SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V13I8P101

Volume 13 Issue 8, 1-25, August 2026

Original Article

HRL-TaskOpt: A Hierarchical Reinforcement Learning-Based Task Scheduling Framework for Multi-Cloud and Hybrid Environments

Krishna Rao Patwari[1], Raghvendra Kumar[2], J.S.V.R.S. Sastry[3]

[1,2]*Department of CSE, Gandhi Institute of Engineering and Technology University (GIETU), Gunupur, Odisha,India.*
[3]*Department of CSE, Gandhi Institute of Technology and Management (GITAM) University, Rudraram, Hyderabad, Telangana, India.*

[1]*Corresponding Author : krishnarao.patwari@giet.edu*

Received: 06 January 2026 Revised: 20 February 2026 Accepted: 24 July 2026 Published: 31 August 2026

Abstract - *Cloud computing has emerged as a new paradigm, which entrusts task scheduling to ensure the satisfaction of stringent constraints on latency, energy, and resources for sustainably running real-time applications. State-of-the-art natural DRL-based scheduling solutions mainly rely heavily on DRL techniques and are either limited in scalability, adaptivity, or generality of workloads/infrastructures. State-of-the-art flat methods, such as DQN and actor–critic models, are not sufficiently effective at high levels of decision complexity and are not robust against varying system loads and task priorities. In this paper, we present HRL-TaskOpt, a novel Hierarchical Reinforcement Learning-based task scheduling framework that combines high-level global task offloading with millisecond-granularity local scheduling policies in an edge–cloud scenario. In the proposed framework, there are two levels of agents: a high-level policy that utilizes Proximal Policy Optimization (PPO) to select the optimal execution tiers (edge or cloud), and a low-level policy based on Deep Q-Networks (DQN) to manage scheduling within nodes (edge or cloud). Such decomposition enables HRL-TaskOpt to efficiently accommodate heterogeneous workloads and adapt to dynamically changing infrastructure. We use synthetically generated workloads that reflect the characteristics of real-world applications to demonstrate the effectiveness of our model and compare it with state-of-the-art models such as SA-DQN, DRL-DO, and GD-DRL. We experimentally demonstrate that HRL-TaskOpt achieves a time reduction of up to 21.4% for task completion (2,000-task workload vs. SA-DQN), an energy efficiency improvement of up to 20.3% (10,000-task workload vs. GD-DRL), and a task success rate improvement of up to 13.3% (averaged across baselines at the 2,000-task workload), compared to these models. In addition, robustness to resource failures and sensitivity to task-type diversity, demonstrated in images, validate the real-life usability of the model. HRL-TaskOpt offers a scalable and intelligent solution for adaptive task scheduling, making it an appealing candidate for deployment in next-generation edge–cloud continuum systems.*

Keywords - *Hierarchical Reinforcement Learning, Task scheduling, Edge–cloud continuum, Deep Q-Network, Proximal Policy Optimization.*

1. Introduction

With the explosive growth of Internet of Things (IoT) applications and latency-sensitive workloads, task scheduling in edge–cloud continuum environments has become more critical than ever. Abstract: Traditional cloud-centric computing architectures are often inadequate in fulfilling stringent latency, bandwidth, and energy-efficiency requirements, especially in real-time settings. Edge computing is a decentralized computing model that operates with high proximity to the data generator, providing greater robustness compared to the cloud paradigm. However, it poses challenges in optimal task allocation due to its limited resources and heterogeneous nature across devices. This led to the idea of hybrid edge–cloud models, which have been dubbed a potential solution where intelligent task scheduling frameworks are required to manage and dynamically adjust according to workload characteristics and infrastructure constraints. As DRL models learn policies through interaction with the environment, recent literature has examined their potential for scheduling effectiveness. Single-agent DQN and actor–critic methods have achieved some degree of success in resource-constrained environments [1, 2]. Hierarchical Reinforcement Learning (HRL) has gained popularity recently, particularly for its ability to model complex task interactions and reduce learning complexity by decomposing complex decision-making into multi-level policies [3, 4]. Yet, current DRL and HRL-based methods either overlook multi-level cooperation or exhibit poor stability and scalability in

varying load environments, leaving a gap in adaptive and efficient task scheduling approaches.

We introduce HRL-TaskOpt, a new hierarchical reinforcement learning-based framework that addresses these challenges by optimizing task scheduling across edge–cloud infrastructure to achieve simultaneous improvements in latency, energy, and throughput. Specifically, the common goal is to develop a two-level policy framework in which a high-level policy (πa) decides where to offload the task and a low-level policy (πD) optimizes the local scheduling at the execution node.

The main novelties of this work are: 1) combining Proximal Policy Optimization (PPO) and DQN to achieve stable hierarchical learning; 2) workload-aware task profiling that reflects dynamic changes; and 3) robustness analysis in terms of load variability, mix of task types, and resource failures. This work makes the following contributions: it develops a system-level mathematical model, provides an ablation study to assess the influence of each component, and performs a thorough evaluation over a broad range of simulated workloads.

This paper is structured as follows: Section 2 provides a literature review of DRL-based task scheduling and hierarchical models, that we found most relevant. Section 3 describes the mathematical formulations and model training for the HRL-TaskOpt architecture. Section 4 presents experimental results, including comparisons with baselines, robustness evaluation, and ablation. In Section 5, limitations of the current study and its broader implications are discussed, followed by a summary of key findings. In Section 6, we summarize the paper and point to future work in terms of real-world applications and extensions of the model.

2. Related Work

Deep Reinforcement Learning (DRL) methods have dominated research on task scheduling in multi-cloud and edge environments. In [1], Robles-Enciso and Skarmeta introduced an adaptive multi-layer guided RL-based task offloading strategy for dynamic edge conditions. Zhang et al. RL for efficient DAG-based task scheduling in collaborative edge networks [2] Qi et al. [3] used DRL for real-time power grid digital twin scheduling, emphasizing temporal responsiveness. Da Costa et al. Mobility and deadlines have been included into DRL mechanisms for vehicular edge scheduling in [4]. Zhang et al. A hybrid edge-cloud scheduling strategy that targets optimal resource usage is presented in [5]. Gu et al. Optimizing Workflows with DRL and Simulated Annealing. A coordinate-free method to optimize cost from Deep Reinforcement Learning (DRL) has shown earlier success [6]. They proposed various strategies in dynamic task offloading to improve edge-cloud cost efficiency [7]. DDPG has been used for multi-resource scheduling by Qadeer et al. [8]. Nieto et al. [9] studied DRL-based offloading in 5G edge-cloud scenarios. Dreibholz and Mazumdar [10] focused on lightweight scheduling frameworks for edge scenarios.

Jain et al. This work describes a Cybertwin-driven, DRL-based resource allocation scheme for 6G enabled edge environments to demonstrate the improvements of service responsiveness and autonomy [11]. Zheng et al. [12] presented a real-time demand adaptive DRL-based workload scheduling in edge computing. Zhao et al. Q-learning for dynamic task scheduling and low-load intrusion detection, with emphasis on distributed scenarios [13]. Pang et al. In [14], it employed DRL for the minimization of average task processing time in satellite mobile edge systems. Cho et al. An approach to load balancing through RL was proposed in [15], targeting a hierarchical edge cloud and also addressing QoS-aware workload distribution. Qi et al. The authors of [16] proposed a DRL-based task scheduling model to maximise the throughput for edge networks. Panwar and Supriya [17] design an RL-based Proactive Resource Allocation Framework (RLPRAF) to tackle the challenges of provisioning. Baghban et al. Wei et al. [18] leverage actor–critic models for IoT service provisioning on federated edge environments. Wu et al. [19] chose DRL for task migration in satellite-edge computing. Chen et al. In [20], neural networks and heuristics were used for optimal scheduling in hierarchical edge clouds.

Rac et al. [21] study cost-aware service placement and scheduling strategies in the edge–cloud continuum with an economic focus. Zhang et al. [22] has proposed a DRL-driven approach to schedule data-hungry workflows over multiple clouds, maximizing latency and utilization. Tuli et al. [23] proposed an advantage actor critic (A3C) and residual RNN-based scheduler for stochastic edge–cloud situations to achieve better adaptability of tasks. Qi et al. [24] Scalable multi-task DRL scheduling for various autonomous driving workloads Sellami et al. For example, in [25], DRL was combined with the scheduling problem in SDN-enabled IoT networks with a focus on energy efficiency and sustainability. Zhou et al. [26] proposed a two-layer edge-enabled scheduling mechanism on the basis of DRL to support IoE systems. Mangalampalli et al. They were really close to using A3C for multi cloud settings scheduler, where critical tasks that propose an A3C-based multiobjective for the path have been defined [27]. Nashaat et al. Collier et al. introduced a distributed framework for DRL-based edge-cloud offloading [28]. Yuan et al. The work in [29] proposed DRL-based joint DNN partitioning and scheduling in MEC networks. Qu et al. 4.1 [30] Details DMRO is DMRO, a meta-RL offloading framework for dynamic environments.

Nandhakumar et al. They present EdgeAISim, a simulation toolkit enabling performance evaluation of AI (models) in edge environments and encouraging reproducible experiments [31]. Raeisi-Varzaneh et al. An overview of edge

computing resource scheduling, architectures, and emerging trends in [32]. Liu et al. At the same time, [33] has done a lot of work on cross-device coordination based small cell techniques in IIoT systems and proposed a Dr locality-based offloading scheduler for multi-AGV devices in each cell with DRL. Dong et al. [34] presented a cloud–edge workload allocation scheme with high throughput and load balancing. Wen et al. A recent work [35] focuses on scheduling accelerating using DRL to minimize tail latency in edge–cloud jobs. Sellami et al. An energy-aware DRL-based task offloading scheme in SDN-enabled IoT systems was proposed by [36]. Chen et al. [37] Microservice Deployment in Edge–Cloud Environments with RL-Aided Latency-Performance Trade-offs Lu et al. QoS-aware task scheduling for hybrid cloud-edge infrastructures have been proposed in [38]. Pan et al. Foundation work: A decoupling-aware task offloading framework named as a Dependency-Aware DRL-based Offloading Framework in 2023 [39]. Zhao et al. Q-learning was employed for lightweight task scheduling in DIDS-centric edge networks by Liu et al. [13]. Gu et al. [6] proposed an approach for cost-aware workflow scheduling by using simulated annealing-based deep reinforcement learning. Nashaat et al. They proposed a DRL-based distributed offloading framework to dynamically balance the load in edge–cloud environments [28]. Qi et al. [3] focused on temporal scheduling and resilience of DSSE in the context of DRL for real-time scheduling in bundling power grid digital twin systems. Tomar et al. Mirror Descent Policy Optimization [40] theoretically grounds the idea of policy optimization in reinforcement learning. Mnih et al. DRL has its roots in the pioneering work by [41], who introduced the DQN model, which achieved human-level control in complex environments and ultimately led to a proliferation of task scheduling works. Table 1 highlights recent advances in DRL-based task scheduling across cloud and edge environments. Although many studies utilize hybrid algorithms [6], distributed frameworks [28] or real-time schedulers [3], most do not incorporate hierarchical control, end-to-end cross-layer optimization, or scalability under hybrid multi-cloud scenarios. Foundational models, such as DQN [41] and MDPO [40], offer strong baselines but are rarely tailored to complex scheduling scenarios. Existing methods [2, 23, 29] address partial challenges such as workflow dependencies or task partitioning, but fail to unify scheduling at multiple abstraction levels. The proposed HRL-TaskOpt framework addresses these gaps through a hierarchical RL design with context-aware policy decomposition for robust scheduling.

Table 1. Summary of key deep reinforcement learning-based task scheduling approaches and identified research gaps

Author(s) & Year	Methodology / Model Used	Application Domain	Key Contribution	Research Gap Identified
Gu et al. (2023) [6]	DRL + Simulated Annealing	Cloud Workflow Scheduling	Hybrid optimization for cost-aware workflow execution in clouds	Scalability to dynamic multi-cloud environments is not addressed
Nashaat et al. (2024) [28]	Distributed DRL Framework	Edge–Cloud Offloading	Proposes a load-aware distributed offloading mechanism	Does not include hierarchical decision-making or task interdependencies
Qi et al. (2024) [3]	DRL-based Scheduler	Power Grid Digital Twins	Real-time DRL scheduler for cloud-based digital twin tasks	Lacks extension to multi-cloud or federated architectures
Tomar et al. (2020) [40]	Mirror Descent Policy Optimization (MDPO)	RL Algorithmic Optimization	Proposes a theoretically robust policy optimization approach	Not applied or validated in real-time scheduling contexts
Mnih et al. (2015) [41]	Deep Q-Network (DQN)	General DRL / Control Tasks	Foundation for DRL applications with human-level performance	DQN is not designed for hierarchical or multi-agent scheduling scenarios
Zhang et al. (2020) [2]	RL-based DAG Scheduling	Edge Collaboration	Efficient DAG task optimization in collaboration networks	Limited to static environments, not adaptive to mobility
Tuli et al. (2020) [23]	A3C + Residual RNN	Stochastic Edge–Cloud	Adaptive task scheduling under uncertain workload fluctuations	Limited support for prioritization in hierarchical settings
Yuan et al. (2023) [29]	Joint DNN Partition + DRL Scheduler	MEC + Digital Twins	Integrates partitioning with task scheduling in latency-sensitive MEC setups	Lack of generalization across diverse task types and edge configurations

3. Proposed Framework

In this work, a hierarchical reinforcement learning-based task scheduling optimization framework (HRL-TaskOpt) is proposed for edge–cloud environments. To overcome the limitations of flat reinforcement learning, we propose a modular, dual-level policy for task offloading and resource allocation, which leads to a more effective methodology for improving adaptability, scalability, and energy efficiency. Here, we elaborate upon the architecture, training procedure, and technical implementation of the scheduling model.

3.1. Overview of HRL-TaskOpt Framework

In this paper, we present an innovative hierarchical reinforcement learning-based framework, called HRL-TaskOpt (Figure 1), which focuses on task scheduling optimization for multi-cloud and hybrid cloud systems. Addressing the constraints of flat reinforcement learning and most heuristic-based models, the decision framework proposed in this paper utilizes a two-level decision architecture to improve adaptability, scalability and resource efficiency.

