Ingénierie des Systèmes d'Information
Vol. 31, No. 3, March, 2026, pp. 889-899
Journal homepage: http://iieta.org/journals/isi

Cost–Reliability-Aware Multi-Workflow Scheduling Using Multi-Agent Deep Reinforcement Learning in Edge–Cloud Systems

Praveen K S[1*], L. Manjunatha Rao[2], Rajshekhar Ghogge[3]

[1] Department of Master of Computer Application, East West Institute of Technology (Affiliated to Visvesvaraya Technological University), Bangalore 560091, India
[2] Department of Information Science and Engineering, Dr. Ambedkar Institute of Technology (Affiliated to Visvesvaraya Technological University), Bangalore 560056, India
[3] Department of Master of Computer Application, Dr. Ambedkar Institute of Technology (Affiliated to Visvesvaraya Technological University), Bangalore 560056, India

Corresponding Author Email: praveenksphd@gmail.com

https://doi.org/10.18280/isi.310320

Received: 21 December 2025
Revised: 20 February 2026
Accepted: 20 March 2026
Available online: 31 March 2026

Keywords:
edge-cloud computing, energy and cost optimization, multi-workflow scheduling, multi-agent deep reinforcement learning, reliability-aware resource allocation, virtual resource function reuse

ABSTRACT

Efficient workflow scheduling across heterogeneous cloud–edge environments demands a balanced trade-off among energy consumption, reliability, and operational cost. Existing solutions often overlook service reuse and multi-agent coordination, leading to redundant resource utilization and unstable convergence. This paper introduces CR-VRF-MW-MADRL, a Cost–Reliability–aware Multi-Workflow multi-agent deep reinforcement learning (MADRL) framework that enables hierarchical global–local scheduling and virtual resource function (VRF) reuse. The proposed model employs a global agent for datacenter and service function chain (SFC) selection, while local agents optimize task-to-server mapping with dynamic voltage and frequency scaling (DVFS) for energy–latency balance. A reward-driven policy update ensures adaptive learning under variable workloads and reliability constraints. Experimental evaluation across heterogeneous datacenters demonstrates significant improvements in energy efficiency, cost reduction, and reliability compared with existing DRL-based and heuristic schedulers, validating the scalability and convergence stability of the proposed CR-VRF-MW-MADRL model. Extended stress scenario analysis, statistical validation using ANOVA, and computational complexity evaluation further confirm robustness, significance, and scalability under high-load and failure conditions.

1. INTRODUCTION

The exponential growth of data-driven applications and the widespread adoption of cloud–edge computing ecosystems have introduced significant challenges. These include balancing energy efficiency, cost, reliability, and execution latency in large-scale workflow scheduling. As scientific and industrial workloads continue to grow in volume and heterogeneity, efficient task orchestration across distributed computational tiers has become essential for sustainable computing environments. Traditional heuristic and metaheuristic approaches perform well in static settings. However, they often struggle in dynamic multi-workflow scenarios, where resource heterogeneity, service function reuse, and reliability trade-offs must be handled simultaneously [1, 2].

Recent research has shifted toward intelligent and energy-aware scheduling frameworks. These approaches integrate predictive models, virtualization, and reinforcement learning to achieve multi-objective optimization. Shivanandappa et al. [1] proposed an energy- and cost-aware workload scheduler that adapts to heterogeneous cloud infrastructures. Their model improves energy utilization under budget constraints. Similarly, Chen et al. [2] introduced a cross-view intelligent scheduling method (CSCR). This method reduces redundant workflow dependencies using computational graph reduction and task clustering, thereby enhancing scalability in complex cloud systems. These studies highlight the need for adaptive scheduling strategies that move beyond static optimization toward context-aware and hierarchical orchestration.

In distributed systems, task decomposition and hierarchical scheduling play a crucial role. They help balance computation across cloud, edge, and end-device layers. Cai et al. [3] proposed a hierarchical model that decomposes workflow tasks across collaborative tiers. Their approach optimizes bandwidth and latency under inter-tier constraints. Divyaprabha and Sudarshan [4] and Divyaprabha and Sudarshan [5] further advanced this area by focusing on failure minimization and energy-deadline optimization. Their work ensures stable task execution in unreliable and heterogeneous environments. Uma and Shukla [6] extended this concept by incorporating dynamic virtual resource optimization in edge–cloud workflows, demonstrating notable improvements in both performance and energy efficiency.

The existing literature on workflow scheduling can be broadly categorized into Deep Reinforcement Learning (DRL)-based, Multi-Agent DRL (MADRL)-based, and Hybrid evolutionary–learning approaches, each addressing different aspects of multi-objective optimization. DRL-based approaches have significantly advanced adaptive workflow scheduling through intelligent policy learning. Mangalampalli et al. [7] proposed a prioritized DRL-based scheduler to balance latency and resource allocation, while Zhang et al. [8] focused on reliability-aware scheduling under energy constraints. Similarly, Fan et al. [9] and Tao et al. [10] addressed budget and deadline constraints using heuristic-enhanced learning. Huang et al. [11] further extended DRL for online multi-workflow scheduling under dynamic environments. Although effective in adaptability, these approaches primarily operate in single-agent settings, limiting scalability in distributed systems.

MADRL-based methods extend this paradigm by enabling cooperative decision-making across multiple agents. Jayanetti et al. [12] introduced a multi-agent DRL framework for energy-aware scheduling in distributed cloud environments, demonstrating improved coordination. However, issues such as convergence instability and limited inter-agent coordination mechanisms remain significant challenges, particularly under large-scale heterogeneous workloads.

Hybrid and evolutionary approaches, including metaheuristics combined with learning, have also been widely explored. Ijaz et al. [13] utilized differential evolution for cost–time optimization, while Liu et al. [14] proposed a critical-task-driven evolutionary algorithm for large-scale workflows. Chen et al. [15] integrated DRL with adaptive scheduling for flexible workflow execution. Despite their scalability, these approaches often suffer from high computational complexity and lack of real-time adaptability.

In parallel, NFV and SFC-based scheduling frameworks [16-20] have emphasized resource virtualization, service chaining, and delay-aware optimization. While these methods improve resource utilization [21] and service continuity, they do not incorporate cost–reliability trade-offs for multi-workflow-level coordination [22, 23].

