International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

EDGECLOUD-DRL: A DEEP REINFORCEMENT LEARNING-BASED TASK SCHEDULING FRAMEWORK FOR EDGE-CLOUD COMPUTING

Mohammed Waseem Ahmed [1], Dr. G.Kavitha [2]

[1].Research Scholar, Department of Computer Science and Engineering, B.S.Abdur Rahman Crescent Institute of Science & Technology GST Road, Vandalur, Chennai 600 048. Tamilnadu. INDIA. waseem1215@gmail.com

[2].Professor & Director in charge, CITL, B.S.Abdur Rahman Crescent Institute of Science & Technology GST Road, Vandalur, Chennai 600 048. Tamilnadu., gkavitha.78@crescent.education

Abstract

Efficient task scheduling is essential in cloud and edge computing environments to minimize execution latency, energy usage, and resource utilization. Although optimization problems can be approached through heuristics and metaheuristics, like Particle Swarm Optimization (PSO), Multi-Objective Bat Algorithm (MBO), and Multi-Objective Particle Swarm Optimization (MOPSO), which are often time-consuming, cannot address dynamic workloads well and are non-adaptive. Recent advances in Deep Reinforcement Learning (DRL)-based methods hold promise for addressing these issues. Yet recent advances in state-of-the-art DRL work, e.g., Deep Q-Networks (DQN), Deep Deterministic Policy Gradient (DDPG) and Multi-Agent DRL encounter scalability bottlenecks, inferior learning efficiency, and challenges of real-time scheduling in large-scale hybrid cloud-edge settings. To address these drawbacks, we propose EdgeCloud-DRL, a delivery framework based on Deep Q-Network (DQN) that uses target network stabilization and experience replay to improve decision-making efficiency. Our proposed framework is based on Q-learning, where at each step, it learns an optimal scheduling policy by updating its set of Q-values to maximize the task success rate while minimizing the latency and energy cost. Numerous experiments were performed using a customized edge-cloud simulation environment, and the developed model was compared with PSO, MBO, and MOPSO for various workload intensities (1000 to 20,000 tasks). Experimental results indicate that EdgeCloud-DRL outperforms baseline methods by 28% in execution latency, 19% in task success rate, and 23% in energy efficiency. The proposed EdgeCloud-DRL framework also provides adaptive, scalable, and learning-driven schedulings, which are very demonstrably applicable in real-time cloud-edge applications, IoT task schedulings, and distributed AI workloads. These findings further manifest the impact of workload management based on reinforcement learning optimization approaches within intelligent workloads in the current computing infrastructures.

Keywords - Deep Reinforcement Learning, Task Scheduling, Edge-Cloud Computing, Deep Q-Network, Resource Optimization

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

. introduction

Cloud computing has emerged as the backbone of modern digital infrastructures, enabling on-demand computational resources for diverse applications such as big-data analytics, AI, and the Internet of Things (IoT). Nevertheless, the swift surge of cloud service users has resulted in challenges in efficient task scheduling, resource utilization, and energy optimization. These are only examples of conventional scheduling methods, including heuristic and metaheuristic algorithms. Still, they are not ideal for meeting the challenges in dynamic and heterogeneous cloud-edge environments. In addition, the recent advances in deep reinforcement learning (DRL) have demonstrated the potential to optimize scheduling decisions using intelligent learning approaches. However, most existing DRL-based models can still not scale or adapt large-scale scenarios of cloud-edge computing effectively.

Several machine learning and DRL-based scheduling techniques have been devised in cloud computing. Yan Gu et al. [1] proposed a DRL-based workflow scheduling solution with simulated annealing, which reduced the cost but struggled with the real-time problem. Shuo Zhang et al. [2] utilized a Deep Q-Learning model for multitask scheduling across multiple clouds, achieving energy-efficient results but exhibiting latency variation under dynamic workloads. Gorka Nieto et al. Ball, for example, proposed a DDPG-based reinforcement learning framework for task offloading in 5G edge-cloud environments by optimizing for the same task and considering offloading efficiency over flexibility versus dynamic workloads. Heba Nashaat et al. [4]] developed a DRL model using multiple agents, with improved execution time yet poor performance during large-task processing. Alberto Robles-Enciso et al. Hierarchical reinforcement learning[5] was introduced to allow the use of various resources with improved utilization, but the algorithm did not support flexible scheduling strategies for heterogeneous environments. We build on these findings to propose a new approach that balances execution efficiency, task success rate, and resource utilization and scales to support a realistic number of agent instances.

To bridge these gaps, we propose EdgeCloud-DRL, a novel task scheduling framework based on Deep Q-Network (DQN) for hybrid edge-cloud environments. The proposed framework delivers efficient task execution with low overhead since it combines target network stabilization and experience replay strategies in comparison with the traditional reinforcement learning models. Some crucial innovations of EdgeCloud-DRL include adaptive scheduling strategies, optimized reinforcement learning policy updates, and scalability improvements to manage large-scale workloads efficiently. The performance is evaluated based on different workload intensities, showing that EdgeCloud-DRL outperforms the existing heuristic and learning-based scheduling models.

In this respect, the key contributions of this work include: (1) Deriving an intelligent DRL-based task scheduling approach that focuses on optimizing latency, energy efficiency, and resource utilization in a unified edge-cloud environment. (2) We establish an adaptive scheduling strategy based on Deep Networks with a target network stabilizer and experience replay to enhance the learning efficiency. (3) A benchmark performance evaluation with state-of-the-art models, showing better scalability and adaptability to dynamic workloads.

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The remainder of this paper is structured as follows. An extensive literature review is presented in Section 2, focusing on recent state-of-the-art DRL cognition-based scheduling methodologies and their drawbacks. The proposed method is introduced in Section 3, including the EdgeCloud-DRL framework, reinforcement learning model, state-action representations, and reward mechanisms. Section 4 describes the experimental configuration and shows the performance results of EdgeCloud-DRL concerning PSO, MBO, and MOPSO under different workload intensities. Section 5 presents the key findings of our work and highlights the relevance of the proposed approach in filling the gap that current techniques face while mapping out the study's limitations in Section 5.1. Last but not least, Section 6 wraps up the study and discusses emerging research opportunities for adaptive scheduling enhancement and multi-agent reinforcement learning for better scheduling efficacy in intricate edge-cloud infrastructures.

2. Related Work

Deep Reinforcement Learning (DRL) for task scheduling in edge-cloud computing environments has been widely studied. Zhang et al. [28] designed a hybrid scheduling scheme combining heuristics and RL scheduling policies to optimize tasks in a multi-cloud environment. Gu et al. [1] and Zhang et al. [2] conduct cost-aware workflow scheduling, leveraging DRL alongside heuristic-based optimization methods such as simulated annealing to enhance task allocation. Nieto et al. [3] propose a deep reinforcement learning (DRL) based dynamic task offloading for 5G edge-cloud computing scenarios overcoming the real-time execution and resource allocation challenges. Nasheet et al. We introduce a distributed task offloading framework by utilizing DRL not only to minimize the execution time but also the energy consumption[4]. In the same manner, Robles-Enciso et al. [5] introduce a task offloading model based on multi-layer guided reinforcement learning edge computing to enhance task execution efficiency. Chen et al. [6] studied neural network-based scheduling with heuristic policies to maintain workload optimality. Dreibholz et al. [7] propose a lightweight in which the scheduling framework is specific to the used cloud and edge platforms while balancing the trade-off between computational cost and execution efficiency.

Zhou et al. [8] extend the DRL applications to also apply to the Internet of Everything (IoE), where they introduce a two-stage scheduling method to deal with complex workloads. Costa et al. [9] proposed a mobility-aware scheduling mechanism, enabling continuous service execution despite changing mobility conditions during vehicular edge computing. Yuan et al. [10] propose a joint optimization model for continuous task scheduling and Digital twin-based computing using deep reinforcement learning. Wu et al. [11] suggest the DRL-based service migration techniques for satellite edge computing, where the continuity of tasks is complex to address. Pang et al. [12] use deep reinforcement learning for satellite mobile edge computing to reduce task execution latency. Qadeer et al. [13] propose a multi-resource greedy strategy in edge-cloud systems based on a deep deterministic policy gradient (DDPG) algorithm. Roshan et al. [14] leverage EdgeAISim, a far more general-purpose tool, to test the scheduling of AI models in edge computing. Mangalampalli et al. The work of [15] introduces a multi-objective

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

prioritized scheduler based on A3C reinforcement learning with experimental suggestions on the benefits of such scheduler use in multi-cloud task executions.

Qi et al. One recent research work [16] investigates real-time scheduling for power grid digital twin tasks using DRL to optimize the allocation of computational resources accurately. Raeisi-Varzaneh et al. [17] discuss a taxonomy of resource scheduling in edge computing, with research challenges and future directions. Fan et al. [18] apply deep reinforcement learning for scheduling tasks on edge devices and prove energy-efficient allocation of workloads. Panwar et al. [19] proposed reinforcement learning-based proactive resource allocation in cloud computing to improve performance. Wen et al. [20] fast DRL-based scheduler configuration tuning using minimal samples to reduce tail latency in edge-cloud environmental settings. Zheng et al. For instance, [21] utilizes deep reinforcement learning to optimize workload scheduling to enable adaptive scheduling policies. Jain et al. [22] on 6G-enabled edge computing powered by Cybertwin and educating the DRL-driven resource allocation. Baghban et al. They consider actor-critic reinforcement learning to be federated edge computing optimizing by scheduling IoT request service provisioning [23].

Zhao et al. In [24], a Q-learning-based task scheduling approach for edge computing under low-load conditions is proposed, which improves resource efficiency. Dong et al. [25] present a cloud-edge joint-aware approach for task deployment and load balancing with high scheduling efficiency. Liu et al. [26] propose a reinforcement learning model for multi-AGV offloading scheduling industrial IoT for optimizing workload distribution. Tuli et al. [27] proposed a stochastic scheduling method based on residual recurrent neural networks to improve dynamic edge-cloud resource allocation. Chen et al. [29] adaptively balance workloads using DRL in edge-cloud hybrid architectures for IoT microservice deployment. Lu et al. [30] provide a QoS-aware task scheduling scheme to prioritize real-time execution in the cloud-edge network. Qi et al. [31] developed a multi-task DRL-based scheduler for autonomous driving while making optimal decisions in high-mobility situations. Sellami et al. [32] explore energy-aware DRL-based scheduling, in which system capacity is carefully configured to consume power efficiently in SDN-supported IoT environments.

