INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGIES
BULGARIAN ACADEMY OF SCIENCES
CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **26**, No 1
Sofia • 2026 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2026-0004

Energy and Delay Optimized Task Offloading Framework in Edge Computing Using DDPG with Dual Critic Attention and Uncertainty-Aware Experience Replay

Srinivas Byatarayanpura Venkataswamy[1], *Vinutha Krishnaiah*[1], *Veena S.*[2], *Manjula H. Nebagiri*[3]

[1]*Department of Computer Science and Engineering, BMS Institute of Technology and Management, Doddaballapura, Yelahanka, Avalahalli, Bengaluru, Karnataka, 560119 India*
[2]*Department of CSE(AIML) & CSE(CS), Ramaiah Institute of Technology, MSR Nagar, Bengaluru, Karnataka, 560054 India*
[3]*Department of Computer Science and Engineering, Shridevi Institute of Engineering Technology, Lingrajpur, Tumkur, 572106 India*
E-mails: srinivas.bv@bmsit.in vinuthak_ise2014@bmsit.in veenas@msrit.edu manjula.n@shrideviengineering.org

***Abstract*:** *The increase in Internet of Things* (*IoT*) *devices has increased demand for effective and reliable task offloading strategies in edge-computing environments. These systems struggle to balance limited computational resources, varying network conditions, and complex dependencies between subtasks. To address these challenges, this article developed a task offloading framework using the Deep Deterministic Policy Gradient* (*DDPG*) *algorithm with Uncertainty-aware Prioritized Experience Replay* (*UPER*) *and Dual Critic Attention* (*DCA*) *mechanism. The task offloading issue is modeled as a Markov Decision Process* (*MDP*), *in which agents learn optimal policies to minimize delay and energy consumption while ensuring high task success rates. The UPER module enhances learning efficacy by sampling transitions with high epistemic uncertainty and temporal difference errors, allowing the DCA component to dynamically balance energy and latency objectives using dual-critic networks. The simulation results show that the proposed model outperforms existing algorithms in terms of offload success rate and minimization of average delay and energy consumption.*

***Keywords*:** *Deep Deterministic Policy Gradient* (*DDPG*), *Dual Critic Attention* (*DCA*), *Markov Decision Process* (*MDP*), *Task success rates, and Uncertainty-aware Prioritized Experience Replay* (*UPER*).

1. Introduction

The demand for high-quality services has increased with the increasing number of applications. As a new computing method for the Internet of Things (IoT), edge computing has evolved into a highly virtualized platform that enables the storage, computing, and networking services across target devices and conventional

information centers of cloud [1-3]. Edge nodes, as a critical component of edge computing infrastructure, consist of routers, switches, and embedded servers [4]. Owing to the continuous growth of the desired devices on the Internet, smartphones and other desired systems are primarily used [5]. In edge computing, the demand for high-quality mobile services has increased the volume and diversity of data generated by IoT devices [6]. Additionally, because of the growing number of integrated devices of edge nodes, a lack of resources results in high costs and load imbalances among edge nodes [7]. Hence, a complete and effective task offloading approach is essential to the development of edge-computing networks and superior performance [8]. In conventional computing paradigms, tasks that are sensitive to delays have been identified as computations on cloud platforms [9, 10]. This leads to latency issues caused by data transmission among Smart End Devices (SEDs) and distant cloud servers [11]. However, with advancements in edge computing, an effective paradigm that strategically positions computational resources near the network edge has emerged [12]. This data transmission in computing architecture gains significance due to the rapid proliferation of SEDs, which are equipped to produce requests and possess feasible computational abilities supported by advancements in hardware technologies [13-15]. Centralized cloud platforms increase the abundance of resources; however, their geographical remoteness from end-users leads to prolonged transmission times, rendering them unsuitable for low-latency tasks. Conversely, edge computing improves transmission delays by deploying resources that are near users, particularly at the network edge. However, edge computing lacks extensive computational ability in centralized cloud environments [16, 17].

Fig. 1 presents an overview of the system architecture used in this article. System architecture includes three layers: the cloud, edge, and user layers. Numerous edge nodes in the edge layer have various limitations. Connections and positions of edge nodes are fixed through third-party service providers or cloud datacenters. For every user, a connection in some spaces is indicated by a dotted circle in Fig. 1. In these spaces, the user offloads tasks to their respective edge nodes or has them processed by a local device. This article examined application-driven tasks developed through numerous subtasks with high dependencies and subtasks belonging to a single application processed on various devices. In this system architecture, the provisioning procedure of an application involves identifying the positions of subtasks. For instance, utilize A_i represents i-th application, which includes three subtasks: $a_1 \rightarrow a_2 \rightarrow a_3$. The extreme solution is an offloading strategy that reduces process delay for A_i, which processes all the subtasks in the edge nodes. The user broadcasts subtasks to edge nodes for processing via a wireless channel, and outcomes are returned after all subtasks are finished. Still, if the size of a subtask is huge, total energy consumption is correspondingly large because of dependencies among precursor and successor, which minimizes the quality of the user service. This article focuses on a task offloading issue on edge computing by addressing the high interdependencies among subtasks. The most significant aspect of this issue is to define offloading positions of subtasks to users to jointly optimize the whole delay and energy

consumption produced through an application while guaranteeing the quality of services to users.