Table 1. Comparative analysis of recent workflow scheduling models

Ref.	Technique/Framework	Objectives	Key Strength	Limitation
[7]	DRL-based prioritized workflow scheduling	Latency, resource allocation	Adaptive priority-aware scheduling using learning	Single-agent design limits scalability in multi-workflow environments
[8]	Reliability-enhanced workflow mapping	Energy, reliability	Adaptive fault tolerance	Single-workflow scope
[9]	Priority-adjusted budget scheduler	Cost, deadline	Critical-task optimization	Limited reliability modeling
[10]	Deadline–budget constrained ACO-based scheduling (DB-ACO)	Deadline, cost	Efficient constraint handling with heuristic optimization	Convergence speed issues and lack of reliability awareness
[11]	RL-based online scheduler	Latency, adaptability	Real-time resource control	Partial observability
[12]	Multi-agent DRL scheduling	Energy, sustainability	Cooperative learning	Convergence instability
[13]	Multi-objective differential evolution scheduling	Time, cost optimization	Effective global search and Pareto optimization	High computational overhead and limited real-time adaptability
[14]	Multi-objective evolutionary algorithm	Performance, fairness	Scalable scheduling	Slow convergence
[15]	AFAS (DRL adaptive scheduler)	Performance, flexibility	Arbitrary workflow adaptation	No SFC reuse
[16]	NFV-based IoT allocator	Resource, delay	Dynamic VNF migration	Lacks reliability focus
[17]	Cooperative VNF caching (LEO)	Caching, latency	Learning-based coordination	High control cost
[18]	Deterministic VNF-FG scheduler	Delay, reliability	Real-time performance	Poor scalability
[19]	Any casting-based SFC provisioning	Cost, delay	Cloud–edge collaboration	Complex routing

Table 1 indicates that most existing workflow scheduling approaches primarily address isolated optimization objectives, such as cost, energy, or deadline constraints, without adequately capturing their interdependencies in multi-workflow environments. Reliability-focused models [9] enhance fault tolerance but are restricted to single-workflow scenarios, whereas budget-driven approaches [10, 11] lack explicit reliability guarantees. DRL-based methods [7, 19] provide adaptability but often rely on single-agent architectures, limiting scalability and coordinated decision-making. Similarly, evolutionary and hybrid techniques [8, 22] achieve strong optimization performance but suffer from high computational overhead and slow convergence under dynamic workloads. In addition, NFV/SFC-based frameworks [23-27] improve resource utilization and service chaining but do not incorporate cost–reliability trade-offs or workflow-level coordination.

These limitations reveal a critical gap in the design of a unified scheduling framework that simultaneously supports cost–reliability optimization, efficient reuse of virtual resource functions, and scalable multi-agent coordination. In particular, three key challenges persist: (1) lack of integrated cost–reliability optimization for concurrent multi-workflow execution, (2) insufficient reuse of virtual resource functions leading to redundant deployments and increased energy consumption, and (3) absence of hierarchical coordination among distributed agents, affecting scalability and convergence stability.

To address these challenges, the proposed CR-VRF-MW-MADRL framework introduces a hierarchical multi-agent deep reinforcement learning (MADRL) architecture, where a Global Agent manages cross-datacenter orchestration and Local Agents perform fine-grained task scheduling under deadline and reliability constraints. The model incorporates a Cost–Reliability Index (CRI)-driven reward formulation and enables adaptive VRF–SFC reuse, reducing redundant deployments while improving energy efficiency and execution cost. This unified design facilitates coordinated decision-making across distributed resources, ensuring scalable, reliable, and adaptive multi-workflow scheduling in

heterogeneous edge–cloud environments.

The main contributions of this work are summarized as follows:

1. A Cost–Reliability Index (CRI)-driven scheduling policy is introduced to jointly optimize energy consumption, execution cost, and reliability in multi-workflow edge–cloud environments.
2. A Virtual Resource Function–Service Function Chain (VRF–SFC) reuse mechanism is developed to minimize redundant deployments and improve resource utilization across concurrent workflows.
3. A hierarchical multi-agent deep reinforcement learning framework is designed, where global and local agents collaboratively perform inter- and intra-datacenter scheduling under dynamic workload conditions.
4. A unified optimization framework is formulated that integrates the above components, enabling stable convergence and scalable performance in heterogeneous and large-scale environments.

2. PROPOSED METHODOLOGY

Unlike existing approaches, the proposed framework integrates cost–reliability-aware optimization, virtual resource reuse, and hierarchical multi-agent coordination within a unified learning architecture. The Cost–Reliability-aware Virtual Resource Function and Multi-Workflow Scheduling framework (CR-VRF-SFC-MW-MADRL) is designed to support concurrent scientific workflows over heterogeneous edge–cloud environments. The proposed model combines three key mechanisms: (i) Virtual Service Function (VSF) reuse to minimize redundant deployments and reduce execution overhead, (ii) Service Function Chain (SFC) optimization to preserve task dependencies and execution order, and (iii) a hierarchical MADRL strategy for distributed scheduling decisions across federated datacenters. Workflows are modeled as Directed Acyclic Graphs (DAGs), where tasks are dynamically mapped to resources while satisfying multi-level deadline and reliability constraints. By incorporating a cost–reliability-aware reward formulation and enabling adaptive coordination between global and local agents, the framework supports efficient resource utilization and scalable workflow execution under dynamic workload conditions.

2.1 System model

Each workflow is modeled as a DAG $G(V, E)$, where each node $v_i \in V$ denotes a computational task, and edge $(v_i, v_j) \in E$ represents a precedence constraint. The DAG is mapped onto a set of federated datacenters,

$$DC = \{dc_1, dc_2, \dots, dc_n\} \quad (1)$$

each containing heterogeneous hosts,

$$M_i = \{m_1, m_2, \dots, m_\lambda\} \quad (2)$$

with distinct compute frequencies, memory capacities, and power ratings.

Each task can be represented through Virtual Service Functions (VSFs) such as aggregation, preprocessing, or inference. Related VSFs form a Service Function Chain (SFC) that defines an execution pipeline (e.g., preprocessing → feature extraction → classification). Reuse of previously deployed VSFs or SFCs in local caches minimizes data transfer, instantiation delay, and network congestion across the edge–fog–cloud hierarchy. A summary of the key mathematical symbols and their definitions is provided in Table 2 for clarity and reference.

Table 2. Symbols and description

Symbol	Description
$G(V, E)$	Workflow DAG with tasks V and dependencies E
t_j	Task j in workflow
dc_k	Datacenter k
m_i	Physical host i
$u(m_i)$	CPU utilization of host m_i
P_{idle}, P_{dyn}	Idle and dynamic power consumption
$T(t_j)$	Execution time of task t_j
$E(t_j)$	Energy consumption of task t_j
MC	Total execution cost
$A(m_i)$	Availability (reliability) of host m_i
R_{SFC}	Reliability of service function chain
R_{min}	Minimum reliability threshold
SFC_{reuse}	Ratio of reusable virtual functions
RU	Resource utilization
CRI	Cost–Reliability Index
w_R, w_C, w_U, w_{reuse}	Weights for CRI components
S, A	State and action space in MADRL
R_i	Reward function
π_θ	Policy function
Q^π	State-action value function

2.2 VRF–SFC reuse mechanism

To improve resource efficiency and reduce redundant deployments, the proposed framework incorporates a Virtual Resource Function (VRF) reuse mechanism within Service Function Chain (SFC) execution. Each SFC request is mapped to a cached VRF pool available within the datacenter. A reuse score is computed as

$$SFC_{reuse} = \frac{\text{Number of reusable VRFs}}{\text{Total required VRFs}}$$

If the reuse configuration satisfies the reliability constraint $R_{SFC} \geq R_{min}$, the scheduler reuses the available VRFs; otherwise, partial redeployment of missing or unreliable functions is performed. This selective reuse strategy minimizes service instantiation delay, reduces bandwidth consumption, and improves overall energy efficiency while maintaining reliability guarantees.