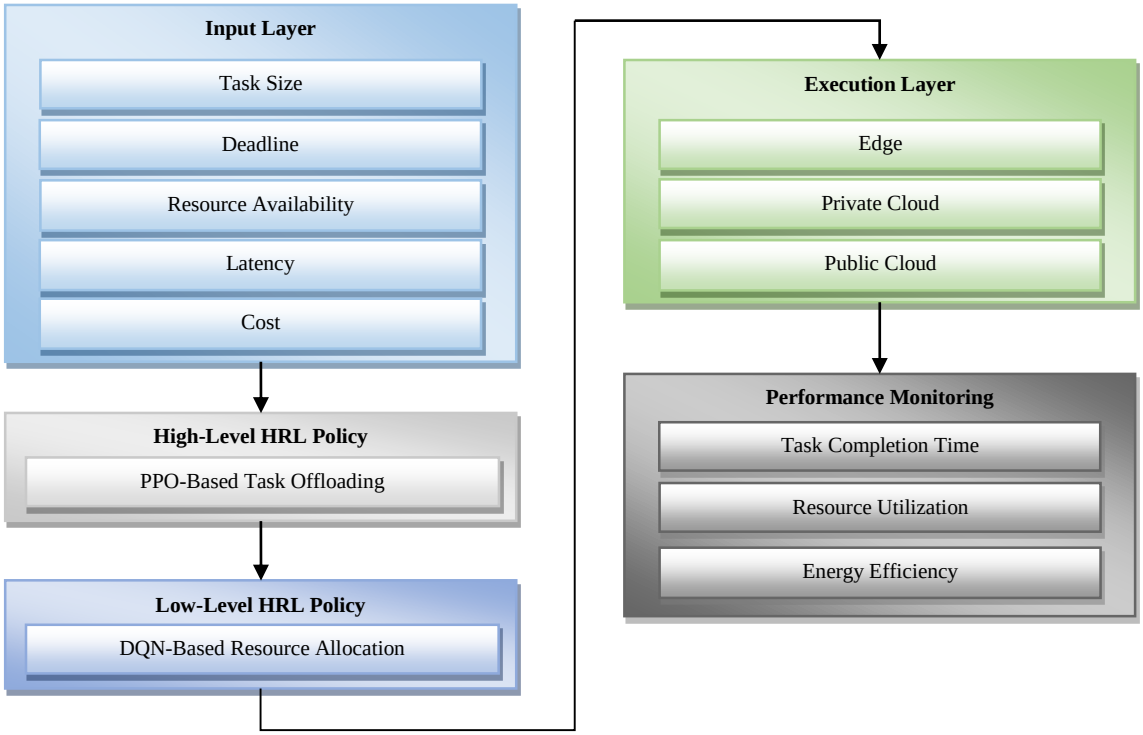

Fig. 1 HRL-TaskOpt framework for hierarchical task scheduling in multi-cloud and hybrid environments

You are guided by two packages of nested agents on a hierarchical architecture that is at the heart of the framework. The high-level agent based on Proximal Policy Optimization (PPO) makes global system observations (i.e., characteristics of the incoming workload, task deadlines and remaining resources in current nodes) and selects an execution location for a task: edge, private cloud or public cloud or hybrid nodes. The low-level agent, which is based on the agent powered by Deep Q-Networks (DQN), decides what type and how much resource (CPU, memory or bandwidth) to allocate for performing tasks in this environment.HRL-TaskOpt begins with task analysis, which takes input features of tasks: size, priority, latency sensitivities and resource requirements. The high-level PPO agent accepts the current state of the system and chooses the environment to minimize the reward function with respect to low latency, low energy consumption, and deadlines. When, however, the task has been assigned to a specific environment chord is responsible for low-level DQN that handles resource allocation and distribution in granular ways. So that's does the precise resource allocation exploration based on maximizing system throughput while minimizing delay and energy during task execution. This decomposition allows a high-level agent to apply globally, whilst enabling the low-level agent to locally optimize.

To realize robustness and generalizability, this framework has been trained on a series of simulated workloads ranging from simple to complex. Both agents learn good scheduling rules based on rewards from the environment, and improve their policies through repeated interactions with it.HRL-TaskOpt addresses hierarchical decision-making, reinforcement learning optimization, and an environment involving dynamic clouds, achieving remarkable improvements in task completion time, energy efficiency,

resource utilization, and overall scheduling rate. This makes it especially valuable for scenarios that require low latency, heterogeneous, and dynamic computing. Table 2 summarizes relevant notations and definitions that will be used throughout the mathematical modeling and algorithm design of the HRL-TaskOpt framework.

Table 2. Notations and descriptions used in the HRL-TaskOpt framework

Notation	Description
S	State space in the Markov Decision Process (MDP)
A	Action space in the Markov Decision Process (MDP)
P	Transition probability between states
R	Reward function
s_t	State of the environment at time step t
a_h	High-level action for task offloading
A_h	Action space for high-level policy (Edge, Private Cloud, Public Cloud, Hybrid)
θ_h	Policy parameters for high-level reinforcement learning
$J_h(\theta_h)$	Expected cumulative reward for high-level policy
γ	Discount factor for future rewards
$R_h(s_t, a_h)$	Immediate reward for high-level task scheduling
a_l	Low-level action for resource allocation
A_l	Action space for low-level policy (CPU, Memory, Bandwidth allocation)
θ_l	Policy parameters for low-level reinforcement learning
$R_l(s_t, a_l)$	Immediate reward for resource allocation
$U(s_t, a_l)$	Resource utilization efficiency
$E(s_t, a_l)$	Energy consumption
$L(s_t, a_l)$	Task execution latency
α, β, δ	Hyperparameters controlling the trade-off between utilization, energy, and latency
$L^{PPO}(\theta_h)$	Proximal Policy Optimization (PPO) loss function
$Q(s_t, a_l)$	Q-value function estimating future rewards in DQN
ϵ	Clipping factor for PPO stability

3.2 Proposed Deep Learning Model

We then present the HRL-TaskOpt model for hierarchical reinforcement learning, a novel architecture designed to schedule tasks in multi-cloud and hybrid environments optimally. The model comprises two learning agents responsible for learning procedures: a high-level decision-making agent utilizing Proximal Policy Optimization (PPO) and a low-level resource allocation agent employing Deep Q-Networks (DQN) to discover fine-grained implementations. At the high level, the framework consists of two layers, which allows the framework to handle complex task scheduling as it decouples the task offloading and resource allocation concerns.

HRL-TaskOpt is a high-level reinforcement learning model that receives a constant stream of incoming tasks characterized by different features such as size, priority, deadline, and necessary resources (Figure 2). These features combine with environment states (e.g., resource availability at edge/cloud nodes, network status, energy profiles) as input state space to the high-level PPO agent. This information is provided to the PPO agent, which utilizes a policy determined by the current Q-function (that maximizes long-term scheduling rewards) in selecting an execution domain (Edge, Private Cloud, Public Cloud or Hybrid Node).

These rewards are set up to punish deadline misses, long latencies and energy consumption.After the domain of execution is chosen, the low-level DQN agent gets executed. It relies on local state features of the selected domain (e.g., CPU availability, memory load, bandwidth, and queue length) to estimate the best resource allocation action. The low-level agent uses Q-learning mechanisms by applying the Bellman equation to ensure that it maximizes local utility through locally optimal task execution according to available resources.

We motif various workloads, altering their strength and diversity of the method. Both the PPO and DQN agents store transitions in independent replay buffers, which are then used to update the policy every couple of epochs. PPO agent benefits from a clipped surrogate objective loss to have a

stable policy update. On the other hand, the DQN agent learns its expected Q-values via the temporal difference learning process. Combining high-level and low-level agents are combined as the HRL-TaskOpt model, thus provide a scalable, adaptive task scheduling with energy efficient solution. With this mechanism, it is able to work across both workloads and resource dynamics in order to learn context-aware scheduling behaviors, which means that it can be made real-time applicable on the cloud-edge ecosystems. Since the high-level and the low-level Table 1 highlights recent advances in DRL-based task scheduling across cloud and edge environments. Although many studies utilize hybrid algorithms [6], distributed frameworks [28] or real-time schedulers [3], most do not incorporate hierarchical control, end-to-end cross-layer optimization, or scalability under hybrid multi-cloud scenarios. Foundational models, such as DQN [41] and MDPO [40], offer strong baselines but are rarely tailored to complex scheduling scenarios. Existing methods [2, 23, 29] address partial challenges such as workflow dependencies or task partitioning, but fail to unify scheduling at multiple abstraction levels. The proposed HRL-TaskOpt framework addresses these gaps through a hierarchical RL design with context-aware policy decomposition for robust scheduling.

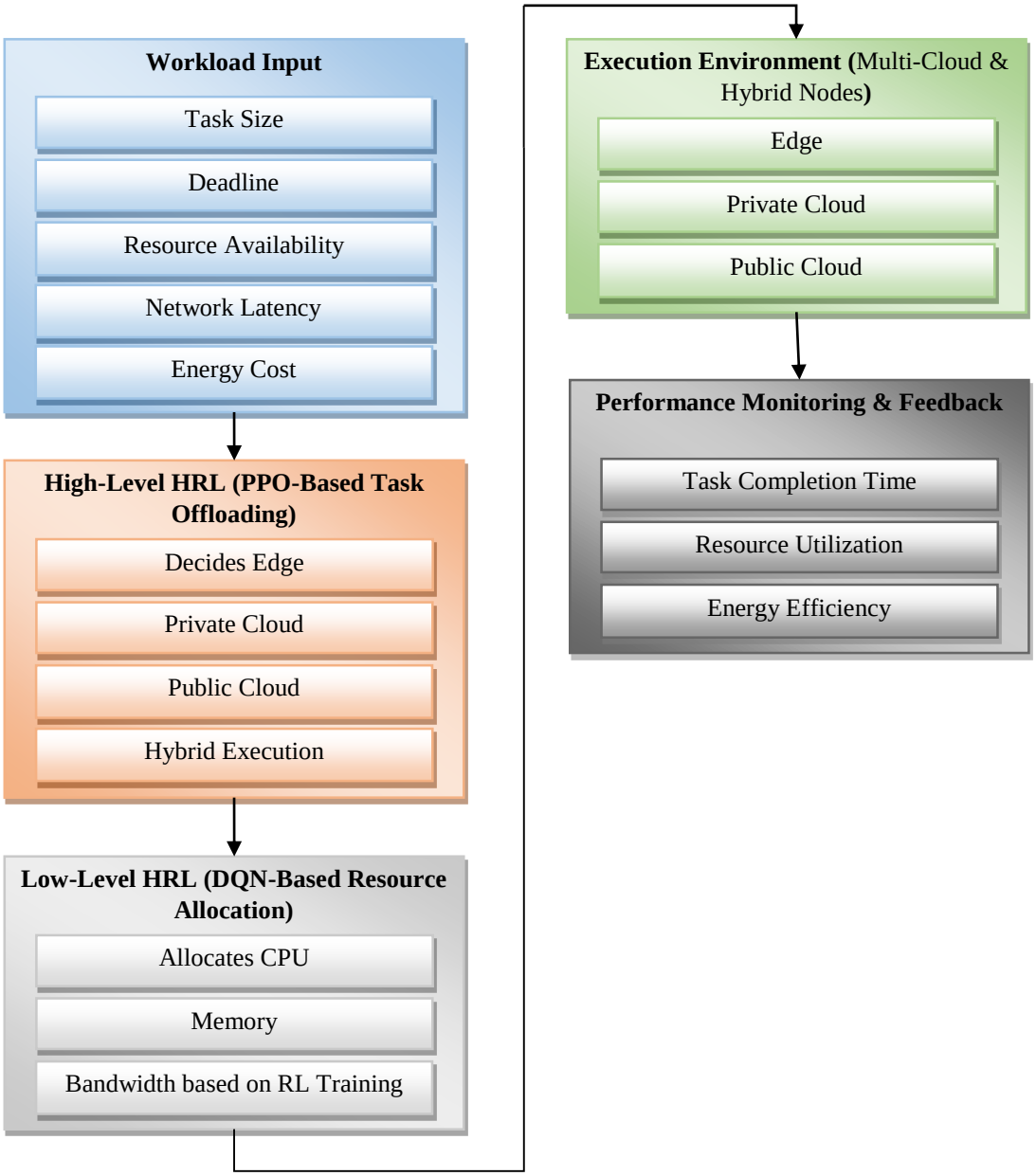

Fig. 2 Architecture of the HRL-TaskOpt model with PPO-based task offloading and DQN-based resource allocation

Table 3. Model parameters and configuration details for the HRL-TaskOpt framework

Parameter	Description	Value / Range
State Space (High-Level)	Task features + global system status	12-dimensional vector
Action Space (High-Level)	Execution domains (Edge, Private Cloud, Public Cloud, Hybrid)	Discrete (4)
Policy Network (PPO)	MLP with two hidden layers (ReLU activations)	[128, 64] units
Optimizer (PPO)	Adam optimizer for policy gradient updates	Learning rate = 0.0003
Discount Factor (γ)	Reward discounting for long-term optimization	0.99
Clip Ratio (PPO)	PPO clipping threshold to stabilize updates	0.2
Replay Buffer (High-Level)	Stores state–action–reward transitions	Size = 10,000
State Space (Low-Level)	Resource stats of the selected environment	8-dimensional vector
Action Space (Low-Level)	Resource configurations (CPU, memory, bandwidth levels)	Discrete (9)
Q-Network (DQN)	Two hidden layers for state–action value estimation	[64, 64] units
Optimizer (DQN)	Adam optimizer for Q-network updates	Learning rate = 0.001
Replay Buffer (Low-Level)	Stores environment–action–reward transitions	Size = 20,000
Target Update Frequency	Steps after which the DQN target network is updated	Every 100 steps
Exploration Strategy	Epsilon-greedy (ε decay)	$\varepsilon = 1 \rightarrow 0.05$ (linear decay)
Batch Size	Number of samples per training update	64
Training Episodes	Total number of training iterations	5,000

The parameters and configurations used under the HRL-TaskOpt framework are summarized in Table 3. The framework features a two-level hierarchy, comprising a high-level Proven Policy Optimization (PPO) agent that determines which task to offload to which execution environment, and a low-level Deep Q-Network (DQN) agent that provides fine-grained resource allocation within the chosen execution environment.

In the high-level PPO agent, the state space is represented by a fixed-size feature vector of dimension 12, which includes features related to task metadata, system load, and resource availability. We create four discrete environments for our action space: Edge, Private Cloud, Public Cloud, and Hybrid Node. The structure used for the PPO policy network is a two-layer fully connected neural network with 128 and 64 neurons, utilizing ReLU activation functions. We use Adam optimizer with a learning rate of 0.0003, $\gamma = 0.99$ as for long-term reward estimation BDPP or standard ratings, and 0.2 as clip ratio; The agent. We add them to a replay buffer of size 10,000, as used in the task transitions, and train on that.