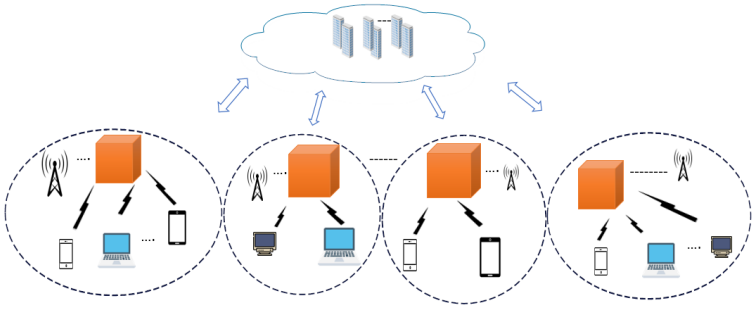

Fig. 1. System architecture of edge computing

1.1. Problem statement

In edge-computing environments, the challenges for effective offloading computation are the energy consumption and delay. These tasks include multiple interdependent subtasks that require intelligence distribution over constrained edge resources. Existing offloading strategies fail to consider the dependencies of subtasks and suffer from high energy consumption and latency because of suboptimal decision-making. In addition, the dynamic nature of edge environments, resource variability, and systems fail to allocate optimal tasks.

1.2. Objective

The main objective of this article is to design an intelligent, energy-efficient, and delay-aware task offloading strategy for edge computing systems by developing a Markov Decision Process and resolving it using a Deep Deterministic Policy Gradient (DDPG-based) reinforcement learning model. To improve the learning efficacy and adaptability of a model in dynamic environments, the proposed model combines Uncertainty-aware Prioritized Experience Replay (UPER) and a Dual Critic Attention (DCA) mechanism. This jointly optimizes energy consumption and execution delay when increasing the offloading success rate, ensuring scalability across varying workloads and device capabilities.

1.3. Contributions

- The main contribution of this manuscript is described below
- The Markov Decision Process (MDP) was developed to model the complex task offloading issue in heterogeneous edge computing environments used for subtask dependencies, delay sensitivity, and energy constraints.
- The Uncertainty-aware Prioritized Experience Replay (UPER) mechanism was incorporated to guide the learning process by focusing on transitions with high temporal-difference errors and epistemic uncertainty by enhancing convergence speed and policy robustness.

- A novel DCA architecture is proposed, in which two separate critic networks estimate latency and energy-based Q-values, dynamically fused by attention weights to balance the objective in the actor updates.

This research manuscript is arranged as follows. Section 2 analyses the existing algorithm. Section 3 provides the system model and details of the proposed model. Section 4 presents the results and compares the existing algorithms with the proposed model. The conclusion of this research paper is given in Section 5.

2. Literature review

Zhai et al. [18] presented a joint task offloading and executing resource allocation for task-based MEC systems. To address this issue, a method is developed for reducing the weighted sum of long-term task computation delay and energy consumption on IoT devices, with a high tolerable task delay considered as a critical constraint. The issue was NP-hard, and the joint task offloading and computing resource allocation model depended on a Deep Q-Network (DQN) for multi-user and multiple tasks (JTOCRA-DQN). The conventional DQN approach includes multi-user, multi-dependent task joint task offloading and executing a resource allocation approach phase before learning to minimize action space.

Huang, Wu and Dong [19] used a multi-objective optimization method appropriate for the executional offloading on dynamic heterogeneous VEC networks. A dynamic multi-objective executional offloading issue is called the Multi-Objective Markov Decision Process (MOMDP). Here, the multi-objective reinforcement learning approach EMOTO minimizes average task computation delay and vehicle energy consumption and increases the revenue of service providers. The preference priority sampling model was developed, and a multi-augmented environment estimator was incorporated for learning environmental methods to multi-objective optimization for learning steadily, leading to highly dynamic changes in the VEC environment to efficiently realize joint optimization on several objectives and enhance the accuracy of decision making and model efficacy.

Zhu et al. [20] introduced the distributed task offloading model EE-A2C, which uses the benefits of an actor-critic approach to improve energy efficacy in edge cloud environments. This model facilitates distributed interactions between multiple agents and edge environments with communication queues, which reduces the average energy consumption and delay for AIGC users. Next, to obtain adaptive and effective task offloading decisions, the developed complex reward-sharing module depends on delay and energy consumption. Finally, included Long Short-Term Memory (LSTM) to improve the capability of a model to capture essential data on energy consumption by enabling robust decision-making.

Lin et al. [21] developed an Offline-to-Online DRL named as O2O-DRL, which utilizes heuristic task logs for the initialization of the DRL method offline. However, offline and online have various distributions by utilizing models to fine-tune the ruin policy learned offline. To avoid this issue, used an on-policy DRL to fine-tune the model and protect the value from evaluation.

Alsadie [22] implemented a Hybrid Task Offloading (HybridTO) approach incorporating a Grey Wolf Optimizer (GWO) and Particle Swarm Optimization (PSO). This model was developed to optimize energy consumption and complete delay constraints in EC environments by considering factors such as capacity, proximity, and latency constraints. By incorporating collaborative abilities into EC servers, HybridTO provides comprehensive advantages for task offloading.

Alhartomi et al. [23] employed RE Predictions with the Deep Reinforcement Learning (named the REP-DRL approach). REP-DRL utilizes an actor-critic model to identify a better algorithm for predicting RE and optimal offloading decisions. The algorithm enhances IoT device processing and expands the system state to offload experiences over time. To store excessive energy in abundance and use it for high demand, a service offloading process was developed based on the predicted RE amount to identify a better service offloading model and enhance energy sustainability in IoT. By defining a highly effective service offloading algorithm based on the predicted RE, this model enhances the energy sustainability of an IoT ecosystem.