2.3 Energy and cost model

The instantaneous power consumed by a host m_i is given by

$$P(m_i) = \begin{cases} P_{m_i}^{idle} + (P_{m_i}^{dyn} - P_{m_i}^{idle}) \cdot u(m_i), & u(m_i) > 0 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where, $u(m_i)$ is CPU utilization, $P_{m_i}^{idle}$ and $P_{m_i}^{dyn}$ denote idle and peak dynamic power, respectively.

The energy consumed by datacenter dc_k during interval Δt is

$$E_{dc}(k) = \sum_{i=1}^{\lambda} P(m_i)(k) \cdot \Delta t \tag{4}$$

The execution time for a task t_j is

$$T(t_j) = \frac{L(t_j)}{F} + WT(t_j) + \max_{t_i \in pred(t_j)} DT(t_i, t_j) \tag{5}$$

where, $L(t_j)$ denotes task workload, F processor frequency, $WT(t_j)$ waiting time, and $DT(t_i, t_j)$ inter-task communication delay. The total task energy is

$$E(t_j) = T(t_j)\,[U \cdot P_{active} + P_{idle}] + ET(t_j) \tag{6}$$

where, $ET(t_j) = \sum_{t_i \in pred(t_j)} \frac{D(t_i,t_j)}{B} P_{comm}$ defines transmission energy. The overall workflow cost is

$$MC = \sum_{t_j \in T} \frac{CT(t_j)}{UC} \tag{7}$$

where, $CT(t_j)$ is compute cost and UC is the unit cost per compute cycle.

2.4 Reliability model

Each host m_i has reliability (availability)

$$A(m_i) = e^{-\lambda_{m_i} \Delta t}, \tag{8}$$

where, λ_{m_i} is the host failure rate. The reliability of a VSF instance s with Rreplicas is

$$R_{VSF}(s) = 1 - \prod_{r=1}^{R} (1 - R_{inst}(s, r)), \tag{9}$$

and the reliability of an SFC c is

$$R_{SFC}(c) = \prod_{s \in c} R_{VSF}(s) \prod_{l \in links(c)} R_{link}(l). \tag{10}$$

2.5 Multi-level deadlines and reliability constraints

Each task t_j satisfies hierarchical deadlines:

$$D_{level}(t_j) = \begin{cases} T_{start}(t_j) + \delta_{edge}, t_j \in \text{Edge} \\ T_{start}(t_j) + \delta_{fog}, t_j \in \text{Fog} \\ T_{start}(t_j) + \delta_{cloud}, t_j \in \text{Cloud} \end{cases} \tag{11}$$

with $\delta_{edge} < \delta_{fog} < \delta_{cloud}$. The global constraints are

$$D_{level}(t_j) = \begin{cases} C_1: \sum_{t_j \in T} T(t_j) \le D_{global}, \\ C_2: E_{dc}(k) \le E_{max}, \\ C_3: U_{node}(m_i) \le 1, \forall m_i, \\ C_4: R_{SFC}(c) \ge R_{min} \end{cases} \tag{12}$$

2.6 Cost-reliability index

To jointly evaluate cost, energy, reliability, and reuse, a Cost–Reliability Index (CRI) is defined as

$$CRI = \frac{w_R \bar{R}_{SFC} - w_C \frac{MC}{MC_{max}} + w_U RU + w_{reuse} SFC_{reuse}}{w_R + w_C + w_U + w_{reuse}}, \tag{13}$$

where, RU is resource reutilization, SFC_{reuse} is the reuse score, and w_* are tunable weights. Higher CRIindicates a more reliable and cost-efficient configuration. The weighting parameters $(w_R, w_C, w_U, w_{reuse})$ are empirically determined through preliminary experiments. Initially, equal weights are assigned, followed by sensitivity-based tuning to balance reliability, cost, and utilization objectives. This ensures stable convergence of the learning model across varying workflow sizes and system conditions.

2.7 Multi-agent deep reinforcement learning model

The system is modeled as a Partially Observable Markov Game (POMG) comprising one Global Agent (GA) and multiple Local Agents (LA). Each agent observes state

$$S = \{utilization, bandwidth, VSF\_cache, reliability, deadline\_slack\}$$

and selects actions from A_{GA}: datacenter and SFC placement, A_{LA}: VM/core selection, reuse decision. The reward function is reliability-aware:

$$\begin{aligned} R_i = -(\alpha E(t_j) + (1-\alpha) T(t_j)) + \beta RU \\ + \gamma SFC_{reuse} - \delta D_{viol} \\ + \kappa (R_{SFC}(c) - R_{min}) \\ - \mu Cost_{deploy}(s). \end{aligned} \tag{14}$$

Agents are trained under a centralized critic–decentralized actor paradigm (MADDPG), where policy gradients are updated as

$$\begin{aligned} \nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\pi_{\theta_i}} [\nabla_{\theta_i} \log \pi_{\theta_i}(a_i \\ | O_i) Q^{\pi}(s, a_1, \ldots, a_N)]. \end{aligned} \tag{15}$$

2.7.1 MADRL architecture and implementation details

To ensure reproducibility and clarity, the architecture of the proposed MADRL model is explicitly defined. Each agent (global and local) is implemented using an actor–critic framework based on Multi-Agent Deep Deterministic Policy Gradient (MADDPG).

The state representation for the Global Agent includes system-wide parameters:

$$S_g = \{E_{dc}, U_{dc}, B_{dc}, SFC_{cache}, R_{est}\}$$

where, E_{dc} denotes datacenter energy, U_{dc} resource utilization, B_{dc} bandwidth, SFC_{cache} available reusable chains, and R_{est} estimated reliability.

The Local Agent observes:

$$S_l = \{CPU, memory, bandwidth, VSF_{cache}, deadline\_slack, A(m_i)\}$$

Each actor network consists of three fully connected layers (128–64–32 neurons) with ReLU activation, followed by a linear output layer for action selection. The critic network adopts a four-layer structure (128–64–32–1) to estimate the Q-value. The learning rate is set to 10^{-4} for the actor and 10^{-3} for the critic, with a discount factor $\gamma = 0.95$. Experience replay and target networks are employed to stabilize training, and mini-batch updates are performed during each training iteration. This architecture ensures efficient learning and convergence across heterogeneous multi-workflow environments.

2.7.2 Global-local agent coordination mechanism

The proposed MADRL framework adopts a hierarchical coordination strategy between the Global Agent (GA) and multiple Local Agents (LAs) to ensure scalable and conflict-aware decision-making.

The Global Agent performs inter-datacenter orchestration, selecting optimal datacenters and SFC placements based on global system states. Local Agents execute intra-datacenter scheduling, mapping tasks to virtual machines and managing resource allocation.

Coordination is achieved through a centralized critic with decentralized actors, where all agents share a global reward signal while maintaining independent policies. The Global Agent periodically receives feedback from Local Agents in terms of energy consumption, reliability, and reuse efficiency, enabling policy refinement.

Potential conflicts, such as resource contention or competing task allocations, are resolved through:

1. Priority-aware reward shaping, where tasks with tighter deadlines receive higher importance
2. Reliability constraints, ensuring $R_{SFC} \geq R_{min}$
3. Resource utilization thresholds, preventing overloading of nodes

Additionally, a penalty term in the reward function discourages conflicting allocations and deadline violations. This cooperative learning mechanism enables agents to converge toward globally optimal policies while maintaining local efficiency.

