SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I6P125

Volume 12 Issue 6, 315-325, June 2025

Original Article

Deep Deterministic Policy Gradient Algorithm for Dynamic Task Scheduling in Edge-Cloud Environment Using Reinforcement Learning

D. Mamatha Rani[1], Supreethi K P[2], Bipin Bihari Jayasingh[3]

[1]*Department of Computer Science, TSWRDCW, Nizamabad, Telangana, India.*
[2]*Department of Computer Science and Engineering, Jawaharlal Nehru Technological University, Hyderabad, Telangana, India.*
[3]*Professor and Head of Department, CVR College of Engineering/IT Department, Hyderabad, Telangana, India.*

[1]*Corresponding Author : mamatha3004@gmail.com*

Received: 13 April 2025 Revised: 15 May 2025 Accepted: 16 June 2025 Published: 27 June 2025

Abstract *- In the contemporary era, cloud computing is helping high-performance computing applications by providing scalable and affordable computing resources. However, the latency of cloud resources is relatively less when compared with edge computing. In this context, using edge cloud for task scheduling has become indispensable in reaping latency performance benefits with edge cloud. However, assigning every task to the edge cloud is impossible, and resource-intensive tasks should be scheduled to the cloud. An edge-cloud environment becomes very complex, and scheduling is an NP-hard problem. Many existing methods based on reinforcement learning are found to have shortcomings in dealing with an ample action space in the presence of a state space. This paper proposes an algorithm known as the Deep Deterministic Policy Gradient Algorithm for Dynamic Task Scheduling (DDPGA-TS). Our algorithm has a novel pruning strategy that continuously monitors the action space and reduces it to improve overall performance in task scheduling. Our method uses three scales of environments. Several performance indicators are used to evaluate the proposed algorithm's performance. In the experimental findings, the suggested algorithm outperforms existing methods such as DDPG-NN and DDPG-CNN.*

Keywords *- Cloud computing, Edge computing, Dynamic task scheduling, Reinforcement learning, Deep Learning.*

1. Introduction

With the emergence of the Internet of Things (IoT) workflow applications, edge cloud has become significant for latency reasons. With edge cloud and cloud infrastructures available, decisions are to be made to schedule a task in the edge cloud or the cloud based on latency or other Service Level Agreements (SLAs).

Despite the advances in RL-based schedulers for edge-cloud systems, DDPG, DQN, A3C, etc., existing approaches find it particularly challenging to efficiently explore the large action spaces with growing state spaces based on dynamically changing quality of service requirements and task complexities. This causes slow convergence, posting of tasks at sub-optimal places, and increased overhead in the operation in heterogeneous resource environments. Also, relatively little work exists on adaptive action space pruning with the real-time scheduling problem. It, however, requires intelligent task scheduling that scales in different environments and adapts to the dynamics of the system in such a way that policy learning is maximized. This paper fills this gap by presenting a Deep Deterministic Policy Gradient Algorithm with a dedicated pruning technique to enhance the scheduling efficiency of edge-cloud systems.

Towards edge-cloud-based task scheduling, many researchers contributed. Almutairi et al. [1] observed that rising IoT devices demand efficient Edge-Cloud task offloading. Fuzzy logic algorithms improve latency-sensitive application service time, enhancing resource utilization. Xu et al. [7] focused on edge-cloud computing that empowers IoV by optimizing offloading using MDP and LFGO with deep Q-learning, improving efficiency and adaptability.

Tuli et al. [13] stated that fog computing optimizes IoT application tasks via A3C and R2N2 models, enhancing quality of service and reducing costs significantly.

Yahia et al. [22] focused on energy-efficient and low-latency task scheduling in Fog-IoT networks using Deep Reinforcement Learning. With IoT workloads, it was observed that the edge-cloud environment is beneficial compared to relying on the cloud alone.

The literature observed that many existing methods based on reinforcement learning have shortcomings in dealing with large action spaces in the presence of state spaces.

The main novelty of the proposed work is that it incorporates a pruning-enhanced Deep Deterministic Policy Gradient Algorithm (DDPGA-TS) for dynamic task scheduling, which dynamically prunes the action space on the fly according to the resource utilization and characteristics of the job. Unlike typical models such as DDPG-NN and DDPG-CNN, which suffer from slow convergence rates and scalability limitations, our approach presents a resource-aware pruning mechanism to enhance convergence efficiency and flexibility in large-scale edge-cloud networks. A hybrid deep learning approach is introduced with Conv1D and GRU layers with attention to capturing both spatial and temporal information to improve the decision of the actor-network. The proposed method is empirically compared across three scales of simulated environments. It shows that it outperforms existing DDPG variants in terms of operation cost, task rejection rate, and Quality of Experience (QoE) by a large margin.

These are our contributions to this paper. Proposed a Deep Deterministic Policy Gradient Algorithm for Dynamic Task Scheduling (DDPGA-TS). Our algorithm has a novel pruning strategy that continuously monitors the action space and reduces it to improve overall performance in task scheduling. Three scales of environments are used in our experiments. Several performance indicators are used to evaluate the experimental results, which outperform existing methods such as DDPG-NN and DDPG-CNN. This paper is structured. Section 2 reviews the literature on current methods in task scheduling in edge-cloud environments. Section 3 presents our methodology for efficient task scheduling in edge-cloud environments. Section 4 presents the results of the experiments. Section 5 discusses the significance of this work and provides limitations. Section 6 concludes our work and gives directions for the future scope of the research.