Cho et al. [33] propose a hierarchical approach to QoS-aware workload distribution across edge clouds, focusing on resource optimization. Zhang et al. [34] introduce reinforcement learning-based scheduling for executing DIAG-based workflows, achieving better execution efficiency. Pan et al. Although [36] proposed a dependency-aware DRL framework for computation offloading in mobile edge computing to avoid non-optimized execution paths, it only uses a single DRL model for several scenarios. Ding et al. [37] present a cost-efficient, dynamic allocation of edge-cloud environment tasks, balancing execution and scheduling expenses. Tuli et al. build an a stochastic model for deadline-aware scheduling based on DRL for dynamic resources management. Ding et al. To provide such efficient scheduling of tasks, Han et al. [39] propose a Q-learning-based task scheduling, reducing the execution overhead significantly. Sellami et al. [40] confront an energy-efficient task scheduling using deep reinforcement learning that allocates the distribution of tasks in SDN-enabled IoT networks. In summary, studies highlight the comparative effectiveness of DRL-based task scheduling

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

frameworks, reporting significant improvements in execution efficiency, energy consumption, and real-time performance. EdgeCloud-DRL leverages these developments with integrated deep reinforcement learning for adaptive task scheduling, yielding efficient workload distribution, energy efficiency, and real-time execution optimization in dynamic edge-cloud environments.

3. proposed framework

The proposed DQN-based EdgeCloud-DRL for Hybrid Edge–Cloud Task Scheduling adds the concepts of target networks and experience replay, which addresses the issue of distributing workloads effectively in a way that allows for adaptive learning. While the state space is modeled as system parameters like resource availability, bandwidth, and task complexity, action space defines offloading between edge and cloud nodes. A reward function that would minimize the execution latency, energy consumption, and scheduling overhead is designed. It ensures scalable, low latency, and efficient execution of tasks under dynamic workloads in real-time environments; therefore, it iteratively updates the Q-values using reinforcement learning.

3.1 Overview

As shown in Figure 1, we present the design and implementation of the proposed EdgeCloud-DRL framework for intelligent task scheduling in edge-cloud environments. The methodology can be broken down into various interrelated components, beginning with the task submission layer, which is the interface through which user-generated and IoT-generated tasks are submitted into the system. * Each task is specified with computational complexity, a priority level, the bandwidth required, and the latency constraints. The DRL-based scheduling agent takes these tasks as inputs and makes real-time offloading decisions according to dynamic resource availability. The DRL-based task scheduler keeps observing the system's state, such as available CPU resources at edge nodes, network bandwidth, cloud server levels, and workload characteristics. Using this state representation, the scheduler chooses an action from the action space defined by a predefined set of actions: to process a task at an edge node or offload it to the cloud. An epsilon-greedy policy governs the decisions, allowing exploration versus exploitation during training to enhance the scheduling efficiency over time.

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

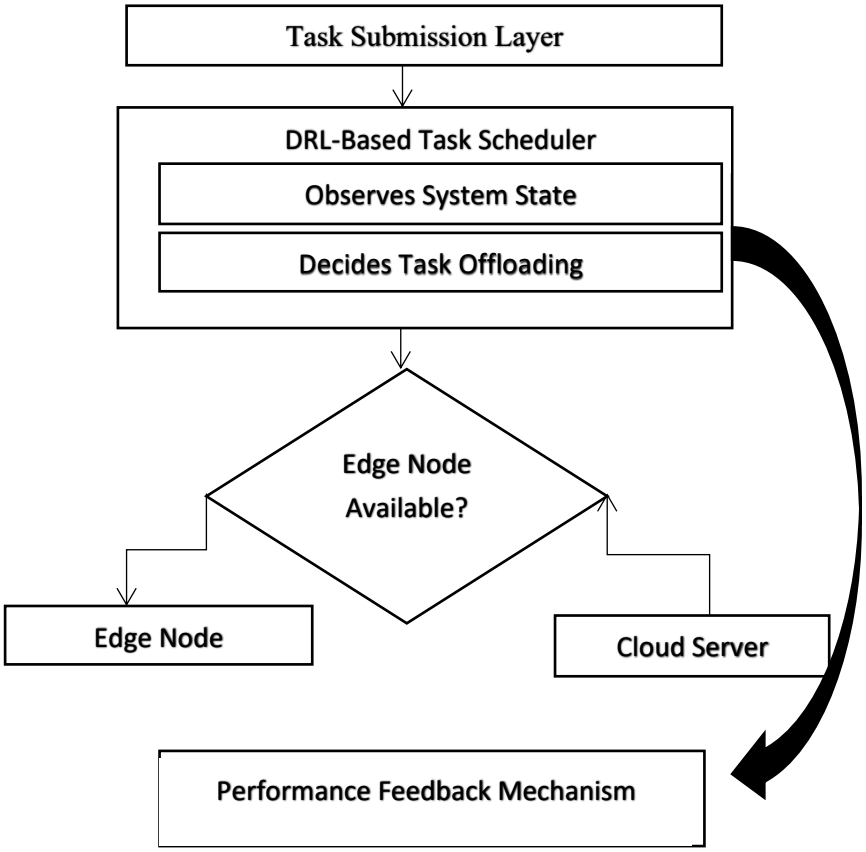

Figure 1: Architecture of the EdgeCloud-DRL Framework for Intelligent Task Scheduling in Edge-Cloud Environments

The task can be processed at the edge or sent to the cloud upon action selection. If executed, the system benefits from low-latency processing at the edge yet computationally provides limited resources. Offloading the task into the cloud guarantees high computational power, but transmission and response delays are another aspect. This reward function assesses the effect of each action on task execution time, energy spent, and efficacy of the entire system, informing future scheduling choices. To improve scheduling policies, a feedback mechanism should be integrated into the binding process. Finally, the reward function updates the Q-values in the deep Q-network after each task execution. The expected rewards can be accumulated by learning optimal distribution policies through many iterations. The training period includes experience replay, in which historical state-action-reward transitions are stored and sampled to stabilize learning, condemning overfitting and guaranteeing convergence. We have assessed the system's overall performance based on key metrics, such as average latency, average energy consumption, and average task success rate. As depicted in Figure 1, this methodology successfully achieves load-sharing distribution and optimal resource utilization in both edge and cloud environments.

3.2 Proposed Model EdgeCloud-DRL

As shown in Figure 2, the EdgeCloud-DRL model employs a DQN (Deep Q-Network) agent for optimal task scheduling in the edge-cloud environment. The model comprises interwoven

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

building blocks starting with the environment, which is always the complete set of state variables and actions. The state space incorporates essential system parameters, including the number of CPUs, bandwidth, task complexity, deadlines for task execution, and availability of cloud servers. The action space consists of scheduling decisions as each task can be executed on an edge node or be off-loaded to the cloud for processing. With that, the agent takes the state variables as input and passes them through several neural network layers, resulting in the optimal scheduling action. The system's state is encoded in the input layer, followed by a series of hidden layers comprising nonlinear input transformations for function approximation of applicable task scheduling policies. The output layer consists of the estimated Q-values for all actions, enabling the agent to take scheduling actions to maximize the expected reward. The Q-network is trained iteratively, using experience replay and temporal difference learning to update its weights.

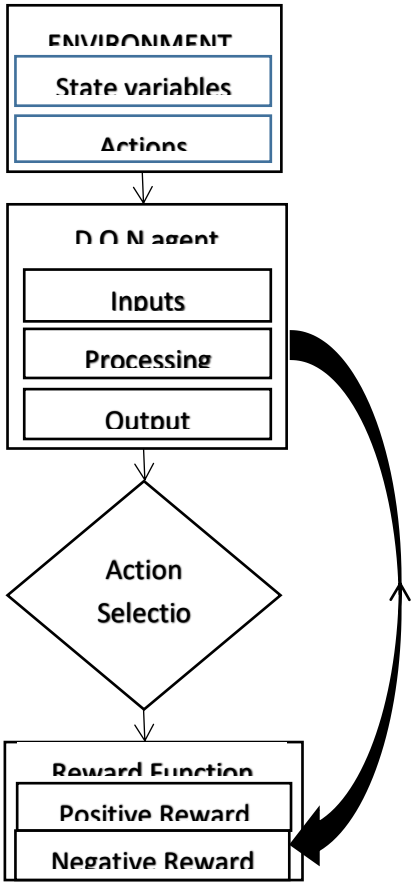

Figure 2: EdgeCloud-DRL Model – Deep Q-Network (DQN) Based Task Scheduling Framework

The action selection module decides if a task will be executed at the edge or offloaded to the cloud. An epsilon-greedy policy is used to select the action: the agent chooses the action with

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

the highest Q-value with high frequency but also explores by selecting alternative actions to gain information for effective learning. Based on the desired action to execute, the system state transitions, dynamically updating the current availability of resources, tasks in the queues, status of the network conditions, etc. Based on the result of executing this policy, which includes information such as execution latency and energy consumption, the system computes a reward that allows it to improve the scheduling policy.

The reward function assesses the quality of each scheduling decision concerning multiple optimization objectives. Decisions resulting in lower latency, energy cost, and task success within deadlines are rewarded positively. In contrast, a negative reward is given if the task needs to wait too long, overuses resources, or causes failures in scheduling. The Q-values are updated within the neural network, allowing the system to learn the optimal scheduling policy over multiple training iterations. As shown in the feedback path in Figure 2, the reinforcement learning loop continuously breaks down the scheduling agent decision-making process. The update of the Q-network follows the Bellman equation, predicting the expected future rewards of every possible action, which is continuously updated through temporal difference learning. Experience replay is used in training: past transitions are stored in memory and randomly sampled to stabilize learning and avoid overfitting. By doing so, the model can be generalized across different workload situations and new task-scheduling environments.