3. System model

In this article, the heterogeneous edge networks designed as connected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ depicted a group of edge nodes, $\mathcal{V} = \{i | 1 \leq i \leq N\}$ and the $\mathcal{E}$ represents a group of links integrating various edge nodes, $\mathcal{E} = \{e_{i,j} | i, j \in \mathcal{V}\}$. The time is slotted with a consistent slot duration, and the index of a time the slot is, represented as t, where $t \in \{1, 2, \ldots, T\}$. Every edge node $i \in \mathcal{V}$ receives computation-intensive and delay-sensitive information from the end users in every time slot. Consider that the task arriving at edge node i in time slot t follows Bernoulli distribution with a probability λ_i. $K_i(t)$ represents a task arriving at the edge node i in time slot t described by $\{s_i(t),\ c_i(t),\ d_i(t)\}$, and: $s_i(t)$ represents size of the input data; and $c_i(t) = \delta_i(t) \times s_i(t)$ represents number of CPU cycles needed to complete a process, where $\delta_i(t)$ is the computation intensity that represents the required CP cycles per b_i; $d_i(t)$ represents the task deadline. Every edge node i processes the arrived task locally or offloads it to a different edge node. Every edge node i handles the execution queue with a task scheduled according to the First-In-First-Out (FIFO) order [24, 25]. Recent tasks should wait in an execution queue in which other tasks occupy computation resources. Here, F_i is the computational capacity of edge node i. Hence, the execution time of task $K_i(t)$ processed through edge node j is measured using the equation

$$T_{ij}^{c}(t) = \frac{c_i(t)}{F_j}, \tag{1}$$

where, r_{ij} is the transmission rate between edge nodes i and j. Hence, the transmission time to send the task $K_i(t)$ from the edge node i to j is calculated using the equation

$$T_{ij}^{t}(t) = \frac{s_i(t)}{r_{ij}}. \tag{2}$$

Every edge node handles $N-1$ received queues to receive tasks from different edge nodes and the $N-1$ sent queues to transmit tasks to different edge nodes. The tasks are transmitted according to FIFO order. After the task, $K_i(t)$ is received through the edge node j, which waits in the execution queue to process and no longer forwards a task to different edge nodes. This is because in a fully connected network, the output of a multiple forwarding task is completed through single forwarding, minimizing the transmission time. The reliability of a system is based on the possibility of failure of every processing unit during processing tasks, such as edge nodes and transmission links. Failure of these elements includes software and hardware. Any failure is normalized to a Poisson distribution driven by the failure rate. In this article, the execution time taken through the edge node j to finish the task $K_i(t)$ without failure is represented as $T_{ij}^c(t)$. In the time interval $\left[0, T_{ij}^c(t)\right]$, failure occurred and is a Poisson process with failure rate parameter α_j. The total number of failures that occurred in the edge node j while processing the task $K_i(t)$ is represented as $N_j\left(T_{ij}^c(t)\right)$. Hence, the probability Pr of $N_j\left(T_{ij}^c(t)\right) = k$ in a time interval $\left[0, T_{ij}^c(t)\right]$ is calculated using the equation

$$\Pr\left\{N_j\left(T_{ij}^c(t)\right) = k\right\} = \frac{\alpha_j T_{ij}^c(t)}{k!} e^{-\alpha_j T_{ij}^c(t)},\ k \geq 0, \tag{3}$$

where $k = 0$, whether task $K_i(t)$ is processed in a time interval $\left[0,\ T_{ij}^c(t)\right]$. Hence, the task reliability $K_i(t)$ processed through the edge node j is calculated using the equation

$$R_{ij}^c(t) = e^{-T_{ij}^c(t)\alpha_j}. \tag{4}$$

Consider failure as a one-off incident that does not impact subsequent tasks. Like task reliability $K_i(t)$ while transmitted from the edge node i to j is measured by the equation

$$R_{ij}^t(t) = e^{-T_{ij}^t(t)\beta_{ij}}. \tag{5}$$

In this equation, β_{ij} represents the transmission link failure rate among edge nodes i and j.

3.1. Proposed model

This section explains the task offloading issue by designing an optimization challenge as an MDP. The actions, states, and reward functions are clearly defined in the MDP model. Subsequently, the proposed deep reinforcement learning depended on DDGP to address previously identified optimization issues. In this framework, Edge Computing is structured as a reinforcement learning environment, with every Edge User acting as an agent. These agents interact with the environment, learn by observing the state of their corresponding edge-computing environments, and make decisions to increase the long-term rewards. Fig. 2 represents the overall process of task-offloading in edge computing using the proposed model.

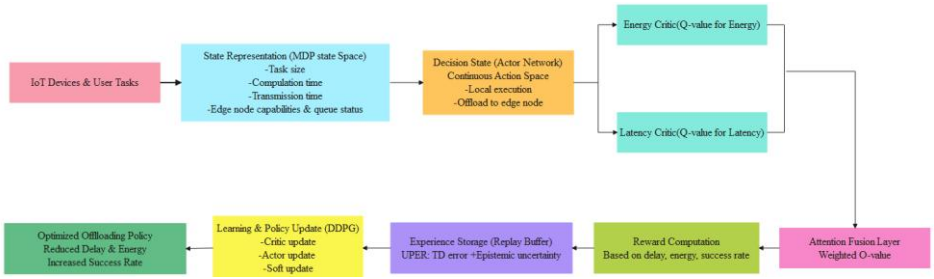

Fig. 2. Overall process of task-offloading in edge computing using the proposed model

3.1.1. Markov Decision Process (MDP)

The MDP is determined as four sets $M = \{N, S, A, R\}$, in which N =1, 2, …, I, is the set of all agents, $S = S_1, S_2, \dots, S_I$ is the state space of every agent, $A = a_1, a_2, \dots, a_I$ contains a set of actions available for every agent, and R represents an aggregate of rewards that agents gain depending on their actions.

3.1.1.1. State space

At the time point t, agent i identifies the status of the user device, which includes task-relevant data, edge-node characteristics, and queue status. The mathematical expression for describing the status is given by the equation

(6) $$s(t) = \left\{\Omega_i^t, Q_i^t, \mathrm{EC}_{\mathrm{list}}\right\}.$$

At time t, Ω_i^t is the task size, the $Q_i^t = \left\{\mathrm{CQ}_i^t,\ \mathrm{TQ}_i^t\right\}$ represents the estimated time required to compute and the time required to transmit the task. The Edge Computing ($\mathrm{EC}_{\mathrm{list}}$) status data of edge nodes, including their computing abilities and the number of tasks currently waiting and processing on an Edge Server (ES).