2. Related Work

This section reviews the literature on prior work related to cloud task scheduling. Almutairi et al. [1] observed that rising IoT devices demand efficient Edge-Cloud task offloading. Fuzzy logic algorithms improve latency-sensitive application service time, enhancing resource utilization. Ding et al. [2] address the challenge of dynamically partitioning stateful data stream applications by proposing a performance-aware partitioning framework that adapts to workload variations and resource dynamics in edge-cloud systems. Their work highlights the importance of minimizing inter-node communication and balancing computational loads to enhance system responsiveness and resource utilization. Compared to earlier static partitioning approaches, their dynamic strategy provides improved adaptability and scalability in heterogeneous and changing environments. Bulej et al. [3] found that CPS demands real-time cloud and edge-cloud operations. The paper introduces an approach with minimal developer impact, ensuring performance guarantees. Zhao et al. [4] stated that edge computing faces security challenges amid rapid growth. Proposed Q-learning-based DIDS scheduling balances load and detection rates efficiently. Luo et al. [5] introduced CPS-driven smart factories that adopt edge-cloud collaboration, demanding efficient real-time task scheduling. Proposed DSOTS and TSGS algorithms reduce latency and costs, enhancing user satisfaction. Zmij et al. [6] observed that IoT applications' growing demand for low latency prompts innovative joint offloading and scheduling with the JTOS framework.

Xu et al. [7] focused on edge-cloud computing that empowers IoV by optimizing offloading using MDP and LFGO with deep Q-learning, improving efficiency and adaptability. Asghari et al. [8] focus on optimizing task scheduling, resource provisioning, and load balancing for scientific workflows in cloud environments. They propose a hybrid approach integrating parallel SARSA reinforcement learning agents with a genetic algorithm to achieve adaptive and efficient workflow execution. Their method enables dynamic decision-making in complex environments by learning optimal strategies through interaction, while the genetic algorithm ensures global optimization. This work demonstrates improved execution time and resource utilization compared to traditional heuristic-based approaches, highlighting the potential of reinforcement learning in workflow management. Guo et al. [9] introduced the SMDQN algorithm, which optimizes service migration in MEC, effectively balancing delay and migration costs and is adaptable to diverse patterns.

Yahia et al. [10] SDN-based DRL manages IoT traffic at the edge, ensuring low latency and high efficiency. Future work focuses on FedML implementation. Wu et al. [11] proposed a DMRO that optimizes edge offloading decisions using deep meta-reinforcement learning, improving adaptability and portability significantly. Hao et al. [12] found that the Industrial CPS demands edge services. The proposed DQN-based service placement effectively minimizes response time. Tuli et al. [13] stated that fog computing optimizes IoT application tasks via A3C and R2N2 models, enhancing quality of service and reducing costs significantly. Gui et al. [14] tackled SMDs' computation overload via edge and cloud offloading, presenting algorithms for hybrid edge-cloud networks. Future work includes optimizing ETRC through multi-agent reinforcement learning and exploring service caching and joint resource scheduling.

Liu et al. [15] found that the SFC embedding in dynamic edge-cloud scenarios is challenging. Two DRL-based methods efficiently handle SFC-DMP, outperforming previous methods in delay reduction. Chen et al. [16] explored the complex network state that challenges fog resource provisioning. A learning-based mobile fog scheme utilizing DDPG is proposed, achieving significant improvements. Mekala et al. [17] proposed a CCR-RL algorithm that optimally balances resource utilization, processing speed, and cost, achieving substantial latency reduction.

Lin et al. [18] used an SA-DQN-based strategy that optimizes computation offloading for CAVs in the VEC environment, effectively reducing energy consumption and failure rates. Han et al. [19] studied the EdgeSlice system that utilizes decentralized deep reinforcement learning for dynamic network slicing in wireless edge computing. Zhang et al. [20] proposed a resource allocation scheme using deep reinforcement learning for IoV in MEC scenarios, ensuring low latency and overhead. Zhou et al. [21] introduced a DRL-TSS method to optimize resource utilization for IoE applications. It proposes Johnson's rule-based presorting and an improved DRL scheme to minimize the makespan. Experimental results show a 1.1-approximation to the optimal schedule. Future research will focus on enhancing the deep learning scheme for scheduling optimization.

Yahia et al. [22] focused on energy-efficient and low-latency task scheduling in Fog-IoT networks using Deep Reinforcement Learning. The proposed algorithm outperforms others, achieving up to 87% energy savings and 50% reduction in time delay. Future work aims to extend the algorithm for Ultra-Dense Edge Computing and Federated Machine Learning to address data privacy concerns. Hakiri et al. [23] highlighted blockchain integration in IoT networks, emphasizing low latency, improved reliability, and privacy. The study introduces a Blockchain-based DRL approach for efficient task scheduling and offloading in SDN-enabled IoT networks, achieving 50% better energy efficiency. Future work aims to achieve interoperability, implement federated learning, and enforce data privacy using homomorphic encryption.

Ye et al. [24] employed deep reinforcement learning and a Markov decision process to jointly improve computation offloading and resource allocation in IoV. The proposed CORA algorithm outperforms non-DRL and DRL algorithms in processing delay and cost. Future work involves considering resource competition among vehicles for edge server computing resources. Zhang et al. [25] proposed a distributed real-time scheduling framework for cloud manufacturing using cloud-edge collaboration and distributed deep reinforcement learning.

The D3QN algorithm outperforms other methods, showing potential for efficient decision-making. Their future work will address uncertainties using digital twins in cloud manufacturing. Based on the surveyed literature, one can observe that even if several deep reinforcement learning models, e.g., A3C [13], DQN [12], distributed DRL [25], have been used for task offloading and resource optimization in edge-cloud and IoT systems, they are still facing the problems of dealing with large-scale dynamic workloads effectively. In particular, the high dimension of the action space, static learning architectures, or lack of adaptive decision-making mechanisms detract from scalability and reactivity. In addition, most existing methods are based on the assumption of a perfect network environment and ignore the action pruning in real-time to cut down the scheduling latency. These constraints drive the demand for a more adaptive and scalable task scheduling scheme based on deep RL and resource-aware pruning to enhance the system performance for different deployment scales.