2.8 Energy and utilization optimization

The continuous energy minimization objective for executing dependent tasks is

$$\min \sum_{m=1}^{p} \int_{x_t}^{y_t} (u_m r_m^{\uparrow} a_m^v + \frac{(1-u_m) r_m^{\uparrow}}{(h_m^{\uparrow})^3} (h_m^f)^3) dt, \quad (16)$$

where, $r_m^{\uparrow}$ is resource utilization, h_m^f the host frequency, and a_m^v VM activity coefficient.

The system utilization is

$$\max \frac{\sum_{k=1}^{o} \sum_{l=1}^{|V_k|} (resource_l^k \cdot V_l^k)}{\sum_{m=1}^{p} (h_m^{\uparrow} \cdot B_m)}. \quad (17)$$

The final multi-objective function is:

$$\min_{\pi} (w_1 MC + w_2 E_{dc} + w_3 MT - w_4 RU - w_5 SFC_{reuse} - w_6 \bar{R}_{SFC}), \quad (18)$$

subject to constraints (12), ensuring energy-efficient, reliable, and reuse-optimized scheduling.

2.9 CR-VRF-SFC-MW-MADRL scheduling algorithm

The CR-VRF-SFC-MW-MADRL framework operates through two hierarchically coordinated algorithms that function at the global and local levels of the edge–cloud platform.

2.9.1 Global scheduler

The Global CR-VRF-SFC-MW-MADRL scheduler in Algorithm 1 is designed to make high-level decisions regarding datacenter selection and Service Function Chain (SFC) placement to maximize the Cloud Reliability Index (CRI) while minimizing cost–energy trade-offs. The global agent observes the system-wide state parameters, including datacenter energy consumption, bandwidth availability, resource utilization, and cached SFC instances. Based on the observed global state, the agent predicts an optimal action using its policy network parameterized by θ_g. The decision involves whether to reuse an existing SFC deployment or to redeploy a subset of components when reliability or resource constraints are violated. The CRI-based reuse and deploy metrics are compared to ensure service reliability ($R_{SFC} \geq R_{min}$) while minimizing reallocation overhead. The global reward function, formulated in Eq. (14), integrates energy efficiency, reliability, and execution cost. The global critic updates its policy through reinforcement learning, continuously refining decisions using feedback from multiple local agents, ensuring adaptive and optimal SFC distribution across heterogeneous datacenters.

Algorithm 1. Global CR-VRF-SFC-MW-MADRL Scheduler

Objective: Select datacenter and SFC placement to maximize CRI and minimize cost–energy trade-offs.

Step 1. **Initialize** Global Agent (GA) with parameters θ_g and reliability threshold R_{min}.

Step 2. **For each** ready task t_j:
Observe global state $S_g = \{E_{dc}, utilization, bandwidth, SFC_{cache}, R_{est}\}$.
Predict action $a_g = \pi_{\theta_g}(S_g) \rightarrow dc$.
Compute CRI_{reuse} and CRI_{deploy}.
Prefer reuse if $CRI_{reuse} \geq CRI_{deploy}$ and $R_{SFC} \geq R_{min}$; else redeploy minimal subset.
Compute global reward using Eq. (14) and update θ_g.

Step 3 Receive feedback from Local Agents and refine critic parameters.

End For

2.9.2 Local scheduler

The Local Scheduler in Algorithm 2 operates within each datacenter, optimizing task-to-server mapping under constraints of energy, reliability, and task deadlines. The local agent observes intra-datacenter state features such as CPU and memory utilization, network bandwidth, virtual SFC cache availability, and deadline slack. Using its local policy $\pi_{\theta_l}(S_l)$, it selects appropriate VMs or cores for execution, leveraging cached VSFs when available to reduce latency. If resources are unavailable, the agent dynamically instantiates new instances

while applying Dynamic Voltage and Frequency Scaling (DVFS) for energy–latency trade-off management. Rewards are computed using Eq. (14) based on local energy consumption, reliability, and reuse efficiency. The aggregated local metrics — including energy, reliability, and reuse rate are reported back to the global agent, enabling hierarchical coordination for end-to-end workflow optimization.

Algorithm 2. Local CR-VRF-SFC-MW-MADRL Scheduler

Objective: Optimize intra-datacenter task-to-server mapping under energy, deadline, and reliability constraints.

Step 1. **Initialize** Local Agent (LA) with parameters θ_l.

Step 2. **For each** incoming task t_j:
Observe local state $S_l = \{CPU, memory, bandwidth, VSF_{cache}, deadline$.
Select action $a_l = \pi_{\theta_l}(S_l) \rightarrow$ allocate VM/core.
Reuse VSF if cached; otherwise instantiate new instance.
Apply DVFS for energy–latency balance.
Compute reward via Eq. (14); update critic and policy.

Step 3 Report local metrics $\{E(t_j), R_{SFC}, SFC_{reuse}\}$ to GA for synchronization.
End For

The global agent coordinates inter-datacenter SFC placement and VSF reuse, while local agents perform fine-grained optimization within datacenters. Reliability telemetry (availability, MTTR, failure rate) is continuously updated to guide reinforcement learning. The hierarchical cooperation enables adaptive, reliability-aware resource utilization that achieves significant improvement in workflow throughput, energy efficiency, cost reduction, and VSF reuse compared to static or single-agent schedulers. In summary, the CR-VRF-SFC-MW-MADRL model introduces a cost-reliability-driven learning framework that adaptively balances performance, dependability, and resource reuse for complex scientific workflows across heterogeneous edge–cloud infrastructures.

3 SIMULATION ANALYSIS AND RESULT

This section presents an extensive performance evaluation of the proposed CR-VRF-SFC-MW-MADRL framework for scheduling scientific workflows over heterogeneous edge–cloud environments. The experimental study primarily aims to validate the framework's ability to achieve improved energy efficiency, reduced makespan, lower execution cost, enhanced resource utilization, better reliability when compared with existing baseline scheduling techniques.

3.1 Baseline models

To ensure a comprehensive and fair performance evaluation, four state-of-the-art workflow scheduling frameworks were selected as baseline models for comparison with the proposed CR-VRF-MW-MADRL framework. These baselines represent leading research directions in energy-aware, reliability-aware, and deep reinforcement learning (DRL)-based scheduling for cloud and edge–cloud environments. AFAS-DRL [15]: *Arbitrary-Freedom Adaptive Scheduling* introduces a DRL-driven multiworkflow scheduling framework that dynamically allocates cloud resources based on workflow priorities. Although highly adaptive, AFAS-DRL lacks explicit multi-agent coordination and cost–reliability coupling, limiting scalability in federated environments. REWS [8]: *Reliability Enhancement Workflow Scheduling* focuses on optimizing workflow reliability under energy constraints using a hybrid reliability–energy optimization model. However, it operates under static energy boundaries and does not incorporate reinforcement learning–based adaptability. CTMOEA [14]: *Critical-Task-Driven Multi-Objective Evolutionary Algorithm* addresses large-scale workflow scheduling by prioritizing critical tasks to minimize makespan and energy cost. Despite strong optimization capability, its evolutionary nature results in high computational complexity and limited responsiveness to dynamic workloads. BWS [9]: *Budget-Constrained Workflow Scheduling* integrates priority adjustment and cost-aware optimization using heuristic rules. While effective for constrained budgets, it does not account for service function reuse or hierarchical learning among distributed agents.

