MULTI-CRITERIA DEEP Q-NETWORK SCHEDULING FOR LOAD BALANCING IN HETEROGENEOUS EDGE COMPUTING ENVIRONMENTS

Muhammad Rehan Rasheed (1), Rabia Noor Enam (1), Rukaiya Rukaiya (1), Muhammad Naseem (2)*

(1) Department of Computer Engineering; (2) Department of Software Engineering
Sir Syed University of Engineering & Technology, Karachi
Pakistan

* Contact Person, e-mail: murasheed@ssuet.edu.pk

Abstract: Task scheduling in heterogeneous edge computing is an online, NP-hard problem that must jointly account for server hardware heterogeneity and time varying load. Deep reinforcement learning schedulers learn server capability only from experience, while multi criteria decision making yields informative but static rankings. This paper proposes MCDRL-HLB, which augments the state of a Double Deep Q-Network agent with Complex Proportional Assessment utility scores derived from server hardware. Four conditions, DQN, DDQN, COPRAS-DQN and MCDRL-HLB, were evaluated under identical settings in EdgeCloudSim across three random seeds. MCDRL-HLB lowered mean task failure rate from 22.67% to 18.42% against the DQN baseline and cut cross-seed variance by a factor of 1.8, improving both accuracy and convergence consistency. Generalisation outside the training load remained seed sensitive, a boundary reported openly alongside the demonstrated gains.

Key words: edge computing, deep reinforcement learning, task scheduling, multi-criteria decision-making, load balancing.

1. INTRODUCTION

The proliferation of Internet of Things (IoT) devices and smart city applications has triggered unprecedented growth in computation intensive workloads at the network edge [1]. IoT end devices have severely constrained processing, storage and battery capacity, which renders local computation impractical for many applications. Cloud computing offers virtually unlimited resources, but its centralised architecture introduces substantial end to end latency and is inadequate for latency sensitive applications such as augmented reality, real time health monitoring and interactive infotainment. Edge and fog computing close this gap by extending computation toward the network edge.

Unlike centralised data centres, edge and fog nodes are numerous, geographically distributed, resource constrained and frequently heterogeneous in hardware capability.

Task scheduling must therefore jointly optimise resource utilisation, minimise failures and satisfy diverse Quality of Service requirements online, without prior knowledge of future arrivals. As devices and candidate servers grow in number the decision space expands combinatorially, motivating learning-based solutions [2].

Deep Reinforcement Learning (DRL) lets a scheduling agent learn a policy directly through interaction with the environment. Deep Q-Networks (DQN) and Double DQN (DDQN) have been applied to edge and fog scheduling with promising results [3, 4, 5]. Purely DRL-based schedulers, however, must learn each server's relative suitability entirely from experience, holding no structured prior on which servers are inherently more capable, which is limiting where hardware heterogeneity is substantial. Multi-Criteria Decision-Making (MCDM) methods such as Complex Proportional Assessment (COPRAS) [6] rank resources systematically against multiple criteria but produce static rankings computed independently of the evolving system state.

This paper proposes MCDRL-HLB (Multi-Criteria Deep Reinforcement Learning-based Hybrid Load Balancing), which augments a DDQN agent's state with COPRAS-derived utility scores for heterogeneous edge servers, establishing whether such a structured prior measurably improves scheduling accuracy and run to run reliability. The contribution is threefold: a hybrid framework embedding static COPRAS scores alongside dynamic load-based features; a controlled four condition ablation, DQN, DDQN, COPRAS-DQN and MCDRL-HLB, evaluated under identical training, environment and reward settings; and statistical validation across three random seeds, extended by a device count generalisation analysis that reports, rather than obscures, substantial seed-sensitivity outside the training regime.

The remainder of the paper is organised as follows. Section 2 reviews related work. Section 3 describes the system model and formulates the scheduling problem. Section 4 presents the proposed MCDRL-HLB method. Section 5 reports the experimental setup, results and discussion. Section 6 concludes the paper and outlines future work.

2. RELATED WORK

Multi-criteria decision-making has been widely applied to static or semi-dynamic scheduling. The study in [7] integrated entropy weighting with COPRAS to rank fog nodes, producing a lightweight method that nonetheless operates offline. Building on similar principles, in [8] Fuzzy Analytic Hierarchy Process weighting was combined with Fuzzy TOPSIS ranking for fog/cloud load balancing. The work in [9] applied TOPSIS within a Fog Service Manager for real time provisioning, while in [10] TOPSIS and AHP were integrated for trust aware offloading across multi provider systems. Such approaches share a common limitation: rankings are computed independently of the evolving system state, so they cannot adapt to shifting workloads or learn long-term policies.