3. Proposed System

In this paper, the rationale behind edge cloud is that it could render services faster to minimize latency, which is essential to honour SLAs. At the same time, the cloud also plays a crucial role, without which most IoT and other use cases cannot be realized [16]. In such an environment, it is essential to achieve context-aware resource provisioning. Some bandwidth-hungry applications need minimal end-to-end latency, as discussed in [1]. In such applications, the edge cloud plays a crucial role. Based on the request received, the proposed system needs to determine whether to use cloud or edge cloud to run it by considering different factors like deadline, resource availability, and bandwidth, to name a few.

3.1. Our Framework

Our system is based on the framework illustrated in Figure 1. The framework is known as the Edge-Cloud-based Deep Learning Framework (EC-DLF) for automatic, efficient task scheduling. The framework is designed to run in the edge cloud for easy access and learning. It is based on an actor-critic model that is part of RL. The actor is synonymous with a policy function that continuously monitors s_t (state)of the environment and is involved in taking appropriate action (a_t) based on s_t. The actor receives r_t (reward) for every action. Critic makes use of $Q(s_t, a_t)$ (a function with action and state) to change policy parameters. Each dynamic task scheduling action at every given time produces an appropriate reward. However, the framework's goal is to maximize reward through policy optimization.

The state space associated with the framework is described here. At any given time, t system state s_t Reflects the resources, their current utility in the edge cloud and the cloud, and the current job assigned by user u is denoted as j_u. This job is expected to be scheduled in either the cloud or the on-premises cloud. The parameters associated with the state are j_u (the job of the given user), Ur_N (utility rate of a given node in the cloud), Ur_M (utility rate of a given node in edge cloud) and Ur_W (total utility rate of all resources). These parameters help an actor learn strategies for optimal and dynamic task scheduling. Concerning action space a_t, based on s_t, the actor determines the resources required to run the job successfully and meet the deadline. The parameters of action space include the bandwidth ($ch_{h)}$, CPU cycles (vm_k) and whether the task is assigned to the cloud or the edge cloud (1 indicates cloud and 0 indicates edge cloud). The reward is computed as in Eq. 1, which considers the gain and cost involved in task scheduling.

$$r_t(s_t, a_t) = I_u(s_t, a_t) - cost_u(s_t, a_t) \times rtt(j_u) \quad (1)$$

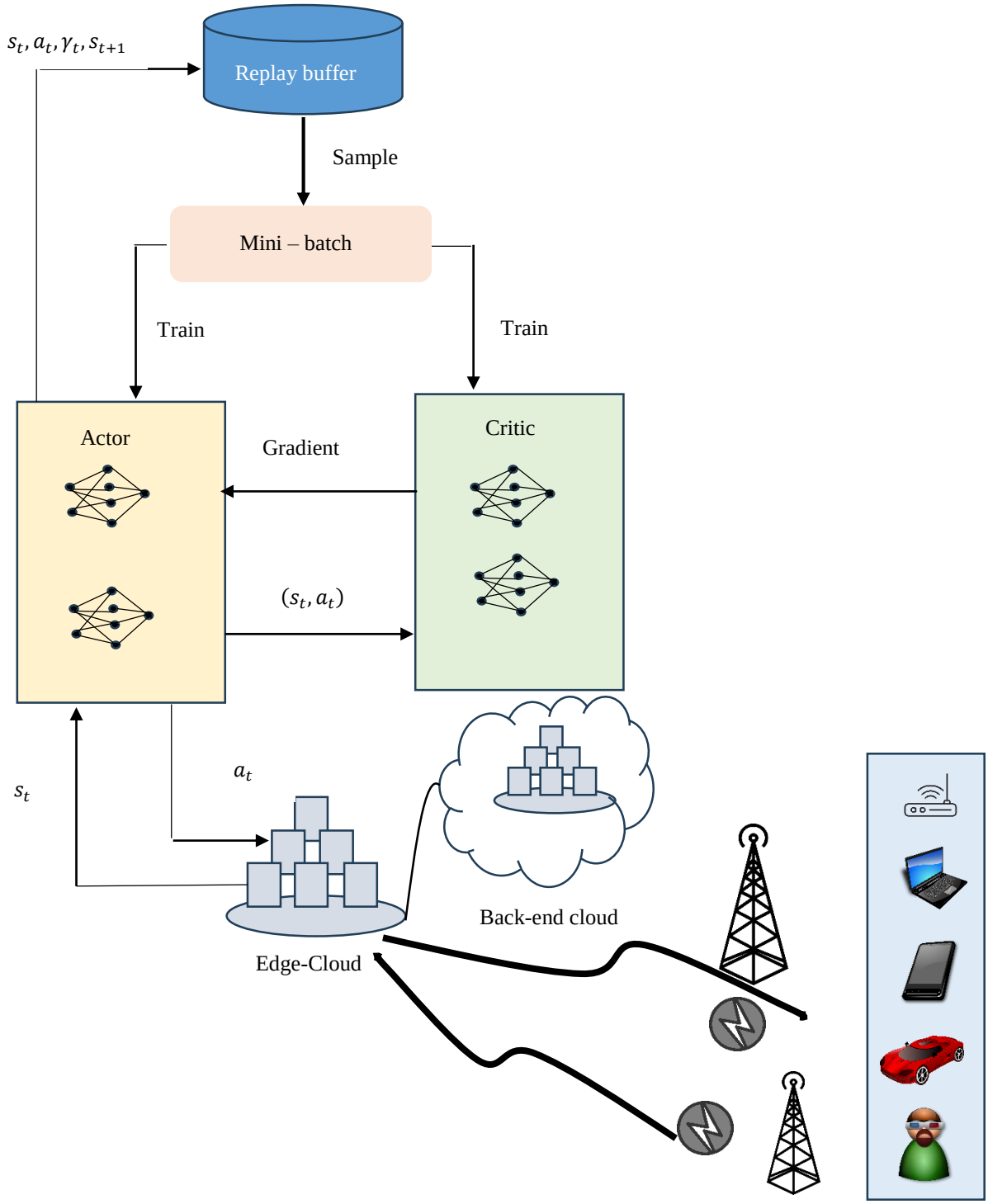

Fig. 1 Proposed edge-cloud-based deep learning framework

The net gain of executing j_u is denoted as $I_u(s_t, a_t)$, while the total cost of executing j_u is denoted as $cost_u(s_t, a_t)$ and time taken to process j_u is denoted as $rtt(j_u)$. Considering computing resources, the completion time of a given job, and wireless resources, the gain and cost are redefined as in Equation (2) and Equation (3), respectively.