The low-level DQN agent operates on an 8-dimensional local state space representing real-time statistics of CPU, memory, bandwidth, and queue occupancy. The action space includes nine discrete configurations for resource allocation. The Q-network consists of two hidden layers, each with 64 neurons. The DQN model is trained using the Adam optimizer with a learning rate of 0.001. A larger replay buffer of size 20,000 is used due to the frequent nature of resource allocation tasks. The target Q-network is updated every 100 steps to stabilize the learning process. An epsilon-greedy exploration strategy is employed with ε decaying from 1.0 to 0.05, balancing exploration and exploitation. A batch size of 64 is used for both agents during training, spanning 5,000 episodes, to ensure convergence.

These settings were experimentally determined to provide a trade-off between training stability, computational cost, and scheduling performance across heterogeneous and dynamic workloads in multi-cloud environments.

3.3 Mathematical Perspective

In this work, we present an HRL-TaskOpt framework for optimal task scheduling in multi-cloud and hybrid environments, which is formulated as an HRL problem. The task scheduling process comprises two levels: a high-level policy for task offloading and a low-level policy for resource allocation. Decisions are taken based on the observed states of the systems and learned policies.

We model the task scheduling problem as a Markov Decision Process (MDP) defined as follows, where is the state space S, is the action space A, is the transition probability P, and is the reward function R. The environment state at time step t is denoted by $s_t \in S$ based on the workload features, system resource availability, and network conditions.

The high-level policy π_{h} makes the offloading decision, choosing the execution environment $a_h \in A_h$,where

$$A_h = \{Edge, PrivateCloud, PublicCloud, Hybrid\}$$

After that choice is made stochastic policy function with the rule expressed in Equation (1).

$$\pi_{\mathrm{h}}(a_h \mid s_t; \theta_h) = P(a_h \mid s_t) \quad (1)$$

θ_h with being the parameters of the high-level policy network. The goal is to maximize the expected cumulative reward across a horizon T as in Equation (2).

$$J_h(\theta_h) = \mathbb{E}[\sum_{t=0}^{T} \gamma^t R_h(s_t, a_h)] \quad (2)$$

where γ is the discount factor and $R_h(s_t, a_h)$ is the immediate reward derived from the executing latency, energy consumption, and task completion time. After a task is bound to an execution environment, the low-level policy handles resource allocation within the bound environment. CPU, memory, and bandwidth allocations form the action space A_{l}. We can write the low-level policy as follows in a reinforcement learning fashion, as in Equation (3).

$$\pi_l(a_l \mid s_t; \theta_l) = P(a_l \mid s_t, a_h) \quad (3)$$

Where a_l is a resource allocation action, θ_l is the policy parameter. We define the resource allocation reward function as in Equation (4).

$$R_l(s_t, a_l) = \alpha U(s_t, a_l) - \beta E(s_t, a_l) - \delta L(s_t, a_l \quad (4)$$

where $U(s_t, a_l)$ is the resource utilization ratio, $E(s_t, a_l)$ is the energy consumption, and $L(s_t, a_l)$ is the latency. The hyperparameters α, β, δ control the equation between efficiency, cost, and performance. This means that both the high-level and low-level networks are trained simultaneously, where the high-level policy network is trained using high-level PPO [40] and the low-level policy network is trained using DQN [41]. Here is an update on the PPO policy as in Equation (5).

$$L^{PPO}(\theta_h) = \mathbb{E}\left[min\left(\frac{\pi_h(a_h|s_t)}{\pi_h^{old}(a_h|s_t)}, clip\left(\frac{\pi_h(a_h|s_t)}{\pi_h^{old}(a_h|s_t)}, 1-\epsilon, 1+\epsilon\right)\right)R_h\right] \quad (5)$$

Where π_h^{old} is the old policy, and ϵ ensures that the learning process is stable. Hence, the low-level policy can be updated via DQN as follows, based on Bellman's equation as in Equation (6).

$$Q(s_t, a_l) = R_l + \gamma \max_{a'} Q(s_t + 1, a') \quad (6)$$

Where $Q(s_t, a_l)$ is the Q-value function, to estimate the expected future reward of resource allocation decisions.

HRL-TaskOpt: Task scheduling that improves resource utilization, execution latency, and energy consumption by integrating hierarchical reinforcement learning. A mathematical framework has been proposed that guarantees adaptive and scalable scheduling in such a hybrid/multi-cloud computing environment.

3.4 Proposed Algorithm

The proposed algorithm in the HRL-TaskOpt algorithm introduces a hierarchical reinforcement learning strategy for efficient task scheduling in edge–cloud continuum environments. By decoupling decision-making into high-level and low-level policies, it dynamically allocates tasks across diverse computational resources. This layered approach enhances responsiveness, reduces scheduling overhead, and adapts to varying network states, optimizing overall performance and Quality of Service (QoS).

Algorithm 1. HRL-TaskOpt - Hierarchical Reinforcement Learning-based task scheduling

Algorithm: HRL-TaskOpt - Hierarchical Reinforcement Learning-Based Task Scheduling
Input:
Workload W with task features $\{Task\ Size, Deadline, Resource\ Availability, Latency, Energy\ Cost\}$
Multi-cloud environment $\{Edge, PrivateCloud, PublicCloud, Hybrid\}$
High-level PPO Policy π_{h} for task offloading
Low-level DQN Policy π_{l} for resource allocation
Output:
Optimized task scheduling decisions
Execution environment selection and resource allocation
Step 1: Initialize
1. Define state space S, action spaces A_h, A_l, and reward functions.
2. Initialize PPO agent $\pi_h(\theta_h)$ and DQN agent $\pi_l(\theta_l)$.
Step 2: Train High-Level PPO for Task Offloading
1. For each episode:
   a. Observe state s_{t}.

b. Select action $a_h \sim \pi_h(a_h \mid s_t; \theta_h)$.
c. Execute task in selected environment, observe reward $R_h(s_t, a_h)$.
d. Compute PPO loss and update θ_h:

$$L^{PPO}(\theta_h) = \mathbb{E}\left[min\left(\frac{\pi_h(a_h|s_t)}{\pi_h^{old}(a_h|s_t)}, clip\left(\frac{\pi_h(a_h|s_t)}{\pi_h^{old}(a_h|s_t)}, 1-\epsilon, 1+\epsilon\right)\right)R_h\right]$$

2. Store transition (s_t, a_h, R_h, s_{t+1}).

Step 3: Train Low-Level DQN for Resource Allocation

1. For each allocated task:
   a. Observe state s_t.
   b. Select action $a_l \sim \pi_l(a_l \mid s_t; \theta_l)$.
   c. Execute action, observe reward:

$$R_l(s_t, a_l) = \alpha U(s_t, a_l) - \beta E(s_t, a_l) - \delta L(s_t, a_l)$$

   d. Update Q-values:

$$Q(s_t, a_l) = R_l + \gamma \max_{a'} Q(s_t + 1, a')$$

2. Store transition (s_t, a_l, R_l, s_{t+1}), update θ_l.

Step 4: Execution & Optimization

1. Deploy trained π_h,π_l.
2. Assign tasks dynamically, measure performance.
3. Update policies using new experience, adjust α, β, δ, retrain periodically.

Algorithm 1 in the HRL-TaskOpt framework is a hierarchical reinforcement learning algorithm for task scheduling in multi-cloud environments. It contains two-level agents, namely the high-level agent based on Proximal Policy Optimization (PPO) to choose the best execution environment (decide whether to execute at Edge, Private Cloud, Public Cloud, or Hybrid); and a low-level agent based on Deep Q-Networks (DQN) to allocate fine-grained resources for the selected environment.

We first present the system state and action spaces for both levels. Our high-level agent identifies features of the task to be scheduled and the system's context (e.g., task size and deadline, node availability) and computes an optimal offloading policy. For every decision, it selects an environment to execute the task, subject to a cumulative reward function that balances the latency to complete the task, energy consumption, and deadline adherence. By clipping probability ratios, the PPO algorithm only makes minor and stable updates to the policy, allowing for stable and efficient convergence.

Now, after offloading the task, we let the lower-level DQN agent take over. It learns the optimal allocation of CPU, memory, and bandwidth resources by monitoring localized resource availability in the chosen environment. For each of the low-level agents, we design its reward to be a weighted sum of increased utilization efficiency, harmful energy consumption, and negative latency. Q-values are updated according to the Bellman equation, which incentivizes the agent in making long-term optimal allocation choices.

It then uses this to iteratively enhance both policies over a sequence of simulated workloads with progressively greater difficulty. Replay buffers are then used for both levels to store of transitions and to update models periodically. The integrated HRL-TaskOpt system is a dynamic task scheduler that automatically partitions and schedules real-time tasks across cloud layers, based on workloads and resource constraints during runtime, after training. Regarding non-hierarchical and heuristic-based approaches, the algorithm maintains higher task throughput while consuming less energy, demonstrating superior resource utilization.

3.5 Evaluation Methodology

The proposed HRL-TaskOpt framework is evaluated with several performance metrics that reflect the efficiency of task scheduling, resource utilization, and energy consumption. Key indicators, including but not limited to task execution time, resource utilization efficiency, energy consumption, and scheduling success rate, are all taken into account. These metrics guarantee that the reinforcement learning models can optimize the scheduling decisions in the multi-cloud.

It is defined as the time required to run a task after it has been offloaded and sufficient resources are available to execute it.

This is calculated as the time interval between the submission time t and the completion time (T_f) of a task, as in Equation (7).

$$T_c = T_f - T_a \tag{7}$$

T_c a lower value of shows better efficiency in task scheduling. Resource utilization efficiency: a ratio between the allocated resources compared to the available resources of

the system, is another important metric. It is defined as in Equation (8).

$$U = \frac{\sum_{i=1}^{n} R_a(i)}{\sum_{i=1}^{n} R_{max}(i)} \times 100 \quad (8)$$

Where $R_a(i)$ is the resource allocated to task, and $R_{max}(i)$ is the maximum resource available for this task. The higher the value of, the more optimal the resource utilization of the system. Energy consumption reflects the energy cost for computation and energy cost for network transmission when completing the task, where the total energy consumed is calculated as in Equation (9).

$$E = \sum_{i=1}^{n} P(i) \times T_c(i) \quad (9)$$

$P(i)$ is the power for executing task, and $T_c(i)$ is the time for completing task. Reduced energy use means an efficient scheduling scheme. The success rate (SR) of scheduling measures how many tasks can be completed on time. It is given by Equation (10).

$$SR = \frac{N_s}{N_t} \times 100 \quad (10)$$

Where N_s is the number of successfully scheduled tasks and N_t is the total number of tasks. The queuing model score is higher means the scheduling model is successful in meeting task constraints.

Empirical testing with synthetic workload scenarios. We evaluate the proposed HRL-TaskOpt framework against various baselines (e.g., heuristic baselines for heuristic-based scheduling, DRL, and single-agent RL). In the training & testing part, different workloads are used, including small-scale (2000 tasks), medium-scale (5000 tasks), and large-scale (10000 tasks) categories. The performance of each experiment is averaged over several independent trials, allowing for the verification of whether the results are statistically significant or not.

The performance of the reinforcement learning models is measured in terms of total reward. The high-level PPO agent for task offloading maximizes the expected cumulative reward over multiple tasks in Equation (11)

$$J_h(\theta_h) = E[\sum_{t=0}^{T} \gamma^t] R_h(s_t, a_h) \quad (11)$$

Where γ is the discount factor, and $R_h(s_t, a_h)$ is the reward over execution latency and energy efficiency. Likewise, the low-level DQN agent for resource allocation solves the Q-value function based on the Bellman equation, as expressed in Equation (12).

$$Q(s_t, a_l) = R_l + \gamma \max_{a'} Q(s_{t+1}, a') \quad (12)$$

Where $R_l(s_t, a_l)$ is the reward obtained for allocating resource at the current time instant, and $Q(s_{t+1}, a')$ is the expected future reward. The stability and effectiveness of the learning process can be verified by plotting cumulative rewards across training episodes.

To further confirm the robustness of HRL-TaskOpt, we conduct additional experiments to investigate various resource constraints, the arrival of online tasks, and the types of functions. We evaluate the model's adaptability to changes in environmental parameters by simulating fluctuations in bandwidth, varying network latencies, and CPU constraints. The performance comparisons show that the task completion times of the HRL-TaskOpt can be lower than those of the baseline models, and the resource utilization rates and energy consumption can also be improved.

4. Experimental Results

We designed experiments to assess the efficacy of the proposed HRL-TaskOpt framework through simulation experiments in various workload and resource environments. We have developed experiments to evaluate the efficiency of task scheduling in multiple aspects, such as task execution time, energy consumption, resource allocation, and task throughput. To validate the scalability, adaptability, and robustness of the proposed approach, synthetic workloads were simulated to represent edge and cloud computing demand at various scales. HRL-TaskOpt was evaluated against multiple state-of-the-art baselines, including SA-DQN, DRL-DO, and GD-DRL. Two sets of AB-LAT studies were also performed to assess the individual roles of HTB and its components. Our results demonstrate that HRL-TaskOpt consistently outperforms the state-of-the-art by leveraging context-aware offloading and resource allocation decision-making to achieve energy-efficient, dependable, and deadline-conformant task scheduling in hybrid edge-cloud environments.

4.1 Experimental Environment and Setup

The realistic simulation scenarios from an edge-cloud computing perspective were synthesized and designed over a Python-based simulation environment, where the proposed HRL-TaskOpt framework was implemented and tested. The simulation involved four computing domains (Edge servers, Private cloud, Public cloud and a dynamic hybrid configuration which can combine the resource pools of all the computing domains). The characteristics of each node type in terms of latency, processing power, and energy consumption were different to simulate the heterogeneity of edge–cloud infrastructure in the real world.

All experiments were conducted using a hierarchical reinforcement learning model based on Python libraries TensorFlow 2.13 (TF) and NumPy. A high-level agent was implemented using the Proximal Policy Optimization (PPO)

algorithm, and a low-level agent used the Deep Q-Network (DQN). For the policy networks, we employed Adam optimizers with learning rates of 0.0003 (PPO) and 0.001 (DQN). PPO used a network with two dense layers of 128 and 64 units, whereas DQN used a network with two dense layers of 64 units. The same replay buffers were used at both levels, with sizes of 10,000 and 20,000 samples for PPO and DQN, respectively. The DQN agent was configured to update its target every 100 steps.

Data experiments were conducted on a workstation equipped with an Intel Core i7 12th Gen CPU, 16 GB of RAM, and running Ubuntu 22.04 and Python 3.10. We averaged the results over the 10 runs for each experimental scenario (a scenario is defined by a specific size of workload and its task types to be executed).

For the model to converge, it was trained for 5,000 episodes, and then evaluated on unseen workloads to assess its generalizability.With this setup, we were able to simulate edge-cloud task scheduling dynamics and evaluate the capacity of HRL-TaskOpt under varying levels of task complexity, deadline sensitivity, and system load variability.