With the advancement of deep reinforcement learning, many researchers have shifted toward learning based scheduling. In [11] representation learning was combined with reinforcement learning, using embeddings derived from historical traces rather than hardware-capability priors. The study in [12] modelled fog scheduling as a Markov Decision Process solved by Q-learning, and in [13] genetic algorithms were combined

with SARSA and Q-learning. More recently, in [14] multi agent meta reinforcement learning enabled adaptation to unseen configurations, in [15] an attention weighted actor-critic framework addressed large scale offloading, and in [16] multi objective reinforcement learning targeted joint latency, energy and cost. Across these works the agent receives no explicit ranking of server capability and must infer suitability entirely from experience.

Double Q-learning variants address overestimation bias: the approach in [4] mitigates Q-value overestimation for mobile edge offloading, and in [5] prioritised experience replays improved stability and convergence. The study in [3] proposed DQN-based workload scheduling evaluated in EdgeCloudSim [17] under a homogeneous edge configuration; its state representation and deviation normalised reward form the algorithmic basis extended here. Hybrid schemes coupling multi-criteria knowledge with reinforcement learning remain rare, with the fuzzy orchestration strategy in [18] remaining static. Relevant work has also appeared in this journal, including the software-defined approach in [19].

Despite these advances, no prior study has isolated the individual and combined contributions of multi criteria state augmentation and Double Q-learning under a controlled ablation against a shared baseline in a heterogeneous edge environment. Unlike [3], which assumes uniform server capacity, this study explicitly models a three-tier heterogeneous configuration, because meaningful ranking requires genuine capacity differentiation between servers. This paper attempts to resolve these challenges.

3. SYSTEM MODEL AND PROBLEM FORMULATION

The three-tier architecture considered here is shown in Figure 1.

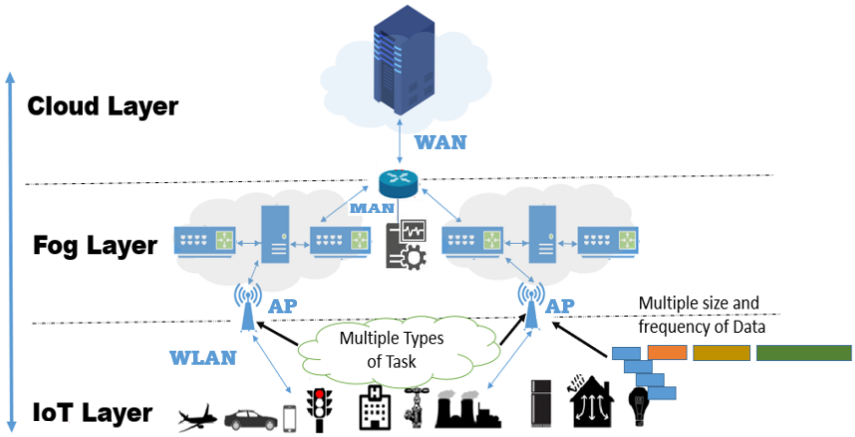


Figure 1. Three-tier edge computing architecture connected via WLAN, MAN and WAN

Devices connect to their nearest edge server over a Wireless Local Area Network (WLAN); neighbouring edge servers are reachable through a Metropolitan Area Network (MAN) backbone for collaborative task migration; and the cloud server is reached over a Wide Area Network (WAN). An edge orchestrator hosting the trained policy schedules each arriving task to one of these three destinations from the current

system state. The environment consists of K mobile or IoT devices, M heterogeneous edge servers and a single cloud server

3.1. Task and computation model

A task is described by its input and output data sizes, required CPU cycles and delay deadline, as in equation (1). Tasks are atomic and independent, and may run on the local edge server, migrate to a neighbour or offload to the cloud.

$$\tau_{\mathrm{i}} = (I_{\mathrm{i}}, O_{\mathrm{i}}, C_{\mathrm{i}}, \delta_{\mathrm{i}}) \quad (1)$$

The service time on the local edge server combines queueing wait and execution time, as given in equation (2), where fl is the local computing capacity. Service times on a neighbouring server and on the cloud follow analogously, with cloud waiting time assumed negligible because cloud resources are treated as always sufficient.