$$I_u(s_t, a_t) = \begin{cases} (dl_u - rtt_{ec}(j_u)).\delta, if\ cloud_t = 0 \\ (dl_u - rtt_{bc}(j_u)).\delta, if\ cloud_t = 1 \end{cases} \quad (2)$$

Where the service provides gain is denoted as δ for successful completion of the job with completion time $rtt(j_u)$ in given deadline dl_u. If the deadline is not met, the gain will be in negative value, which also influences the reward. This condition motivates the proposed framework to decide whether to satisfy SLA or deadlines.

$$cost_u(s_t, a_t) = \begin{cases} Ur_w(t).c_h + Ur_M(t).c_k, if\ cloud_t = 0 \\ Ur_w(t).c_h + Ur_N(t).c_k, if\ cloud_t = 1 \end{cases} \quad (3)$$

On the contrary, to gain, cost includes the resource and bandwidth of the channel. The bandwidth and VM (node) cost per unit time are denoted as c_h and c_k, respectively. Unlike the work in [18], each job's computations for cost, gain, and reward are performed. As each job can have its quality of service needs, this approach has been proven to be more helpful. It also helps in optimizing policies associated with job scheduling. Optimization of resource utilization based on the quality of service needs of a job is expressed in Equation (4).

$$maximize \sum_{t=1}^{T} r_t(s_t, a_t) \quad (4)$$

This optimization is subject to conditions such as $Ur_W(t) \leq 1, Ur_M(t) \leq 1$ (utility of bandwidth) and $Ur_N(t) \leq 1, \forall t \in T$ (utilization should not exceed capacity available). Another constraint indicates that the job should be completed without exceeding the deadline. Table 1 shows the notations used in this paper.

Inspired by the work in [13], deep learning usage was optimized in our RL model. It makes use of Conv1D for local feature extraction. GRU learns temporal dependencies, and an attention procedure is used to gain information that impacts the prediction process. GRU uses minimal parameters and does not need a memory unit, leading to optimized performance compared to LSTM [20].

The notion of experience replay for a better training process is exploited. The system can choose bandwidth appropriately along with the VM to execute the given job. To select a node or server that is less busy, a pruning strategy is incorporated that continuously and actively monitors and reduces the action space significantly. The proposed mechanism also takes care of load balancing. The optimal policy required for scheduling is achieved with the pruning strategy.

Table 1. Notations used in the proposed system

Notation	Meaning
dl_u	Denote a deadline given by the user.
d_u	Data of the user to be processed
j_u	Denotes the user's current job
$j_u(dl_u, d_u)$	Denotes tuple associated with user's job
$\mu_h^w(t)$	Number of channels at a given time t
t	Denotes time
a_t	Denotes a continuous action
s_t	Denotes state
r_t	Denotes a reward at a given time
$Q(s_t, a_t)$	Denotes action value function
ch_h	Denotes bandwidth
vm_k	Denotes CPU cycles
$I_u(s_t, a_t)$	Gain of job execution
$cost_u(s_t, a_t)$	Cost of resource
c_h	Cost of bandwidth of the wireless channel
c_k	Allocation of VM per unit of time

Algorithm: Deep Deterministic Policy Gradient Algorithm for Dynamic Task Scheduling (DDPGA-TS)
Input: User jobs with quality of service needs
Output: Optimal task scheduling

```
Begin
Initialize replay memory M
For each scheduling interval
    Reset environment
    For each time instance t in T
        Observe the state
        Prune action space
        Predict an action
        Receive reward and next state
        Store transition in M
        Train the actor and critic
        Update weight vectors
    End For
    Schedule tasks based on weight vectors
End For
End
```

As presented in Algorithm, it inputs user jobs with quality of service needs and results in optimal task scheduling. It has an iterative process to prune the action space for every scheduling interval with the underlying RL-based approach. The workload in the proposed system is denoted as J, and all users are denoted as U. Each job is denoted as a tuple, denoted as $j_u(dl_u, d_u)$ consisting of deadline and data to be processed. Therefore, the entire workload of all users is denoted as $J = \{j_1(dl_1, d_1), j_2(dl_2, d_2), \dots, j_U(dl_U, d_U)\}$. Every user's job requires memory and CPU processing. Our approach in CPU cycle modelling is similar to that of [12]. Our bandwidth model is described here. In edge computing, many nodes support wireless access. B denotes the bandwidth of all nodes. Each node supports many wireless channels denoted as $\{ch_1^w, ch_2^w, ch_3^w, \dots, ch_H^w\}$. Our computation model includes a number of nodes, denoted as M, in edge computing. Each node can have a set of VMs denoted as $K = \{vm_1^m, vm_2^m, vm_3^m, \dots, vm_K^m\}$. The computation model also has a cloud with a number of servers consisting of VMs in each server. A set of VMs is denoted as $K = \{vm_1^m, vm_2^m, vm_3^m, \dots, vm_K^m\}$. Our delay model is based on the definition of Round-Trip Time (RTT), which includes end-to-end time right from sending the job to completing the job and sending the results back. This delay model is somewhat similar to that of [10]. The propagation time of a given job is considered 5ms for the edge cloud and 50ms for the cloud. The transmission time for a given job concerning the edge cloud is computed as in Equation (5).