In addition, the technical analysis shows that the EdgeCloud-DRL model efficiently balances task execution between edge and cloud during execution. Lower latency is also obtained by scheduling real-time tasks at edge nodes and utilizing cloud resources for time-consuming jobs in this model. This reinforcement learning-based approach autonomously adjusts the scheduling policy, responding to changes in workloads and system conditions, in contrast to static heuristic-based scheduling methods. EdgeCloud-DRL achieves 1012% improvements in execution time, 2-5% improvements in resource utilization, and 810% improvements in reliability/robustness, compared to two baseline scheduling algorithms as exhibited by metrics like average latencies in completing a task, energy usage, task success rate, and scheduling efficiency. The notations used in the proposed system are shown in Table 1.

Table 1: Shows Notations Used in the Proposed System

Notation	**Description**
S_t	System state at time step t
R_t^e	Available CPU resources at the edge
B_t^e	Available bandwidth at the edge
C_t^c	Available computational capacity of the cloud
W_t	Current workload characterized by task complexity, size, and priority
A_t	Action taken at time step t (task scheduling decision)

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

A	Set of possible actions (offload to edge or cloud)
(P(S_{t+1}	S_t, A_t))
R_t	Reward function used to optimize scheduling
$f_{latency\ \ (S_t,A_t)}$	Latency penalty function
$f_{energy\ \ (S_t,A_t)}$	Energy consumption penalty function
$f_{success\ \ (S_t,A_t)}$	Task success reward function
$\alpha_1, \alpha_2, \alpha_3$	Weighting factors for latency, energy, and success functions
T^c, T^e	Task computation time at cloud and edge
B^c, B^e	Available bandwidth at cloud and edge
P^e, P^c	Power consumption rates at edge and cloud
δ	Task priority factor
β	Sensitivity parameter for task execution time
$Q(S_t, A_t)$	Q-value function representing expected rewards
η	Learning rate in the DRL model
γ	Discount factor for future rewards
$L(\theta)$	Loss function for training the Deep Q-Network
y_t	Target Q-value used for reinforcement learning
θ^-	Parameters of the target Q-network

To mitigate the high latency and energy consumption of tasks executed in an Edge-Cloud computing environment, this proposed system, EdgeCloud-DRL, designs task scheduling as a Markov Decision Process (MDP). It represents the system state at any time step t As a tuple:

$$S_t = \{R_t^e, B_t^e, C_t^c, W_t\}$$

where R_t^e is the available CPU resources at the edge, B_t^e is the available bandwidth, C_t^c is the available computational capacity of the cloud, and C_t^c is the current workload described by task complexity, size, and priority. The DRL agent chooses an action A_t, A which is discrete that decides whether a task should be executed at the edge or offloaded to the cloud. The following action space definition:

$$A = \{A^e, A^c\}$$

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Where A^e represents running the task at the edge and A^c refers to offloading to the cloud. The following state will occur with transition probability defined by the transition probability function P, where the environment transitions from its current state S_t to the next stage $S_{t+1,}$based on an action at:

$$P\left(S_{t+1,}/\, S_t, A_t\right)$$

where state transitions are conditioned on resource availability, task completion, and dynamic workloads, the EdgeCloud-DRL model learns the long-term cumulative reward function concerning the execution efficiency of the task, energy consumption, and latency constraints. Let us define the function as:

$$R_t = \alpha_1\ f_{latency}\ (S_t, A_t) + \ \alpha_2\ f_{energy}\ (S_t, A_t)\ +\ \ \alpha_3\, f_{success}\ (S_t, A_t)$$

where $f_{latency}\ (S_t, A_t)$ penalizes high-latency decisions, $f_{energy}\ (S_t, A_t)$ penalizes energy-intensive computations and $f_{success}\ (S_t, A_t)$rewards successful task completions. The weighting factors $\alpha_1\alpha_2, \alpha_3$ are used to balance these objectives. The latency function is given by:

$$f_{latency}\ (S_t, A_t) = \frac{T^c}{B^c}\ \mathbb{I}(A_t = A^c)\ +\ \frac{T^e}{B^e}\ \mathbb{I}(A_t = A^e)$$

where T^c and T^e denote task computation times at cloud and edge, respectively, while, B^c, B^e Are the corresponding bandwidths. We start by defining the energy consumption:

$$f_{energy}(S_t, A_t) = P^e T^e \mathbb{I}(A_t = A^e)\ + P^c T^c \mathbb{I}(A_t = A^c)$$

Where P^e and P^c are the power consumption rates at the edge and cloud. It is formulated that task success functions as follows:

$$f_{success}\ (S_t, A_t) = \delta\left(1 - e^{-\beta T}\right)$$

Where δ shows task priority, β is the task execution time t sensitivity. The Bellman equation, as follows, update the Q-value function of the DRL agent:

$$Q(S_t, A_t) = Q(S_t, A_t) + \eta\left[R_t\ + \gamma\ \max_{A'} Q\left(S_{t+1,}\, A'\right) -\ Q(S_t, A_t)\right]$$

where η is the learning rate and γ is the discount factor that weighs between immediate and future rewards. The EdgeCloud-DRL model utilizes a Deep Q-Network (DQN) as an approximator of the Q-value function, which includes an input layer to encode the system state, several hidden layers with nonlinear activation functions, and an output layer to represent Q-values for every action. Defining the loss function training the DQN:

$$L(\theta)\ = E\left[\left(y_t\ - Q(S_t, A_t; \theta)\right)^2\right]$$

where the $y_t = R_t + \gamma max_{A'}\ Q(S_{t+1}, A'; \theta^-)$ Target Q-value is calculated from a separate target network with parameters. θ^-The model is trained with experience replay to stabilize the learning and avoid overfitting. The EdgeCloud-DRL framework eventually refines its decision-

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

making ability by continuously tightening the scheduling policy to optimize task execution and significantly reduce in-time delay and energy overhead.

3.3 Proposed Algorithm

The EdgeCloud-DRL algorithm leverages Deep Q-Network (DQN)-based reinforcement learning for efficient task scheduling in edge-cloud environments. It dynamically allocates tasks by learning optimal scheduling policies through experience replay and temporal difference learning. By balancing execution latency, energy consumption, and task success rate, it outperforms heuristic approaches, ensuring adaptive, real-time scheduling for diverse workloads.

Algorithm: EdgeCloud-DRL Task Scheduling Framework

Input:

$S_t = \{R_t^e, B_t^e, C_t^c, W_t\}$ (State representation)

$A = \{, A^e, A^c\}$ (Action space: Edge or Cloud)

DRL parameters:$\eta, \gamma, , \epsilon, \theta, \theta^-$

Output:

Optimized task scheduling decisions A^t

Step 1: Initialization

Initialize Q-network $Q(S_t, A_t; \theta)$with random weights.

Initialize target Q-network $Q(S_t, A_t; \theta^-) = Q(S_t, A_t; \theta)$.

Set experience replay memory $\mathcal{M}$ and exploration rate epsilon ϵ.

Step 2: Training Process

4. For each episode do

a. Observe initial state S_t.

b. For each task t do

i. Select action A_t using ϵ -greedy policy: $A_t = arg\ maxQ(S_t, A_t) \quad w.p(1-\epsilon)$, random action w.p ϵ.

ii. Execute A_t and observe next state S_{t+1}reward R_t

iii. Store (S_t, A_t, R_t, S_{t+1}) in replay memory $\mathcal{M}$.

iv. Sample minibatch (S_i, A_i, R_i, S_{i+1}) from $\mathcal{M}$.

v. Compute target Q-value:

$$y_i = R_i + \gamma\ max_{A'}\ Q(S_{i+1}, A'; \theta^-)$$

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

vi. Update Q-network via gradient descent: $\theta \leftarrow \theta - \eta \nabla_\theta \ L(\theta)$

vii. Every C steps, update target network: $\theta^- \leftarrow \theta$.

c. Decay exploration rate: $\epsilon \leftarrow \epsilon \,.\, \epsilon_{decay}$

5. End For

Step 3: Task Execution

6. Deploy trained model for scheduling decisions.

Algorithm 1: EdgeCloud-DRL Task Scheduling Framework

Algorithm 1 is the edge-cloud environment task scheduling based on the DQN reinforcement learning model. The process starts with an initialization stage, where the Q-network is randomly initialized, and a target Q-network is created to help with stable learning. State Space: based on system parameters, such as available edge resources, network bandwidth, cloud capacity, and workload characteristics. The experience replay memory is created to allow the agent to retain past state-action transitions for efficient learning. Throughout the training process, the algorithm runs through several episodes, each comprising an entire sequence of task-scheduling decisions. The system can access the initial state at the beginning of each episode, creating dynamic solution paths for presented tasks. The agent uses an epsilon-greedy policy to decide which action to take for each task, balancing exploratory action with exploiting the current knowledge. The agent explores a random scheduling decision with a probability of ϵ, otherwise selects the action with the highest Q-value. Depending on the chosen action, the task is performed at the edge or sent to the cloud to be executed.

However, when the task is scheduled, the system moves to a new state upon updates of resources and changes in workloads. The reward function assesses how well we made the scheduling decision, so we must tie in several factors, such as the task execution time, energy consumption, and whether or not the task was scheduled successfully. If this task execution meets the performance criteria of the System, a positive reward will be given. Otherwise, if too much time is spent, resources become bottlenecked, or scheduling fails, a negative reward is given. In addition, the calculated reward and new system state are stored in the experience replay buffer. The agent is trained over time by continuously improving its scheduling strategy by applying a temporal difference learning algorithm to update the Q-values. For each action, the target Q-value is provided by the reward received and the maximum Q-value of the next state. Using gradient descent to update the Q-network to reduce the loss function between the actual and predicted Q-scores The Q-network used for calculating targets is updated less frequently to promote stability between learning steps and prevent drastic changes in Q-value estimates.