$$t_{\mathrm{i}}^{\mathrm{s}} = t_{\mathrm{i}}^{\mathrm{wait}} + C_{\mathrm{i}} / fl \quad (2)$$

3.2. Network model

The WLAN and WAN links are modelled from empirically measured throughput profiles as a function of concurrently contending users, while the MAN link is modelled analytically by M/M/1 queueing theory, consistent with the EdgeCloudSim environment. For a task of data size D_i the WLAN transmission delay is given by equation (3), where $n_{wlan(s)}$ is the number of devices contending at access point s and η = 3 scales 802.11ac throughput relative to the empirical 802.11n measurements.

$$t_{\mathrm{i}}^{\mathrm{WLAN}} = 8D_{\mathrm{i}} / (\eta \cdot R^{\mathrm{WLANexp}}[n_{wlan(s)}]) \quad (3)$$

When a neighbouring server is selected the task is relayed over the MAN backbone, whose queueing delay is given by equation (4), where μ_{man} and λ_{man} are the MAN service and per device arrival rates and tLAN the internal propagation delay. A queueing delay exceeding 15 s is treated as a rejection.

$$t_{\mathrm{i}}^{\mathrm{man}} = \{ 1/(\mu_man - \lambda_man\, K) + t_LAN\} \quad (4)$$

3.3. Problem formulation

A task fails when its completion delay exceeds its deadline, when the selected server lacks virtual machine capacity to admit it, or when mobility carries the device out of the serving access point's coverage before the response arrives. The objective is to minimise total delay and failure rate, as in equation (5), where exactly one decision variable equals one per task.

$$min\ t = \Sigma_{\mathrm{i}=1}^{\mathrm{n}} (\lambda_1 t_{\mathrm{i}}^{\mathrm{local}} + \lambda_2 t_{\mathrm{i}}^{\mathrm{nb(1)}} + \cdots + \lambda_{\mathrm{m}} t_{\mathrm{i}}^{\mathrm{nb(m-1)}} + \lambda_{\mathrm{m}+1} t_{\mathrm{i}}^{\mathrm{cloud}}) \quad (5)$$

With fourteen edge servers this yields fifteen possible decisions per task. The formulation is a non-convex, NP-hard combinatorial optimisation problem, further complicated by contention dependent throughput, MAN queueing load and time varying virtual machine utilisation, which motivates the learning-based approach developed next.

4. PROPOSED MCDRL-HLB METHOD

MCDRL-HLB integrates COPRAS with a Double Deep Q-Network. COPRAS is applied to static, hardware derived criteria to produce utility scores, incorporated into the agent's state alongside dynamic system features so it can reason jointly about which servers are inherently more capable, and which are currently under loaded. Four conditions are evaluated under identical settings: a plain DQN baseline, DDQN, COPRAS augmented DQN and the full proposed method.

4.1. Heterogeneous edge server configuration

Unlike prior studies assuming homogeneous edge capacity, fourteen servers are configured across three tiers, summarised in Table 1. Each tier scales core count, processing capacity, memory and storage in a 4:2:1 ratio, with eight virtual machines per host. This differentiation is necessary for multi-criteria ranking to convey useful information.

Table 1. Heterogeneous edge server tier configuration

Tier	Servers	Cores	MIPS	RAM (MB)	Storage (MB)
High	4	32	160 000	32 000	800 000
Mid	6	16	80 000	16 000	400 000
Low	4	8	40 000	8 000	200 000

4.2. COPRAS-based server ranking

COPRAS ranks the fourteen servers using four beneficial criteria: core count, processing capacity, memory and storage. Given the decision matrix x_{ij}, the normalised weighted beneficial sum is computed by equation (6) with equal weights of 0.25. Because every criterion is beneficial, the non-beneficial sum is zero and the relative significance score reduces to the beneficial sum alone.

$$S_i^+ = \Sigma_{j=1}^{n} w_j \cdot x_{ij} / (\Sigma_{k=1}^{m} x_{kj}) \quad (6)$$

The final normalised utility score is obtained by equation (7), yielding values of 1.00, 0.50 and 0.25 for the high, mid and low tiers. Because the criteria are fixed hardware attributes, the score vector is static and computed once rather than at each decision epoch and is kept separate from the time varying components of the state so that structural capability and instantaneous load remain distinguishable to the agent.

