FOREX Publication
Open Access | Rapid and quality publishing

International Journal of Computer Science Research (IJCSR)
Survey Article | Volume 3, Issue 2 | Pages 71-78 | ISSN XXXX-XXXX

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

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

[1]*Department of Computer Science,TSWRDCW, Nizamabad,Telangana 503003,India; mamatha3004@gmail.com*
[2]*Department of Computer science and Engineering, Jawaharlal Nehru Technological University, Hyderabad, Telangana-500085, India; supreethi.pujari@jntuh.ac.in*
[3]*Professor and Head of Department , CVR College of Engineering/IT Department,Hyderabad,Telangana-501510,India; bipinbjayasingh@cvr.ac.in*

*Correspondence: mamatha3004@gmail.com

ABSTRACT- In 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, usage of edge cloud for task scheduling became indispensable to reap benefits of latency performance with edge cloud. However, it is not possible to assign every task to edge cloud and resource-intensive tasks should be scheduled to cloud. With edge-cloud environment, it becomes very complex and scheduling is a NP-hard problem. Many existing methods based on reinforcement learning are found to have shortcoming in dealing with large action-space in presence of state-space. In this paper we proposed an algorithm known as Deep Deterministic Policy Gradient Algorithm for Dynamic Task Scheduling (DDPGA-TS). Our algorithm has a novel pruning strategy that continuously monitors action space and reduces it to improve overall performance in task scheduling. We used three scales of environments with our experiments. Number of performance indicators are used to evaluate performance of the proposed algorithm. Experimental results revealed that the proposed algorithm outperforms existing methods such as DDPG-NN and DDPG-CNN.

Keywords: Cloud Computing, Edge Computing, Dynamic Task Scheduling, Reinforcement Learning, Deep Learning.

ARTICLE INFORMATION

Author(s): D Mamatha Rani, Supreethi K P and Bipin Bihari Jayasingh;

Received: 00/00/2024; **Accepted**: 00/00/2024; **Published**: 30/06/2024;
e-ISSN: XXXX-XXXX;
Paper Id: IJCSR-030204;
Citation: 10.37391/IJCSR.030204

Publisher's Note: FOREX Publication stays neutral with regard to Jurisdictional claims in Published maps and institutional affiliations.

1. INTRODUCTION

With the emergence of Internet of Things (IoT) workflow applications, edge cloud became very significant for latency reasons. With edge cloud and cloud infrastructures available, decisions are to be made to schedule a task in edge cloud or cloud based on latency or other Service Level Agreements (SLAs). 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 QoS 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 edge-cloud environment is very useful when compared to relying on cloud alone.

From the literature, it was observed that many existing methods based on reinforcement learning are found to have shortcoming in dealing with large action-space in presence of state-space. Our contributions in this paper are as follows. Proposed an algorithm known as Deep Deterministic Policy Gradient Algorithm for Dynamic Task Scheduling (DDPGA-TS). Our algorithm has a novel pruning strategy that continuously monitors action space and reduces it to improve overall performance in task scheduling. We used three scales of environments with our experiments. Number of performance indicators are used to evaluate performance of the proposed algorithm. Experimental results revealed that the proposed algorithm outperforms existing methods such as DDPG-NN and DDPG-CNN. The remainder of the paper is structured as follows. Section 2 reviews literature on existing methods in task scheduling in edge-cloud environments. Section 3 presents our methodology for efficient task scheduling in edge-cloud environments. Section 4 presents results of experiments. Section 5 discusses significance of this work and provides limitations. Section 6 concludes our work and gives directions for future scope of the research.