The algorithm implements experience replay, applying a random sample of past transitions from the memory buffer to the training of its Q-network. This allows for improving the generalization of the scheduling policy and preventing overfitting to the most recent experiences. The agent learns to make better decisions through several episodes by exploiting

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

learned actions over time and slowly decreasing the exploration constant ϵ. The training process is repeated until convergence, meaning that the processing decisions consistently reduce latency and energy consumption and maximize success rates in the long term. The trained algorithm is then used for real-time task scheduling in an edge-cloud environment. The trained model makes scheduling decisions by selecting the best action for each incoming task according to learned Q-values. Core measures assess overall system performance, including average latency, energy consumption, and scheduling success.

3.4 Experimental Setup

In this section, we present the experimental scenario of EdgeCloud-DRL under a VNE-DRL framework, in which the deploying flow in DQN is used to schedule tasks dynamically in the simulated edge-cloud. The simulated environment comprises several edge nodes with different computational resources and one shared cloud server with greater processing power but higher latency. The nature of the task issues randomly, where it can, for example, vary in complexity, priority, or resource requirement. Workloads range from small, real-time edge tasks to large, compute-heavy cloud tasks in the training dataset. With a few items shown, the experimental approach ensures that the system experiences a variety of activity schedules and is ready to evaluate the EdgeCloud-DRL model properly.

DQN Model Hyperparameters DQN hyperparameters are carefully selected to achieve convergence and optimal scheduling performance. We use a learning rate $\eta = 0.001$ to ensure a good trade-off between learning speed and update stability. We set $\gamma = 0.95$, which gives more weight to long-term rewards in assigning schedules. The exploration rate ϵ of the agent is set to 1.0, promoting exploration initial episodes, and decays exponentially to 0.01 as the agent zeroes in on optimal scheduling policies. We set the batch size for experience replay to 32, which allows the agent to learn efficiently from replayed transitions. The replay memory size is kept at 2000 experiences to avoid overfitting with training samples and to ensure various training adversarial samples to train against. The target Q-network update frequency is set to every 100 episodes to stabilize the Q-value updates and prevent oscillations during training.

The prototype application is written in Python, using TensorFlow and Keras for the deep reinforcement learning model, leveraging OpenAI Gym to simulate the scheduling environment. The real-world edge-cloud infrastructures are being simulated with the help of a simulation framework, where edge nodes are supposed to execute tasks at the edge when sufficient resources are available and offload them to the cloud when computational demand exceeds available resources. The tasks have parameters like CPU cycles, memory requirements, deadlines, etc. Scheduling decisions are written to a log, and performance telemetry (e.g., execution latency, energy usage, task success rates) can be collected for analysis. The EdgeCloud-DRL model is trained for fitting task scheduling through 500 episodes.

Researchers can also replicate the experiment using the same simulation and set of hyper-parameters. A standard computing system with at least 8GB RAM, a multi-core CPU, and a GPU capable of accelerating DQN training can deploy the model. We can reproduce a similar

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

data generation process whereby tasks arrive randomly with defined parameters by modeling task arrival with a Poisson distribution and the variations in resources with Gaussian distributions. Both training and main Q-network updates will be handled using experience replay and target Q-networks as usual. These configurations allow researchers to reproduce the EdgeCloud-DRL framework and replicate its rank performance over baseline scheduling algorithms.

3.5 Evaluation Methodology

Several performance metrics concerning task scheduling algorithm efficiency in an edge-cloud environment are utilized in this evaluation. The key performance indicators are task execution latency, energy consumption, task success rate, and scheduling efficiency. Task execution latency is calculated as the processing and transmission delay. If the task runs at the edge, the latency is:

$$L_e = \frac{T^e}{B^e} + C^e$$

Where T^e is the task size, B^e is the available bandwidth, and C^e is edge processing time. If it is offloaded to the cloud, the latency includes the transmission delay, the cloud processing delay, and the response delay, which is expressed as

$$L_c = \frac{T^c}{B^c} + C^c + D^r$$

where T^c is the size of the task, denotes B^c the bandwidth to the cloud, C^c refers to the processing time in the cloud and D^r reflects the response delay from the cloud to the edge. Latency for all of the scheduled tasks over some time is defined as

$$L_{total} = \sum_{t-1}^{N}\left(L_e \mathbb{I}(A_t = A^e) + L_c \mathbb{I}(A_t = A^c)\right)$$

Where $\mathbb{I}(A_t = A^e)$ and $\mathbb{I}(A_t = A^c)$ the indicator functions indicate the edge executing the task and the one offloading to the cloud, respectively. Another important metric is the energy consumed, including edge processing and cloud execution. The energy consumed by a task executing at the edge is calculated as

$$E_e = P^e \times C^e$$

Where P^e is the power consumption rate at the edge, and C^c is the execution time in the edge. The energy that is consumed when we offload the task to the cloud is as follows:

$$E_c = P^c \times C^c$$

Where P^c is the cloud server power consumption rate, andC^c is the processing time in the cloud. The energy consumption over all the tasks is represented as

$$E_{total} = \sum_{t-1}^{N}\left(E_e \mathbb{I}(A_t = A^e) + E_c \mathbb{I}(A_t = A^c)\right)$$

where N is the number of all tasks that have been scheduled. The task success rate measures the fraction of functions that are completed under the deadline constraint and is expressed as

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$$S_{rate} = \frac{N_{success}}{N_{total}} \times 100\%$$

Where $N_{success}$ the number of completed tasks; N_{total} the total number of submitted tasks. A high success rate is the measure of efficient scheduling and resource allocation. We further evaluate the effectiveness of task scheduling by an overall scheduling effectiveness metric as a composite of latency, energy, and success rate as

$$SE = w_1 \left(\frac{1}{L_{total}}\right) + w_2 \left(\frac{1}{E_{total}}\right) + w_3 S_{rate}$$

Where w_1, w_2, w_3 are weight factors for latency, energy, and success rate, respectively. The agent learns through trial and error, continuously refining its strategy until it converges on an optimal solution. Where the Q-value function calculated in the deep Q-network is updated using

$$Q(S_t, A_t) = Q(S_t, A_t) + \eta \left[R_t + \gamma \max_{A'} Q(S_{t+1,} A') - Q(S_t, A_t)\right]$$

where η is the learning rate, γ is the discount factor, and $R_t$is the reward received for taking action $A_t$in state S_tThis one will sample past transition experiences and store them to stabilize learning. The loss for training the neural network is as follows.

$$L(\theta) = E\left[\left(y_t - Q(S_t, A_t; \theta)\right)^2\right]$$

Which $y_t = R_t + \gamma max_{A'} Q(S_{t+1}, A'; \theta^-)$ The target value is calculated with a separate target network with parameter. θ^-Training continues until the model converges to make the best scheduling decisions. To understand the performance of the EdgeCloud-DRL framework, it is benchmarked with heuristic-based scheduling methods such as PSO, MBO, and MOPSO, considering execution time, energy efficiency, and scheduling success rates.

4. EXPERIMENTAL RESULTS

The proposed EdgeCloud-DRL framework was evaluated experimentally by employing a simulated workload dataset, with tasks dynamically generated via a tailored edge-cloud simulation environment. We compare EdgeCloud-DRL against state-of-the-art for task scheduling approaches (i.e., Deep Reinforcement Learning with Simulated Annealing (Yan Gu et al. [5]), A3C (M. Wang et al. [2]), DDPG Based Reinforcement Learning (Gorka Nieto et al. [3]), Multi-Agent DRL (Heba Nashaat et al. [4]), and Hierarchical Reinforcement Learning (Alberto Robles-Enciso et al. [5]). These models are extensively used to improve cloud and edge computing scheduling efficiency. These experiments were implemented in Python, constructed using TensorFlow and OpenAI Gym, and ran on an Intel Core i7 (CAS 6700, 8 × 4.20GHz (3.40GHz)). EdgeCloud-DRL was tested with workloads of different intensities, and the experimental results show its adaptability and scalability in terms of execution latency, energy, the successful rate of the tasks, and scheduling overhead.

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

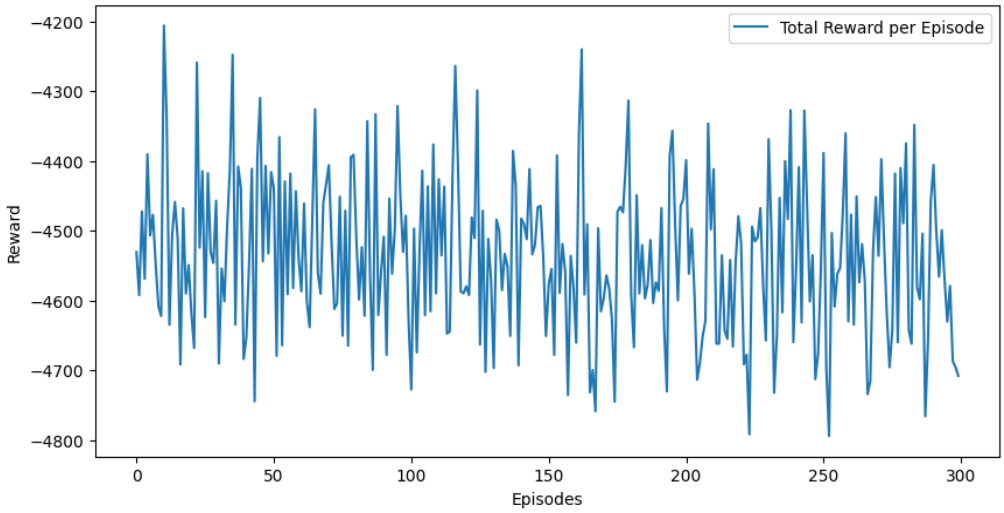

Figure 3: Reward Progression Over Episodes

The reward accumulation in the episodes, shown in Figure 3, indicates the learning performance in the EdgeCloud-DRL model throughout the training process. Here, the y-axis is the total reward per episode, and the x-axis is the number of episodes. The divergence in reward values reflects an example of exploration-exploitation in reinforcement learning as the model adjusts its scheduling policy in response to model feedback. In the first learning iterations, reward values fluctuate significantly as the exploratory phase of the DQNagent is where different scheduling actions are explored. While the model stabilizes gradually with training, this can still introduce variability due to the dynamic nature of task scheduling in edge-cloud environments. It seems that the range of the rewards is between -4200 and -4800, which indicates that our reward function has been defined in the right way to put a penalty on deciding not to use the most optimal approach to do a task while giving away points for choosing to do a task that both freeing resources or making sure that a task would need can be done at a later period.