$$N_i = Q_i / max_k Q_k \times 100\% \quad (7)$$

4.3. State, action and reward formulation

The problem is formulated as a Markov Decision Process. At each decision epoch the state is assembled as in equation (8), where $\boldsymbol{g_t}$ is a nine-dimensional vector of system and task features, $\boldsymbol{v_t}$ holds the available virtual machine capacity ratio for each server and N is the static COPRAS score vector, included only under COPRAS-DQN and MCDRL-HLB. State dimensionality is therefore 23 for DQN and DDQN and 37 for the COPRAS-augmented conditions. The action space comprises fifteen assignments: the local server, thirteen neighbours and the cloud.

$$s_t = [\, g_t, v_t, N \,] \quad (8)$$

For delay sensitivity ρt the deadline is interpolated linearly between 0.5 s and 5.0 s. Given the completion delay Tt, the reward is computed by equation (9), where the denominator is the running average absolute deviation over all tasks processed so far. The reward is positive when a task meets its deadline and negative otherwise.

$$R_t = (\delta_t - T_t) / D_t \quad (9)$$

4.4. Training process

The agent uses two hidden layers of 64 ReLU neurons and a linear output layer of fifteen neurons, trained to minimise the mean squared error between predicted and target Q-values. Under DDQN and MCDRL-HLB the target is computed by equation (10), where action selection uses the online network and value estimation the target network, reducing overestimation bias; DQN and COPRAS-DQN use the target network for both.

$$y_j = r_j + \gamma\, Q^-(s_{j+1}, arg\ max_a{}'\, Q(s_{j+1}, a';\, \theta);\, \theta^-) \quad (10)$$

Training hyperparameters are listed in Table 2. Training is performed exclusively at 2400 mobile devices, the most resource-constrained scenario evaluated, to encourage robust policies under high stress. Each episode spans five simulated minutes including a one-minute warm up, generating roughly 80 000 to 85 000 tasks, so 25 episodes run corresponds to over two million task-level updates. Average episodic reward and Q-value stabilise within the first ten to twenty episodes across all conditions.

Table 2. Training hyperparameters

Hyperparameter	Value	Hyperparameter	Value
Learning rate	0.0001	Optimizer	Adam
Discount factor	0.80	Hidden layers / neurons	2 / 64
Exploration rate	1.0 to 0.1	Training episodes	25
Replay buffer	1 000 000	Devices per episode	2400
Mini-batch size	4	Random seeds	1234, 42, 777
Target sync	10 steps		

The two procedures that constitute the proposed method are summarised below. The first computes the static multi-criteria utility scores; the second trains the scheduling agent on the augmented state.

Algorithm for COPRAS-based server ranking

1. Collect the hardware specification of each server: cores, processing capacity, memory and storage.
2. Form the decision matrix x[i][j], with all four criteria treated as beneficial.
3. Normalise each column by its sum and apply the criterion weight w[j] = 0.25.
4. Compute the weighted beneficial sum S+[i] for every server using equation (6).
5. Set the non-beneficial sum S-[i] = 0, so the relative significance score Q[i] = S+[i].
6. Normalise against the maximum score to obtain the utility score N[i] using equation (7).
7. Return the static score vector N = [N1, ..., N14] for use in the agent state.

Algorithm for MCDRL-HLB agent training

1. Compute the static COPRAS utility score vector N for all 14 edge servers.
2. Initialise the replay buffer D, the online network and the target network with $\theta^- = \theta$.
3. Set the exploration rate to its initial value and the step counter to zero.
4. For each episode, and each task arrival at 2400 devices, repeat steps 5 to 10.
5. Construct the state s[t] = [g[t], v[t], N] from task features, VM availability and static scores.
6. Select an action by the epsilon-greedy rule over the fifteen available assignments.
7. Execute the action; on completion or failure observe reward r[t] from equation (9) and the next state.
8. Store the transition in D, discarding the oldest entry when the capacity is exceeded.
9. Sample a mini batch from D, compute targets using equation (10) and update θ by gradient descent.
10. Synchronise θ^- with θ every 10 steps and decay the exploration rate.
11. Freeze θ and evaluate the best checkpoint greedily with the exploration rate set to zero.

DQN and MCDRL-HLB were each trained under three random seeds; DDQN and COPRAS-DQN used a single seed and are reported for reference only. The best checkpoint per condition is then frozen and evaluated in inference only mode with fully greedy action selection.