2. RELATED WORK

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Computer Science Research (IJCSR)
Survey Article | Volume 3, Issue 2 | Pages 71-78 | ISSN XXXX-XXXX

This section reviews literature on prior works related to task scheduling in cloud. 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. Bulej et al. [3] found that CPS demand 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 effectively 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 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. Guo et al. [9] introduced SMDQN algorithm which optimizes service migration in MEC, balancing delay and migration costs effectively, 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 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 minimizes response time effectively. Tuli et al. [13] stated that fog computing optimizes IoT application tasks via A3C and R2N2 models, enhancing QoS 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] investigated on 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 CCR-RL algorithm optimally balances resource utilization, processing speed, and cost, achieving significant latency reduction. Lin et al. [18] used 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 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 the integration of blockchain 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 focus on achieving interoperability, implementing federated learning, and enforcing data privacy using homomorphic encryption.

Ye et al. [24] focused on joint optimization of computation offloading and resource allocation in IoV using a Markov decision process and deep reinforcement learning. 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 eta l. [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 focus on addressing uncertainties using digital twins in cloud manufacturing. From the literature, it was observed that many existing methods based on reinforcement learning are found to have shortcoming in dealing with large action-space in presence of state-space.

3. PROPOSED SYSTEM

In this paper, we considered dynamic task scheduling in edge-cloud computing. Our system model includes both edge cloud and cloud to optimize performance in task scheduling. 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, cloud also plays crucial role without which most of IoT and other use cases cannot be realized [16]. In such environment, it is important to achieve context aware resource provisioning. Some bandwidth hungry applications need minimal end to end latency as discussed in [1]. In such applications edge cloud plays crucial role. Based on the request arrived, 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 mention few.

3.1 Our Framework

Our system is based on the framework illustrated in Figure 1. The framework is known as Edge-Cloud based Deep Learning Framework (EC-DLF) for automatic efficient task scheduling.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Computer Science Research (IJCSR)
Survey Article | Volume 3, Issue 2 | Pages 71-78 | ISSN XXXX-XXXX

The framework is designed to run in edge-cloud for ease of access and learning. It is based on actor-critic model that is part of RL. The actor is synonymous to a policy function which continuously monitors s_t (state) of environment and involve in making 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 make changes to parameters of policy. At every given time, each dynamic task scheduling action results in appropriate reward. However, the goal of the framework is to maximize reward through policy optimization.

