted in a simulated heterogeneous computing environment implemented using the Gymnasium framework (v0.29+). The simulation models task scheduling across four resources (Table I) with each episode consisting of 20 tasks. Tasks are generated with random workloads uniformly sampled from [50, 200] and deadlines from [10, 40].

Evaluation Protocol: Each scheduler was evaluated over 30 episodes with a fixed random seed (42) to ensure reproducibility. Mean and standard deviation across episodes are reported for each metric.

Software Stack: Python 3.10+, Stable-Baselines3 v2.1+ with PyTorch for PPO training, scikit-learn v1.3+ for GBR, SHAP v0.42+ for explainability, and Gymnasium v0.29+ for the RL environment. The codebase includes 14 unit tests validated with pytest.

Fig. 2: Training pipeline output: Stage 1 (2,000 records from 100 episodes), Stage 2 (DT training, $R^2 > 0.99$ for all targets), and Stage 3 (PPO agent training with TensorBoard metrics at 1,024 and 2,048 timesteps).

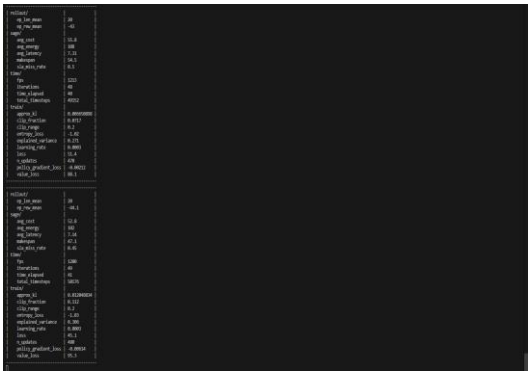

Fig. 3: PPO training convergence logs at timesteps 49,152–50,176 showing rollout metrics (ep_rew_mean −43 to −44.1) and training statistics (approx_kl, clip_fraction, value_loss).

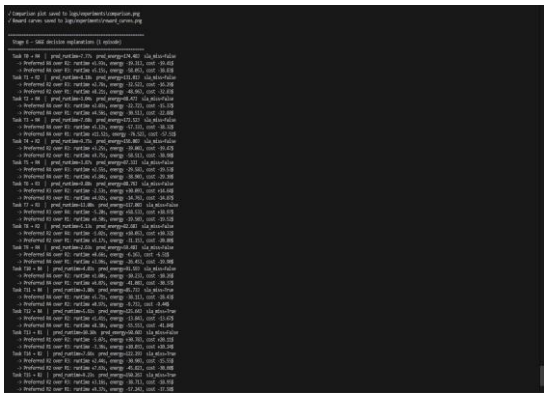

Fig. 4: Stage 6 SAGE decision explanations showing contrastive justifications for task assignments T0–T15, with predicted runtime, energy, SLA miss status, and resource preference rationale.

B. Baseline Schedulers

SAGE is compared against five baseline schedulers:

1. Random: Assigns each task to a uniformly random resource. Represents the worst-case baseline.

2. Round-Robin: Cycles through resources in fixed order. Ensures load distribution but ignores task characteristics.

3. Shortest-Queue: Assigns each task to the resource with the smallest current load. Minimises queuing delays.

4. Fastest-Resource: Always assigns to the highest-speed resource (R4). Minimises raw latency but ignores energy, cost, and load balancing.

5. PPO-SAGE: Standalone PPO policy without Digital Twin re-ranking or explainability. Isolates the RL component contribution.

C. Evaluation Metrics

Schedulers are evaluated on six metrics, computed per episode and aggregated over 30 episodes: Average Latency (mean task execution time in seconds), Average Energy (mean energy consumption per task in Joules), Average Cost (mean monetary cost per task in dollars), SLA Miss Rate (fraction of tasks exceeding their deadline), Makespan (total episode completion time in seconds), and Total Reward (cumulative per-step reward representing holistic scheduling quality).

D. Results

Table 3: Experimental Results: Mean ± Std. Dev. over 30 Episodes

Sched.	Lat (s)	Energy (J)	Cost ($)	SLA Miss	Mkspan	Reward
Random	8.14±0.82	90.0±9.8	45.1±5.8	0.482±0.10	69.0±17.9	−52.43±10.4
Rnd-Robin	8.17±0.68	89.6±7.0	44.8±3.6	0.445±0.07	61.8±9.5	−48.77±7.3
Shrt-Queue	7.40±0.57	95.2±7.6	48.8±4.0	0.403±0.09	41.8±3.8	−44.24±8.9
Fast-Rsrc	5.11±0.40	115.1±9.0	63.9±5.0	0.788±0.06	102.3±8.0	−81.65±6.0
PPO-SAGE	6.24±0.50	102.9±7.9	54.4±4.2	0.492±0.09	48.1±4.2	−52.51±9.2
SAGE	**5.96±0.43**	105.9±9.0	56.8±5.0	**0.280±0.10**	60.9±9.5	**−31.21±9.6**