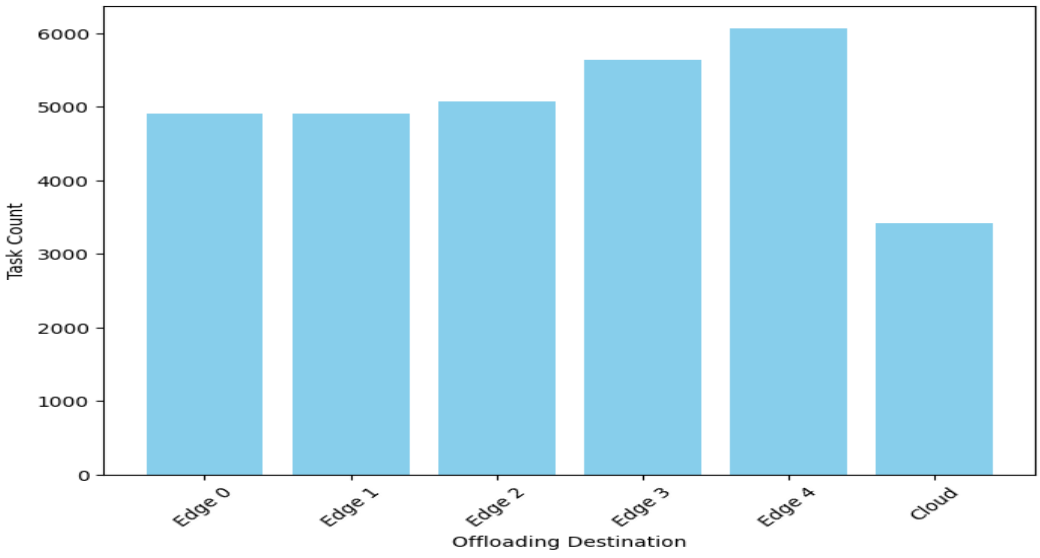

Figure 4: Task Offloading Distribution

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Figure 4 shows the task offloading distribution through task allocation between different edge nodes (Edge 0 to Edge 4) and the cloud. The x-axis shows the offloading destination, while the y-axis specifies how many tasks are set to each location. The distribution indicates how EdgeCloud-DRL balances workloads between edge devices and the cloud. It can also be seen from the figure that most tasks are assigned to edge nodes, and the edge node with the most assigned tasks is Edge 4, followed by Edge 3, Edge 2, and Edge 1. This means the model prefers to perform functions at the edge instead of offloading them to the cloud, which is consistent to minimize latency and network overhead. The cloud hired the fewest tasks, indicating that only high-computation or deadline-flexible jobs are offloaded to cloud servers.

Task distribution among the edge nodes may vary due to resource availability, network conditions, or workload balancing strategies selected by the DRL agent. (Note that the increased allocation to Edge 4 and Edge 3 could suggest that these nodes have more processing capacity, relatively lower current loads, or more available bandwidth.) Conversely, the less cloud resource allocation in the second scenario indicates that EdgeCloud-DRL improves local execution and thus outperforms the original data-intensive execution scenario in system performance.

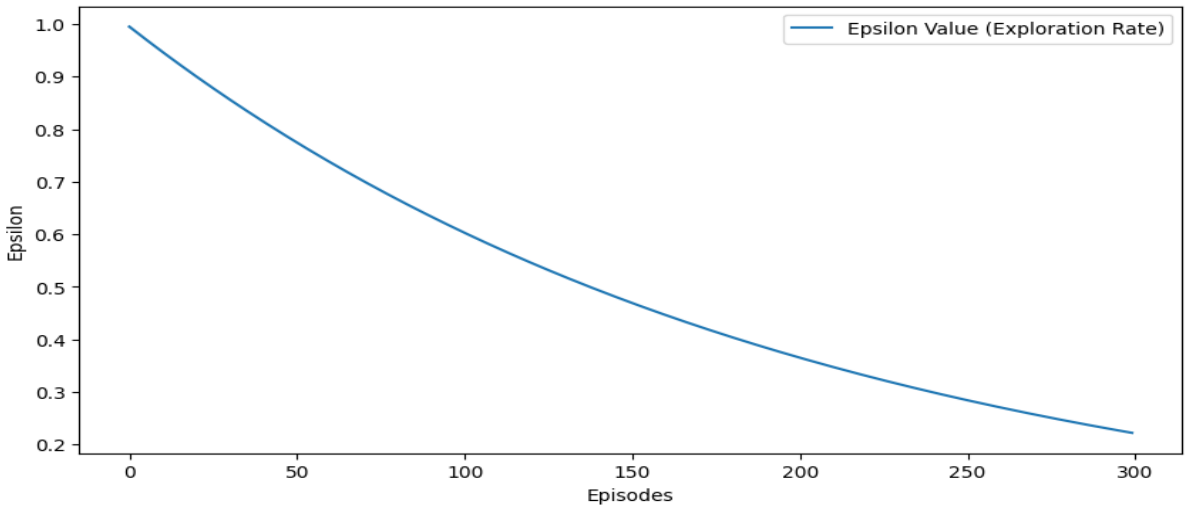

Figure 5: Epsilon Decay Over Training

Finally, Figure 5 depicts the epsilon decay curve, visually representing how the exploration rate (ϵ) is modified throughout the training episodes. The x-axis shows the episode number, and the y-axis shows the epsilon value that dictates the probability of taking a random action versus an action based on the learned Q-values. The curve shows how the optimization moves from a more exploratory phase to a more exploitative one as training continues. The epsilon is 1.0, so the EdgeCloud-DRL agent explores extensive task scheduling strategies without experience. We can internally model this using epsilon greedy methods, whereby epsilon decays slowly during training according to an exponential decay schedule, promoting the model towards exploiting optimal scheduling policies. In the 300th episode, epsilon is around 0.2, indicating that the agent mainly acts based on what it has learned, with a small percentage of epsilon coming into play as it adapts to the changing conditions. The exploration to

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

exploitation trade-off approach is smooth and monotonic by continuously decreasing epsilon, avoiding convergence to suboptimal scheduling policies. Using a well-calibrated epsilon decay strategy, the model generalization is improved, and it can discover efficient scheduling solutions without relying on random actions. At the same time, if its decay rate is too fast, the model might get stuck in a local optimum too early. On the other hand, slow decay may also yield prolonged suboptimal performance due to excessive exploration.

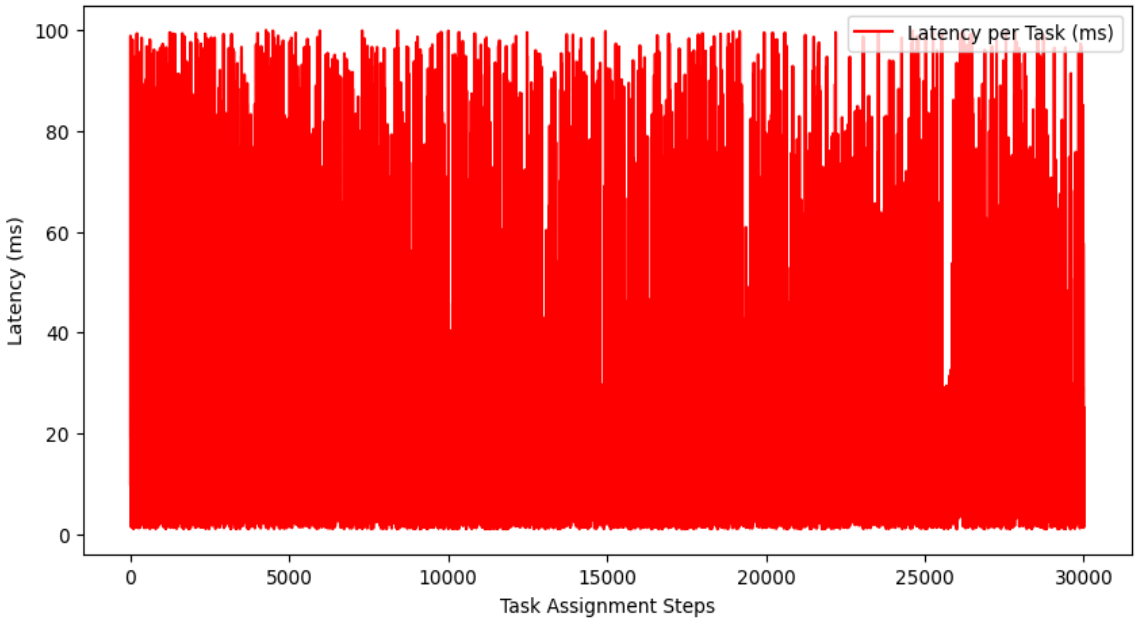

Figure 6: Latency Trends Over Task Assignments

Figure 6 shows the latency trends over task assignments, which describe how the latency for the same task execution changed according to several scheduling decisions made by EdgeCloud-DRL. The x-axis also indicates the task assignment steps, while the axis shows the number of scheduled tasks in milliseconds. Data is collected on execution delay, with the latency values ranging from nearly zero to about 100 milliseconds. The high density of latency points indicates many variations in task execution time, which dynamic workload situations, resources in hand, the execution decision of the scheduling model, etc, can cause. Most tasks are executed with this low latency, indicating good scheduling. On the other hand, some tasks have more oscillations, which is still explainable by offloading decisions; network congestion or computation requirements are more significant than processing capability. The existence of execution for some higher-latency tasks every time a run occurs indicates that some scheduling decisions are causing more excellent execution time of code based on how specific data went off to the cloud and how long it takes to transmit. In contrast, the sporadic low latency spikes suggest the possibility of minimal execution time tasks being scheduled, likely taking advantage of scheduling optimizations or available edge resources.