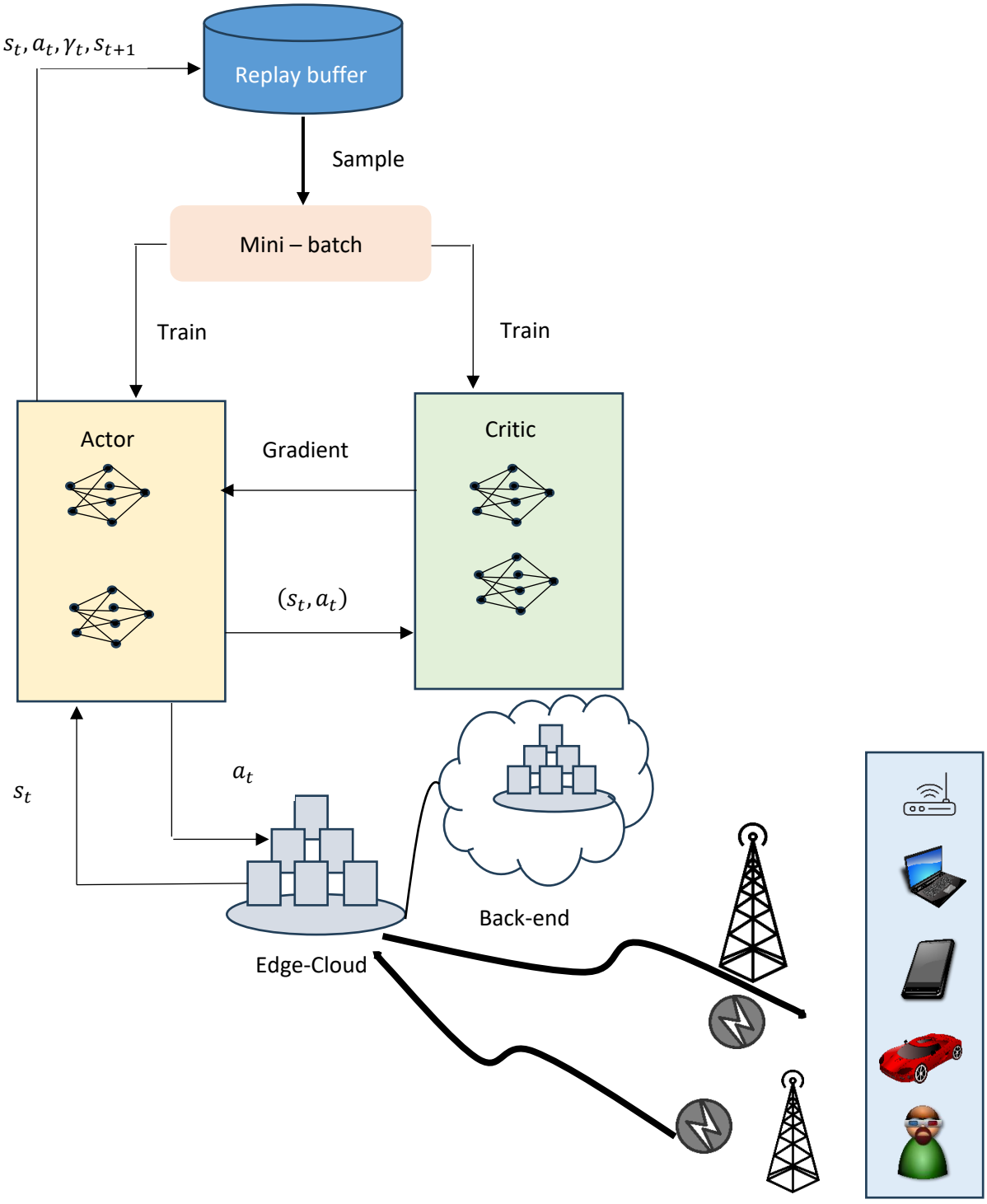

Figure 1: Proposed Edge-Cloud based Deep Learning Framework

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 edge cloud and cloud and the current job at the time given by user u is denoted as j_u. This job is expected to be scheduling in either cloud or edge cloud. The parameters associated with state are j_u (the job of given user), Ur_N (utility

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Computer Science Research (IJCSR)
Survey Article | Volume 3, Issue 2 | Pages 71-78 | ISSN XXXX-XXXX

rate of given node in cloud), Ur_M (utility rate of given node in edge cloud) and Ur_W (total utility rate of all resources). These parameters help an actor learns strategies for optimal and dynamic task scheduling. With respect to action space a_t, based on s_t, the actor determines resources required to run the job successfully and meeting deadline. The parameters of action space include the bandwidth (ch_h), CPU cycles (vm_k) and whether task is assigned to cloud or edge cloud (1 indicates cloud and 0 indicates edge cloud). The reward is computed as in *equation 1* which considers gain and cost involved in task scheduling.

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

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, completion time of given job and wireless resources, we redefine gain and cost as in *equation 2* and *equation. 3* respectively.

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

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

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

On the contrary to gain, cost includes resource and bandwidth of channel. The cost associated with bandwidth and VM (node) per a given unit time is denoted as c_h and c_k respectively. Unlike the work in [18], the computations for cost, gain and reward for each job. As each job can have its QoS needs, this approach is proved to be more useful. It also helps in optimizing policies associated with job scheduling. We considered optimization of resource utilization based on QoS needs of job and it is expressed as in *equation 4.*

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

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). There is another constraint that indicates job should be completed without exceeding deadline. *Table 1* shows the notations used in this paper.