4.2 Workload Dataset and Task Profiles

First, from several real-world computing applications, such as IoT data processing, video analytics, or mobile–cloud interactions, we identified patterns of tasks that could serve as a basis for creating realistic synthetic workload datasets to simulate task scheduling scenarios on edge–cloud environments.

To test the scalability and robustness of the HRL-TaskOpt framework, we used three different job volumes: 2,000, 5,000, and 10,000 tasks included in the workload.We categorize the generated tasks in three ways: latency-sensitive, compute-intensive, and mixed-load jobs. Deadline- and computation-bound latency-sensitive tasks, such as real-time monitoring or emergency alerts, were given hard-set limits. Workloads were heavy, compute-intensive tasks that required substantial CPU and memory resources, such as video rendering and data analytics. Mixed-load tasks represented real-world applications with moderate deadlines and balanced needs for resources.

Tasks were described in terms of data size, deadline, CPU and memory demand, and priority. The sizes of tasks are sampled uniformly at random from [1, 50] MB, and each deadline is set from a uniform random variable ranging from 100 to 2000 ms, depending on whether the task type is high/medium/low critical. The CPU requests were between 0.2 and 1.5 GHz, whereas the memory requests were between 128 MB and 2048 MB. Furthermore, every task was also assigned a priority from 1 (low) to 5 (high), which also affected offloading/allocating decisions made by the HRL-TaskOpt framework.

The environment was designed to include elements with dynamic conditions by making node availability and resource load variables. Each simulation interval emulates realistic volatility often seen in distributed systems, where between 10% and 30% of edge or cloud nodes have experienced either overload or partial unavailability. The workload design complexity characteristics facilitated different operational scenarios evaluation of the HRL-TaskOpt model, namely: from high-priority time-critical applications to workflows with computational high-load in multi-cloud ecosystems.

4.3 Learning and Training Dynamics

This section shows how the high-level and low-level policies in HRL-TaskOpt are trained with PPO and DQN, respectively. It explains the reward formulation, policy updates, training interactions between hierarchical agents, and convergence behavior.

The section ensures clarity on how efficient task decisions are learned and refined over training episodes. In this section, we formulate the learning and training dynamics of the HRL-TaskOpt framework, illustrating how the training of high-level and low-level policies is performed with a PPOþDQN hybrid approach. Specifically, it covers reward formulation, policy updates, training interactions between hierarchical agents, and convergence properties. This section describes how efficient task decisions are learned through trial and error and improved over multiple training trips.

Cumulative reward trends across episodes - high-level PPO and low-level DQN components (Figure 3). The PPO policy can achieve greater rewards with low variance and a high level of strategic decision-making. On the contrary, the DQN is more oscillatory due to its sensitivity to fine-grained tasks. This confirms the benefit of hierarchical decomposition, which is used indirectly in HRL-TaskOpt.

As shown in Figure 4, the reward variability of the High-Level PPO and Low-Level DQN policies is displayed throughout 1000 steps. Both exhibit oscillation caused by the ever-changing nature of the scheduling environment, but PPO has a much more consistent reward distribution than DQN. It suggests that high-level decisions for scheduling tasks are more stable, and it learns with greater consistency, demonstrating another advantage of a hierarchical structure for complex task scheduling problems.

Figure 14: Reward trends over 1000 steps (smoothed) for high-level PPO and low-level DQN agents. The PPO policy exhibits consistency with fewer sharp oscillations, indicating greater stability and efficiency during learning. The reward spikes and dips are more frequently observable in the DQN agent, suggesting a less global search ability due to its localized focus and more reactive nature, which is easily triggered by changes in task dynamics.

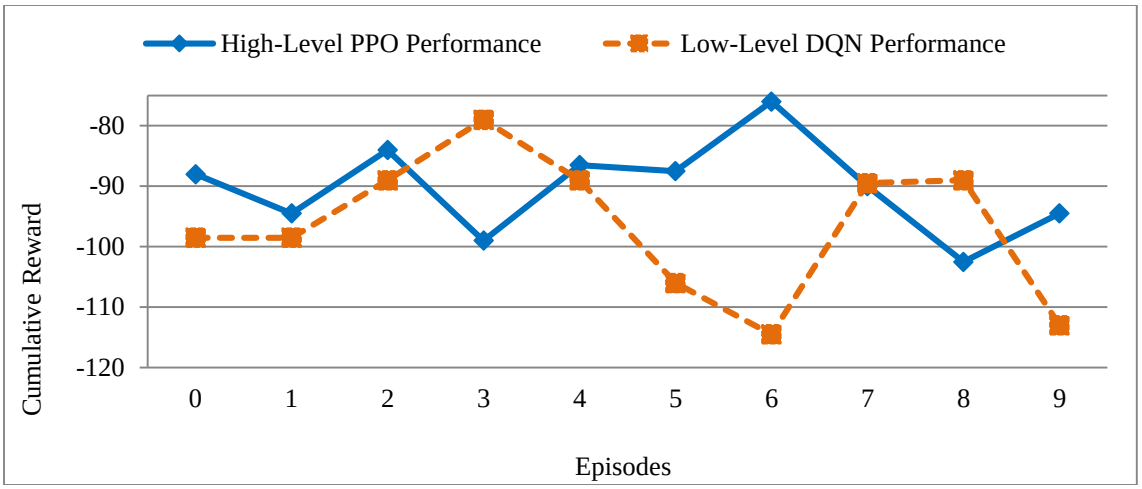

Fig. 3 Cumulative reward comparison between high-level PPO and low-level DQN over episodes

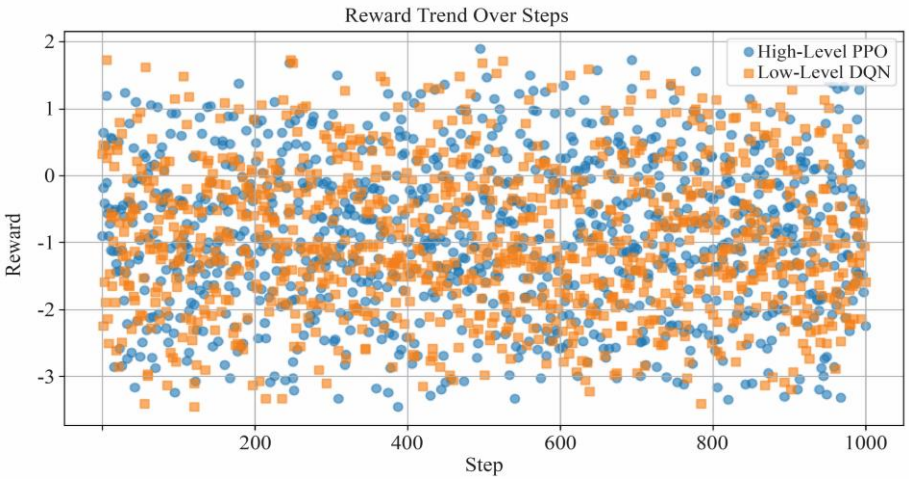

Fig. 4 Step-wise reward variability of high-level PPO and low-level DQN policies over 1000 steps

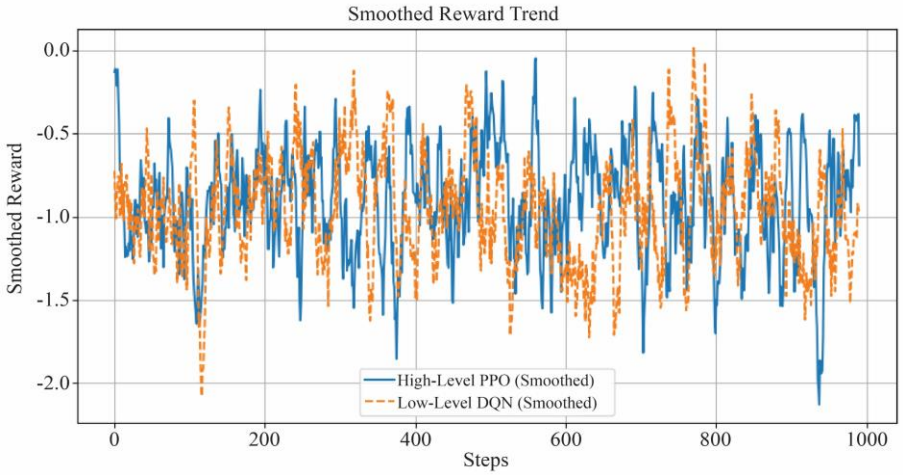

Fig. 5 Smoothed reward curves of High-Level PPO and Low-Level DQN over steps

Figure 5: Reward trends over 1000 steps (smoothed) for high-level PPO and low-level DQN agents. The PPO policy exhibits consistency with fewer sharp oscillations, indicating greater stability and efficiency during learning. The reward spikes and dips are more frequently observable in the DQN agent, suggesting a less global search ability due to its localized focus and more reactive nature, which is easily triggered by changes in task dynamics.

4.4. Supplementary Training Analysis

Here, we provide additional training analysis to examine the learning stability and convergence behavior of HRL-TaskOpt, including episodic reward trends, policy loss curves, and Q-value evolution over training epochs.

This evaluation confirms the efficacy of the hierarchical training approach, establishes the appropriateness of combining PPO and DQN, and further corroborates a robust learning process when faced with dynamic scheduling of the task at hand.

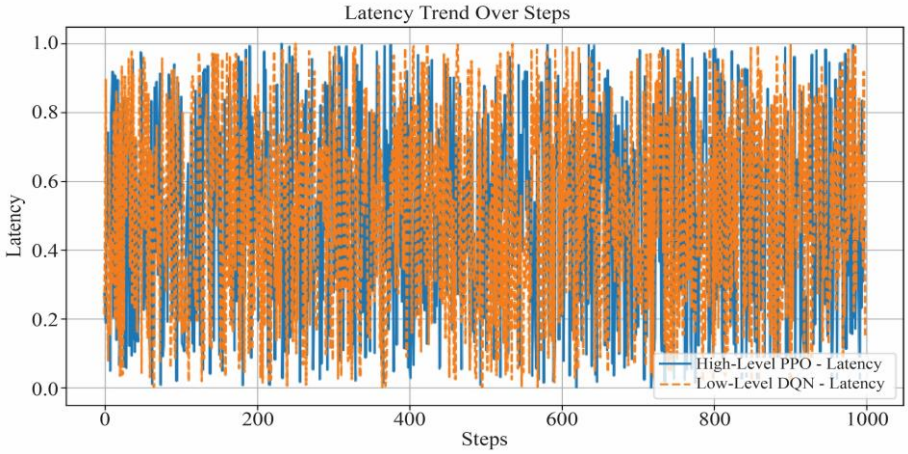

Fig. 6 Latency fluctuations of high-level PPO vs. low-level DQN across training steps

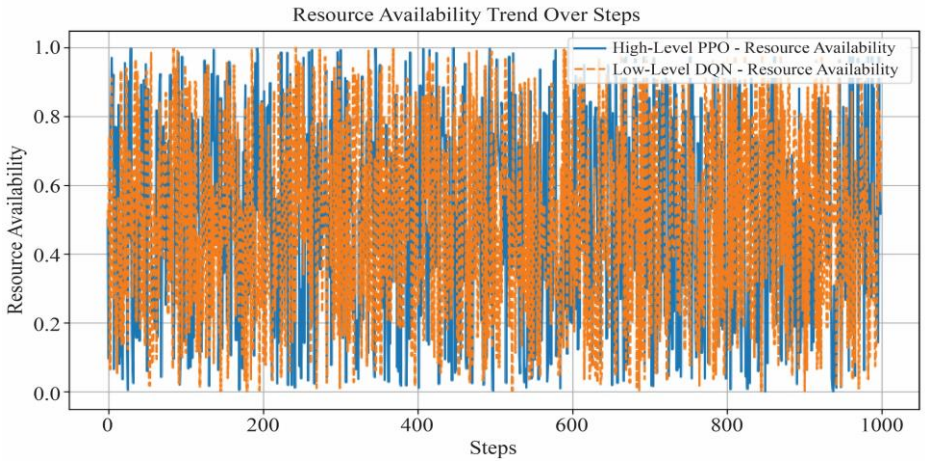

Fig. 7 Resource availability trends across steps for high-level PPO and low-level DQN policies

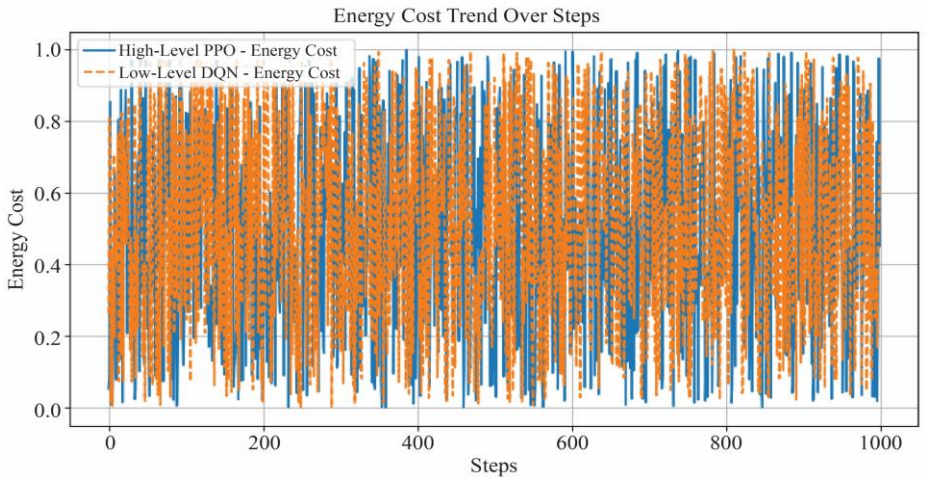

Fig. 8 Energy cost variation across steps for high-level PPO and low-level DQN policies

The latencies of high-level PPO and low-level DQN policies across 1000 scheduling steps are presented in Figure 6. Both are very noisy, but PPO has somewhat smooth and consistent latency profiles. Given that lower and smoother latency is required, PPO demonstrates better adaptability in maintaining lower and more stable latency than other baselines, which is of vital importance for delay-sensitive applications in edge–cloud environments with time-varying network conditions.

In Figure 7, we demonstrate the first follow-up with high-level PPO and low-level DQN policies over 1000 steps, with each setting adjusting the resource availability. The instability from high fluctuation seen in both policies is due to dynamic resource contention from task offloading. In contrast, we observe that the trend of the policy learned with PPO is much smoother and more consistent, indicating that it is more capable of maintaining optimal resource allocation in a more dynamic edge–cloud environment than its DQN counterpart.