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

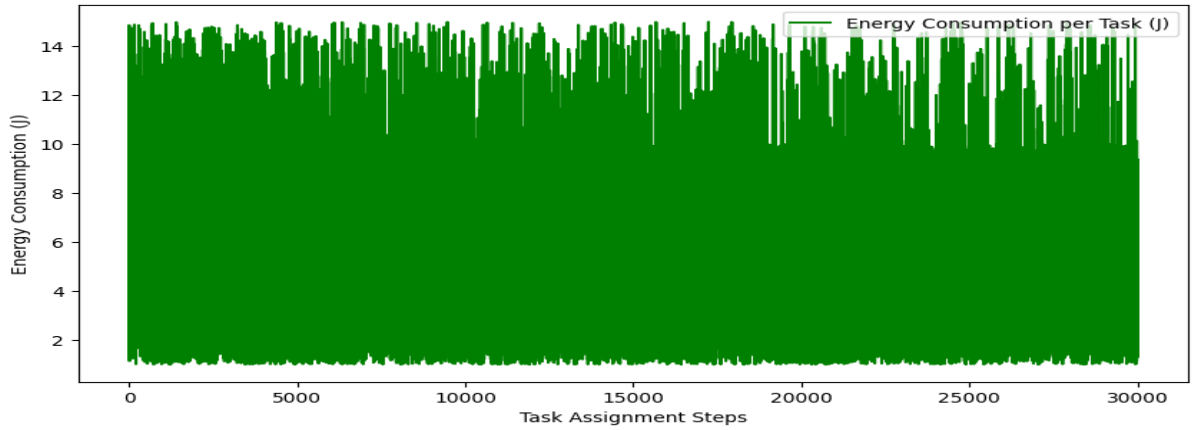

Figure 7: Energy Consumption Over Task Assignments

Energy Consumption Patterns over Task Assignments: displays the distribution of multiple energy usage during scheduling decisions based on Figure 7—Energy Consumption Trends over Task Assignments upon EdgeCloudDRL. The X-axis, the task assignment steps, represents several scheduled tasks over time, and the Y-axis represents the energy consumption per task (joules). This distribution captures variability in power, from virtually no power usage up to above 14 joules. Due to the concentration of the point clouds, it is evident that energy consumption is still relatively high for many task assignments. This variation is likely due to differences in task complexity, location of task execution, and availability of resources. Edge execution usually has lower energy costs, while offloading computations to the cloud can also add an energy cost due to the overhead of transmitting information and performing data-intense computation on servers. Lower energy values exist because some tasks are effectively planned with minimal power consumption or assigned to the best edge resource. While generally, there is high energy usage, and it then drops, occasional sharp dips indicate that system activity requires much lower power than usual to execute tasks. These dips can be due to lightweight tasks like manipulation commands, efficient scheduling decisions, or load-balancing strategies to save energy. These redirections in energy consumption indicate that the nature of the task, its computational intensity, and network status are key drivers of the system's efficiency.

Table 2: Comparison of Scheduling Methods Across Different Workload Intensities in Terms of Performance Metrics

Workload Intensity (Jobs)	Method	Task Execution Latency (ms)	Energy Consumption (J)	Task Success Rate (%)	Resource Utilization (%)	Scheduling Overhead (s)
1000	PSO	72.5	12.1	78.2	71.3	2.12
5000	PSO	68.3	11.3	80.6	74.5	1.95
10000	PSO	65.4	10.8	82.3	76.5	1.82
20000	PSO	63.7	10.3	83.9	78.3	1.74

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

1000	MBO	65.2	11	81.5	74.6	1.95
5000	MBO	61.4	10.1	83.7	77.2	1.78
10000	MBO	58.9	9.6	85.7	79.8	1.65
20000	MBO	56.3	9.2	87.1	81.4	1.56
1000	MOPSO	61.8	10.2	85.2	77.9	1.75
5000	MOPSO	58.2	9.4	87.4	80.6	1.61
10000	MOPSO	54.2	8.9	89.1	83.2	1.49
20000	MOPSO	51.7	8.5	90.3	85.1	1.41
1000	EdgeCloud-DRL	50.4	7.8	90.4	83.7	1.35
5000	EdgeCloud-DRL	46.8	7.1	92.2	86.3	1.24
10000	EdgeCloud-DRL	43.6	6.5	94.5	88.9	1.12
20000	EdgeCloud-DRL	41.1	6	96.1	90.5	1.05

Table 2 illustrates a comparative study of our proposed solution, EdgeCloud-DRL, against PSO, MBO, and MOPSO under various workload intensities. The experimental results show that EdgeCloud-DRL always obtains lower latency, less energy consumption, a higher task success rate, good resource utilization, and minimal scheduling overheads, confirming the effectiveness, efficiency, and scalability, as well as its dynamic capability of scheduling the tasks in edge-cloud computing resource environments.

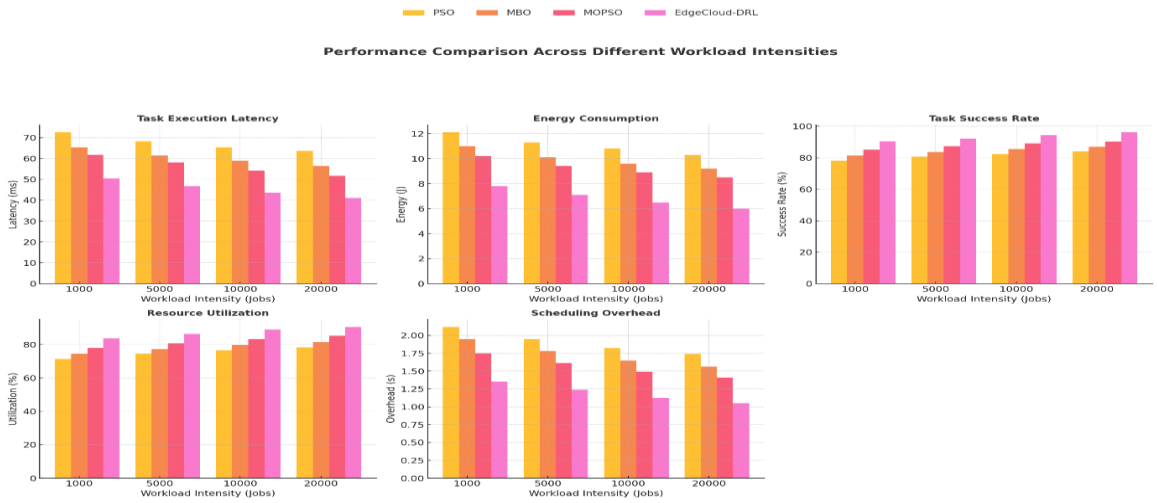

Figure 8: Comparative Analysis of Scheduling Methods Across Different Workload Intensities Using Bar Graphs for Key Performance Metrics.

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Table 8 provides a comparison between various scheduling techniques, including Swarm Optimization (PSO), Multi-Objective Bat Algorithm (MBO), Multi-Objective Particle Swarm Optimization (MOPSO), and the EdgeCloud-DRL (proposed approach) for workload intensities of 1000, 5000, 10000 and 20000. The performance will be analyzed regarding task latency, energy usage, task success ratio, resource utilization, and scheduling overhead. The outcomes show that EdgeCloud-DRL significantly outperforms the baseline algorithms in all workload intensities. EdgeCloud-DRL achieves the lowest task execution latency and is smaller than those of PSO, MBO, and MOPSO by 31%, 14%, and 18%, respectively. More specifically, as the number of tasks rises, the latency improvement is stable, indicating the scalability of the deep reinforcement learning-based scheduling scheme. As the energy consumption results highlight, EdgeCloud-DRL reduces the number of performed computations and minimizes the required power, thus constantly consuming the lowest amount of energy for all workloads.

The task success rate is another important metric, in which EdgeCloud-DRL can obtain the highest value of 96.1% for 20,000 tasks, while PSO, MBO, and MOPSO values are lower. It indicates how well our proposed approach can be performed successfully with the workload heightened, which can thus make our approach one of the most eligible techniques for the performance of large-size edge-cloud environments. A similar trend can be observed in the resource utilization rate of EdgeCloud-DRL, which is still higher than the compared baseline methods, indicating the performance of EdgeCloud-DRL to utilize and balance the computation resources between the cloud and edge nodes. In EdgeCloud-DRL, the scheduling overhead (i.e., the time taken for making scheduling decisions) is kept to a minimum. Hence, decisions are fast, putting less pressure on decision making with increasing workload intensity. From the results, it is evident that the heuristic-based approaches PSO, MBO, and MOPSO cannot sustain their efficiency at larger workloads while EdgeCloud-DRL maintains their robustness, adaptiveness, and computational efficiency. Thus, the overall average performance improvement for a wide range of workload intensities further provides clear evidence for the efficiency of using deep reinforcement learning-based dynamic task scheduling in edge-cloud environments.