Table 1: Notations used in the proposed system

Notation	Meaning
dl_u	Denote a deadline given by user
d_u	Data of 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 given time t
t	Denotes time
a_t	Denotes a continuous action
s_t	Denotes state
r_t	Denotes a reward at 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 wireless channel
c_k	Allocation of VM per unit time

Inspired by the work in [13], we optimized deep learning usage in our RL model. It makes use of Conv1D for local feature extraction. Temporal dependencies are learned by GRU besides an attention procedure is used to gain information that impacts prediction process. GRU uses minimal parameters and does not need memory unit leading to optimized performance when compared with LSTM [20]. We exploited the notion of experience replay for better training process. Our system has provision to choose bandwidth appropriately along with VM to execute given job. In order to choose a node or server that is less busy, we incorporated a pruning strategy which continuously and actively monitors and reduces action space significantly. The proposed mechanism also takes care of load balancing. Optimal policy required for scheduling is achieved with the pruning strategy.

Algorithm 1: Deep Deterministic Policy Gradient Algorithm for Dynamic Task Scheduling

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

1. Begin
2. Initialize replay memory M
3. For each scheduling interval
4. Reset environment
5. For each time instance t in T
6. Observe state
7. Prune action space
8. Predict an action
9. Receive reward and next state
10. Store transition in M
11. Train actor and critic
12. Update weight vectors
13. End For
14. Schedule tasks based on weight vectors
15. End For
16. End

As presented in Algorithm 1, it takes user jobs with QoS needs as input and results in optimal task scheduling. It has an iterative process to prune 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 is denoted as U.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Computer Science Research (IJCSR)
Survey Article | Volume 3, Issue 2 | Pages 71-78 | ISSN XXXX-XXXX

Each job is denoted as a tuple, denoted as $j_u(dl_u, d_u)$ consisting of deadline and data to be processed. Therefor 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 job needs memory and CPU for processing. Our approach in CPU cycles modelling is similar to that of [12]. Our bandwidth model is described here. In edge computing there are many nodes that support wireless access. B denotes 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 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 cloud with 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) that includes end-to-end time right from sending job to completion of job and sending results back. This delay model is somewhat similar to that of [10]. The propagation time of given job is considered as 5ms for edge cloud and 50ms for cloud. The transmission time for given job with respect to 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 given job with respect to 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 edge cloud and cloud is computed as in *equation 7* and *equation 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 *equation 10* for edge cloud and cloud respectively.

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

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

We also proposed our utility model which indicates utilization of resources at given time t. 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, for each node in edge cloud and cloud, the utility rate is computed as in *equation 12* and *equation 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 important role in the proposed RL based system which enables efficient scheduling of dynamic workloads in edge-cloud environment.

4. RESULTS AND DISCUSSION

We built a prototype application to simulate the proposed framework and the underlying algorithm. We make all experiments using a Dell PC with i7 processor of 2.9 GHz, 128GB RAM and 64-bit Windows 11 OS. Results of our experiments are compared with two existing methods known as DDPG-NN and DDPG-CNN taken from [19]. We used three different environments as shown in *table 2* for experiments.

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

Between edge cloud and cloud the bandwidth availability is 1Gbps. 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 number of iterations for each episode is set to 1000.

4.1 Convergence Comparison

Convergence of different models is discussed here in terms of normalized reward and normalized training loss. The convergence experiments are made in small scale environment as in *table 2*.

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

As presented in *table 3*, normalized reward is provided against number of episodes in the process of task scheduling in 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

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Computer Science Research (IJCSR)
Survey Article | Volume 3, Issue 2 | Pages 71-78 | ISSN XXXX-XXXX

As presented in *table 4*, normalized training loss is provided against number of episodes in the process of task scheduling in edge-cloud environment.