3.1.1.2. Action space

The observed agents present an environmental state and make decisions depending on the observation to optimize the task offloading process. This intelligent decision-making process is essential for differentiating the primary characteristics of various user-device agents. The mathematical expression for defining the decision is given in the equation

(7) $$a(t) = \{x_{i,t}^m | x_{i,t}^m \in \{0, 1, 2, \dots, M\},$$

where $x_{i,t}^m$ is an offloading decision for the task Ω_i^t made by the user i in a present time slot t. When $x_{i,t}^m = 0$ the task is processed using a local user device. Otherwise, the task is offloaded to an edge server ES_m.

3.1.1.3. Reward function

To obtain an optimum task offloading that reduces energy consumption and delay while increasing the success rate of task offloading, a reward function is considered that considers task offloading delay, energy consumption, and the number of tasks that are successfully offloaded. Initially, define r_i^{d} and r_i^{e} as sets of delay and energy consumption information for the user i before time t. Equation bellow provides the mathematical expression for this process:

(8) $$\begin{cases} r_i^{\mathrm{d}} = [D_i(1), D_i(2), \dots, D_i(t)], \\ r_i^{\mathrm{e}} = [E_i(1), E_i(2), \dots, E_i(t)], \end{cases}$$

where $D_i(t)$ and $E_i(t)$ are the delay and energy consumption of the user device i at the time point t. The mean values of delay and energy consumption are represented as φ_{d} (d – delay) and φ_{e} (e – energy), (f – ???) and their mathematical expressions are given by the equations

(9) $$\begin{cases} \varphi_{\mathrm{d}} = \frac{1}{t}\sum_{k=0}^{t} r_i^{\mathrm{d}}[k], \\ \varphi_{\mathrm{e}} = \frac{1}{t}\sum_{k=0}^{t} r_i^{\mathrm{e}}[k]. \end{cases}$$

Here, the weight penalty value of the energy consumption, delay, and success rate of task offloading, where Unf is the total number of tasks that failed to offload, ETC is the total number of tasks that attempted to offload, and $D_{\max}$ is the maximum delay time. The $[-1]$ is the last component in a collection, a data value that represents the present user state and respective point in time, and its mathematical expression is given by the equation

(10) $$\begin{cases} \omega_{\mathrm{f}} = 1 - \frac{\mathrm{Unf}}{\mathrm{ETC}}, \\ \omega_{\mathrm{e}} = -\log_2\left(\frac{r_i^{\mathrm{e}}[-1]}{\varphi_{\mathrm{e}}} + l\mathrm{e} - 10\right), \\ \omega_{\mathrm{d}} = 1 - \frac{\varphi_{\mathrm{d}}}{D_{\max}}. \end{cases}$$

Depending on the task offloading completion, delay, and energy consumption, the next equation correctly measures r:

(11) $$r(t) = \alpha\omega_{\mathrm{f}} + \beta\omega_{\mathrm{e}} + \eta\omega_{\mathrm{d}}.$$

Among these parameters, α, β and η are the weight coefficients of task completion, energy consumption, and delay, respectively.

3.1.2. Deep Deterministic Policy Gradient (DDGP)

The DQN approach offers a solution to high-dimensional state-space issues, but it cannot address the issue of continuous action spaces. DDGP learns policies on continuous action spaces. This is a model-free off-policy actor-critic approach that utilises a DNN. Rather than utilizing value-based algorithms, DDPG uses a policy gradient approach that directly processes the policy to identify the optimal policy. In RL, assumptions or constraints are placed on the environments the agent interacts with, and these constraints occur in the action spaces. In general, agents act with discrete actions, although sometimes they handle continuous actions as well. The highest function is not feasible for calculating on continuous action spaces, although utilizing a policy gradient approach, such as DDGP, directly learns the optimum policy for an agent. DDPG falls under the subclass of an actor–critic approach. Here, the process begins with the general RL framework, where the agent interacts with the environment by performing actions and receiving corresponding rewards. The objective of an agent is to consider optimum actions and increase rewards. In an actor-critic approach, an agent contains two units: the actor and critic. The actor acts in the environment, and critics offer feedback to the actor depending on the action it takes. When a critic receives a reward from an agent, it transmits that to the feedback signal and sends that to the actor. The DDPG

framework incorporates two Deep Neural Networks: The actor network and $\mu(s|\theta^u)$ which presents policy function and a critic network $Q(s,a|\theta^Q)$ which estimates the Q-value function. These actor and critic networks also include the target network. The network of an actor target is determined as μ' which includes the parameter $\theta^{\mu'}$. The critical target network is determined as Q' to the parameter $\theta^{Q'}$. Its mathematical expression is given by the next equation:

(12) $Q(s,a|\theta^Q)$ is updated as

$$L(\theta^Q) = E_{\mu^l}\left[\left(y_i - Q(s_i,a_i|\theta^Q)\right)^2\right] \text{ where } y_i = r_i + \gamma Q(s_{i+1},\mu(s_{i+1})|\theta^Q.$$

DDPG is a model-free off-policy actor-critic that utilizes the approximations of a deep function. This method is particularly efficient in continuous action spaces. The mathematical expression for the policy update is given by the equation

(13) $$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta),$$

where α represents the learning rate, $J(\theta)$ represents an objective function, and its mathematical expression is given by the equation

(14) $$J(\theta) = E_{s\sim D}\left[Q\left(s,\pi_\theta(s)\right)\right].$$

The mathematical expression for the *Q*-function update is given by the equation

(15) $$\phi \leftarrow \phi + \beta\nabla_\emptyset(Q_\emptyset(s,a) - y)^2,$$

where β represents the learning rate and the target value y is given by the equation)

(16) $$y = r + \gamma Q_{\emptyset'}\left(s',\ \pi_{\theta'}(s')\right),$$

where γ represents the discount factor and the $\emptyset'$, θ' are parameters of the target networks, r represents reward, and the s' represents the next state.