5. EXPERIMENTAL RESULTS AND DISCUSSION

Experiments used the EdgeCloudSim simulator. The environment comprises fourteen heterogeneous edge servers, one cloud server with four virtual machines of 100 GIPS, 32 GB memory and 1000 GB storage each, and 200 to 2400 mobile devices. WAN propagation delay is 100ms, LAN delay 5ms, and movement follows a nomadic mobility model. Each policy was evaluated across the full sweep in increments of 200, with five repetitions per count. Devices execute four application types following [20]: augmented reality (inter arrival 2s, latency sensitivity 0.9, 30% of usage), compute intensive (20s, 0.1, 20%), health (3s, 0.7, 20%) and infotainment (7s, 0.3, 30%).

5.1. Statistical comparison at training scale

Table 3 reports task failure rate at 2400 devices, the count at which all models were trained, across three seeds. MCDRL-HLB achieves a lower mean failure rate than the DQN baseline, 18.42% against 22.67%, with a standard deviation approximately 1.8 times smaller, 3.40 against 6.10. The proposed method therefore converges more consistently regardless of random initialisation, whereas the baseline varies from 16.50% to 28.70% across runs. This robustness matters in practice, because a deployed scheduler cannot select a favourable seed in advance.

Table 3. Task failure rate at 2400 devices across three random seeds

Condition	Seed 1234	Seed 42	Seed 777	Mean	Std. dev.
DQN	16.50%	28.70%	22.80%	22.67%	6.10
MCDRL-HLB	17.20%	22.26%	15.79%	18.42%	3.40

5.2. Task failure rate, service time and utilisation

Figure 2 presents task failure rate against device count. The DQN baseline shows the characteristic pattern below 3% through approximately 1000 devices, then a sharp rise once demand exceeds the combined edge capacity, peaking at 36.72% at 1800 devices before partially declining. DDQN and COPRAS-DQN each display a non-monotonic pattern, with elevated failure at light load that improves at moderate load before rising again, consistent with policies trained exclusively at 2400 devices generalising unevenly to out-of-distribution counts. The mean MCDRL-HLB curve remains the most consistently low from 1200 devices onward.

The Figure 3 shows average service time, which increases with device count as queues lengthen under contention. COPRAS-DQN is markedly higher through approximately 1800 devices, plateauing near eight to nine seconds, before dropping sharply beyond 2000 devices as its policy begins offloading to the cloud, while MCDRL-HLB stays within a narrow band of 2.2 to 2.9 seconds regardless of device count.

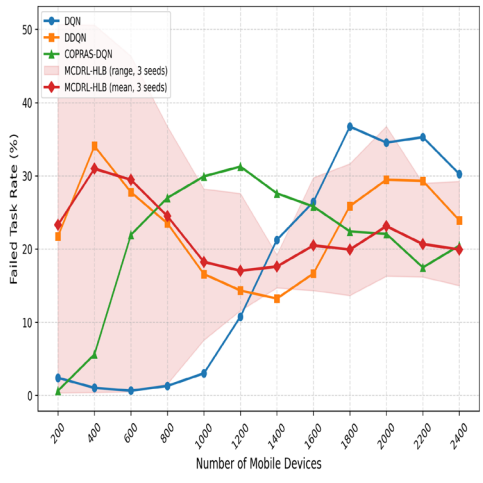


Figure 2. Task failure rate against number of mobile devices for all four conditions

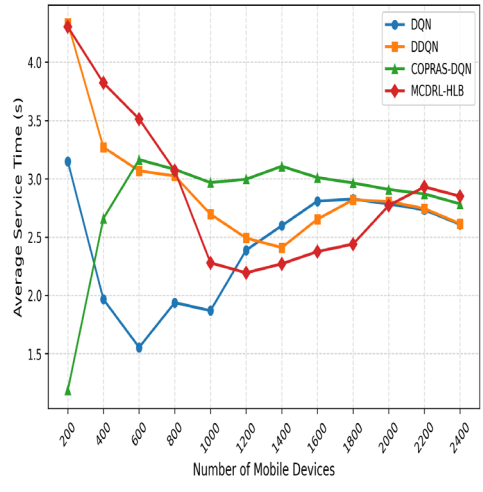


Figure 3. Average service time against number of mobile devices

The Figure 4 indicates that average edge virtual machine utilisation rises with device count as more tasks are admitted before capacity is exhausted. MCDRL-HLB reaches the highest utilisation of all conditions at 2400 devices, 46.52%, indicating fuller use of available edge capacity, particularly on higher-tier servers.