$$t_u(trans_{ec}) = \frac{d_u}{ch_h^w} + \frac{R_u}{ch_h^w} \quad (5)$$

The transmission time for a given job concerning the cloud is computed as in Equation (6).

$$t_u(trans_{bc}) = t_u(trans_{ec}) + \frac{d_u}{b} + \frac{R_u}{b} \quad (6)$$

The processing time required by the edge cloud and the cloud is computed as in Equation (7) and (8), respectively.

$$t_u(proc_{ec}) = \frac{L_u}{vm_k^m} \quad (7)$$

$$t_u(proc_{bc}) = \frac{L_u}{vm_k^n} \quad (8)$$

Based on this, the RTT can be computed as in Equation (9) and (10) for edge cloud and cloud, respectively.

$$rtt_{ec}(j_u) = t_u(prop_{ec}) + t_u(trans_{ec}) + t_u(proc_{ec}) \quad (9)$$

$$rtt_{bc}(j_u) = t_u(prop_{bc}) + t_u(trans_{bc}) + t_u(proc_{bc}) \quad (10)$$

A proposed utility model indicates resource utilization at a given time t. The utility rate of all nodes used in the computation is computed as in Equation (11).

$$Ur_w(t) = \frac{\sum_{w=1}^{W}(\sum_{h=1}^{H} ch_h^w . \mu_h^w(t))}{B} \quad (11)$$

Similarly, the utility rate is computed for each node in the edge cloud and the cloud according to Equation (12) and (13), respectively.

$$Ur_M(t) = \frac{\sum_{n=1}^{M}(\sum_{k=1}^{K} vm_k^m . \mu_k^m(t))}{C_{ec}} \quad (12)$$

$$Ur_N(t) = \frac{\sum_{n=1}^{N}(\sum_{k=1}^{K} vm_k^m . \mu_k^n(t))}{C_{bc}} \quad (13)$$

These computations play an essential role in the proposed RL-based system, which enables efficient scheduling of dynamic workloads in an edge-cloud environment.

4. Results and Discussion

A prototype application is built to simulate the proposed framework and the underlying algorithm. All experiments use a Dell PC with a 2.9 GHz i7 processor, 128GB RAM, and a 64-bit Windows 11 OS. The results of our experiments are compared with two existing methods, DDPG-NN and DDPG-CNN, taken from [19]. Three different environments are used for experiments, as shown in Table 2. The bandwidth availability between the edge cloud and the cloud is 1 Gbps. The learning rate set for the actor and critic is 0.0001 and 0.001, respectively. The γ value is set to 0.99, while the number of iterations for each episode is 1000.

Table 2. Environments used for experiments

Scale of Environment	#Edge Computing Nodes	#Cloud Nodes	#Jobs
Small	10	10	10000
Medium	30	30	100000
Large	50	50	1000000

Table 3. Comparison of convergence in terms of normalized reward

Episodes	Normalized Reward		
	Proposed (DDPGA-TS)	DDPG-NN	DDPG-CNN
0	0.0	0.0	0.0
200	0.9	0.8	0.7
400	0.9	0.8	0.7
600	0.9	0.8	0.7
800	0.9	0.8	0.7
1000	0.9	0.8	0.7

4.1. Convergence Comparison

The convergence of different models is discussed here in terms of normalized reward and normalized training loss. The convergence experiments are made in a small-scale environment, as in Table 2. Table 3 shows that normalized reward is provided against the number of episodes in task scheduling in an edge-cloud environment. Table 4 shows that the normalized training loss is provided against the number of episodes in the task scheduling process in an edge-cloud environment.

Table 4. Comparison of convergence in terms of normalized training loss

Episodes	Normalized Training Loss		
	Proposed (DDPGA-TS)	DDPG-NN	DDPG-CNN
0	0.3	0.7	0.9
200	0.1	0.2	1.0
400	0.1	0.2	0.3
600	0.1	0.2	0.3
800	0.1	0.2	0.3
1000	0.1	0.2	0.3