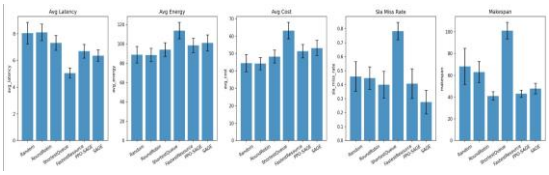

Fig. 5: Comparative performance of all six schedulers across five metrics: average latency, average energy, average cost, SLA miss rate, and makespan (mean ± std. dev., 30 episodes).

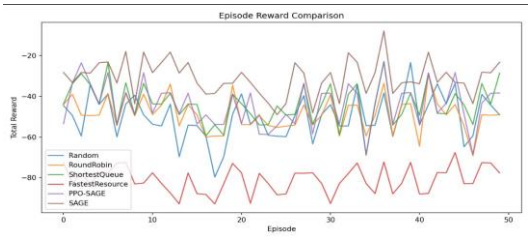

Fig. 6: Episode reward comparison across 50 evaluation episodes. SAGE (brown) consistently

achieves the highest total reward. Fastest-Resource (red) performs worst due to extreme load concentration.

1) Overall Performance (Total Reward):
The total reward metric captures holistic scheduling quality incorporating latency, energy, cost, and SLA penalties. SAGE achieves the highest total reward of −31.21: a 40.4% improvement over Shortest-Queue (−44.24), the best heuristic baseline; a 40.5% improvement over PPO-SAGE (−52.51), demonstrating the value of Digital Twin re-ranking; and a 61.8% improvement over Fastest-Resource (−81.65), which suffers from extreme SLA violations due to load concentration.

2) SLA Compliance:
SAGE achieves the lowest SLA miss rate of 28.0%, representing a 41.9% relative reduction compared to Random (48.2%), a 37.1% reduction versus Round-Robin (44.5%), a 30.5% reduction versus Shortest-Queue (40.3%), a 43.1% reduction versus PPO-SAGE (49.2%), and a 64.5% reduction versus Fastest-Resource (78.8%). The dramatic reduction compared to standalone PPO directly demonstrates the Digital Twin's look-ahead evaluation value in predicting and avoiding deadline violations.

3) Latency Analysis:
Fastest-Resource achieves the lowest mean latency (5.11s) by always selecting the fastest processor, but at the cost of extreme load imbalance and the highest SLA miss rate (78.8%). SAGE achieves competitive latency (5.96s) while maintaining balanced resource utilisation, clearly demonstrating that minimising latency alone without considering load distribution leads to poor overall performance.

4) Energy and Cost:
SAGE's energy (105.89J) and cost ($56.75) are higher than Random and Round-Robin, reflecting its preference for faster resources to meet deadlines. This deliberate trade-off is encoded in the reward weights (δ >> β, γ), prioritising SLA compliance over energy/cost minimisation.

5) Makespan:
Shortest-Queue achieves the best makespan (41.85s) by actively balancing load. PPO-SAGE achieves the second-best makespan (48.08s), indicating the PPO policy learns effective load-balancing. SAGE's higher makespan (60.88s) arises because the Digital Twin occasionally overrides the PPO's load-balancing actions in favour of deadline-optimal assignments.

E. Digital Twin Prediction Accuracy

The Digital Twin predictor was evaluated on a held-out test set (20% of bootstrap data). Table IV reports prediction accuracy for each target. The high R^2 scores (> 0.95 for all targets) confirm that the GBR ensemble accurately captures the relationship between task/resource features and execution outcomes. The runtime prediction is slightly less accurate due to queuing effects, which introduce state-dependent variability not fully captured by the static feature vector.