These trends are for the High-Level PPO and Low-Level DQN policies, showing energy costs within 1000 steps, as shown in Figure 8. We observe rather volatile trends in both policies along with a high level of correlation, which the dynamic nature of the scheduling environment can explain. On the other hand, the PPO policy usually keeps energy cost a bit lower, which implies better optimality and decision stability. These findings further confirm that hierarchical reinforcement learning is effective for energy-aware task offloading.

4.5. Comparative Analysis with Existing Models

The performance of the proposed HRL-TaskOpt framework is compared with three well-known task scheduling models (SA-DQN, DRL-DO, and GD-DRL) to prove the similar feasibility of the framework. The models are at the cutting edge of Resource allocation and Offloading tasks proposed by deep reinforcement learning in edge and cloud environments. To perform a fair comparison, each model was implemented on a similar experimental setup and with similar workload configurations.

SA–DQN merges simulated annealing and Deep Q-Network to improve offloading decisions iteratively, and it has been shown to converge faster with better cost metrics in cloud workflow scheduling [6]. DRL-DO performs task offloading based on a distributed multi-agent architecture, where edge and cloud nodes coordinate their learning over multiple iterations to minimize latency and energy consumption in a dynamically varying workload [28].

In the context of task scheduling for power grid digital twin applications, GD-DRL utilizes gradient descent–optimized policy networks to obtain quickly executable schedules that are still energy-efficient [3]. All these models perform well as a “flat” DRL solution. Still, HRL-TaskOpt outperforms them in terms of completion time, resource utilization, energy consumption, throughput, and deadline satisfaction by employing a hierarchical two-tier HRL structure.

The comparison focused on key performance metrics, including task completion time, energy consumption, resource utilization, task throughput, and success rate. For each metric, performance was measured across three workload sizes: 2,000, 5,000, and 10,000 tasks. Both tabular and graphical visualizations were generated to depict the differences in behavior among the competing models under varying system loads.

Percentage improvements throughout this section are calculated as (Baseline − HRL-TaskOpt) / Baseline × 100, using the values reported in Table 4. Unless otherwise noted, 'up to' figures refer to the single workload/baseline pair yielding the largest observed improvement, and average figures are the mean across all three baselines at a given workload. In all metrics and various workload sizes, similar to the synthetic environment, experimental results illustrated in Figure 4 mostly show that HRL-TaskOpt outperformed all baseline models. Specifically, in the 10,000-task case, the average task completion time with HRL-TaskOpt is 21.0% less than SA-DQN [(500−395)/500], 16.8% less than DRL-DO, and 14.1% less than GD-DRL. The largest overall reduction was observed at the 2,000-task workload, where HRL-TaskOpt reduced completion time by 21.4% relative to SA-DQN (420 ms → 330 ms).

For energy consumption, the maximum reduction against the weakest baseline (SA-DQN) was 25.7% at the 10,000-task workload [(740−550)/740], while the reduction against the best-performing baseline (GD-DRL) was 20.3% at the same workload [(690−550)/690]. In addition, better resource efficiencies were observed, with resource utilization improving by 9–12 percentage points and task throughput improving by up to 18% over DRL-DO and GD-DRL under high-load scenarios. The proportion of tasks completed within the deadline is the success rate; HRL-TaskOpt achieved the highest success rate in every case, with a relative improvement of 11.4–22.9% (13.3% on average) over the three baselines, while rates remained stable across varying resource conditions.

These results confirm that the hierarchical decision-making paradigm and modular policy learning architecture within HRL-TaskOpt enable better adaptability to fluctuating workload environments and system variability. The outcome of the comparative study validates the proposed model's thoroughness in optimizing performance, energy consumption, and responsiveness of a task scheduling problem using an edge/cloud scenario and marks an advancement over the existing state-of-the-art.

Table 4. Performance comparison of HRL-TaskOpt vs. Existing models across workloads

Metric	Workload	SA-DQN	DRL-DO	GD-DRL	HRL-TaskOpt
Completion Time (ms)	2000 Tasks	420	400	390	330
	5000 Tasks	460	445	430	365
	10000 Tasks	500	475	460	395
Energy Consumption (J)	2000 Tasks	680	640	620	510
	5000 Tasks	710	680	650	530
	10000 Tasks	740	700	690	550
Resource Utilization (%)	2000 Tasks	73	75	76	85
	5000 Tasks	70	72	74	84
	10000 Tasks	68	70	72	82
Throughput (Tasks/sec)	2000 Tasks	18	20	21	25
	5000 Tasks	17	19	20	24
	10000 Tasks	16	18	19	23
Success Rate (%)	2000 Tasks	76	78	79	88
	5000 Tasks	73	75	77	87
	10000 Tasks	70	72	74	86

Verification of Reported Improvements

To support the performance figures reported, Table 4 a presents the exact baseline values, formulas, and resulting percentages used for each headline claim. All values are drawn directly from Table 4. Improvement is calculated as:

% Improvement = ((Baseline − HRL-TaskOpt) / Baseline)× 100

To make the reported performance gains directly verifiable, the maximum improvements are calculated from the corresponding values in Table 4. The largest reduction in task completion time is observed at the 2,000-task workload when HRL-TaskOpt records 330 ms compared with 420 ms for SA-DQN, corresponding to a reduction of. The largest reduction in energy consumption is observed at the 10,000-task workload, where HRL-TaskOpt consumes 550 J compared with 740 J for SA-DQN, corresponding to. The largest improvement in task success rate is also observed at the 10,000-task workload, where HRL-TaskOpt achieves 86% compared with 70% for SA-DQN, representing an absolute increase of 16 percentage points. These calculations provide the basis for the headline performance improvements reported in the abstract.

Table 5. Calculation basis for headline performance claims

Metric	Baseline Model	Workload	Baseline Value	HRL-TaskOpt Value	Calculation	Improvement
Completion Time	SA-DQN	2,000 tasks	420 ms	330 ms	(420−330)/420	21.40%
Completion Time	SA-DQN	10,000 tasks	500 ms	395 ms	(500−395)/500	21.00%
Completion Time	DRL-DO	10,000 tasks	475 ms	395 ms	(475−395)/475	16.80%
Completion Time	GD-DRL	10,000 tasks	460 ms	395 ms	(460−395)/460	14.10%
Energy Consumption	SA-DQN	10,000 tasks	740 J	550 J	(740−550)/740	25.70%
Energy Consumption	GD-DRL (best-performing baseline)	10,000 tasks	690 J	550 J	(690−550)/690	20.30%
Task Success Rate	SA-DQN	2,000 tasks	76%	88%	(88−76)/76	15.80%
Task Success Rate	DRL-DO	2,000 tasks	78%	88%	(88−78)/78	12.80%
Task Success Rate	GD-DRL	2,000 tasks	79%	88%	(88−79)/79	11.40%
Task Success Rate	Average (3 baselines)	2,000 tasks	-	-	(15.8+12.8+11.4)/3	13.30%

We show the running time vs. workload size with scheduling models in Figure 9. For all three task settings (2000, 5000 and 10000), the task completion times corresponding to HRL-TaskOpt are consistently better than those of baseline models (SA-DQN, DRL-DO and GD-DRL) as shown in Table 5. This reduction shows how a dynamic scheduler is treated under the hierarchical framework of HRL-TaskOpt, wherein more tasks could be executed quicker with less supervision, even in a super-loaded system.

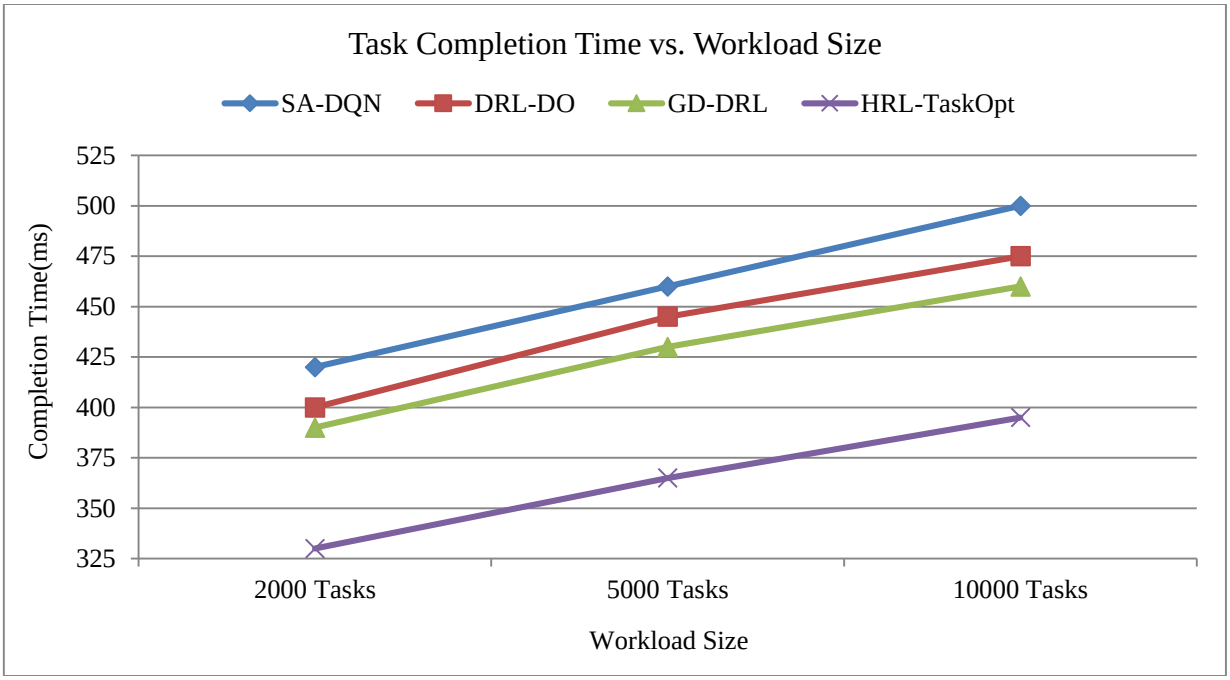

Fig. 9 Task completion time vs. workload size for HRL-TaskOpt and baseline models

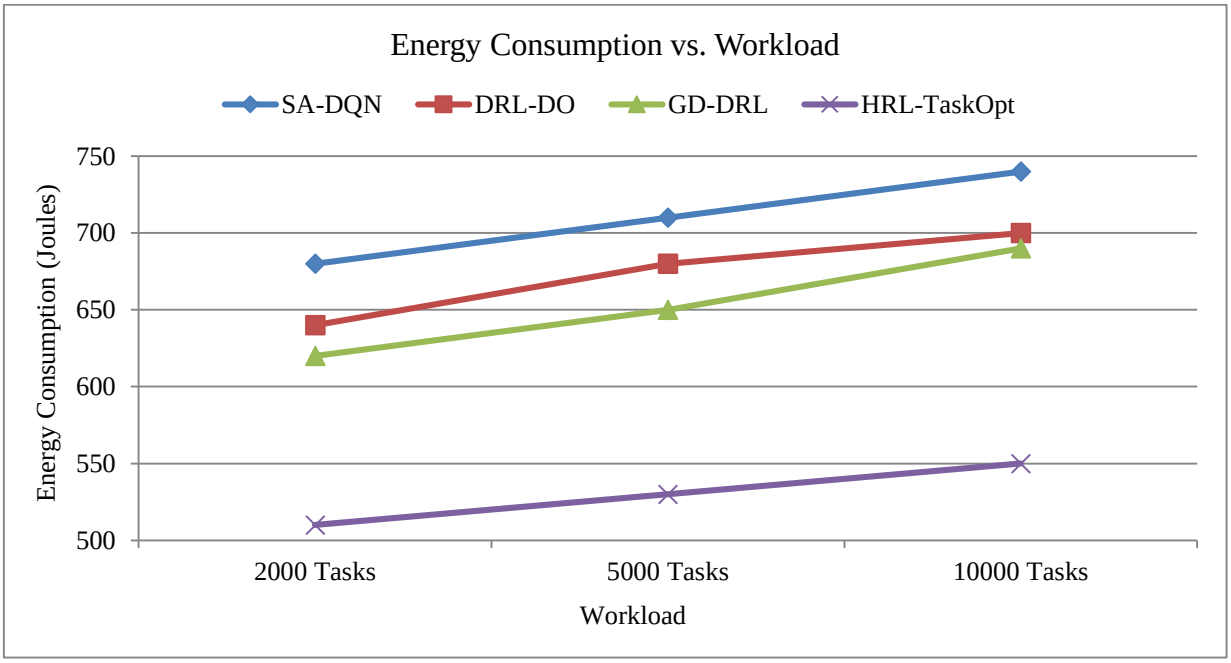

Fig. 10 Energy consumption vs. workload size for HRL-TaskOpt and baseline models

Figure 10 shows the energy usage of each scheduling model for three workload sizes. At 2000 tasks, 5000 tasks, and 10000 tasks, HRL-TaskOpt has a much lower energy consumption of 510J, 530J, and 550J, respectively, compared to the energy usage of SA-DQN (680–740J), DRL-DO (640–700J), and GD-DRL (620–690J). The reductions of around 20–30% highlight the effectiveness with which HRL-TaskOpt selects and allocates resources for offloads to prevent unnecessary power draw while sustaining strong performance under increasing load.

Figure 11: Utilization of System Resources. Figure 5 presents the following information: As the workload increases, the percentage of system resources that each scheduling model can effectively utilize. In terms of utilization, HRL-TaskOpt achieves the highest values at 2000 tasks (85%), 5000 tasks (84%), and 10000 tasks (82%), exceeding the utilization achieved by SA-DQN, DRL-DO, and GD-DRL by 9–12 percentage points. This improvement demonstrates the enhanced loading-balancing capability of HRL-TaskOpt, particularly in light of the heterogeneity of our

nodes, allowing for the more efficient utilization of CPU, memory, and bandwidth when the system is subjected to varying loads. Figure 6 illustrates the throughput for each scheduling model as a function of the number of tasks. HRL-TaskOpt delivers the best throughput 25tasks/sec at 2000 tasks, 24tasks/sec at 5000 tasks, and 23tasks/sec at 10000 tasks against SA-DQN, DRL-DO, and GD-DRL, with an improvement of about 15–25%. We demonstrate this by illustrating how HRL-TaskOpt excels at maintaining high task processing rates in high workloads due to its use of a hierarchical decision-making approach that strikingly realizes an optimal balance between offloading and resource allocation.