3.1.3. Uncertainty-aware Prioritized Experience Replay (UPER) – Dual Critic Attention (DCA)

Traditional DDPG-based offloading models struggle to balance multiple objectives, such as energy and latency, which degrades effective learning in large action spaces. This also minimises convergence because of the uniform sampling and static critical design. To address these challenges, UPER-DCA was proposed, which improves the traditional DDPG framework process. To improve the learning efficacy and adaptability of a DDPG framework in dynamic edge environments, an integrated UPER-DCA mechanism that incorporates uncertainty-aware prioritized experience replay with a dual-critic attention network is proposed. In a UPER, every transition is estimated based on the Temporal Difference (TD) error of epistemic uncertainty of a critic's prediction acquired through dropout-based variance estimation. This ensures a transition from agent to sample, which is uncertain and impactful, by enhancing convergence speed. Moreover, introduced a dual-critic architecture, where one critic was optimized for latency and the other for energy efficacy. The attention fusion layer adaptively weighs its contribution to actor training, resulting in a robust and flexible policy that balances the multiple objectives. This mechanism ensures that the model makes stable and informed task-offloading decisions, particularly under highly dynamic user and resource conditions.

3.1.3.1. Uncertainty-aware prioritized experience replay

RL agents enhance their performance by learning from previous experience and storing it as sets. $M = \{N, S, A, R\}$ in a replay buffer. In traditional DDPG, samples are uniformly drawn from an ineffective buffer, and several experiences are redundant or inappropriate. Prioritized Experience Replay (PER) enhances learning through transitions of sampling with high Temporal Difference (TD) error, which considers high-error samples to be informative. However, TD error does not capture the uncertainty that is essential in edge computing, where the environment and system load change rapidly. The UPER replaces epistemic uncertainty using a Monte Carlo Dropout. Based on the priority score, the mathematical expression is given by the equation

(17) $$P(i) = \lambda \times \text{TD} - \text{error}(i) + (1 - \lambda) \times \text{Uncertainty}\ (i),$$

where $\lambda \in [0, 1]$ balances the uncertainty and error. This ensures that the model focuses on uncertain transitions that increase stability and learning speed.

3.1.3.2. Dual Critic Attention (DCA)

In the traditional DDPG, a single critic is used to evaluate the action-value function. $Q(s, a)$, which learns expected return under a fixed policy. In multi-objective decision-making, such as reducing energy consumption and latency, a single critic struggles to compete for offloading, particularly when trade-offs are nonlinear. The DCA introduced two individual criticisms, as described below.

- $Q_1(s, a)$ estimate expected return from minimisation of latency,
- $Q_2(s, a)$ estimates the expected return for energy efficacy.

The attention mechanism is used to dynamically fuse them based on the present state, and its mathematical expression is given by the equation

(18) $$Q(s, a) = \alpha(s) \times Q_1(s, a) + \big(1 - \alpha(s)\big) \times Q_2(s, a),$$

where α represents the learned through attention weights from the state embedding. It learns a much-nuanced *Q*-function that adopts system priorities.

Algorithm 1. DDPG with UPER-DCA.

Initialize Actor network $\mu(s|\theta\mu)$ and the critic network $Q(s, a|\theta Q)$ with random weights

Initialize target network $\theta\mu, \theta Q$

Initialize reply buffer $D \leftarrow \emptyset$

Initialize UPER-DCA priority buffer $P \leftarrow \emptyset$

For episode 1 to MaxEpisodes do:

Initialize system state s_0 with node status, task features, and MEC capacity

For t = 1 to T do:

$a_t \leftarrow \mu(s_t|\theta\mu)$

Observe reward r_t, next state s_{t+1} and uncertainty score u_t

Compute confidence score $C_t \leftarrow 1 - u_t$

Compute experience priority $p_t \leftarrow C_t \times |r_t|$

Store (s_t, a_t, r_t, s_{t+1}) into buffer D with priority p_t into P

Sample minibatch $B \leftarrow \{(s_i, a_i, r_i, s_{i+1})\}$ from D using priorities P

For every sample in B:

Calculate target Q-value:

$$y_i \leftarrow r_i + \gamma \times Q'(s_{i+1}, \mu'(s_{i+1}|\theta\mu')|\theta Q'$$

Update the critic by reducing the loss.

$$L \leftarrow (1/N) \times \sum (Q(s_i, a_i|\theta Q) - y_i)^2$$

Update the actor by using policy gradient.

$$\nabla\theta\mu\, J \approx (1/N) \times \sum \nabla a\, Q(s, a|\theta Q)|a = \mu(s) \times \nabla\theta\mu\ \ \mu(s|\theta\mu)$$

Soft update of target networks

$$\theta\mu' \leftarrow \tau\theta\mu + (1-\tau)\theta\mu'$$

$$\theta Q' \leftarrow \tau\theta Q + (1-\tau)\theta Q'$$

End for

End for

Return optimized policy $\mu(s|\theta\mu)$ for task offloading

4. Experimental results

In this manuscript, a grid search over the primary hyperparameters is conducted to ensure stable and effective learning in DDPG using the UPER-DCA-based offloading model. In this manuscript, identified that an actor learning rate of 10-2 allows the policy network to converge rapidly without oscillation, whereas the critical learning rate of 10 power –2 suffered from a balance between the value estimate and update speed. A discount factor of 0.9 provides an optimum trade-off between delay and long-term performance, when lower and higher values lead to variance amplification.