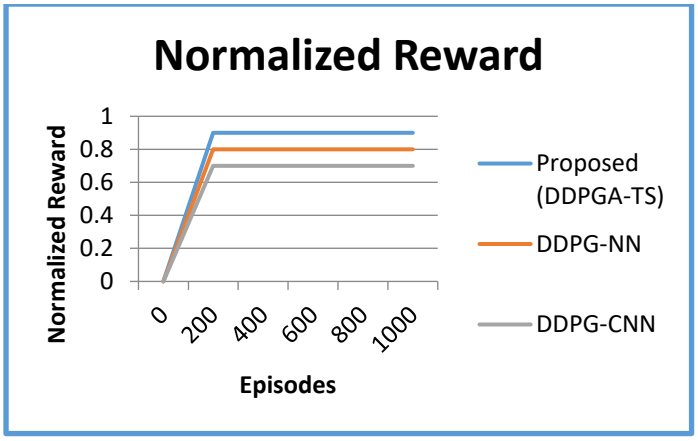

Figure 2: Normalized reward against episodes

The convergence rate, as presented in *figure 2*, in terms of normalized reward is observed against number of episodes. The normalization process is carried out with simple min-min approach. The performance of the proposed model is better as it is capable of learning local features and also long term features.

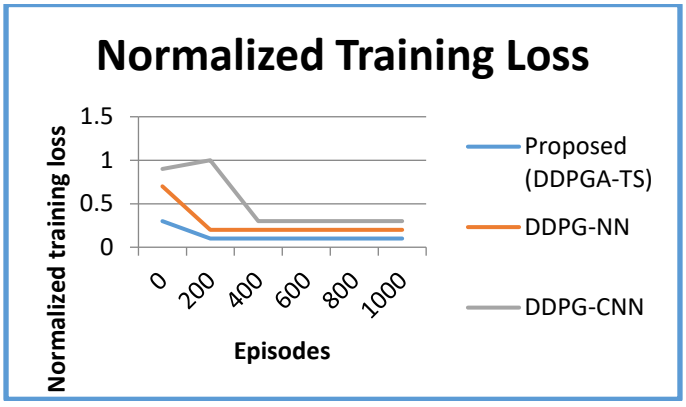

Figure 3: Normalized training loss against episodes

The convergence rate, as presented in *figure 3*, in terms of normalized training is observed against number of episodes. The normalization process is carried out with simple min-min approach. The performance of the proposed model is better as it is capable of learning local features and also long term features. The training loss reflects superior convergence with the proposed model.

4.2 Performance Evaluation

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

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

As presented in *table 5*, operational cost of the three models is provided. Less average operational cost indicates better performance.

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

As presented in *table 6*, rejection rate of the three models is provided. Less average rejection rate indicates better performance.

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

As presented in *table 6*, QoE of the three models is provided. High QoE indicates better performance.

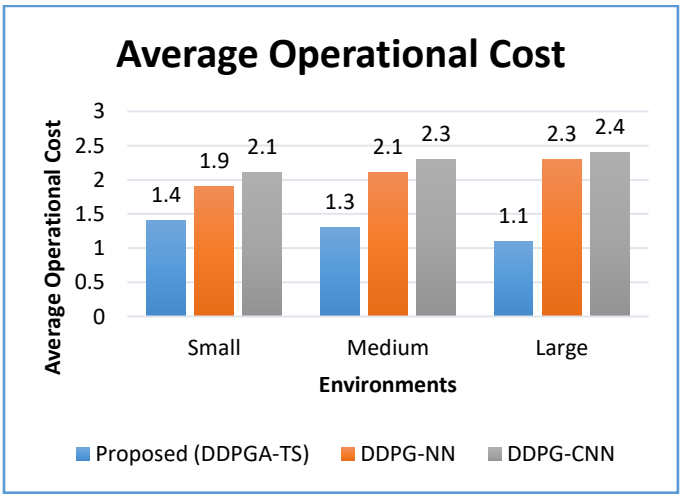

Figure 4: Average operational cost among the models

As presented in *figure 4*, the average operational cost is provided against the three environments. When small environment is used for experiments, the cost of DDPG-CNN is 2.1, DDPG-NN is 1.9 while the proposed DDPGA-TS is 1.4. When medium environment is used for experiments, the cost of DDPG-CNN is 2.3, DDPG-NN is 2.1 while the proposed DDPGA-TS is 1.3. When large environment is used for experiments, the cost of DDPG-CNN is 2.4, DDPG-NN is 2.3

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Computer Science Research (IJCSR)
Survey Article | Volume 3, Issue 2 | Pages 71-78 | ISSN XXXX-XXXX

while the proposed DDPGA-TS is 1.1. From the results, it is observed that the proposed model exhibits better performance over the existing models.