Collectively, these baselines represent leading benchmarks across evolutionary, heuristic, and single-agent DRL paradigms. Their comparative evaluation against CR-VRF-MW-MADRL highlights the advantages of hierarchical multi-agent coordination, VRF–SFC reuse, and joint cost–reliability optimization in heterogeneous edge–cloud environments. All baseline models are implemented under a unified simulation framework and evaluated using identical datasets, system configurations, and performance metrics to ensure a fair and consistent comparison.

3.2 Experiment setup

The simulation environment is implemented using an enhanced CloudSim 3.0 toolkit [24, 25], integrated with the Keras deep learning library [26] to facilitate MADRL model training and inference. The simulated edge–cloud infrastructure comprises 10 datacenters, each hosting 25 heterogeneous physical servers, modeled according to the SPECpower_ssj2008 benchmark suite [27].

Each physical host is provisioned with 32 GB RAM and 1 TB storage, while virtual nodes offer 32 GB storage and 5 Mbps bandwidth, ensuring realistic edge–cloud heterogeneity. The infrastructure operates on an Ubuntu Linux environment with a Xen hypervisor, enabling efficient resource virtualization and scheduling. Both small-core and large-core virtual machines are instantiated to capture diversity in computational capacity across nodes.

The experimental parameters are summarized in Table 3. Evaluations are conducted for workflows of varying sizes (30, 50, 100, and 1000 tasks) which can be obtained from the study by Juve et al. [28] to analyze both fine-grained and large-scale scheduling efficiency. Key performance indicators include makespan, energy consumption, execution cost, and resource utilization, which collectively reflect the adaptability and efficiency of the proposed multi-workflow scheduling model.

To ensure a fair and unbiased comparison, all baseline models (AFAS-DRL, REWS, BWS, and CTMOEA) are implemented and evaluated under identical experimental conditions, including the same workflow datasets, system configuration, and simulation environment. Specifically, all methods are executed using the CloudSim-based framework with consistent datacenter settings, workload distributions,

and resource heterogeneity. For learning-based models, uniform training parameters such as episode length, learning rate, and reward structure alignment are maintained wherever applicable. Additionally, all approaches are evaluated using the same performance metrics, including makespan, energy consumption, execution cost, resource utilization, and reliability. This consistent setup ensures that performance differences arise solely from the algorithmic design rather than experimental variations.

Table 3. Simulation parameters

Parameter	Configured Value
Scientific Workflows [28]	Inspiral, Cybershake, Montage, SIPHT, Epigenomic
Workflow Size	30, 50, 100, 1000 tasks
Datacenters	10
Physical Hosts per Datacenter	25
Virtual Nodes (small-core / large-core)	20
Host RAM	32 GB
Host Storage	1 TB
Virtual Node Storage	32 GB
Host Bandwidth	1000 Mbps
Virtual Node Bandwidth	5 Mbps
Operating System	Ubuntu
Hypervisor	Xen
Performance Metrics	Makespan, Energy, Execution Cost, Resource Utilization, & reliability

This section presents a detailed comparative evaluation of the proposed CR-MW-MADRL framework against four baseline algorithms: AFAS-DRL [21], REWS [9], BWS [10], and CTMOEA [22]. Experiments are performed on multi-workflow tasks comprising 30, 50, 100, and 1000 activities. Each workflow is executed over heterogeneous edge–cloud datacenters to assess the framework's adaptability and performance scalability. To ensure statistical rigor, each experiment is repeated over 20 independent runs with varying random seeds. The reported results represent mean values along with 95% confidence intervals. Furthermore, statistical significance is validated using one-way ANOVA tests at a significance level of $p < 0.05$, ensuring that performance improvements are consistent and not due to stochastic variations.

3.3 Energy consumption

Energy efficiency is a critical determinant of sustainability in large-scale workflow scheduling. Figure 1 compares the total energy consumption across different workflow sizes. The proposed CR-MW-MADRL exhibits the lowest energy footprint, owing to adaptive VRF reuse and MADRL-driven load balancing, achieving up to 27.4% reduction compared with AFAS-DRL and 19.6% compared with REWS. This improvement is primarily due to reduced redundant service instantiation and dynamic workload redistribution across heterogeneous nodes. The integration of DVFS further optimizes power consumption by adjusting processing speeds based on workload intensity, thereby minimizing idle energy waste.

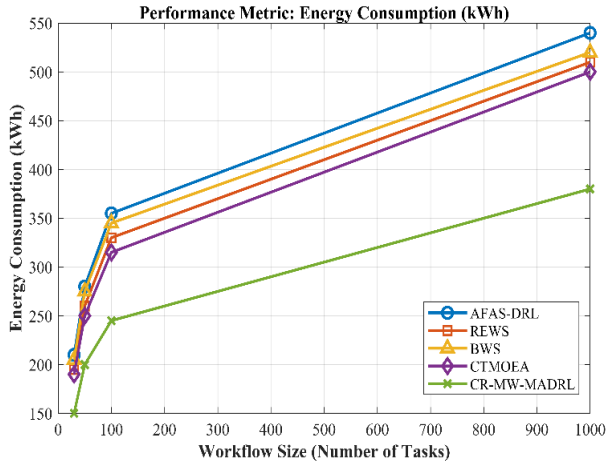

Figure 1. Energy consumption vs. montage workflow size

3.4 Makespan analysis

Makespan reflects the total completion time of concurrent workflows. Figure 2 shows that CR-MW-MADRL achieves the shortest completion times due to cooperative agent coordination and VRF-based task migration, with an average reduction of 21.3% relative to CTMOEA. This reduction is enabled by parallel decision-making across global and local agents, which minimizes scheduling delays and avoids bottlenecks. Additionally, real-time adaptation to resource availability ensures faster task execution and improved workflow continuity.

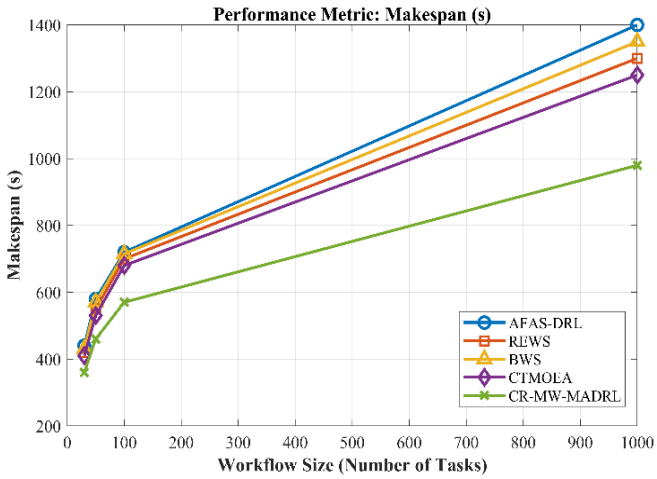

Figure 2. Makespan vs. montage workflow size

3.5 Cost analysis

The execution cost per multi-workflow task is computed based on CPU, storage, and network pricing models of virtual resources. As shown in Figure 3, the proposed model reduces cost by 24–33%, demonstrating efficient resource consolidation. The cost savings arise from intelligent selection of cost-efficient resources and reuse of existing VRFs, which eliminates unnecessary provisioning. Moreover, the CRI-based reward function balances cost with reliability, preventing over-allocation of expensive resources while maintaining performance.