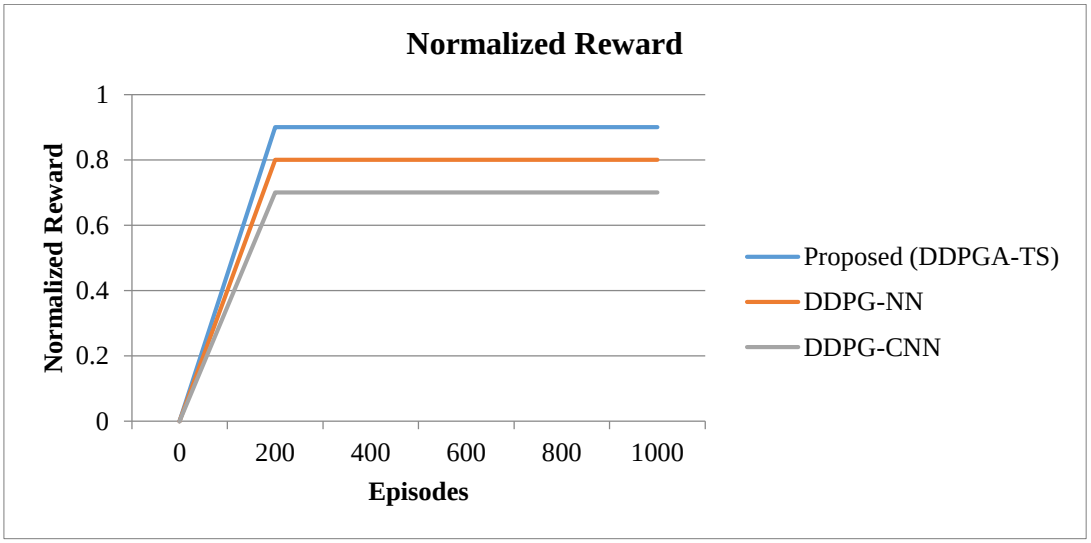

Fig. 2 Normalized reward against episodes

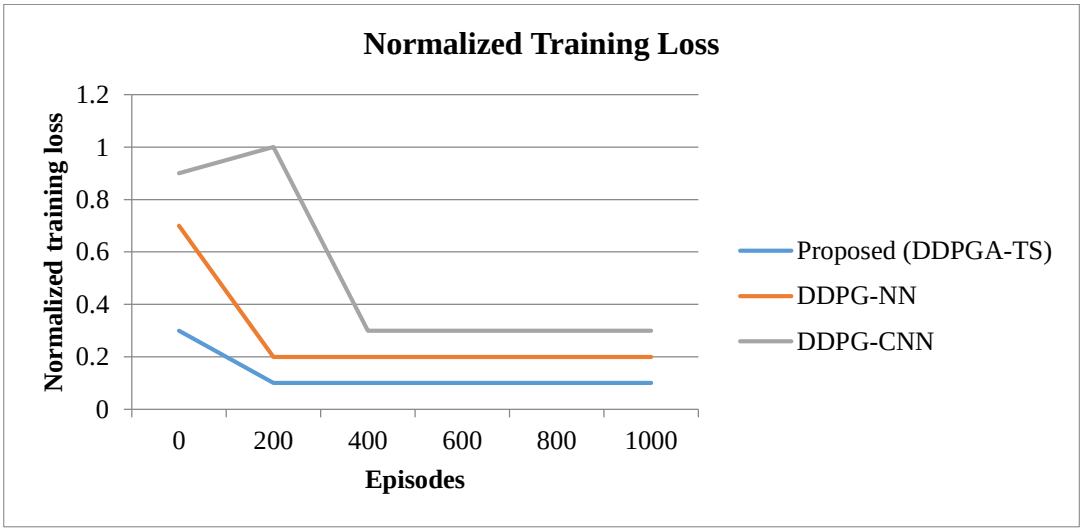

Fig. 3 Normalized training loss against episodes

As presented in Figure 2, the convergence rate in terms of normalized reward is observed against the number of episodes. The normalization process is carried out with a simple min-min approach. The proposed model performs better as it can learn local and long-term features.

As presented in Figure 3, the convergence rate in terms of normalized training is observed against the number of episodes. The normalization process is carried out with a simple min-min approach. The proposed model performs better as it can learn local and long-term features. The training loss reflects superior convergence with the proposed model.

4.2. Performance Evaluation

The performance of the proposed model and two existing models is evaluated using three metrics in three environments.

Table 5 presents the operational costs of the three models. Lower average operational costs indicate better performance.

Table 6 presents the rejection rate of the three models. A lower average rejection rate indicates better performance.

Table 7 provides the three models of QoE. A High QoE indicates better performance.

Table 5. Average operational cost of different models

Models	Average Operational Cost		
	Small	Medium	Large
Proposed (DDPGA-TS)	1.4	1.3	1.1
DDPG-NN	1.9	2.1	2.3
DDPG-CNN	2.1	2.3	2.4

Table 6. Average rejection rate of different models

Models	Average Rejection Rate		
	Small	Medium	Large
Proposed (DDPGA-TS)	4	3	2
DDPG-NN	15	17	23
DDPG-CNN	16	20	24

Table 7. Quality of experience of different models

Models	Quality of Experience		
	Small	Medium	Large
Proposed (DDPGA-TS)	0.62	0.69	0.7
DDPG-NN	0.54	0.48	0.4
DDPG-CNN	0.48	0.4	0.28

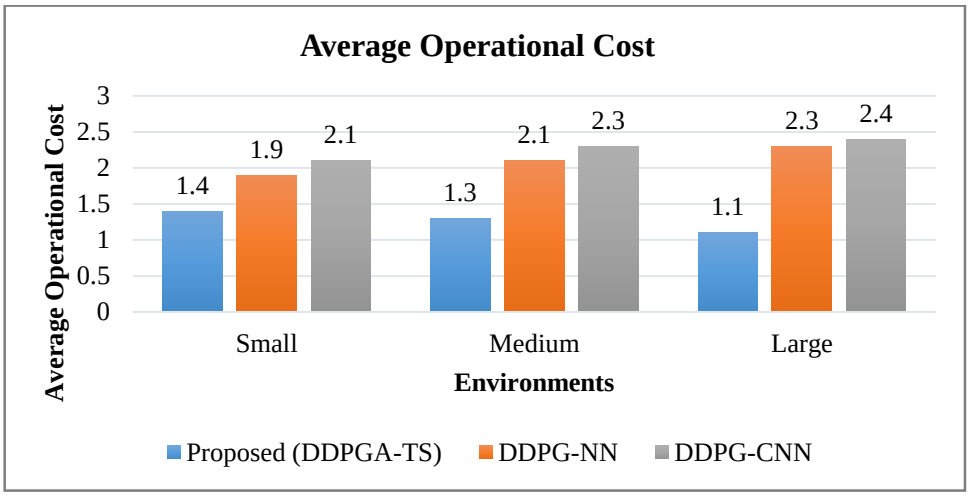

Fig. 4 Average operational cost among the models

As presented in Figure 4, the average operational cost is provided against the three environments. When the small environment is used for experiments, DDPG-CNN costs 2.1, DDPG-NN is 1.9, and the proposed DDPGA-TS is 1.4. When the medium environment is used for experiments, DDPG-CNN costs 2.3, DDPG-NN is 2.1, while the proposed DDPGA-TS is 1.3. When the large environment is used for experiments, DDPG-CNN costs 2.4, DDPG-NN is 2.3, and the proposed DDPGA-TS is 1.1. From the results, it is observed that the proposed model exhibits better performance than the existing models.