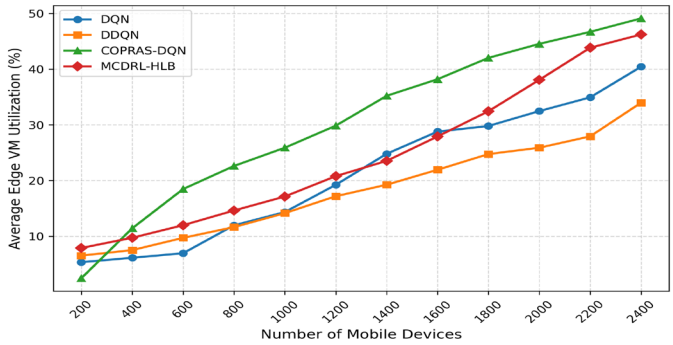


Figure 4. Average edge virtual machine utilisation against number of nodes

5.3. Causes of task failure

The Figure 5 decomposes the causes of task failure at 2400 devices, summed across five repetitions. Capacity exhaustion dominates across all conditions, reflecting the constraint imposed by the edge tier at peak load. MCDRL-HLB records the lowest capacity-related failure count, 58 889 against 118 884 for the DQN baseline, indicating that the COPRAS-derived prior helps the agent avoid over-concentrating tasks on exhausted servers. DDQN shows by far the highest network-related count, 27 100, and MCDRL-HLB the lowest, 179.

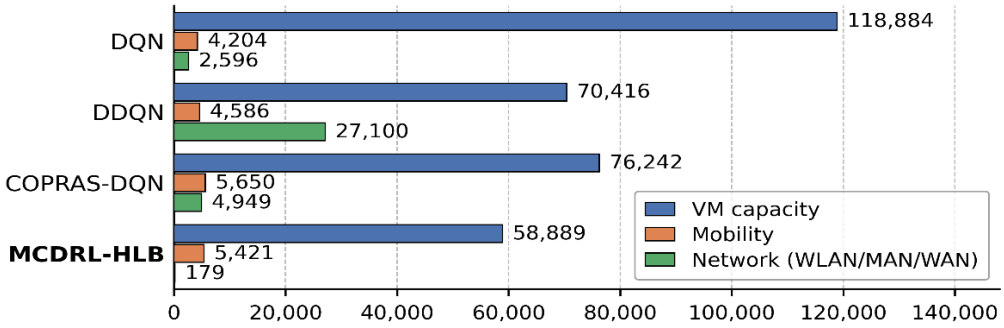


Figure 5. Reasons for task failure at 2400 devices across five repetitions

5.4. Application type and edge-cloud distribution

The relative ordering of conditions is broadly preserved across all four application types. Augmented reality and health tasks, carrying the highest delay sensitivity, show substantially higher failure rates than infotainment and compute-intensive tasks. The effect is most extreme for COPRAS-DQN under augmented reality, reaching 91.36% at 1800 devices, suggesting that COPRAS augmentation without Double Q-learning is particularly detrimental to delay-sensitive workloads.

The Figure 6 decomposes average service time into its edge and cloud components. Edge service time generally exceeds cloud service time, consistent with the limited capacity of the edge tier. COPRAS-DQN records zero cloud service time from 200

through 1800 devices, an extreme edge-concentration behaviour, whereas DDQN and MCDRL-HLB both use the cloud from the lightest load onward, giving the proposed method a more balanced edge-cloud distribution.