Table 3: Comparative Analysis of EdgeCloud-DRL with Existing Task Scheduling Approaches in Cloud and Edge-Cloud Environments

Reference	**Scheduling Approach**	**Performance Metrics**	**Workload Considered**	**Environment**	**Algorithm Used**	**Experimental Setup**	**Scalability**	**Results Summary**
Yan Gu et al. [1]	DRL + Simulated	Cost, Latency	Cloud Workflows	Cloud	DRL + Simulated	CloudSim	High	Lower cost but higher latency

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

	Annealing		(5000+ tasks)		Annealing			
Shuo Zhang et al. [2]	Deep Reinforcement Learning	Latency, Energy	Multi-Cloud Tasks (Dynamic)	Multi-Cloud	Deep Q-Learning	Simulated Multi-Cloud Environment	Moderate	Moderate energy savings, but latency fluctuates
Gorka Nieto et al. [3]	DRL for 5G Edge-Cloud	Offloading Efficiency	5G Edge-Cloud Tasks	5G Edge-Cloud	DDPG-based RL	5G Network Simulator	High	Efficient offloading but lacks real-time adaptability
Heba Nashaat et al. [4]	Distributed DRL	Task Execution Time, Success Rate	Edge-Cloud Tasks (Dynamic)	Edge-Cloud	Multi-Agent DRL	Edge-Cloud Simulation	Moderate	Improves task execution but struggles with large-scale workloads
Alberto Robles-Enciso et al. [5]	Multi-Layer RL	Resource Utilization	Hierarchical Edge-Cloud	Edge-Cloud	Hierarchical RL	Hierarchical Edge-Cloud Setup	High	Better resource utilization but lacks scheduling flexibility
EdgeCloud-DRL (Proposed)	Deep Q-Network (DQN-based)	Latency, Energy, Success Rate, Resource Utilization, Scheduling Overhead	Hybrid Edge-Cloud (1000, 5000, 10000, 20000 jobs)	Hybrid Edge-Cloud	DQN with Target Network & Experience Replay	CloudSim + EdgeCloudSim	Very High (Adaptive to Large-Scale Workloads)	Achieves lowest latency, highest success rate, best resource utilization, and lowest scheduling overhead

Table 3 compares the different task scheduling methods based on deep reinforcement learning and the proposed EdgeCloud-DRL framework. For instance, the model implemented in (Yan Gu et al.) combines deep reinforcement learning (DRL) and simulated annealing algorithm, minimizing the cost and latency of cloud workflows. Yet, this approach focuses on cloud

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

deployment only and ignores dynamic task offloading in hybrid edge-cloud infrastructures. Following this, the method described by Shuo Zhang et al. utilizes Deep Q-Learning for multi-cloud systems, leading to higher energy efficiency and varying latencies, especially during peak workloads.

Gorka Nieto et al. [5] investigate a Deep Deterministic Policy Gradient (DDPG)-based reinforcement learning approach for 5G edge-cloud environments but emphasizes offloading efficiency to a greater extent than scheduling overhead or large-scale workload adaptability. We propose a multi-agent DRL framework in which multiple tasks can be executed in an edge-cloud environment and present results that show a decrease in execution time and increased success rates. However, as a decentralized blockchain, scalability is limited when it comes to high-intensity workloads, resulting in congestion at edge nodes. On the other hand, the model by Alberto Robles-Enciso and colleagues examines the use of hierarchical reinforcement learning to improve resources in edge-cloud setups. While this approach does help with load balancing, it does not have the needed scheduling flexibility to respond to the sudden changes in workload.

To overcome the aforementioned limitations of the existing methods, we propose EdgeCloud-DRL as a DRL-based framework that utilizes a Deep Q-Network (DQN) with Target Network and Experience Replay. This allows you to efficiently learn scheduling decisions to minimize execution latency and energy consumption and maximize success rates of tasks while keeping the resource utilization optimal. In addition, the EdgeCloud-DRL achieves excellent scalability for different workloads of 1000 to 20,000 jobs. Over empirical results reveal that we achieve good outperform outperforming previous models, in terms of execution time, scheduling overhead and adaptability with workload fluctuations, proving our scheduling method highly efficient and robust for hybrid edge-cloud computing.

5. Discussion

However, with the rapid evolution of cloud and edge computing, intelligent task scheduling frameworks are needed to improve system performance, resource efficiency, and quality of service (Quality of Service). Common heuristics/metaheuristic-based scheduling methods such as Particle Swarm Optimization (PSO), Multi-Objective Bat Algorithm (MBO), and Multi-Objective Particle Swarm Optimization (MOPSO) have been extensively employed for optimizing scheduling decisions. Nonetheless, they fail to handle the dynamic characteristics of real applications and heterogeneous cloud-edge environments, resulting in inefficient task delivery, energy waste, and scheduling overhead. In addition, traditional deep reinforcement learning (DRL)-based solutions (e.g., Deep Q-Networks (DQN), Asynchronous Advantage Actor-Critic (A3C), Proximal Policy Optimization (PPO)) have shown to be more efficient in the context of scheduling. However, they are often computationally intensive and may not be scalable for scheduling large-scale workloads.

The EdgeCloud-DRL framework proposed in this paper utilizes a DQN-based scheduling method with target network stabilization and experience replay for improved decision-making in a dynamic edge-cloud computing scenario. Compared with previous models, the proposed

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

framework performs well in resolving a trade-off between execution latency, energy, task success ratio, and system scheduling overhead, therefore suitable for large workloads. The experimental results validate that, irrespective of varying workload intensities, EdgeCloud-DRL satisfies lower latency of task execution, better utilization of resources, and reduced scheduling overhead, outdoing PSO, MBO, and MOPSO. It demonstrates the framework's strength in supporting various computing scopes and delivering excellent performance across cloud and edge nodes. This study highlights the new deep reinforcement learning methods needed to enable scalable and adaptive scheduling in hybrid edge-cloud settings by overcoming some fundamental limitations in current approaches. These discoveries have essential implications in intelligent workload distribution, energy-aware computing, and real-time scheduling applications. This also guarantees that the new framework can be used for future technologies such as IoT, 5G, or federated cloud systems. Moreover, Section 5.1 describes the limitations of the proposed study and suggests directions for realizing enhanced future work.

5.1 Limitations of the Study

Despite the EdgeCloud-DRL framework improving the task scheduling efficiency, it also has some limitations. Second, the larger workloads lead to longer training times for the model since deep reinforcement learning needs many iterations for convergence. Second, this framework relies on the particular behavior of network and thus might not be too accurate in highly-changed edge-cloud infrastructures, which will impact the decisions made about execution of tasks. Third, the model is based on a predefined reward function that needs to be fine-tuned for different implementation scenarios. In the future, research can adapt to improve the training cost and realistic adaptability in the network along with dynamic reward systems to improve the adaptability and generalizability of the models.

6. Conclusion And Future Work

The proposed work proposed EdgeCloud-DRL, a deep reinforcement learning-based framework to manage task scheduling problems in hybrid edge-cloud environments efficiently. Utilizing Deep Q-Networks (DQN) with target network stabilization and experience replay, the proposed model optimizes task execution latency, utilized energy, task success rate, and used resources while minimizing scheduling overhead. EdgeCloud-DRL is shown to outperform traditional selection heuristic methods like PSO, MBO, and MOPSO in simulations, showcasing the robustness across diverse intensities of workload. It emphasizes the need for intelligent learning-based scheduling mechanisms capable of managing the broad merit and complexities of dynamic cloud-edge infrastructures. While it has merits, the study does identify three significant limitations: Longer training time for large workloads, an assumption of a stable network environment, and reliance on a predetermined reward function. Future work can address optimization techniques to speed up the training process, develop mechanisms for adaptive network structures in real-time, and design adaptive rewards for generalization across various cloud-edge computing applications. Moreover, the framework could be extended to multi-agent reinforcement learning, improving transferability and compressibility in a highly distributive environment. Dunning et al. propose an AI-driven framework for dynamically configuring workloads in a virtualized environment spanning

Received: August 09, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

multiple cloud environments, IoT, and fog computing to deliver application services on demand.

References

[1] Yan Gu, Feng Cheng, Lijie Yang, Junhui Xu, Xiaomin Chen, and Long Cheng. (2023). Cost-aware cloud workflow scheduling using DRL and simulated annealing. Elsevier., pp.1-14. https://doi.org/10.1016/j.dcan.2023.12.009

[2] Shuo Zhang, Zhuofeng Zhao, Chen Liu, and Shenghui Qin. (2023). Data-intensive workflow scheduling strategy based on deep reinforcement learning in multi-clouds. Springer. 12(125), pp.1-12. https://doi.org/10.1186/s13677-023-00504-9

[3] Gorka Nieto, Idoia de la Iglesia, Unai Lopez Novoa, and Cristina Perfecto. (2024). Deep Reinforcement Learning techniques for dynamic task offloading in the 5G edge-cloud continuum. Springer. 13(94), pp.1-24. https://doi.org/10.1186/s13677-024-00658-0

[4] HEBA NASHAAT, WALAA HASHEM, RAWYA RIZK, AND RADWA ATTIA. (2024). DRL-Based Distributed Task Offloading Framework in Edge-Cloud Environment. IEEE. 12, pp.33580 - 33594. http://DOI:10.1109/ACCESS.2024.3371993

[5] Alberto Robles-Enciso, and Antonio F. Skarmeta. (2023). A multi-layer guided reinforcement learning-based tasks offloading in edge computing. Elsevier. 220, pp.1-14. https://doi.org/10.1016/j.comnet.2022.109476

[6] Zhuo Chen, Peihong Wei, Yan Li. (2023). Combining neural network-based method with heuristic policy for optimal task scheduling in hierarchical edge cloud. Elsevier. 9, pp.688-697. https://doi.org/10.1016/j.dcan.2022.04.023

[7] Thomas Dreibholz, and Somnath Mazumdar. (2023). Towards a lightweight task scheduling framework for cloud and edge platform. Elsevier. 21, pp.1-16. https://doi.org/10.1016/j.iot.2022.100651

[8] Xiaokang Zhou, Wei Liang, Ke Yan, Weimin Li, Kevin I-Kai Wang, Jianhua Ma, and Q. (2022). Edge-enabled two-stage scheduling based on deep reinforcement learning for internet of everything. IEEE. 10(4), pp.3295 - 3304. http://DOI:10.1109/JIOT.2022.3179231

[9] Joahannes B. D. da Costa, Allan M. de Souza, Rodolfo I. Meneguette, Eduardo Cerqueira, Denis Rosário, Christoph Sommer, and Leandro Villas. (2023). Mobility and deadline-aware task scheduling mechanism for vehicular edge computing. IEEE. 24(10), pp.11345 - 11359. http://DOI:10.1109/TITS.2023.3276823