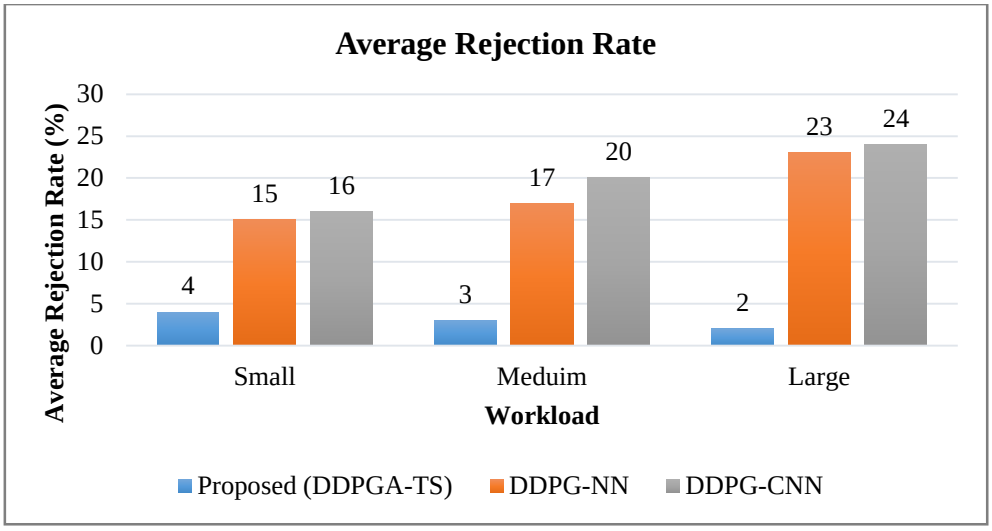

Fig. 5 Average rejection rate among the models

As presented in Figure 5, the average rejection rate is provided against the three environments. When the small environment is used for experiments, the rejection rate of DDPG-CNN is 16, DDPG-NN is 15, and the proposed DDPGA-TS is 4. When the medium environment is used for experiments, the rejection rate of DDPG-CNN is 20, DDPG-NN is 17, and the proposed DDPGA-TS is 3. When the large environment is used for experiments, the rejection rate of DDPG-CNN is 24, DDPG-NN is 23, and the proposed DDPGA-TS is 2. The results show that they are better than existing models in terms of rejection rate.

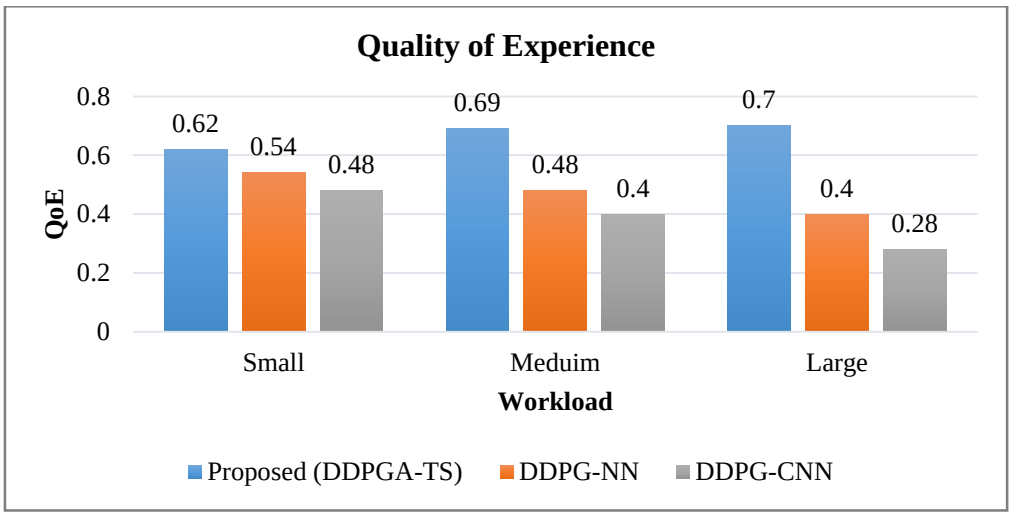

Fig. 6 QoE among the models

As presented in Figure 6, the QoE is provided against the three environments. When the small environment is used for experiments, the QoE of DDPG-CNN is 0.48, DDPG-NN is 0.54, and the proposed DDPGA-TS is 0.62. When the medium environment is used for experiments, the QoE of DDPG-CNN is 0.4, DDPG-NN is 0.48, and the proposed DDPGA-TS is 0.69. When the large environment is used for experiments, the QoE of DDPG-CNN is 0.28, DDPG-NN is 0.4, and the proposed DDPGA-TS is 0.7. The results regarding QoE are better than those of the existing models. In summary, the proposed DDPGA-TS model is an adaptive approach to choosing suitable VMs and wireless channels and optimizing the scheduling of tasks in the edge-cloud environment. The pruning strategy involved in the proposed methodology helps reduce the rejection rate. In all the environments, the proposed model could outperform all existing methods.

The remarkable effectiveness of the proposed DDPGA-TS algorithm is mainly attributed to two specific improvements: leveraging a dynamic pruning strategy and a hybrid neural structure. The pruning policy greatly simplifies the action space by pruning away the unfavourable or loaded nodes in real time, making the model pay more attention to the scheduling decisions that have the potential for better performance. This results in faster convergence, as verified by the lower normalized training loss and higher normalized reward during the training period than DDPG-NN and DDPG-CNN. Furthermore, the combination of Conv1D and GRU layers makes it possible to extract both local and temporal features effectively, enhancing the model for predicting suitable task placements in a time-varying environment. Consequently, the DDPGA-TS model maintains lower average operation costs, lower task rejection ratio, and

better Quality of Experience (QoE) in different-size scenarios. Our results confirm this and provide practical evidence of the benefit of our adaptive method over nonadaptive DDPG variants.