Table 1. Hyperparameters of the proposed model

Parameter	Tested values	Best value
Actor learning rate	10 power –2, 10 power –3, 10 power –4	10 power –2
Critic learning rate	10 power –3, 10 power –4	10 power –3
Discount factor (γ)	0.3, 0.6, 0.9, 0.95, 0.99	0.9
Batch size	32, 64, 128	32

A batch size of 32 yielded fast and stable training in a dynamic edge environment, and outperformed large minibatches in terms of convergence speed and reward consistency. Table 1 lists the hyperparameters of a proposed model.

The performance comparison of the different offloading approaches in terms of the average delay and energy consumption determined the superiority of the proposed model, as presented in Table 2. Conventional rule-based strategies, such as random, round robin, and greedy, provide the highest delay and energy values because of their lack of state awareness and optimization. Among reinforcement learning-based algorithms, DQN and its variants, such as Double DQN, Duelling DQN, and D3QN, show enhancements by addressing value overestimation and stability, and the actor–critic algorithm provides fewer gains by policy-based learning. The proposed model significantly outperformed all existing algorithms and obtained an average delay of 1.40 s and energy consumption of 0.81 J. This is owing to the effective task offloading policy learned by the DDPG model, which is improved by the UPER-DCA mechanism. By efficiently balancing latency and energy objectives and focusing on learning informative experiences, the proposed

model ensures optimum and consistent decision-making under dynamic edge computing conditions.

Table 2. Performance comparison of different offloading approaches in terms of average delay and energy consumption

Algorithm	Average delay (s)	Energy consumption (J)
Random	2.65	3.45
Round Robin	2.61	3.38
Greedy	2.53	3.27
DQN	2.30	2.42
Double DQN	2.21	2.31
Dueling DQN	2.18	2.27
D3QN	2.15	2.20
Actor-Critic	2.14	2.19
Proposed DDPG+UPER-DCA model	1.40	0.81

The connection between device computational ability and system performance is analyzed through varying CPU frequency from 2.0 GHz to 4.5 GHz in Table 3. The results showed that increasing CPU ability significantly reduced average task delay and energy consumption. This enhancement contributes to the fast local processing abilities of high-frequency CPUs, which reduces queuing and computation time. Moreover, DDPG with the UPER-DCA model learns to dynamically exploit devices capable of local computation, thereby minimizing reliance on energy consumption offloading. These results confirm that the proposed model efficiently adapts to device heterogeneity and uses more computational resources to optimize task offloading decisions.

Table 3. Performance of the proposed model by varying CPU frequency

CPU (GHz)	Average delay (s)	Energy consumption (J)
2.0	1.74	1.14
3.0	1.61	0.97
4.5	1.32	0.74

A comparison of task drops and offload success rates across different approaches shows the reliability and robustness of the proposed offloading strategy, as shown in Table 4. The proposed model obtained the best outcomes, with a task drop rate and offload success, compared with traditional algorithms. Integrating the UPER and DCA enables a model to focus on high uncertainty and transitions when adaptively balancing multiple objectives. Consequently, it consistently makes informed and successful offloading decisions in dynamic edge computing.

Table 4. Comparison of the task drop rate and offload success rate of the proposed model

Algorithm	Task drop rate (%)	Offload success rate (%)
Random	20.4	79.6
Round Robin	20.1	79.9
Greedy	18.9	81.1
DQN	15.2	84.8
Double DQN	14.1	85.9
Dueling DQN	13.9	86.1
D3QN	13.5	86.5
Actor-Critic	13.4	86.9
Proposed DDPG+UPER-DCA model	10.6	89.4

A statistical analysis of the proposed model is presented in Table 5, which evaluates the performance of the different task offloading algorithms. The descriptive statistics show that the proposed model obtained a lower average delay and energy consumption and demonstrated high efficacy. The scalability analysis shows a linear enhancement in performance in maximizing the CPU frequency with statistically significant energy minimization, indicating the adaptability of the model.

Table 5. Statistical analysis of the proposed model

Metrics	Mean	Standard deviation
Average delay (s)	2.24	0.37
Energy consumption (J)	2.48	0.82

Table 6 presents the ablation study, which systematically evaluates the individual and combined contributions of the UPER and DCA models to the overall performance of the proposed model. Four variants – DDPG only, DDPG + UPER, DDPG + DCA, and the proposed DDPG + UPER + DCA – are compared across four primary measures: average delay, energy consumption, offload success rate, and task drop rate. The baseline DDPG model showed lower performance with high delay, energy consumption, and a lower offload success rate. When UPER is included, the capability of the model to learn from high uncertainty and informative experience maximizes, causing a reduction in delay and energy when the offload success rate also increases. The incorporation of DCA improves the performance by managing the trade-off between energy and latency, resulting in delay and energy minimisation. The completely proposed model outperformed all other combinations, obtaining less delay, minimum energy consumption, a high offload success rate, and a lower task drop rate. These results show that every component independently enhances model efficacy, and its combination offers advantages that cause robust, scalable, and intelligent task offloading suitable for dynamic edge environments.