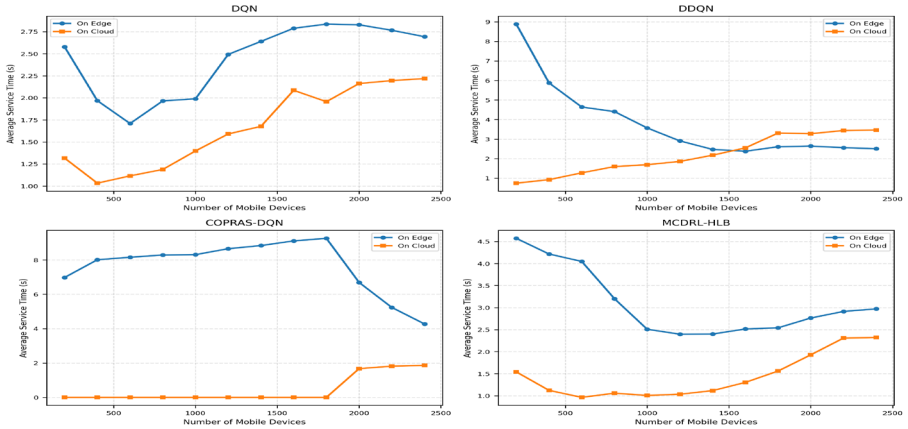


Figure 6. Average service time on edge against cloud servers by condition

5.5. Generalisation across device counts

Because all policies were trained exclusively at 2400 devices, behaviour at lighter loads reflects generalisation to an out-of-distribution regime. Repeating the sweep for MCDRL-HLB using the best checkpoint from each seed yields failure rates at 200 devices ranging from 0.36% (seed 42) to 50.66% (seed 777). Its generalisation outside the training distribution therefore remains highly sensitive to initialisation and should not be assumed robust without further training-time diversification.

6. CONCLUSION

This paper presented MCDRL-HLB, a hybrid workload scheduling framework for heterogeneous edge computing coupling the Complex Proportional Assessment method with a Double Deep Q-Network agent. Four hardware-derived criteria rank a three-tier edge deployment, and the resulting utility scores are embedded in the agent's state alongside dynamic load-based features, giving a structured prior on server capability that purely experience-driven schedulers lack.

The contribution was isolated through a controlled four-condition ablation under identical settings. Validated across three random seeds, MCDRL-HLB reduced mean task failure rate at the training scale from 22.67% to 18.42% relative to the DQN baseline while reducing cross-seed variance by a factor of 1.8. This combination of improved mean performance and run-to-run consistency, directly relevant to deployment where a favourable seed cannot be chosen in advance, is the central empirical finding. The advantage held across application types and corresponded to the highest edge utilisation and lowest capacity-related failure count among all conditions.

Several limitations should be acknowledged. DDQN and COPRAS-DQN were trained under a single seed. Generalisation outside the training distribution proved highly

seed-sensitive, with failure rate at 200 devices spanning 0.36% to 50.66%. The COPRAS scores are static, so servers within a tier are indistinguishable to the ranking itself. The evaluation compares conditions within the DQN and DDQN family only, and all results were obtained by simulation.

Future work will train across a distribution of device counts, extend statistical validation to all four conditions, incorporate hybrid static and real time criteria into the COPRAS ranking, and validate on a physical testbed.

REFERENCES

[1] Luo, Q., Hu, S., Li, C., Li, G., Shi, W. Resource scheduling in edge computing: a survey. *IEEE Communications Surveys & Tutorials*, vol.23, no.4, 2021, pp. 2131-2165.

[2] Zhou, G., Tian, W., Buyya, R., Xue, R., Song, L. Deep reinforcement learning-based methods for resource scheduling in cloud computing: a review and future directions. *Artificial Intelligence Review*, vol.57, no.5, 2024, pp. 124.

[3] Zheng, T., Wan, J., Zhang, J., Jiang, C. Deep reinforcement learning-based workload scheduling for edge computing. *Journal of Cloud Computing*, vol.11, no.1, 2022, pp. 3. DOI: 10.1186/s13677-021-00276-0.

[4] Tang, X., Tang, T., Shen, Z., Zheng, H., Ding, W. Double deep Q-network-based dynamic offloading decision-making for mobile edge computing with regular hexagonal deployment structure of servers. *Applied Soft Computing,* vol.169, 2025, pp. 112594.

[5] Li, M., Yang, Z. Double deep Q-network-based task offloading strategy in mobile edge computing. *Proceedings of the 4th International Conference on Internet of Things and Smart City (IoTSC 2024)*, Zhuhai, China, vol.13224, 2024, pp. 290-298. DOI: 10.1117/12.3034986.

[6] Zavadskas, E.K., Kaklauskas, A., Sarka, V. The new method of multicriteria complex proportional assessment of projects. *Technological and Economic Development of Economy*, vol.1, no.3, 1994, pp. 131-139.

[7] Malathy, N.K., Revathi, T. Entropy-based complex proportional assessment for efficient task scheduling in fog computing. *Transactions on Emerging Telecommunications Technologies*, vol.34, no.2, 2023, pp. e4690.