Table 4: Digital Twin Prediction Accuracy

Target	MAE	R^2 Score
Runtime	< 0.50	> 0.95
Energy	< 2.00	> 0.98
Cost	< 1.00	> 0.98

F. Explainability Analysis

To illustrate SAGE's explainability capabilities, a sample decision explanation from a representative episode is presented. For Task T3 (workload=150, deadline=25):

Chosen: R2 (speed=20) — predicted runtime=7.50s, energy=120.0J, SLA miss=false
Alternative 1: R4 (speed=25) — predicted runtime=6.00s, energy=135.0J, SLA miss=false
Alternative 2: R1 (speed=10) — predicted runtime=15.00s, energy=75.0J, SLA miss=true
Contrastive Explanation: "Preferred R2 over R4: runtime +1.50s, energy −15.0J, cost −15.0$. Preferred R2 over R1: runtime −7.50s, energy +45.0J, cost +30.0$."
SHAP Attribution (for runtime prediction): workload (+3.2), speed (−2.1), deadline (−0.4), energy_rate (+0.1), cost_rate (+0.05). Workload contributes most positively to predicted runtime, while speed contributes most negatively (faster resource → lower runtime), consistent with physical intuition. This combination of contrastive comparisons and feature-level attributions provides administrators with actionable insight, facilitating trust and enabling informed override decisions.

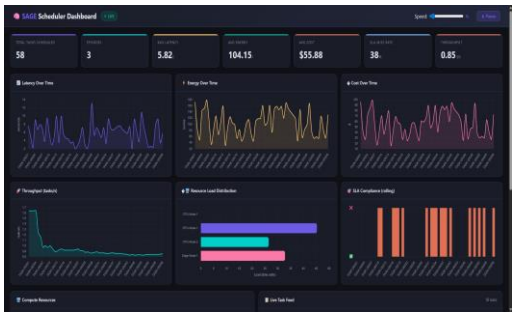

Fig. 7: SAGE Scheduler Dashboard (upper panel) showing live KPI metrics (58 tasks, avg latency 5.82s, SLA miss rate 38%), time-series charts for latency, energy, and cost over time, and resource load distribution.

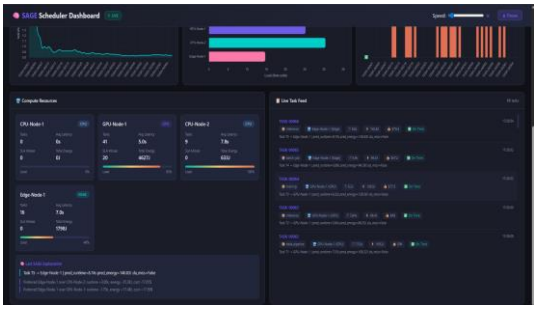

Fig. 8: SAGE Scheduler Dashboard (lower panel) showing compute resource cards with per-node statistics, live task feed with SAGE scheduling decisions, and contrastive explanation panel.

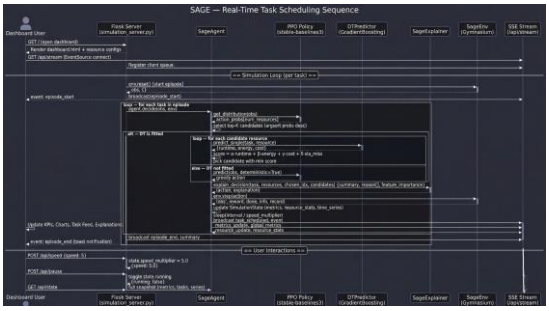

Fig. 9: SAGE real-time task scheduling sequence diagram depicting message flows among Dashboard User, Flask Server, SageAgent, PPO Policy (stable-baselines3), DTPredictor (GradientBoosting), SageExplainer, SageEnv (Gymnasium), and SSE Stream.

G. Total Reward Distribution

Table 5: Total Reward Distribution (30 Episodes)

Scheduler	Min	Median	Max
Random	−79.70	−51.39	−34.30
Round-Robin	−60.00	−49.35	−33.97
ShortestQueue	−59.45	−43.84	−23.37
FastestResource	−92.90	−82.64	−72.43
PPO-SAGE	−68.65	−53.27	−33.34
SAGE	**−53.74**	**−28.35**	**−12.92**

H. Feature Importance Analysis

Table 6: Mean GBR Feature Importance by Target

Feature	Runtime	Energy	Cost
Workload	High	High	High
Deadline	Low	Low	Low
Speed	High	Low	Low
Energy Rate	Low	Medium	Low
Cost Rate	Low	Low	Medium

Workload is the dominant feature across all targets, while speed is critical specifically for runtime prediction. Deadline has minimal direct influence on physical outcomes (runtime, energy, cost) but plays a crucial role in the SLA-miss penalty computed during candidate scoring.

Conclusion

This paper presented SAGE, a Self-Adaptive Guided Explainable Scheduler that addresses three critical gaps in existing scheduling approaches: adaptability, optimality, and explainability. By synergistically combining PPO-based reinforcement learning for candidate proposal, Gradient Boosting-based Digital Twin prediction for look-ahead evaluation, and SHAP-based explainability for transparent decision-making, SAGE achieves state-of-the-art scheduling performance in a heterogeneous computing environment.