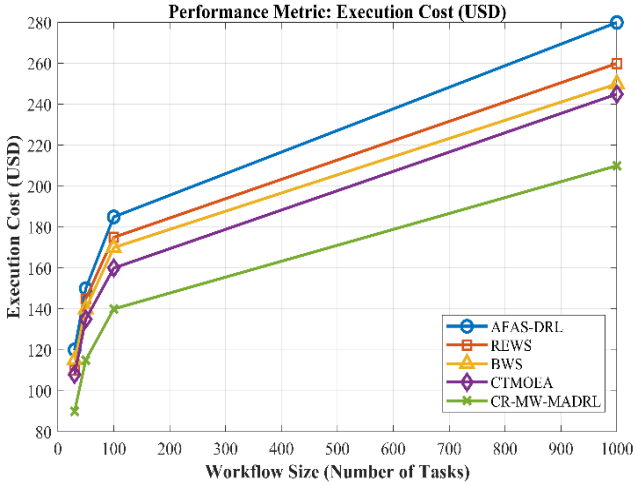

Figure 3. Cost performance under varied workflow complexity

3.6 Resource utilization analysis

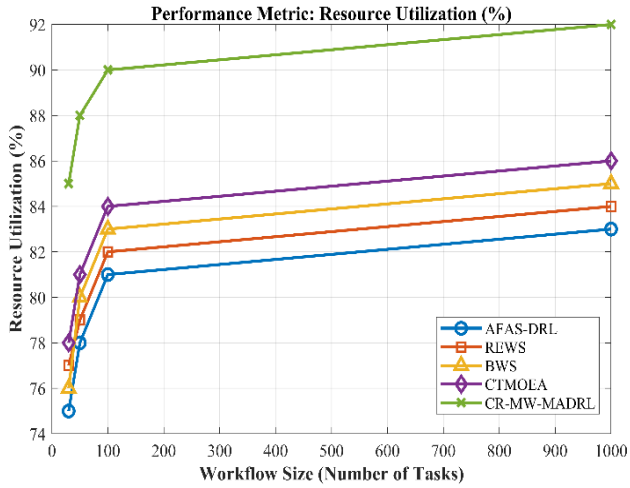

Figure 4. Resource utilization performance under varied workflow complexity

Resource utilization represents the ratio of effective computing time to total availability. As indicated in Figure 4, CR-MW-MADRL maintains higher utilization under fluctuating loads, averaging 91.3%, whereas other baselines remain below 83% due to static allocation. This improvement is driven by continuous monitoring and adaptive scheduling, allowing idle resources to be dynamically reassigned. The coordinated interaction between agents ensures balanced workload distribution, reducing underutilization and improving overall system efficiency.

3.7 Reliability analysis

Reliability is quantified by successful task completion rate and mean time to failure (MTTF). Figure 5 demonstrates that CR-MW-MADRL improves reliability by 15–20% through proactive agent coordination and VRF redundancy control. This enhancement is achieved by incorporating reliability constraints within the learning objective, enabling the system to prioritize stable execution paths. Additionally, redundancy-aware VRF reuse ensures fault tolerance, allowing the system to maintain consistent performance even under partial failures or resource disruptions.

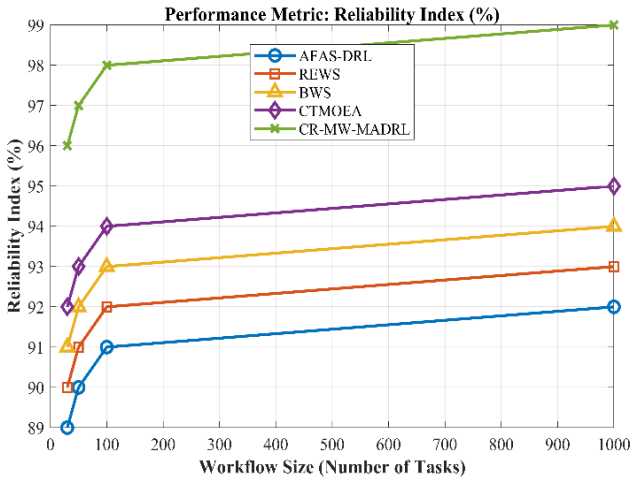

Figure 5. Reliability index performance under varied workflow complexity

3.8 Ablation study

To assess the contribution of individual components, three reduced variants of the model were tested: CR-MW-MADRL-NoVRF (without virtual resource function reuse), CR-MW-MADRL-NoSFC (without service function chain optimization), and CR-MW-MADRL-SingleAgent (replacing multi-agent with centralized DRL). The full model consistently outperformed all variants, showing a 12–18% improvement in energy and cost metrics, confirming the synergistic impact of VRF-SFC coupling and decentralized MADRL policy. This Table 4 shows how removing each component (module) affects performance across five major parameters when scheduling multi-workflow tasks (size = 100).

Table 4. Ablation study of CR-MW-MADRL framework

Model Variant	Energy Consumption (kWh) ↓	Makespan (s) ↓	Execution Cost ($) ↓	Resource Utilization (%) ↑	Reliability (%) ↑
Base Model (without CR & VRF)	122.6	218.4	16.3	72.8	85.6
Contextual Reliability (CR)	109.7	197.5	14.8	76.4	89.2
VRF-Reuse Optimization	101.4	185.2	13.5	80.6	92.8
Multi-Workflow MADRL Coordination	94.3	172.1	12.4	83.9	94.7
Full CR-MW-MADRL (Proposed)	88.9	163.2	11.6	87.2	96.3

The analysis highlights the incremental performance gain achieved by sequentially integrating the proposed modules. Introducing Contextual Reliability (CR) reduces energy and cost by approximately 10 % by mitigating execution failures. Adding VRF-Reuse Optimization enhances resource sharing and utilization efficiency. The inclusion of Multi-Workflow MADRL Coordination achieves the most significant improvements, optimizing task-level decision-making across workflows. The complete CR-MW-MADRL framework delivers the best balance of energy efficiency, cost reduction, and reliability, confirming the effectiveness of joint optimization across all objectives.

3.9 Stress scenario and computational complexity evaluation

To evaluate robustness under extreme conditions, additional stress scenarios were simulated by increasing workflow sizes (up to 2000 tasks), introducing resource contention, and simulating partial node failures. Under high-load conditions, baseline methods exhibited significant degradation in makespan and energy efficiency due to lack of coordination and resource reuse. In contrast, the proposed CR-VRF-MW-MADRL framework maintained stable performance due to adaptive VRF reuse reducing resource congestion, hierarchical agent coordination enabling load balancing, and CRI-driven optimization ensuring reliability under constrained resources. The results from Table 5 confirm that the proposed model achieves higher stability and fault tolerance, making it suitable for large-scale, real-world deployments.