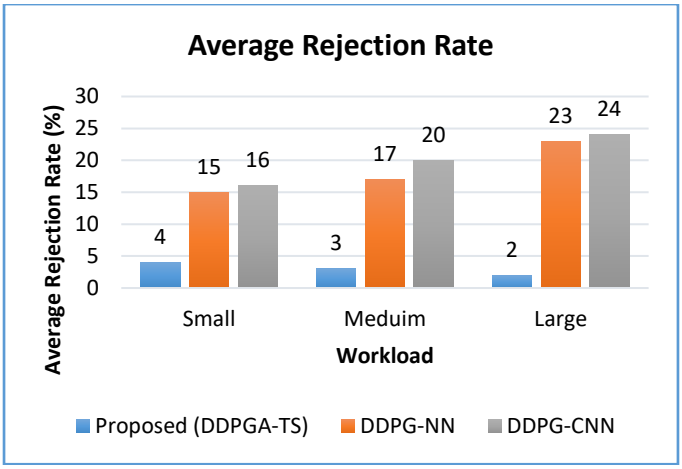

Figure 5: Average rejection rate among the models

As presented in *figure 5*, the average rejection rate is provided against the three environments. When small environment is used for experiments, the rejection rate of DDPG-CNN is 16, DDPG-NN is 15 while the proposed DDPGA-TS is 4. When medium environment is used for experiments, the rejection rate of DDPG-CNN is 20, DDPG-NN is 17 while the proposed DDPGA-TS is 3. When large environment is used for experiments, the rejection rate of DDPG-CNN is 24, DDPG-NN is 23 while the proposed DDPGA-TS is 2. From the results, it is observed that the proposed model exhibits better performance over the existing models in terms of rejection rate.

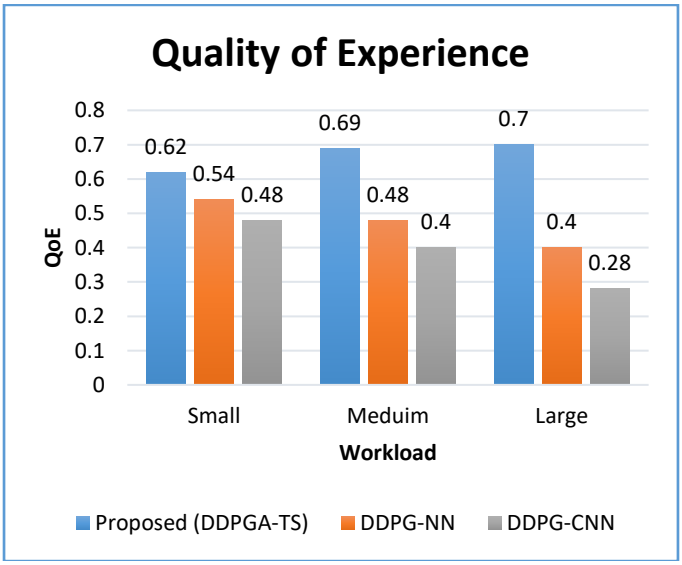

Figure 6: QoE among the models

As presented in *figure 6*, the QoE is provided against the three environments. When small environment is used for experiments, the QoE of DDPG-CNN is 0.48, DDPG-NN is 0.54 while the proposed DDPGA-TS is 0.62. When medium environment is used for experiments, the QoE of DDPG-CNN is 0.4, DDPG-NN is 0.48 while the proposed DDPGA-TS is 0.69. When large environment is used for experiments, the QoE of DDPG-CNN is 0.28, DDPG-NN is 0.4 while the proposed DDPGA-TS is 0.7. From the results, it is observed that the proposed model exhibits better performance over the existing models in terms of QoE. In summary, the proposed DDPGA-TS model as adaptive approach in choosing suitable VMs, wireless channels and optimize in scheduling of tasks to edge-cloud environment. The pruning strategy involved in the proposed methodology helps it in reduction of rejection rate. In all the environments, the proposed model could outperform all existing methods.