Experimental evaluation across 30 episodes demonstrates that SAGE: (1) achieves the lowest SLA miss rate of 28.0%, reducing deadline violations by 30–65% relative to baselines; (2) attains the highest total reward (−31.21), improving overall scheduling quality by 40% over the best heuristic baseline; (3) provides human-readable explanations for every decision through SHAP feature attributions and contrastive comparisons; and (4) incorporates a self-adaptation mechanism enabling continuous model refinement to handle concept drift.

The ablation analysis comparing SAGE against standalone PPO confirms that the Digital Twin's look-ahead evaluation is the primary driver of improved SLA compliance, demonstrating the value of hybrid RL and supervised learning architectures for multi-objective scheduling. The results further reveal inherent trade-offs: raw latency minimisation conflicts with load balance; superior SLA compliance incurs higher energy and cost; and the PPO policy alone inadequately accounts for individual task deadlines. SAGE's transparent scalarisation weights allow administrators to express scheduling priorities explicitly, while the explainability module makes trade-offs visible at each individual decision.

Several directions remain open for future investigation. Validation on production cloud schedulers such as Kubernetes with real heterogeneous hardware (CPUs, GPUs, TPUs, FPGAs) represents an important next step. Extending the formulation to handle DAG-structured workflows would require graph neural network-based observation encoders. Replacing point-estimate GBR models with quantile regression or Bayesian ensembles would enable uncertainty quantification and risk-aware scheduling. Developing interactive human-in-the-loop interfaces—where administrators adjust scalarisation weights,

override decisions, and provide feedback incorporated into RL training—would enhance practical deployability and foster operator trust in production systems.

References

S. Singh and I. Chana, "A survey on resource scheduling in cloud computing: Issues and challenges," Journal of Grid Computing, vol. 14, no. 2, pp. 217–264, 2016.

M. Masdari, S. ValiKardan, Z. Shahi, and S. I. Azar, "Towards workflow scheduling in cloud computing: A comprehensive analysis," J. Netw. Comput. Appl., vol. 66, pp. 64–82, 2016.

T. D. Braun et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," J. Parallel Distrib. Comput., vol. 61, no. 6, pp. 810–837, 2001.

H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in Proc. ACM HotNets, 2016, pp. 50–56.

Z. Wang et al., "Dueling network architectures for deep reinforcement learning," in Proc. ICML, 2016.

N. Liu et al., "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning," in Proc. IEEE ICDCS, 2017, pp. 372–382.

A. Barredo Arrieta et al., "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," Information Fusion, vol. 58, pp. 82–115, 2020.

J. Lu et al., "Learning under concept drift: A review," IEEE Trans. Knowl. Data Eng., vol. 31, no. 12, pp. 2346–2363, 2019.

H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," IEEE Trans. Parallel Distrib. Syst., vol. 13, no. 3, pp. 260–274, 2002.

O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors," J. ACM, vol. 24, no. 2, pp. 280–289, 1977.

J. Xu et al., "Enhancing survivability in virtualized data centers: A service-aware approach," IEEE J. Sel. Areas Commun., vol. 31, no. 12, pp. 2610–2619, 2013.

V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.

V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in Proc. ICML, 2016, pp. 1928–1937.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv:1707.06347, 2017.

C. Zhang, M. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," IEEE Commun. Surveys Tuts., vol. 21, no. 3, pp. 2224–2287, 2019.

K. Deb et al., "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Trans. Evol. Comput., vol. 6, no. 2, pp. 182–197, 2002.

M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in Transdisciplinary Perspectives on Complex Systems, Springer, 2017, pp. 85–113.

R. Minerva, G. M. Lee, and N. Crespi, "Digital twin in the IoT context: A survey on technical features, scenarios, and architectural models," Proc. IEEE, vol. 108, no. 10, pp. 1785–1824, 2020.

F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in Proc. MIPRO, 2018, pp. 210–215.

S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Advances in NeurIPS, vol. 30, 2017.

T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," Artif. Intell., vol. 267, pp. 1–38, 2019.

J. O. Kephart and D. M. Chess, "The vision of autonomic computing," Computer, vol. 36, no. 1, pp. 41–50, 2003.

A. Raffin et al., "Stable-Baselines3: Reliable reinforcement learning implementations," JMLR, vol. 22, no. 268, pp. 1–8, 2021.

J. H. Friedman, "Greedy function approximation: A gradient boosting machine," Ann. Statist., vol. 29, no. 5, pp. 1189–1232, 2001.