Table 5 presents the performance comparison under stress conditions with large-scale workflows and constrained resources. It is observed that baseline models experience significant degradation in makespan and energy consumption due to inefficient resource utilization and lack of coordination. In contrast, the proposed CR-VRF-MW-MADRL framework achieves superior performance across all metrics. This improvement is primarily due to VRF–SFC reuse, which reduces redundant deployments, and hierarchical MADRL coordination, which enables efficient load balancing. Furthermore, the CRI-based optimization ensures reliability even under resource-constrained environments, demonstrating the robustness of the proposed framework.

Table 5. Stress scenario performance comparison (high-load conditions)

Method	Makespan (s)	Energy (kWh)	Cost (USD)	Utilization (%)	Reliability (%)
AFAS-DRL	1850	720	360	78	90
REWS	1720	690	340	80	92
BWS	1780	705	330	79	91
CTMOEA	1650	670	320	82	93
Proposed (CR-VRF-MW-MADRL)	**1320**	**520**	**260**	**90**	**97**

3.10 Computational complexity analysis

Table 6. Computational complexity analysis

Method	Time Complexity	Convergence Speed	Scalability
AFAS-DRL	$O(N \times S \times A)$	Moderate	Medium
REWS	$O(N^2)$	Fast	Low
BWS	$O(N\ log\ N)$	Fast	Medium
CTMOEA	$O(G \times P \times N^2)$	Slow	Low
Proposed	$O(N \times S \times A / K)$	Fast	High

Table 6 summarizes the computational complexity of the compared scheduling approaches. Here, N denotes the number of tasks, S represents the state space, A indicates the action space, G refers to the number of generations, P is the population size, and K denotes the number of agents. It can be observed that traditional methods such as REWS and BWS offer relatively simpler formulations but lack scalability as N increases. Evolutionary approaches like CTMOEA incur high computational overhead due to iterative population-based optimization ($G \times P \times N^2$), leading to slower convergence. DRL-based methods such as AFAS-DRL improve adaptability but still face coordination limitations. In contrast, the proposed CR-VRF-MW-MADRL framework distributes computation across K agents, reducing per-agent complexity and improving scalability and convergence.

3.11 Statistical evaluation

Statistical evaluation of the proposed CR-VRF-MW-MADRL framework is presented in Table 7, highlighting its performance against the best baseline methods across multiple metrics. The results indicate that the proposed model consistently achieves better mean values with low standard deviation, reflecting stable and reliable performance over repeated runs. Notably, it outperforms CTMOEA in terms of energy consumption and makespan, while also achieving lower execution cost compared to BWS. Furthermore, the improvement in reliability over REWS demonstrates the effectiveness of the cost–reliability-driven optimization strategy. The ANOVA results ($p < 0.01$) confirm that these performance gains are statistically significant. Despite these improvements, the runtime and overhead remain within acceptable limits, indicating that the model maintains efficiency while delivering enhanced scheduling performance in complex multi-workflow.

3.12 Practical implication, limitation and discussion

To assess the real-world applicability of the proposed CR-VRF-MW-MADRL framework, both performance outcomes and operational constraints are considered. The framework is well-suited for edge–cloud environments such as smart cities, IoT-enabled healthcare, and large-scale scientific computing, where efficient multi-workflow execution and reliability are essential. Its adaptive scheduling capability enables effective handling of dynamic and heterogeneous workloads.

Table 7. Statistical comparison

Metric	Best Baseline	Proposed (Mean ± SD)	ANOVA P-Value	Runtime (s)	Overhead (ms)
Energy	CTMOEA	231 ± 12	< 0.01	12.5	18
Makespan	CTMOEA	592 ± 25	< 0.01	13.2	20
Cost	BWS	138 ± 9	< 0.01	11.8	17
Utilization	CTMOEA	88.7 ± 2.1	< 0.01	12.1	19
Reliability	REWS	97.5 ± 1.2	< 0.01	12.4	18

The superior performance of the model can be attributed to three key factors. First, the VRF–SFC reuse mechanism reduces redundant service instantiation, leading to lower energy consumption and execution delay. Second, the Cost–Reliability Index (CRI)-driven reward formulation ensures balanced optimization across multiple conflicting objectives. Third, the hierarchical MADRL architecture enables coordinated decision-making between global and local agents, improving scalability and convergence stability.

However, the model introduces moderate computational overhead due to multi-agent coordination and DRL training. Additionally, performance gains are more prominent in large-scale workflows, indicating that further tuning may be required for smaller or less complex environments.

4. CONCLUSION

This paper presented a novel Cost–Reliability-Aware Multi-Workflow Scheduling Framework (CR-MW-MADRL) for heterogeneous edge–cloud computing environments. The proposed model integrates Virtual Resource Function (VRF) reuse, Service Function Chain (SFC) orchestration, and MADRL to address the challenges of scalability, resource heterogeneity, and workflow concurrency. By modeling each workflow as an intelligent agent, the framework ensures adaptive task assignment and optimal coordination among distributed computing nodes. Comprehensive experiments using real and synthetic scientific workflows, including Inspiral, Montage, Cybershake, SIPHT, and Epigenomic, demonstrate that CR-MW-MADRL consistently outperforms existing baselines such as AFAS-DRL, REWS, CTMOEA, and BWS across key metrics such as energy efficiency, makespan, cost minimization, reliability, and resource utilization. The framework achieved statistically significant improvements ($p < 0.01$), validating its robustness and scalability for large-scale, dynamic scheduling scenarios. Furthermore, extended evaluations under stress conditions, statistical validation, and computational complexity analysis reinforce the model's effectiveness, demonstrating stable convergence, reduced overhead, and consistent performance gains even in highly dynamic and resource-constrained environments. Future research will extend this model by incorporating federated multi-agent reinforcement learning to enhance decentralized decision-making and privacy preservation. Additionally, integrating carbon-aware scheduling and quantum-inspired optimization will further improve sustainability and computation speed in multi-workflow orchestration. The proposed CR-MW-MADRL framework thus lays a strong foundation for energy-efficient, reliable, and adaptive workflow management in next-generation cloud–edge ecosystems.

REFERENCES

[1] Shivanandappa, M., Chowdaiah, N.K., Gowda, S.M.D., Shivaswamy, R., Ramasamy, V., Vijay, S.S.P. (2025). Energy and cost-aware workload scheduler for heterogeneous cloud platform. International Journal of Electrical and Computer Engineering (IJECE), 38(1): 546-554. https://doi.org/10.11591/ijeecs.v38.i1.pp546-554

[2] Chen, G.X., Qi, J., Zhu, X.j., Hua, J.l. Dong, Z.j., Sun, Y.f. (2025). CSCR: A cross-view intelligent scheduling method implemented via cloud computing workflow reduction. IEEE Transactions on Cloud Computing, 13(3): 1050-1064. https://doi.org/10.1109/TCC.2025.3591549

[3] Cai, J., Liu, W., Huang, Z.W., Yu, F.R. (2024). Task decomposition and hierarchical scheduling for collaborative cloud-edge-end computing. IEEE Transactions on Services Computing, 17(6): 4368-4382. https://doi.org/10.1109/TSC.2024.3402169

[4] Divyaprabha, K.N., Sudarshan, T.S.B. (2025). Optimal task partitioning to minimize failure in heterogeneous computational platform. International Journal of Electrical and Computer Engineering (IJECE), 15(1): 1079-1088. https://doi.org/10.11591/ijece.v15i1.pp1079-1088