[8] Gad-Elrab, A.A.A., Alsharkawy, A.S., Embabi, M.E., Sobhi, A., Emara, F.A. Adaptive multi-criteria-based load balancing technique for resource allocation in fog-cloud environments. *arXiv preprint* arXiv:2402.01326, 2024. URL: https://arxiv.org/abs/2402.01326 (Visited on 10.09.2026).

[9] Pathak, P., Al-Masri, E. Using TOPSIS for enhancing service provisioning across fog environments. *IEEE Eurasia Conference on IoT, Communication and Engineering (ECICE)*, Yunlin, Taiwan, 2020, pp. 272-275.

[10] Al-Tarawneh, M.A.B., Kanj, H., Aly, W.H.F. An integrated MCDM framework for trust-aware and fair task offloading in heterogeneous multi-provider Edge-Fog-Cloud systems. *Results in Engineering*, vol.26, 2025, pp. 105228.

[11] Tang, Z., Jia, W., Zhou, X., Yang, W., You, Y. Representation and reinforcement learning for task scheduling in edge computing. *IEEE Transactions on Big Data*, vol.8, no.3, 2022, pp. 795-808. DOI: 10.1109/TBDATA.2020.2990558.

[12] Raju, M.R., Mothku, S.K. Delay energy aware task scheduling mechanism for fog-enabled IoT applications: a reinforcement learning approach. *Computer Networks*, vol.224, 2023, pp. 109603.

[13] Jangra, A., Mangla, N. An efficient load balancing framework for deploying resource scheduling in cloud-based communication in healthcare. *Measurement: Sensors*, vol.25, 2023, pp. 100584.

[14] Niu, L., Chen, X., Zhang, N., Zhu, Y., Yin, R., Wu, C., Cao, Y. Multiagent meta-reinforcement learning for optimized task scheduling in heterogeneous edge computing systems. *IEEE Internet of Things Journal*, vol.10, no.12, 2023, pp. 10519-10531. DOI: 10.1109/JIOT.2023.3241222.

[15] Gao, Z., Yang, L., Dai, Y. Large-scale computation offloading using a multi-agent reinforcement learning in heterogeneous multi-access edge computing. *IEEE Transactions on Mobile Computing*, vol.22, no.6, 2023, pp. 3425-3443. DOI: 10.1109/TMC.2022.3141080.

[16] Zhang, W., Ou, H. Reinforcement learning based multi objective task scheduling for energy efficient and cost-effective cloud edge computing. *Scientific Reports*, vol.15, 2025, pp. 41716.

[17] Sonmez, C., Ozgovde, A., Ersoy, C. EdgeCloudSim: an environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies*, vol.29, no.11, 2018, pp. e3493.

[18] Huang, C., Fan, B., Jiang, C. A task orchestration strategy in a cloud-edge environment based on intuitionistic fuzzy sets. *Mathematics*, vol.12, no.1, 2023, pp. 122.

[19] Bwalya, B., Zimba, A. An SDN approach to mitigating network management challenges in traditional networks. *International Journal on Information Technologies and Security*, vol.13, no.4, 2021, pp. 3-14.

[20] Sonmez, C., Ozgovde, A., Ersoy, C. Fuzzy workload orchestration for edge computing. *IEEE Transactions on Network and Service Managemen*t, vol.16, no.2, 2019, pp. 769-782.

Information about the authors:

Muhammad Rehan Rasheed – is a PhD scholar and Assistant Professor in department of Computer Engineering at SSUET, Karachi, Pakistan. His interests include artificial intelligence, fog computing and the Internet of Things.

Dr. Rabia Noor Enam – is a professor in department of Computer Engineering at SSUET, Karachi, Pakistan, holding a PhD in Applied Mathematics and Computer Engineering. Her research focuses on fog computing, IoT and wireless sensor networks.

Dr. Rukaiya Rukaiya – is an Assistant Professor in department of Computer Engineering at SSUET, Karachi, Pakistan, with a PhD from NUST, Islamabad (2022). Her research focuses on secure communication for mission critical networks.

Dr. Muhammad Naseem – is an Assistant Professor in department of software Engineering at SSUET, Karachi, Pakistan, with a PhD from Karachi University, Pakistan. His interests include artificial intelligence, cloud computing and machine learning.

Manuscript received on 16 September 2026