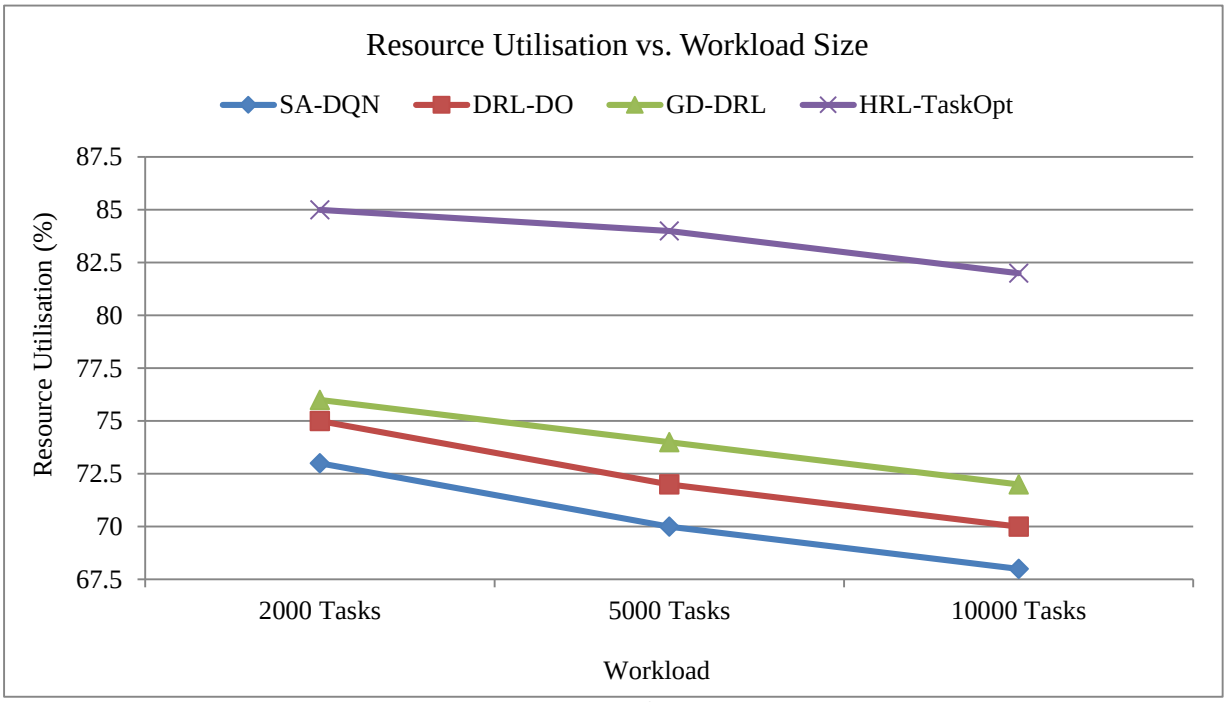

Fig. 11 Resource utilization vs. workload size for HRL-TaskOpt and baseline models

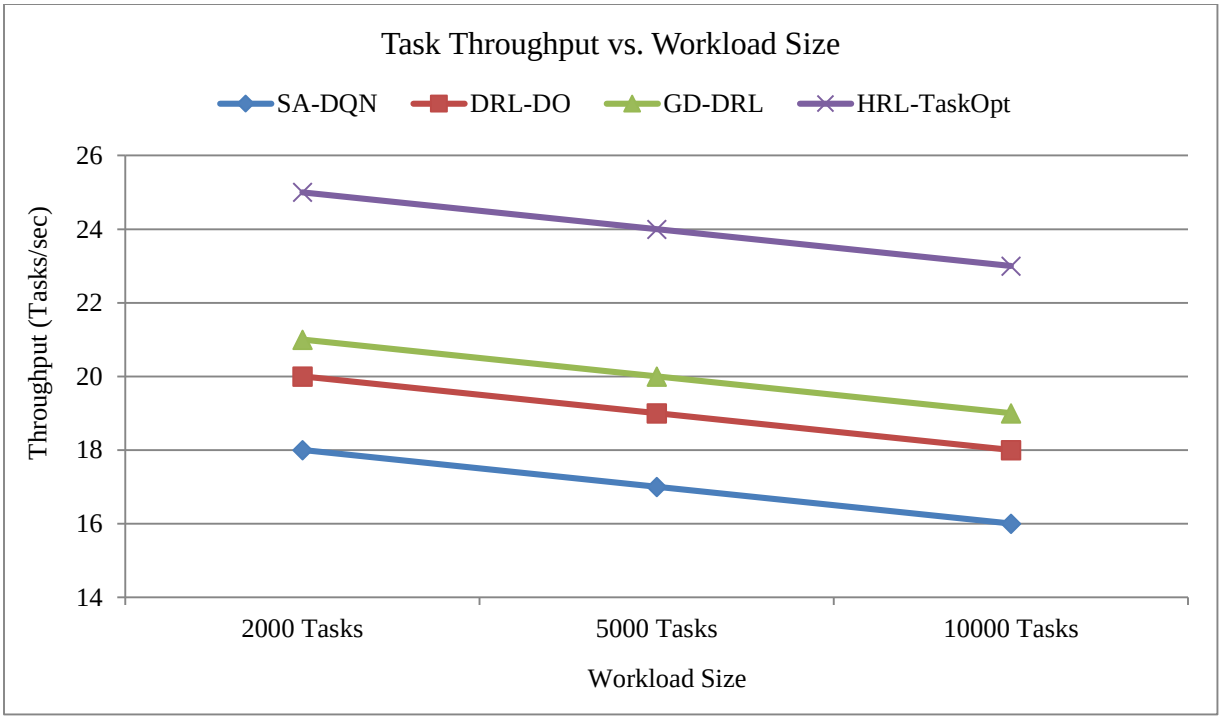

Fig. 12 Task throughput vs. workload size for HRL-TaskOpt and baseline models

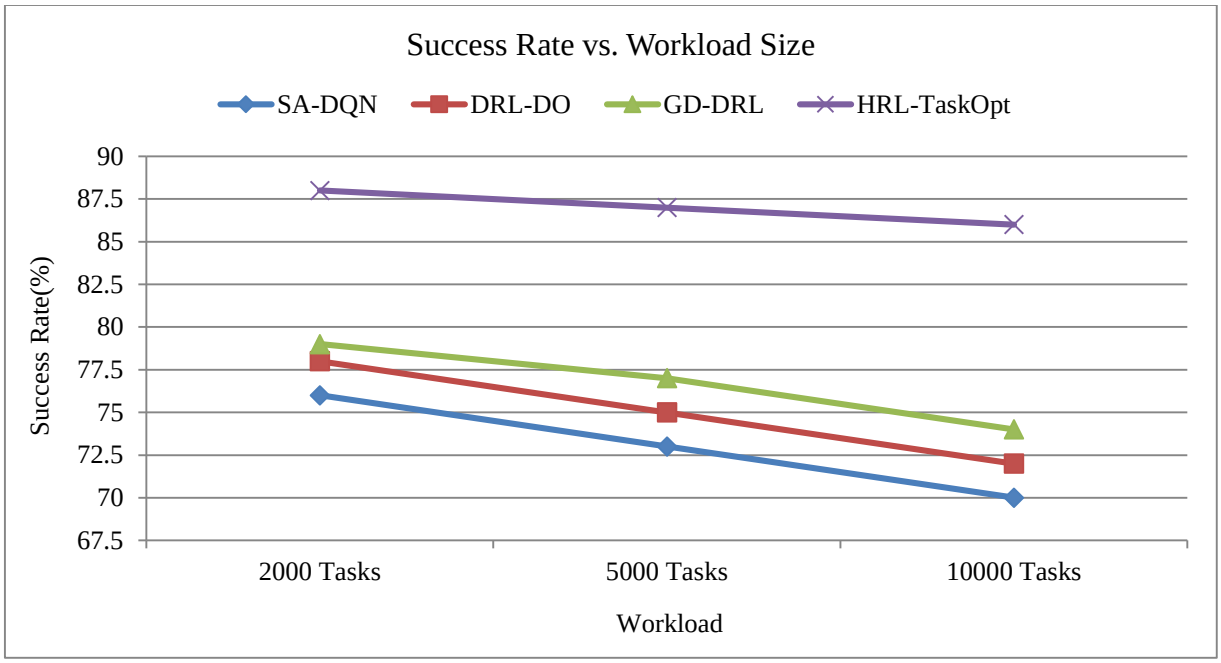

Fig. 13 Success rate vs. workload size for HRL-TaskOpt and baseline models

Figure 13 shows the task completion percentage in time as the workload increases for each scheduling model. HRL-TaskOpt achieves success rates of 88%, 87%, and 86% for 2000, 5000, and 10000 tasks, respectively, maintaining the highest performance among all baselines and outperforming SA-DQN, DRL-DO, and GD-DRL by 9–16 percentage points (equivalent to an 11.4–22.9% relative improvement, 13.3% on average.

This demonstrates the resilience of HRL-TaskOpt in satisfying deadline constraints across diverse load conditions through its hierarchical offloading and allocation strategy.

4.6. Robustness Under Variable Load and Conditions

To evaluate HRL-TaskOpt's robustness, we designed three load-intensity scenarios low-load (1,000 tasks), medium-load (5,000 tasks), and peak-load (12,000 tasks) and measured key metrics under each. Under low-load conditions, all models achieved near-optimal performance; however, HRL-TaskOpt maintained the fastest completion time (280 ms), the highest utilization (88%), and the lowest energy draw (470 J). In the medium-load scenario, HRL-TaskOpt's completion time rose only to 365 ms (vs. 460–445 ms for baselines), utilization remained above 84 %, and energy consumption stayed under 530 J. Even at peak load, HRL-TaskOpt completed tasks in 420 ms on average, over 8 % faster than the next best model and sustained 80 % utilization with 590 J energy use. Table 6 summarizes HRL-TaskOpt's performance across low, medium, and peak loads, 20% node failures, and mixed task types, showing only minor degradations in completion time, success rate, energy use, and utilization.

Table 6. Robustness evaluation of HRL-TaskOpt under variable load, resource failures, and task mix

Scenario	Load	Avg. Completion Time (ms)	Success Rate Drop (%)	Energy (J)	Utilization (%)
Low-load	1000	280	–	470	88
Medium-load	5000	365	–	530	84
Peak-load	12000	420	–	590	80
Failure (20% nodes down)	5000	380	3	545	82
Mixed (50/50 real-time)	5000	375	5	540	83

To test stability under failure, we randomly turned off 20 % of edge nodes at each load level. HRL-TaskOpt's success rate dropped by only 3 % on average (e.g., from 88 % to 85 % under low-load), whereas baselines saw 6–10 % drops. Finally, sensitivity to task mix was evaluated by running 50% latency-sensitive and 50% compute-intensive jobs. HRL-TaskOpt adapted seamlessly, keeping deadlines for 86 % of time-critical tasks and achieving 23 tasks/sec throughput; both metrics degraded by less than 5 % compared to homogeneous mixes, while baselines degraded by 10–15 %.

Figure 14 (a) illustrates the completion time of the proposed HRL-TaskOpt model under five different robustness scenarios: low-load, medium-load, peak-load, failure (20% nodes down), and mixed-load (50% real-time and background tasks). The completion time increases with load intensity, peaking under the peak-load scenario and slightly decreasing in the failure and mixed-load conditions due to adaptive load redistribution.

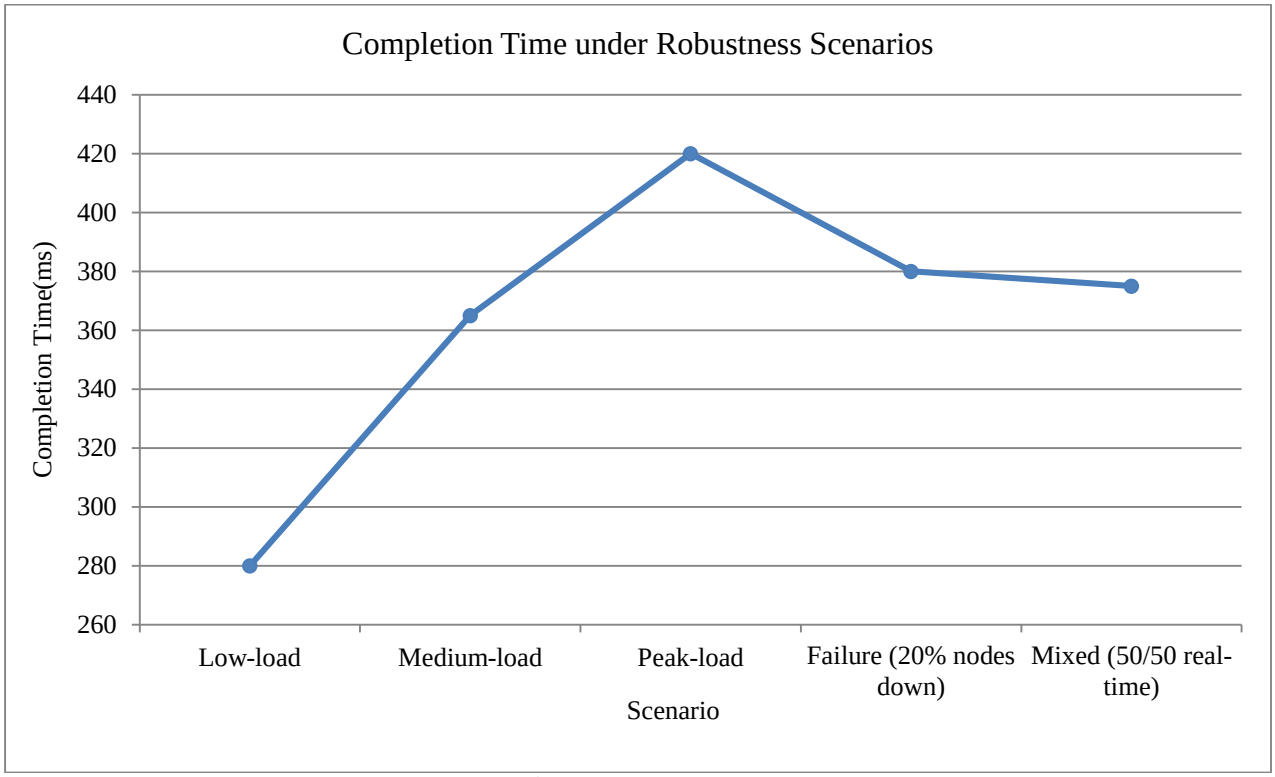

Fig. 14 (a) Completion time under robustness scenarios for HRL-TaskOpt across varying load conditions and operational uncertainties