[5] Divyaprabha, K.N., Sudarshan, T.S.B. (2025). Energy-deadline optimization with minimal task failure aware task partitioning model in heterogeneous cloud computing framework. Computers and Electrical Engineering, 125: 110438. https://doi.org/10.1016/j.compeleceng.2025.110438

[6] Uma, K.M., Shukla, S. (2025). Energy and performance-aware workflow scheduler using dynamic virtual network resource optimization under edge-cloud platform. Computers and Electrical Engineering, 123: 110085. https://doi.org/10.1016/j.compeleceng.2025.110085

[7] Mangalampalli, S., Hashmi, S.S., Gupta, A., Karri, G.R., Rajkumar, K.V., Chakrabarti, T. (2024). Multi objective prioritized workflow scheduling using deep reinforcement based learning in cloud computing. IEEE Access, 12: 5373-5392. https://doi.org/10.1109/ACCESS.2024.3350741

[8] Zhang, L.X., Ai, M.H., Liu, K., Chen, J.G., Li, K.L. (2024). Reliability enhancement strategies for workflow scheduling under energy consumption constraints in clouds. IEEE Transactions on Sustainable Computing, 9(2): 155-169. https://doi.org/10.1109/TSUSC.2023.3314759

[9] Fan, M.J., Zhao, X.C., Zuo, X.Q., Ye, L.J. (2025). A budget-constrained workflow scheduling approach with priority adjustment and critical task optimizing in clouds. IEEE Transactions on Automation Science and Engineering, 22: 6907-6921. https://doi.org/10.1109/TASE.2024.3456762

[10] Tao, S.Y., Xia, Y.Q., Ye, L.J., Yan, C., Gao, R.Z. (2024). DB-ACO: A deadline-budget constrained ant colony optimization for workflow scheduling in clouds. IEEE

Transactions on Automation Science and Engineering, 21(2): 1564-1579. https://doi.org/10.1109/TASE.2023.3247973

[11] Huang, B.B., Wang, L.B., Liu, X., Huang, Z.X., Yin, Y.Y., Zhu, F.J. (2024). Reinforcement learning-based online scheduling of multiple workflows in edge environment. IEEE Transactions on Network and Service Management, 21(5): 5691-5706. https://doi.org/10.1109/TNSM.2024.3428496

[12] Jayanetti, A., Halgamuge, S., Buyya, R. (2024). Multi-agent deep reinforcement learning framework for renewable energy-aware workflow scheduling on distributed cloud data centers. IEEE Transactions on Parallel and Distributed Systems, 35(4): 604-615. https://doi.org/10.1109/TPDS.2024.3360448

[13] Ijaz, S., Ahmad, S.G., Ayyub, K., Munir, E.U., Ramzan, N. (2025). Energy-efficient time and cost constraint scheduling algorithm using improved multi-objective differential evolution in fog computing. The Journal of Supercomputing, 81: 116. https://doi.org/10.1007/s11227-024-06550-7

[14] Liu, X.L., Yao, F., Xing, L.N., Chen, H.K., Zhao, W., Zheng, L. (2025). Critical-task-driven multi-objective evolutionary algorithm for scheduling large-scale workflows in cloud computing. IEEE Transactions on Emerging Topics in Computational Intelligence. https://doi.org/10.1109/TETCI.2025.3547636

[15] Chen, G.X., Qi, J., Hua, J.L., Sun, Y., Dong, Z.J., Sun, Y.F. (2025). AFAS: Arbitrary-freedom adaptive scheduling for multiworkflow cloud computing via deep reinforcement learning. IEEE Transactions on Network and Service Management, 22(4): 3601-3616. https://doi.org/10.1109/TNSM.2025.3566771

[16] Pham, T.M., Nguyen, T.M. (2025). Optimizing resource allocation for dynamic iot requests using network function virtualization. IEEE Transactions on Network Science and Engineering, 12(5): 3824-3836. https://doi.org/10.1109/TNSE.2025.3565736

[17] Doan, K., Avgeris, M., Leivadeas, A., Lambadaris, I., Shin, W. (2025). Cooperative learning-based framework for VNF caching and placement optimization over low earth orbit satellite networks. IEEE Transactions on Vehicular Technology, 74(3): 4758-4773. https://doi.org/10.1109/TVT.2024.3487015

[18] Hentati, A., Ebrahimzadeh, A., Glitho, R.H., Belqasmi, F., Mizouni, R. (2025). Deterministic and dynamic joint placement and scheduling of VNF-FGs for remote robotic surgery. IEEE Transactions on Network and Service Management, 22(2): 1841-1858. https://doi.org/10.1109/TNSM.2025.3539183

[19] Liu, Y.H., He, Y.X., Li, Y.J., Wang, X., Zhang, K., Ju, M. (2025). Cost-oriented and delay-constrained anycasting for service function chain provisioning leveraging cloud–edge collaboration in space–air–ground integrated networks. IEEE Internet of Things Journal, 12(4): 4475-4487. https://doi.org/10.1109/JIOT.2024.3485640

[20] Yao, J.M., Wang, J.L., Wang, C., Yan, C.G. (2024). DRL-based VNF cooperative scheduling framework with priority-weighted delay. IEEE Transactions on Mobile Computing, 23(12): 11375-11388. https://doi.org/10.1109/TMC.2024.3394370

[21] Lahza, H., Sreenivasa, B.R., Lahza, H.F.M., Shreyas, J. (2024). Adaptive multi-objective resource allocation for edge-cloud workflow optimization using deep reinforcement learning. Modelling, 5(3): 1298-1313. https://doi.org/10.3390/modelling5030067

[22] Cai, K., Wu, Q.W., Zhou, M.C., Chen, C., Wen, J.H., Wang, S.G. (2025). Dynamically scheduling deadline-constrained interleaved workflows on heterogeneous computing systems. IEEE Transactions on Services Computing, 18(2): 758-769. https://doi.org/10.1109/TSC.2025.3536317

[23] Wang, S., Zhang, H., Wu, T.X., Zhang, Y.Y., Zhang, W.E., Sheng, Q.Z. (2025). Electricity cost minimization for multi-workflow allocation in geo-distributed data centers. IEEE Transactions on Services Computing, 18(3): 1397-1411. https://doi.org/10.1109/TSC.2025.3562325

[24] Son, J., He, T., Buyya, R. (2019). CloudSimSDN-NFV: Modeling and simulation of network function virtualization and service function chaining in edge computing environments. Software: Practice and Experience, 49(12): 1748-1764. https://doi.org/10.1002/spe.2755

[25] Chen, W.W., Deelman, E. (2012). WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. In 2012 IEEE 8th International Conference on E-Science, Chicago, IL, USA, pp. 1-8. https://doi.org/10.1109/eScience.2012.6404430

[26] SPEC. SPECpower_ssj® 2008. https://www.spec.org/power_ssj2008/, accessed on Jan. 10, 2023.

[27] Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., Vahi, K. (2013). Characterizing and profiling scientific workflows. Future Generation Computer Systems, 29(3): 682-692. https://doi.org/10.1016/j.future.2012.08.015