Table 6. Ablation study to evaluate the impact of UPER and DCA modules on task offloading performance in edge computing

Variant models	Average delay (s)	Energy consumption (J)	Offload success rate (%)	Task drop rate (%)
DDPG only	2.02	1.96	84.2	15.8
DDPG + UPER	1.76	1.39	86.5	13.5
DDPG + DCA	1.61	1.11	87.9	12.1
DDPG + UPER + DCA	1.40	0.81	89.4	10.6

4.1. Comparative analysis

To ensure a fair comparison, the proposed model is simulated using the same experimental parameters as those in the JTOCRA-DQN [18] and EMOTO [19] models. The simulation parameters for JTOCRA-DQN [18] are the number of base stations (four), the area of 500×500 m, coverage radius (150 m), and $\lambda = 1$. The simulation parameters for EMOTO [19] are a road section length of 4200 m, number of base stations of 30 m, vehicle speed of [5, 30] m/s, and transmission power of 4 W. The results in Table 7 compare the execution times and total costs for increasing numbers of IoT devices. The proposed model demonstrated consistent superiority over the JTOCRA-DQN [18] by reducing the execution time

and total cost. This performance is obtained by learning effective task allocation using DDPG with UPER-DCA, which allows the model to adaptively learn cost-effective offloading strategies under high-load conditions.

Table 8 presents a comparison with EMOTO [19] in terms of average task delay and energy consumption by varying the number of tasks. EMOTO [19] performed well in terms of energy efficacy as the number of tasks increased. The proposed model maintains a balanced delay and energy performance. These enhancements demonstrate the superior adaptability of the proposed model by varying the workloads based on its ability to capture short- and long-term dependencies in task offloading when making energy-aware decisions in edge environments. These comparisons demonstrate that the proposed model achieves better scalability and resource efficacy, determines generalization across various system settings, and provides a reliable solution for dynamic task offloading in edge computing.

Table 7. Comparison of the proposed model with JTOCRA-DQN [18]

Method	Number of IoT devices	Execution time (ms)	Total cost
JTOCRA-DQN [18]	10	99.8706	72.8044
	15	160.9159	108.0406
	20	213.3423	140.242
	25	247.3563	179.8502
	30	298.3993	227.1728
	35	344.3903	296.802
	40	397.8001	336.5364
Proposed DDPG+UPER-DCA	10	87.6509	65.5982
	15	98.7612	94.2190
	20	153.8329	112.7236
	25	207.7410	131.8961
	30	261.3095	201.9570
	35	309.8361	250.1057
	40	349.3793	305.0715

Table 8. Comparison of the proposed model with EMOTO [19]

Method	Number of tasks	Average delay	Average energy
EMOTO [19]	25	1.918	1.741
	50	1.645	1.496
	80	1.658	1.575
	110	1.614	1.462
	140	0.973	2.658
Proposed DDPG+UPER-DCA	25	1.693	1.529
	50	1.409	1.382
	80	1.529	1.661
	110	1.504	1.590
	140	1.429	1.498

4.2. Discussion

The proposed DDPG-based task offloading model, enhanced with UPER and DCA, addresses essential challenges such as energy consumption, latency, and subtask dependency handling in edge computing environments. The integration of UPER

effectively enhances learning stability by prioritizing experience transitions with higher epistemic uncertainty and TD errors, ensuring that the agent focuses on uncertain and informative experiences. The DCA mechanism dynamically balances the energy and delay trade-offs using a dual critic network that adaptively weighs the latency and energy values estimated in the actor updates. This results in a robust policy that generalises well by varying system loads and device abilities. The experimental results validated the efficacy of the model, which consistently outperformed baseline algorithms such as DQN, Actor-Critic, and conventional rule-based strategies. Specifically, it obtains a lower average delay and energy consumption with a higher offloading success rate and reduced task drop rate. Statistical analysis shows that these enhancements are statistically robust. Additionally, scalability analysis determined that the model adapts well to maximizing CPU frequencies and IoT device counts and maintains optimum performance under high-load scenarios. The DDPG+UPER-DCA model provides a scalable, energy-aware, and latency-optimised solution for intelligent task offloading in edge networks.

5. Conclusion

In this manuscript, a DDPG-based task offloading model is improved using UPER and DCA to address the challenges of latency, energy consumption, and task dependencies in edge computing. The proposed model demonstrated superior performance in terms of offload success rate, delay minimisation, and energy efficacy when compared with existing baseline and deep reinforcement learning models. The proposed model exhibits robustness and scalability by efficiently balancing multiple objectives and adapting to dynamic edge environments. The incorporation of UPER effectively enhanced the learning efficacy through a guiding agent to focus on transitions that are uncertain and informative, increasing convergence and avoiding poor local optima. The DCA mechanism allows the model to learn the adaptive tradeoff between energy consumption and latency, thereby enabling robust and flexible offloading decisions under various system conditions. The proposed model handles different workload intensities and determines strong generalization abilities. In the future, the model will be extended to handle adversarial network conditions and incorporate multi-agent collaboration for distributed offloading in edge environments.

References

1. Lim, D., I. Joe. A DRL-Based Task Offloading Scheme for Server Decision-Making in Multi-Access Edge Computing. – Electronics, Vol. **12**, 2023, No 18, 3882.
2. Wu, Z., Z. Jia, X. Pang, S. Zhao. Deep Reinforcement Learning-Based Task Offloading and Load Balancing for Vehicular Edge Computing. – Electronics, Vol. **13**, 2024, No 8, 1511.
3. Tütüncüoğlu, F. G. Dán. Joint Resource Management and Pricing for Task Offloading in Serverless Edge Computing. – IEEE Transactions on Mobile Computing, Vol. **23,** 2023, No 6, pp. 7438-7452.