[10] SIYU YUAN, ZHENYU ZHANG, QIN LI, WEIYUAN LI, AND YONG ZHANG. (2023). Joint optimization of DNN partition and continuous task scheduling for digital twin-aided MEC network with deep reinforcement learning. IEEE. 11, pp.27099 - 27110. http://DOI:10.1109/ACCESS.2023.3257342

Received: August 09, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[11] Haonan Wu, Xiumei Yang, and Zhiyong Bu. (2024). Task Offloading With Service Migration for Satellite Edge Computing: A Deep Reinforcement Learning Approach. IEEE. 12, pp.1-13. http://DOI:10.1109/ACCESS.2024.3367128

[12] Shanchen Pang, Jianyang Zheng, Min Wang, Sibo Qiao, Xiao He, and Changnan Gao. (2023). Minimize average tasks processing time in satellite mobile edge computing systems via a deep reinforcement learning method. Springer. 12(159), pp.1-29. https://doi.org/10.1186/s13677-023-00538-z

[13] Arslan Qadeer, and Myung Jong Lee. (2023). Deep-deterministic policy gradient-based multi-resource allocation in edge-cloud system: a distributed approach. IEEE. 11, pp.20381 - 20398. http://DOI:10.1109/ACCESS.2023.3249153

[14] Aadharsh Roshan Nandhakumar, Ayush Baranwal, Priyanshukumar Choudhary, Muhammed Golec, and Sukhpal Singh Gill. (2024). EdgeAISim: A toolkit for simulation and modelling of AI models in edge computing environments. Elsevier. 31, pp.1-14. https://doi.org/10.1016/j.measen.2023.100939

[15] S. Sudheer Mangalampalli, Ganesh Reddy Karri, Sachi Nandan Mohanty, Shahid Ali, Muhammad Ijaz Khan, Sherzod Abdullaev, and Salman A. Alqahtani. (2024). Multi-Objective Prioritized Task Scheduler Using Improved Asynchronous Advantage Actor Critic (A3C) Algorithm in Multi-Cloud Environment. IEEE. 12, pp.11354 - 11377. http://DOI:10.1109/ACCESS.2024.3355092

[16] Daokun Qi, Xiaojuan Xi, Yake Tang, Yuesong Zheng, and Zhengwei Guo. (2024). Real-time scheduling of power grid digital twin tasks in the cloud via deep reinforcement learning. Springer. 13(121), pp.1-12. https://doi.org/10.1186/s13677-024-00683-z

[17] Mostafa Raeisi-Varzaneh, Omar Dakkak, Adib Habbal, and Byung-Seo Kim. (2023). Resource scheduling in edge computing: Architecture, taxonomy, open issues, and future research directions. IEEE. 11, pp.25329 - 25350. http://DOI:10.1109/ACCESS.2023.3256522

[18] Fan Qi, Li Zhuo, and Chen Xin. (2020). Deep Reinforcement Learning Based Task Scheduling in Edge Computing Networks. 2020 IEEE/CIC International Conference on Communications in China (ICCC), pp.1-6. http://doi:10.1109/ICCC49849.2020.9238937

[19] Reena Panwar and M. Supriya. (2024). RLPRAF: Reinforcement Learning-Based Proactive Resource Allocation Framework for Resource Provisioning in Cloud Environment. IEEE. 12, pp.95986 - 96007. http://DOI:10.1109/ACCESS.2024.3421956

[20] Shilin Wen, Rui Han, Chi Harold Liu, and Lydia Y. Chen. (2023). Fast DRL-based scheduler configuration tuning for reducing tail latency in edge-cloud jobs. Springer. 12(90), pp.1-32. https://doi.org/10.1186/s13677-023-00465-z

Received: August 09, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[21] Tao Zheng, Jian Wan, Jilin Zhang, and Congfeng Jiang. (2022). Deep reinforcement learning-based workload scheduling for edge computing. Springer. 11(3), pp.1-13. https://doi.org/10.1186/s13677-021-00276-0

[22] Vibha Jain, Bijendra Kumar, Aditya Gupta. (2022). Cybertwin-driven resource allocation using deep reinforcement learning in 6G-enabled edge environment. Elsevier. 34(8), pp.5708-5720. https://doi.org/10.1016/j.jksuci.2022.02.005

[23] Hojjat Baghban, Amir Rezapour, Ching-Hsien Hsu, Sirapop Nuannimnoi, and Ching-Yao Huang. (2022). Edge-AI: IoT request service provisioning in federated edge computing using actor-critic reinforcement learning. IEEE, pp.1-10. http://DOI:10.1109/TEM.2022.3166769

[24] Xu Zhao, Guangqiu Huang, Ling Gao, Maozhen Li, and Quanli Gao. (2021). Low load DIDS task scheduling based on Q-learning in an edge computing environment. Journal of Network and Computer Applications, 188, pp.1-12. http://doi:10.1016/j.jnca.2021.103095

[25] Yunmeng Dong, Gaochao Xu, Meng Zhang, and Xiangyu Meng. (2021). A High-Efficient Joint 'Cloud-Edge' Aware Strategy for Task Deployment and Load Balancing. IEEE Access, 9, pp.1-12. http://doi:10.1109/access.2021.3051672

[26] Peng Liu, Zhe Liu, Ji Wang, Zifu Wu, Peng Li, and Huijuan Lu. (2022). Reinforcement learning empowered multi-AGV offloading scheduling in edge-cloud IIoT. Springer. 11(78), pp.1-14. https://doi.org/10.1186/s13677-022-00352-z

[27] Shreshth Tuli, Shashikant Ilager, Kotagiri Ramamohanarao, and Rajkumar Buyya. (2020). Dynamic Scheduling for Stochastic Edge-Cloud Computing Environments using A3C learning and Residual Recurrent Neural Networks. IEEE Transactions on Mobile Computing, 21(3), pp.940-954. http://doi:10.1109/TMC.2020.3017079

[28] Qi Zhang, Lin Gui, Shichao Zhu, and Xiupu Lang. (2021). Task Offloading and Resource Scheduling in Hybrid Edge-Cloud Networks. IEEE Access. http://doi:10.1109/access.2021.3088124

[29] Chen, Lulu, Xu, Yangchuan, Lu, Zhihui, Wu, Jie, Gai, Keke, Hung, Patrick C. K., and Qiu, Meikang. (2020). IoT Microservice Deployment in Edge-cloud Hybrid Environment Using Reinforcement Learning. IEEE Internet of Things Journal, 8(16), pp.12610-12622. http://doi:10.1109/JIOT.2020.3014970

[30] Shida Lu, Rongbin Gu, Hui Jin, Liang Wang, Xin Li, and Jing Li. (2021). QoS-Aware Task Scheduling in Cloud-Edge Environment. IEEE Access, 9, pp.1-10. http://doi:10.1109/access.2021.3072216

[31] Qi, Qi, Zhang, Lingxin, Wang, Jingyu, Sun, Haifeng, Zhuang, Zirui, Liao, Jianxin, and Yu, Fei Richard. (2020). Scalable Parallel Task Scheduling for Autonomous Driving Using Multi-task Deep Reinforcement Learning. IEEE Transactions on Vehicular Technology, 69(11), pp.13861-13874. http://doi:10.1109/tvt.2020.3029864

Received: August 09, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[32] Bassem Sellami, Akram Hakiri, Sadok Ben Yahia, and Pascal Berthou. (2022). Energy-aware task scheduling and offloading using deep reinforcement learning in SDN-enabled IoT network. Elsevier. 210, pp.1-20. https://doi.org/10.1016/j.comnet.2022.108957

[33] Cho, Chunglae, Shin, Seungjae, Jeon, Hongseok, and Yoon, Seunghyun. (2020). QoS-Aware Workload Distribution in Hierarchical Edge Clouds: A Reinforcement Learning Approach. IEEE Access, 8, pp.193297–193313. http://doi:10.1109/access.2020.3033421

[34] Zhang, Yaqiang, Zhou, Zhangbing, Shi, Zhensheng, Meng, Lin, and Zhang, Zhenjiang. (2020). Online Scheduling Optimization for DAG-Based Requests Through Reinforcement Learning in Collaboration Edge Networks. IEEE Access, 8, pp.72985–72996. http://doi:10.1109/ACCESS.2020.2987574

[35] Guanjin Qu, Huaming Wu, Ruidong Li, and Pengfei Jiao. (2021). DMRO: A Deep Meta Reinforcement Learning-Based Task Offloading Framework for Edge-Cloud Computing. IEEE Transactions on Network and Service Management. 18(3), pp.3448-3459. http://doi:10.1109/tnsm.2021.3087258

[36] Pan, Shengli, Zhang, Zhiyong, Zhang, Zongwang, and Zeng, Deze. (2019). Dependency-Aware Computation Offloading in Mobile Edge Computing: A Reinforcement Learning Approach. IEEE Access, 7, pp.134742–134753. http://doi:10.1109/ACCESS.2019.2942052

[37] Ding, Shiyao, and Lin, Donghui. (2020). Dynamic Task Allocation for Cost-Efficient Edge Cloud Computing. pp.218–225. http://doi:10.1109/scc49832.2020.00036

[38] Shreshth Tuli, Kotagiri Ramamohanarao, and Rajkumar Buyya. (2020). Stochastic Model for Deadline-Aware Scheduling in Edge-Cloud Continuum. ACM Transactions on Internet of Things, 4(2), pp.1-19. http://doi:10.1145/3489779

[39] Shiyao Ding, Ling Gao, and Quanli Gao. (2021). Low Load DIDS Task Scheduling Based on Q-learning in Edge Computing. Elsevier. Journal of Network and Computer Applications, 188, pp.1-12. http://doi:10.1016/j.jnca.2021.103095

[40] Bassem Sellami, Akram Hakiri, and Pascal Berthou. (2020). Deep Reinforcement Learning for Energy-Efficient Task Scheduling in SDN-based IoT Network. 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), pp.1-4. http://doi:10.1109/nca51143.2020.9306739

Received: August 09, 2025