5. CONCLUSION AND FUTURE WORK

We proposed a framework named Edge-Cloud based Deep Learning Framework (EC-DLF) for automatic efficient task scheduling. The framework is designed to run in edge-cloud for ease of access and learning. It is based on actor-critic model that is part of RL. The actor is synonymous to a policy function which continuously monitors s_t (state) of environment and involve in making appropriate action (a_t) based on s_t. We proposed an algorithm known as Deep Deterministic Policy Gradient Algorithm for Dynamic Task Scheduling (DDPGA-TS). Our algorithm has a novel pruning strategy that continuously monitors action space and reduces it to improve overall performance in task scheduling. Our system model includes both edge cloud and cloud to optimize performance in task scheduling. The rationale behind edge cloud is that, it could render services faster to minimize latency which is essential to honour SLAs. We used three scales of environments with our experiments. Number of performance indicators are used to evaluate performance of the proposed algorithm. Experimental results revealed that the proposed algorithm outperforms existing methods such as DDPG-NN and DDPG-CNN. In future, we intend to improve our framework with parallelized techniques to leverage training and learning process.

REFERENCES

[1] Jaber Almutairi and Mohammad Aldossary; (2021). A novel approach for IoT tasks offloading in edge-cloud environments . Journal of Cloud Computing, p1-19.

[2] Shaoshuai Ding; Lei Yang; Jiannong Cao; Wei Cai; Mingkui Tan and Zhenyu Wang; (2021). Partitioning Stateful Data Stream Applications in Dynamic Edge Cloud Environments . IEEE Transactions on Services Computing, p1-14.

[3] Bulej, LubomÃ¬r; BureÅ¡, TomÃ¡Å¡; Filandr, Adam; HnÄ›tynka, Petr; HnÄ›tynkovÃ¡, Iveta; PacovskÃ, Jan; Sandor, Gabor and Gerostathopoulos, Ilias (2021). Managing latency in edgeâ€ "cloud environment. Journal of Systems and Software, 172, p1-15.

[4] Xu Zhao; Guangqiu Huang; Ling Gao; Maozhen Li and Quanli Gao; (2021). Low load DIDS task scheduling based on Q-learning in edge computing environment . Journal of Network and Computer Applications, p1-12.

[5] Yu Zhang, Bing Tang, Jincheng Luo and Jiaming Zhang. (2022). Deadline-Aware Dynamic Task Scheduling in Edge–Cloud Collaborative Computing. MDPI. 11, pp.1-24.

[6] Abdullah Lakhan, Mazin Abed Mohammed, Karrar Hameed Abdulkareem and Mu. (2022). Delay Optimal Schemes for Internet of Things Applications in Heterogeneous Edge Cloud Computing Networks. MDPI. 22, pp.1-30.

International Journal of Computer Science Research (IJCSR)
Survey Article | Volume 3, Issue 2 | Pages 71-78 | ISSN XXXX-XXXX

[7] Qihe Huang;Xiaolong Xu and Jinhui Chen; (2021). Learning-aided fine grained offloading for real-time applications in edge-cloud computing . Wireless Networks, p1-16.

[8] Asghari, Ali; Sohrabi, Mohammad Karim and Yaghmaee, Farzin (2020). Task scheduling, resource provisioning, and load balancing on scientific workflows using parallel SARSA reinforcement learning agents and genetic algorithm. The Journal of Supercomputing.