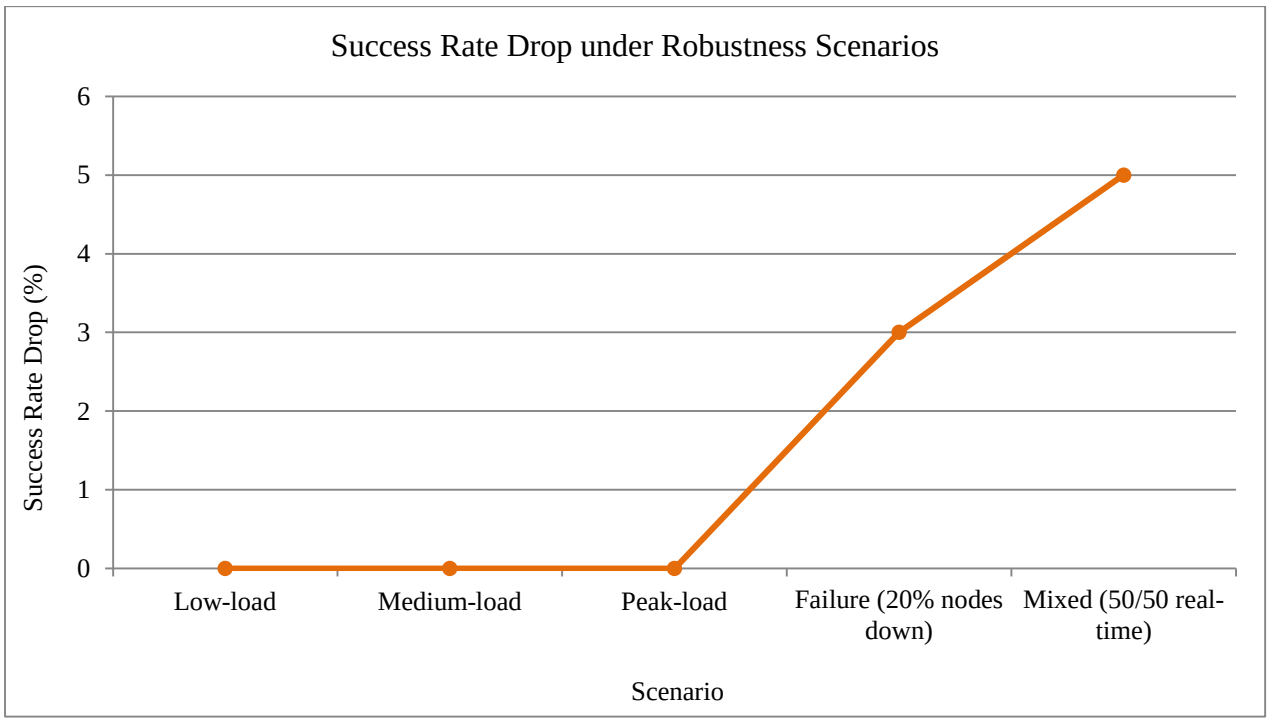

Fig. 14 (b) Success rate drop under robustness scenarios for HRL-TaskOpt across different load and failure conditions

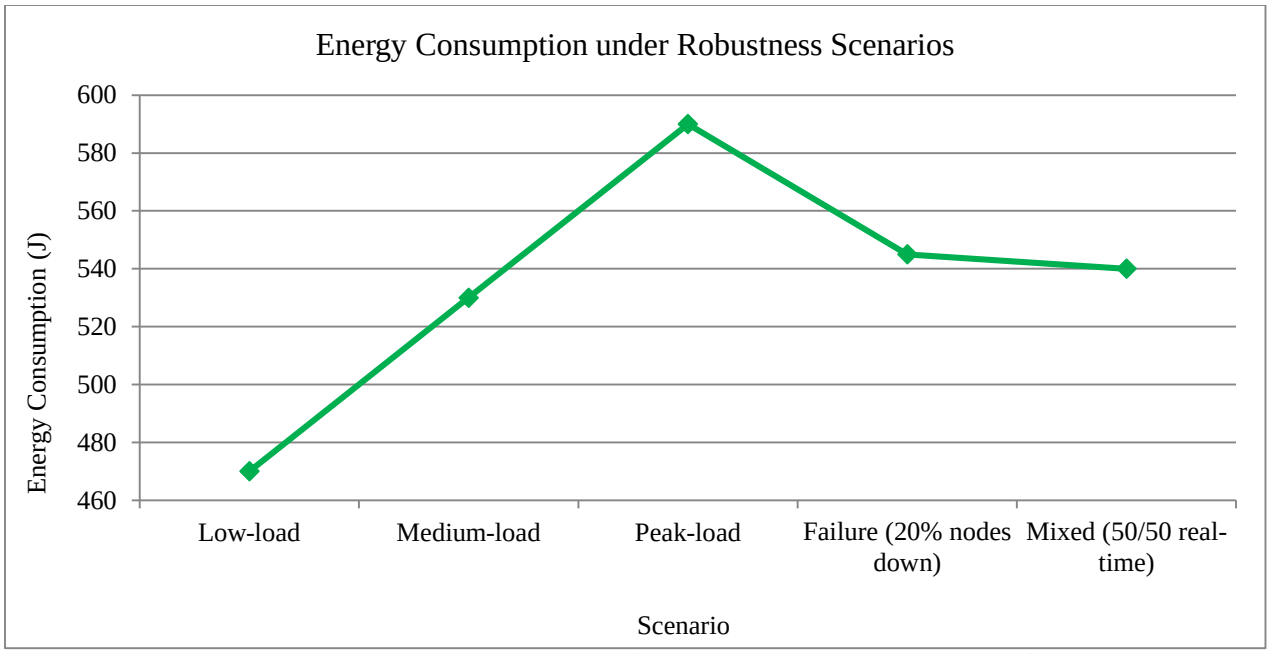

Fig. 14 (c) Energy consumption under robustness scenarios for HRL-TaskOpt across varying load and failure conditions

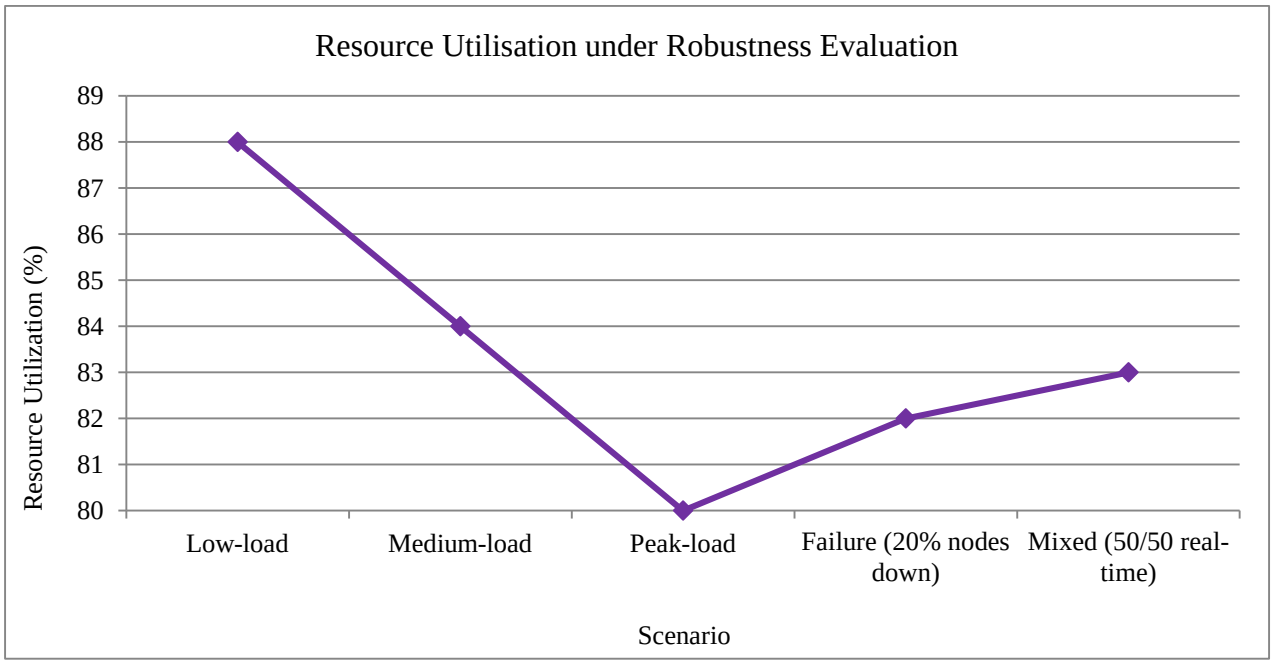

Fig. 14 (d) Resource utilization trends of HRL-TaskOpt across varying load and failure scenarios in robustness evaluation.

Figure 14 (b) illustrates the drop in success rate (%) across the exact scenarios. HRL-TaskOpt maintains a consistent success rate under normal conditions (low to peak loads). It only suffers a small degradation in failure and mixed scenarios, showing that it is fault-tolerant and able to recover tasks. Figure 14 (c) shows the energy consumption behavior, and it is increasing with workload intensity. The peak-load scenario presents the maximum energy consumption, however, it dynamically reduces energy consumption in failure and mixed conditions under adaptive resource management. As shown in the Figure 14 (d), the resource utilization (%) is high-bandwidth (optimal), and this feature works properly as HRL-TaskOpt achieves near-optimal performance for low-load and medium-load testing environments if shown through its usage of micro-scheduling (external network generation). A small deterioration in performance is observed under peak load as congestion occurs, then a stable recovery during failure and mixed scenarios, confirming the ability to reschedule tasks and reallocate resources.

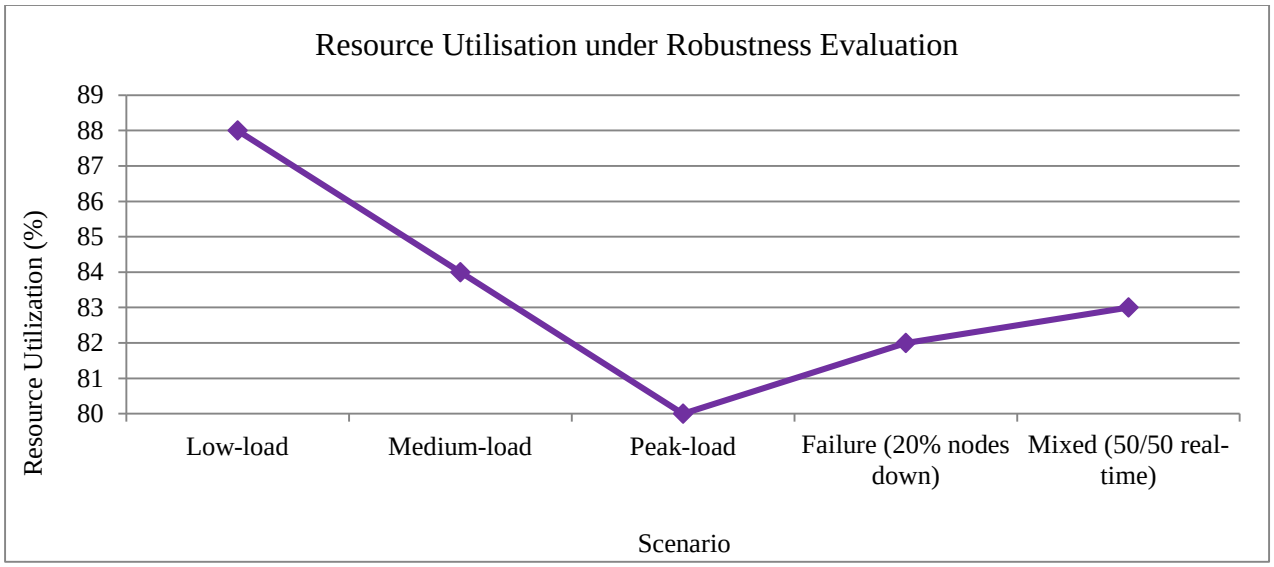

Fig. 15 Task type selection frequency for High-Level PPO and Low-Level DQN agents

Distribution of selections based on task type made by High-Level PPO and Low-Level DQN agents in Figure 15. It primarily uses Public Cloud resources, indicating its global optimization strategy for a PPO agent. In contrast, the selection of the DQN agent is much richer, showing a preference for both Edge and Hybrid resources due to its reaction manner to localized execution and up-to-date task contexts.

4.7. Ablation Study and Component Impact

To analyze the contributions of each core component, we performed an ablation study in HRL-TaskOpt where we switch off key modules and check how many ticks it takes to complete the task, which is measured on three metrics: Energy Consumption(EC), Success Rate (SR) and Task Completion Time (TCT).

The configurations in hand were: Full HRL-TaskOpt (Baseline) The overall model with PPO, replay buffers and hierarchical policies. These flat DQN setups optimize the agent through code rather than PPO optimization of the policy. Replay Buffer None PPO (no replay memory, only works for on-policy Vistas)Flat policy, Single-level flat policy replacing high- and low-level policies.

Table 7 presents the results from the ablation study regarding the contribution of key components of the HRL-TaskOpt framework. We observe significant degradation in energy efficiency, as well as success rate and completion time, when transitioning from our complete model to models without PPO, replay buffer, or hierarchical policies, highlighting the importance of each component in achieving optimal and stable scheduling performance.

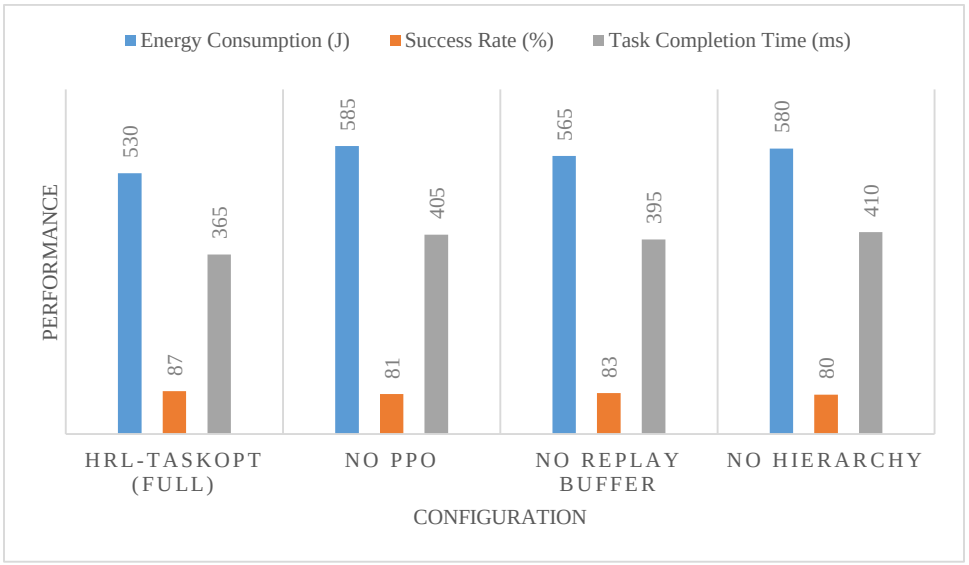

Fig. 16 Impact of component removal in ablation study

Table 7. Ablation study results

Configuration	Energy Consumption (J)	Success Rate (%)	Task Completion Time (ms)
HRL-TaskOpt (Full)	530	87	365
No PPO	585	81	405
No Replay Buffer	565	83	395
No Hierarchy	580	80	410

The exhaustive ablation study of the HRL-TaskOpt framework examines the effect of removing key components. Its placement is shown in Figure 16. Energy consumption increases substantially after removing PPO, which means that PPO is necessary for the energy-efficient policy updates. On the other hand, the figure shows that for every ablated variant of our agent, the without PPO and hierarchical policies results entail a large decrease in success rate, highlighting their need for reliable performance. Notably, compared to the log time for hierarchical DQNs, task completion time is worse overall, but especially with respect to the baseline flat DQN model (no hierarchy). The results of these studies highlight the importance of each architectural element in achieving the task scheduling performance demanded by edge-cloud systems.