4. Hao, H., C. Xu, W. Zhang, S. Yang, G. M. Muntean. Joint Task Offloading, Resource Allocation, and Trajectory Design for Multi-Uav Cooperative Edge Computing with Task Priority. – IEEE Transactions on Mobile Computing, Vol. **23,** 2024, No 9, pp. 8649-8663.
5. Li, N., L. Zhai, Z. Ma, X. Zhu, Y. Li. Lyapunov-Guided Deep Reinforcement Learning for Service Caching and Task Offloading in Mobile Edge Computing. – Computer Networks, Vol. **250**, 2024, 110593.
6. An, X., Y. Li, Y. Chen, T. Li. Joint Task Offloading and Resource Allocation for Multi-User Collaborative Mobile Edge Computing. – Computer Networks, Vol. **250**, 2024, 110604.
7. Cheng, C., L. Zhai, X. Zhu, Y. Jia, Y. Li. Dynamic Task Offloading and Service Caching Based on Game Theory in Vehicular Edge Computing Networks. – Computer Communications, Vol. **224**, 2024, pp. 29-41.
8. Zhang, Z., F. Zhang, M. Cao, C. Feng, D. Chen. Enhancing UAV-Assisted Vehicle Edge Computing Networks through a Digital Twin-Driven Task Offloading Framework. – Wireless Networks, Vol. **31**, 2024, No 1, pp. 965-981.
9. Liu, X., J. Zheng, Y. Li, M. Zhang, R. Wang, Y. He. Multi-Path Serial Tasks Offloading Strategy and Dynamic Scheduling Optimization in Vehicular Edge Computing Networks. – Vehicular Communications, Vol. **49**, 2024, 100827.
10. Min, H., A. M. Rahmani, P. Ghaderkourehpaz, K. Moghaddasi, M. Hosseinzadeh. A Joint Optimization of Resource Allocation Management and Multi-Task Offloading in High-Mobility Vehicular Multi-Access Edge Computing Networks. – Ad Hoc Networks, Vol. **166**, 2025, 103656.
11. Ju, T., L. Li, S. Liu, Y. Zhang. A Multi-UAV-Assisted Task Offloading and Path Optimization for Mobile Edge Computing via Multi-Agent Deep Reinforcement Learning. – Journal of Network and Computer Applications, Vol. **229**, 2024, 103919.
12. Liu, J., Y. Wang, D. Pan, D. Yuan. QoS-Aware Task Offloading and Resource Allocation Optimization in Vehicular Edge Computing Networks via MADDPG. – Computer Networks, Vol. **242**, 2024, 110282.
13. Wang, S., S. Zhao, H. Gui, X. He, Z. Lu, B. Chen, Z. Fan, S. Pang. Energy-Efficient Collaborative Task Offloading in Multi-Access Edge Computing Based on Deep Reinforcement Learning. – Ad Hoc Networks, Vol. **169**, 2025, 103743.
14. Zhang, S. H., J. S. Wang, S. W. Zhang, Y. X. Xing, X. T. Wang, X. F. Sui. MOSO: Multi-Objective Snake Optimizer with Density Estimation and Grid Indexing Mechanism for Edge Computing Task Offloading and Scheduling Optimization. – Cluster Computing, Vol. **28**, 2025, No 4, 244.
15. Tolba, B., M. Abo-Zahhad, M. Elsabrouty, A. Uchiyama, A. H. Abd El-Malek. Joint User Association, Service Caching, and Task Offloading in Multi-Tier Communication/Multi-Tier Edge Computing Heterogeneous Networks. – Ad Hoc Networks, Vol. **160**, 2024, 103500.
16. Wang, J., M. Zhang, Q. Yin, L. Yin, Y. Peng. Multi-Agent Reinforcement Learning for Task Offloading with Hybrid Decision Space in Multi-Access Edge Computing. – Ad Hoc Networks, Vol. **166**, 2025, 103671.
17. Su, X., X. Fang, Z. Cheng, Z. Gong, C. Choi. Deep Reinforcement Learning Based Latency-Energy Minimization in Smart Healthcare Network. – Digital Communications and Networks, Vol. **11**, 2025, No 3, pp. 795-805.
18. Zhai, L., Z. Lu, J. Sun, X. Li. Joint Task Offloading and Computing Resource Allocation with DQN for Task-Dependency in Multi-Access Edge Computing. – Computer Networks, Vol. **263**, 2025, 111222.
19. Huang, Z., X. Wu, S. Dong. Multi-Objective Task Offloading for Highly Dynamic Heterogeneous Vehicular Edge Computing: An Efficient Reinforcement Learning Approach. – Computer Communications, Vol. **225**, 2024, pp. 27-43.
20. Zhu, K., S. Li, X. Zhang, J. Wang, C. Xie, F. Wu, R. Xie. An Energy-Efficient Dynamic Offloading Algorithm for Edge Computing Based on Deep Reinforcement Learning. – IEEE Access, Vol. **12**, 2024, pp. 127489-127506.
21. Lin, H., L. Yang, H. Guo, J. Cao. Decentralized Task Offloading in Edge Computing: An Offline-to-Online Reinforcement Learning Approach. – IEEE Transactions on Computers, Vol. **73**, 2024, No 6, pp. 1603-1615.

22. A l s a d i e, D. Efficient Task Offloading Strategy for Energy-Constrained Edge Computing Environments: A Hybrid Optimization Approach. – IEEE Access, Vol. **12**, 2024, pp. 85089-85102.
23. A l h a r t o m i, M., A. S a l h, L. A u d a h, S. A l z a h r a n i, A. A l z a h m i. Enhancing Sustainable Edge Computing Offloading via Renewable Prediction for Energy Harvesting. – IEEE Access, Vol. **12**, 2024, pp. 74011-74023.
24. A l A s s a f, M. M., M. Q a t a w n e h, A. A l R a d h i. A Cost-Benefit Model for Feasible IoT Edge Resources Scalability to Improve Real-Time Processing Performance. – Cybernetics and Information Technologies, Vol. **24**, 2024, No 4, pp. 59-77.
25. S h e r i f, H., E. A h m e d, A. M. K o t b. Energy-Efficient and Accelerated Resource Allocation in O-RAN Slicing Using Deep Reinforcement Learning and Transfer Learning. – Cybernetics and Information Technologies, Vol. **24**, 2024, No 3, pp. 132-150.

Received: 12.08.2025, Accepted: 03.12.2025