[9] Wang, Hongman; Li, Yingxue; Zhou, Ao; Guo, Yan and Wang, Shangguang (2020). Service migration in mobile edge computing: A deep reinforcement learning approach. International Journal of Communication Systems, e4413–.

[10] Bassem Sellami;Akram Hakiri;Sadok Ben Yahia 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).

[11] 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.

[12] Yixue Hao; Min Chen; Hamid Gharavi; Yin Zhang and Kai Hwang; (2021). Deep Reinforcement Learning for Edge Service Placement in Softwarized Industrial Cyber-Physical System . IEEE Transactions on Industrial Informatics.

[13] Tuli, Shreshth; Ilager, Shashikant; Ramamohanarao, Kotagiri and Buyya, Rajkumar (2020). Dynamic Scheduling for Stochastic Edge-Cloud Computing Environments using A3C learning and Residual Recurrent Neural Networks. IEEE Transactions on Mobile Computing, 1–1.

[14] Qi Zhang; Lin Gui; Shichao Zhu and Xiupu Lang; (2021). Task Offloading and Resource Scheduling in Hybrid Edge-Cloud Networks . IEEE Access.

[15] Yicen Liu;Yu Lu;Xi Li;Wenxin Qiao;Zhiwei Li and Donghao Zhao; (2021). SFC Embedding Meets Machine Learning: Deep Reinforcement Learning Approaches . IEEE Communications Letters.

[16] Miaojiang Chen; Tian Wang; Shaobo Zhang and Anfeng Liu; (2021). Deep reinforcement learning for computation offloading in mobile edge computing environment . Computer Communications.

[17] M. S. Mekala; Alireza Jolfaei; Gautam Srivastava; Xi Zheng; Amjad Anvari-Moghaddam and P. Viswanathan; (2022). Resource Offload Consolidation Based on Deep-Reinforcement Learning Approach in Cyber-Physical Systems . IEEE Transactions on Emerging Topics in Computational Intelligence.

[18] Bing Lin;Kai Lin;Changhang Lin;Yu Lu;Ziqing Huang and Xinwei Chen; (2021). Computation offloading strategy based on deep reinforcement learning for connected and autonomous vehicle in vehicular edge computing . Journal of Cloud Computing,.

[19] Qiang Liu;Tao Han and Ephraim Moges; (2020). EdgeSlice: Slicing Wireless Edge Computing Network with Decentralized Deep Reinforcement Learning . 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS),.

[20] Yiwei Zhang; Min Zhang; Caixia Fan; Fuqiang Li and Baofang Li; (2021). Computing resource allocation scheme of IOV using deep reinforcement learning in edge computing environment . EURASIP Journal on Advances in Signal Processing, .

[21] Xiaokang Zhou, Wei Liang, Ke Yan, Weimin Li, Kevin I-Kai Wang, Jianhua Ma and Qun Jin. (2023). Edge-Enabled Two-Stage Scheduling Based on Deep Reinforcement Learning for Internet of Everything. IEEE. 10(4), pp.3295-3304. https://creativecommons.org/licenses/by/4.0/

[22] 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, pp.1-20.

[23] Bassem Sellami, Akram Hakiri and Sadok Ben Yahia. (2022). Deep Reinforcement Learning for Energy-Aware Task Offloading in Join SDN-Blockchain 5G massive IoT Edge Network. Elsevier, pp.1-19.

[24] Jiwei Huang, Jiangyuan Wan, Bofeng Lv, Qiang Ye and Ying Chen. (2023). Joint Computation Offloading and Resource Allocation for Edge-Cloud Collaboration in Internet of Vehicles via Deep Reinforcement Learning. IEEE. 17(2), pp.2500-2511. http://doi:10.1109/JSYST.2023.3249217

[25] Lixiang Zhang, Chen Yang, Yan Yan and Yaoguang Hu. (2022). Distributed Real-Time Scheduling in Cloud Manufacturing by Deep Reinforcement Learning. IEEE. 18(12), pp.8999-9007. http://doi:10.1109/TII.2022.3178410