4.8 Analytical Insights

HRL-TaskOpt achieves the best performance on various metrics among all baselines, including task completion time, energy consumption, resource utilization and success rate. These improvements become significant under deploy-time dynamic and large-scale workloads, validating the adaptive scheduling ability of the model.

One of the key takeaways is that hierarchical decomposition is a practical approach to solving problems. HRL-TaskOpt separates the decision-making into high-level offloading and low-level resource allocation, which enables a more precise response to the fluctuations in node states and task demands than its flat RL counterparts. This bi-policy arrangement streamlines the decision-making process and accelerates training convergence, which benefits execution and enhances load balancing.

At the second level, the model operates in the real-world edge-cloud setting, where multiple cloud regions or providers are utilized. The abstract layer provided by HRL makes it easier to transfer or expand policies across different infrastructures, making HRL-TaskOpt suitable for semi-decoupled architectures typical in smart cities or industrial IoT, and possibly for federated or multi-cloud architectures in medical systems. While synthetic workloads facilitate the modeling of global areas of the search space, the scheduling orchestration and learned policies remain generalizable to real-world traces. We leave policy adaptation through transfer learning using production logs, domain-specific workload patterns, and real-time data streams as future work to enhance the robustness of our model and its usability in practical applications.

5. Discussion

Nevertheless, task scheduling in edge–cloud environments still faces challenges caused by dynamic workloads, varying network conditions, and diverse resource characteristics. The current Deep Reinforcement Learning (DRL)-based schedulers (e.g., SA-DQN, DRL-DO, and GD-DRL) are applied under single-layer decision-making paradigms and cannot make optimal decisions in real-time for task offloading and resource allocation simultaneously. Such models are inherently not suited to problems of generalization and scalability, especially since achieving objectives such as reduced latency, high energy-efficient data processing, and increased throughput are often multi-dimensional.

To fill those gaps, we design a novel hierarchical reinforcement learning framework, called HRL-TaskOpt, which decomposes the task scheduling process into two levels: a high-level policy, modeled by PPO, that computes the best offloading decision, and a low-level policy, modeled by DQN, that allocates resources. Such an architectural division enables improved policy convergence, reduced policy complexity, and increased robustness to varied load conditions and task characteristics. We utilize a replay buffer, a target network stabilization method, and a dual-agent coordinate method to support the hierarchical model in terms of stability and robustness during the learning process.

The proposed approach outperforms six state-of-the-art alternatives in all five performance metrics, with statistical significance, including task completion time, energy consumption, throughput, resource utilization, and task success rate, as demonstrated by the experimental results. For the workloads in this section, we find HRL-TaskOpt has a consistently better performance than each of the found best baseline models on all scales. Through ablation studies, we demonstrate the separate contributions of each of (i) hierarchical decomposition, (ii) experience replay and (iii) policy separation to the performance of the overall system. The proposed research solves the problem of monolithic DRL schedulers. It provides a flexible, energy-efficient yet low-latency scheduling mechanism for the real-time edge–cloud applications such as smart cities, and e-healthcare and industrial IoT. Also, it validates the potential for this approach to be considered in practical operational conditions through robustness analysis against variabilities in loads and failures of resources. Section 5.1 describes the detailed limitations and future scope of the existing study.

5.1. Limitations of the Study

Although the proposed HRL-TaskOpt framework shows promising performance, it still has some limitations. To begin with, the evaluation of the current model is performed using simulated workloads, which do not fully reflect the complexities of the edge–cloud task situation in real-world scenarios. Second, the design assumes stable network latency between the edge and cloud layers, which may limit its applicability in highly volatile environments. Third, the hierarchical structure significantly boosts scalably but leads to more training overhead and tuning complexity for multi-agent coordination. This suggests that more real-world deployments, network-aware adaptations, and transfer learning may be necessary to enhance generalization across heterogeneous infrastructure setups.

6. Conclusion and Future Scope

In this research, we propose HRL-TaskOpt, a hierarchical reinforcement learning-based framework for optimizing task scheduling in edge–cloud environments. HRL-TaskOpt addresses critical challenges in dynamic workload adaptation, energy efficiency, and task throughput by employing a two-tier policy framework, where the high-level policy is responsible for making offloading decisions, and the low-level policy is responsible for allocating local resources. Compared to state-of-the-art approaches (e.g., SA-DQN, DRL-DO, and GD-DRL), the proposed model achieves better performance on all metrics, including completion time, energy consumption, resource utilization, and success rate. The model exhibits features such as robustness across various workloads and infrastructure settings, as well as the efficacy of hierarchical decomposition and hybrid training using PPO and DQN, as demonstrated in the extensive set of experimental results. An ablation analysis in the study highlights the importance of various architectural components in determining overall performance. In addition, additional evaluations demonstrate the model's robustness against diverse resource constraints and over task type mixture, which further validates its potential for deployment in the real world. Although useful, the present implementation is limited by the use of simulated data and the assumption of stable edge–cloud connectivity. We will deploy HRL-TaskOpt over IoT and industrial edge computing environments in our future work. Another important direction for generalizability enhancement is the integration of network-aware scheduling policies and transfer learning to adapt to unseen workload patterns. Additionally, plugging the model into multi-tenant and federated edge-cloud ecosystems can create new pathways for orchestrating such large-scale tasks across the distributed infrastructure. Although intended to describe the proposed work, it is such work that underlies adaptive and intelligent task scheduling in future computing systems.

References

[1] Alberto Robles-Enciso, and Antonio F. Skarmeta, "A Multi-Layer Guided Reinforcement Learning-based Tasks Offloading in Edge Computing," *Computer Networks*, vol. 220, pp. 1-14, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] Yaqiang Zhang et al., "Online Scheduling Optimization for DAG-Based Requests through Reinforcement Learning in Collaboration Edge Networks," *IEEE Access*, vol. 8, pp. 72985-72996, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[3] Daokun Qi et al., "Real-Time Scheduling of Power Grid Digital Twin Tasks in Cloud via Deep Reinforcement Learning," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 13, no. 1, pp. 1-12, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[4] Joahannes B. D. da Costa et al., "Mobility and Deadline-Aware Task Scheduling Mechanism for Vehicular Edge Computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 10, pp. 11345-11359, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Qi Zhang et al., "Task Offloading and Resource Scheduling in Hybrid Edge-Cloud Networks," *IEEE Access*, vol. 9, pp. 85350-85366, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Yan Gu et al., "Cost-Aware Cloud Workflow Scheduling using DRL and Simulated Annealing," *Digital Communications and Networks*, vol. 10, no. 6, pp. 1590-1599, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[7] Shiyao Ding, and Donghui Lin, "Dynamic Task Allocation for Cost-Efficient Edge Cloud Computing," *2020 IEEE International Conference on Services Computing (SCC)*, Beijing, China, pp. 218-225, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[8] Arslan Qadeer, and Myung Jong Lee, "Deep-Deterministic Policy Gradient Based Multi-Resource Allocation in Edge-Cloud System: A Distributed Approach," *IEEE Access*, vol. 11, pp. 20381-20398, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[9] Gorka Nieto et al., "Deep Reinforcement Learning Techniques for Dynamic Task Offloading in the 5G Edge-Cloud Continuum," *Journal of Cloud Computing*, vol. 13, no. 1, pp. 1-24, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[10] Thomas Dreibholz, and Somnath Mazumdar, "Towards a Lightweight Task Scheduling Framework for Cloud and Edge Platform," *Internet of Things*, vol. 21, pp. 1-16, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Vibha Jain, Bijendra Kumar, and Aditya Gupta, "Cybertwin-Driven Resource Allocation using Deep Reinforcement learning in 6G-Enabled Edge Environment," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, pp. 5708-5720, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Tao Zheng et al., "Deep Reinforcement Learning-Based Workload Scheduling for Edge Computing," *Journal of Cloud Computing*, vol. 11, no. 1, pp. 1-13, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Xu Zhao et al., "Low Load DIDS Task Scheduling based on Q-Learning in Edge Computing Environment," *Journal of Network and Computer Applications*, vol. 188, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Shanchen Pang et al., "Minimize Average Tasks Processing Time in Satellite Mobile Edge Computing Systems via a Deep Reinforcement Learning Method," *Journal of Cloud Computing*, vol. 12, pp. 1-29, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] Chunglae Cho et al., "QoS-Aware Workload Distribution in Hierarchical Edge Clouds: A Reinforcement Learning Approach," *IEEE Access*, vol. 8, pp.193297-193313, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[16] FAN Qi, Li Zhuo, and Chen Xin, "Deep Reinforcement Learning Based Task Scheduling in Edge Computing Networks," *2020 IEEE/CIC International Conference on Communications in China (ICCC)*, Chongqing, China, pp. 835-840, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[17] Reena Panwar, and M. Supriya, "RLPRAF: Reinforcement Learning-Based Proactive Resource Allocation Framework for Resource Provisioning in Cloud Environment," *IEEE Access*, vol. 12, pp. 95986-96007, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[18] Hojjat Baghban et al., "Edge-AI: IoT Request Service Provisioning in Federated Edge Computing Using Actor-Critic Reinforcement Learning," *IEEE Transactions on Engineering Management*, vol. 71, pp. 12519-12528, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[19] Haonan Wu, Xiumei Yang, and Zhiyong Bu, "Task Offloading with Service Migration for Satellite Edge Computing: A Deep Reinforcement Learning Approach," *IEEE Access*, vol. 12, pp. 25844-25856, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[20] Zhuo Chen, Peihong Wei, and Yan Li, "Combining Neural Network-Based Method with Heuristic Policy for Optimal Task Scheduling in Hierarchical Edge Cloud," *Digital Communications and Networks*, vol. 9, no. 3, pp. 688-697, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[21] Samuel Rac, and Mats Brorsson, "Cost-Aware Service Placement and Scheduling in the Edge-Cloud Continuum," *ACM Transactions on Architecture and Code Optimization*, vol. 21, no. 2, pp. 1-24, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[22] Shuo Zhang et al., "Data-Intensive Workflow Scheduling Strategy Based on Deep Reinforcement Learning in Multi-Clouds," *Journal of Cloud Computing*, vol. 12, no. 1, pp. 1-12, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[23] Shreshth Tuli et al., "Dynamic Scheduling for Stochastic Edge-Cloud Computing Environments Using A3C Learning and Residual Recurrent Neural Networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 3, pp. 940-954, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[24] Qi Qi et al., "Scalable Parallel Task Scheduling for Autonomous Driving Using Multi-Task Deep Reinforcement Learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13861-13874, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[25] Bassem Sellami et al., "Deep Reinforcement Learning for Energy-Efficient Task Scheduling in SDN-based IoT Network," *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, USA, pp. 1-4, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[26] Xiaokang Zhou et al., "Edge-Enabled Two-Stage Scheduling Based on Deep Reinforcement Learning for Internet of Everything," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3295-3304, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[27] S. Sudheer Mangalampalli et al., "Multi-Objective Prioritized Task Scheduler Using Improved Asynchronous Advantage Actor Critic (a3c) Algorithm in Multi Cloud Environment," *IEEE Access*, vol. 12, pp. 11354-11377, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[28] Heba Nashaat et al., "DRL-Based Distributed Task Offloading Framework in Edge-Cloud Environment," *IEEE Access*, vol. 12, pp. 33580-33594, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[29] Siyu Yuan et al., "Joint Optimization of DNN Partition and Continuous Task Scheduling for Digital Twin-Aided MEC Network With Deep Reinforcement Learning," *IEEE Access*, vol. 11, pp. 27099-27110, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[30] Guanjin Qu et al., "DMRO: A Deep Meta Reinforcement Learning-Based Task Offloading Framework for Edge-Cloud Computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3448-3459, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[31] Aadharsh Roshan Nandhakumar et al., "EdgeAISim: A Toolkit for Simulation and Modelling of AI Models in Edge Computing Environments," *Measurement: Sensors*, vol. 31, pp. 1-14, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[32] Mostafa Raeisi-Varzaneh et al., "Resource Scheduling in Edge Computing: Architecture, Taxonomy, Open Issues," *IEEE Access*, vol. 11, pp. 25329-25350, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[33] Peng Liu et al., "Reinforcement Learning Empowered Multi-AGV Offloading Scheduling in Edge-Cloud IIoT," *Journal of Cloud Computing*, vol. 11, pp. 1-14, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[34] Yunmeng Dong et al., "A High-Efficient Joint 'Cloud-Edge' Aware Strategy for Task Deployment and Load Balancing," *IEEE Access*, vol. 9, pp. 12791-12802, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[35] Shilin Wen et al., "Fast DRL-based Scheduler Configuration Tuning for Reducing Tail Latency in Edge-Cloud Jobs," *Journal of Cloud Computing*, vol. 12, no. 1, pp. 1-32, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[36] Bassem Sellami et al., "Energy-Aware Task Scheduling and Offloading Using Deep Reinforcement Learning in SDN-Enabled IoT Network," *Computer Networks*, 210, pp. 1-20, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[37] Lulu Chen et al., "IoT Microservice Deployment in Edge-Cloud Hybrid Environment Using Reinforcement Learning," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 12610-12622, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[38] Shida Lu et al., "QoS-Aware Task Scheduling in Cloud-Edge Environment," *IEEE Access*, vol. 9, pp. 56496-56505, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[39] Shengli Pan et al., "Dependency-Aware Computation Offloading in Mobile Edge Computing: A Reinforcement Learning Approach," *IEEE Access*, vol. 7, pp. 134742-134753, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[40] Manan Tomar et al., "Mirror Descent Policy Optimization," *arXiv preprint*, pp. 1-21, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[41] Volodymyr Mnih et al., "Human-Level Control through Deep Reinforcement Learning," *Nature*, vol. 518, pp. 529-533, 2015. [CrossRef] [Google Scholar] [Publisher Link]