5. Conclusion and Future Work

An Edge-Cloud-Based Deep Learning Framework (EC-DLF) is proposed for automatic, efficient task scheduling. The framework is designed to run in the edge cloud for easy access and learning. It is based on an actor-critic model that is part of RL. The actor is synonymous with a policy function that continuously monitors a sub t (state) of the environment and involves making appropriate action (a. sub t) based on a sub t. An algorithm known as the Deep Deterministic Policy Gradient Algorithm for Dynamic Task Scheduling (DDPGA-TS). Our algorithm has a novel pruning strategy that continuously monitors the action space and reduces it to improve overall performance in task scheduling. Our system model includes edge cloud and cloud to optimize performance for task scheduling. The rationale for edge cloud is that it could render services faster to minimize latency, which is essential to honour SLAs. Three scales of environments are used in our experiments. Several performance indicators are used to evaluate the proposed algorithm's performance. The experimental findings showed that the suggested algorithm outperformed existing methods such as DDPG-NN and DDPG-CNN. In the future, our framework is intended to improve with parallelized techniques to leverage the training and learning process.

Conflicts of Interest

Deep Deterministic Policy Gradient Algorithm Dynamic Task Scheduling in Edge-Cloud Environment Using Reinforcement Learning. The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria, educational grants, participation in speakers' bureaus, membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

References

[1] Jaber Almutairi, and Mohammad Aldossar, "A Novel Approach for IoT Tasks Offloading in Edge-Cloud Environments," *Journal of Cloud Computing*, vol. 10, pp. 1-19, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Shaoshuai Ding et al., "Partitioning Stateful Data Stream Applications in Dynamic Edge Cloud Environments," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 2368-2381, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[3] Lubomír Bulej et al., "Managing Latency in Edge–Cloud Environment," *Journal of Systems and Software*, vol. 172, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] 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]

[5] Yu Zhang et al., "Deadline-Aware Dynamic Task Scheduling in Edge-Cloud Collaborative Computing," *Electronics*, vol. 11, no. 15, pp. 1-24, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[6] Abdullah Lakhan et al., "Delay Optimal Schemes for Internet of Things Applications in Heterogeneous Edge Cloud Computing Networks," *Sensors*, vol. 22, no. 16, pp. 1-30, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[7] Qihe Huang, Xiaolong Xu, and Jinhui Chen, "Learning-Aided Fine Grained Offloading for Real-Time Applications in Edge-Cloud Computing," *Wireless Networks*, vol. 30, pp. 3805-3820, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[8] Ali Asghari, Mohammad Karim Sohrabi, and Farzin Yaghmaee, "Task Scheduling, Resource Provisioning, and Load Balancing on Scientific Workflows Using Parallel SARSA Reinforcement Learning Agents and Genetic Algorithm," *The Journal of Supercomputing*, vol. 77, pp. 2800-2828, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[9] Hongman Wang et al., "Service Migration in Mobile Edge Computing: A Deep Reinforcement Learning Approach," *International Journal of Communication Systems*, vol. 36, no. 1, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[10] 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*, Cambridge, MA, USA, pp. 1-4, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] 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]

[12] Yixue Hao et al., "Deep Reinforcement Learning for Edge Service Placement in Softwarized Industrial Cyber-Physical System," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5552-5561, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[13] 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]

[14] 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]

[15] Yicen Liu et al., "SFC Embedding Meets Machine Learning: Deep Reinforcement Learning Approaches," *IEEE Communications Letters*, vol. 25, no. 6, pp. 1926-1930, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] Miaojiang Chen et al., "Deep Reinforcement Learning for Computation Offloading in Mobile Edge Computing Environment," *Computer Communications*, vol. 175, pp. 1-12, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[17] M.S. Mekala et al., "Resource Offload Consolidation Based on Deep-Reinforcement Learning Approach in Cyber-Physical Systems," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 2, pp. 245-254, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[18] Bing Lin et al., "Computation Offloading Strategy Based on Deep Reinforcement Learning for Connected and Autonomous Vehicle in Vehicular Edge Computing," *Journal of Cloud Computing*, vol. 10, pp. 1-17, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[19] Qiang Liu, Tao Han, and Ephraim Moges, "EdgeSlice: Slicing Wireless Edge Computing Network with Decentralized Deep Reinforcement Learning," *2020 IEEE 40th International Conference on Distributed Computing Systems*, Singapore, pp. 234-244, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[20] Yiwei Zhang et al., "Computing Resource Allocation Scheme of IOV Using Deep Reinforcement Learning in Edge Computing Environment," *EURASIP Journal on Advances in Signal Processing*, vol. 2021, pp. 1-19, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[21] 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, 2022. [CrossRef] [Google Scholar] [Publisher Link]

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

[23] Bassem Sellami, Akram Hakiri, and Sadok Ben Yahia, "Deep Reinforcement Learning for Energy-Aware Task Offloading in Join SDN-Blockchain 5G Massive IoT Edge Network," *Future Generation Computer Systems*, vol. 137, pp. 363-379, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[24] Jiwei Huang et al., "Joint Computation Offloading and Resource Allocation for Edge-Cloud Collaboration in Internet of Vehicles via Deep Reinforcement Learning," *IEEE Systems Journal*, vol. 17, no. 2, pp. 2500-2511, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[25] Lixiang Zhang et al., "Distributed Real-Time Scheduling in Cloud Manufacturing by Deep Reinforcement Learning," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 8999-9007, 2022. [CrossRef] [Google Scholar] [Publisher Link]