Optimizing Resource Allocation in MEC-Enabled CR-NOMA-Assisted IoT Networks: A DRL-Driven Strategy

Muhammad Taha Qaiser*, Muhammad Sarmad Sohail*, Minahil Shafqat*, Syed Asad Ullah*, Haejoon Jung†, and Syed Ali Hassan*
*School of Electrical Engineering & Computer Science (SEECS),
National University of Sciences & Technology (NUST), 44000 Islamabad, Pakistan.
†Department of Electronic Engineering, Kyung Hee University (KHU), Yongin 17104, South Korea.
Email: *{mqaiser.bee20seecs,msohail.bee20seecs,mshafqat.bee20seecs,sullah.phdee21seecs,ali.hassan}@seecs.edu.pk,
†haejoonjung@khu.ac.kr

***Abstract*—Mobile edge computing (MEC) has emerged as a promising paradigm to enhance the computational capabilities of resource-constrained secondary devices (RCSDs) in proximity to prescheduled primary devices (PDs). In this context, we introduce a novel framework where an energy harvesting (EH)-enabled RCSD efficiently offloads computational tasks to an MEC server, while employing a cognitive radio-inspired non-orthogonal multiple access (CR-NOMA) scheme for efficient data transmission. The RCSD also harvests energy from the ambient radio frequency (RF) signals of the surrounding PDs. We propose a deep reinforcement learning (DRL)–based optimization strategy, specifically the deep deterministic policy gradient (DDPG) algorithm to minimize the service delay of the RCSD and optimize the time-sharing coefficient for harvesting energy when offloading computational tasks to the MEC server. This dynamic resource allocation strategy intelligently determines the duration for which RCSDs transmit data and allocate time for energy harvesting, thereby ensuring an optimal balance between computation offloading and energy sustainability. Simulations demonstrate the effectiveness of the proposed scheme in maximizing the utility of the RCSDs while minimizing the overall service delay of the RCSD.**

***Index Terms*—Mobile edge computing (MEC), cognitive radio-inspired non-orthogonal multiple access (CR-NOMA), deep reinforcement learning (DRL), and deep deterministic policy gradient (DDPG).**

I. INTRODUCTION

IN the rapidly evolving landscape of modern communication networks, mobile edge computing (MEC) has emerged as a pivotal architectural component, playing a central role in addressing the escalating demands of industrial internet-of-things (IIoT), 5G, and beyond 5G (B5G) networks. MEC strategically places computational resources at the network's edge, revolutionizing the conventional cloud computing model [1]. This strategic positioning facilitates efficient and timely computation, crucial for applications across diverse domains, such as autonomous vehicles [2], smart cities, and augmented reality. By reducing latency and alleviating computational burdens, MEC significantly enhances the overall performance of communication systems for diverse applications.

Meanwhile, deep reinforcement learning (DRL), as a subset of machine learning (ML) stands at the forefront of transformative technologies, playing a pivotal role in shaping the landscape of 5G, 6G, and the industrial Internet-of-things (IIoT). Moreover, in the realm of IIoT, DRL's significance amplifies as it facilitates autonomous decision-making, predictive maintenance, and efficient utilization of resources in industrial processes. The seamless integration of DRL into these advanced technologies not only unlocks unprecedented capabilities but also paves the way for a future where intelligent, self-optimizing systems redefine the possibilities of connectivity and industrial automation. In this work, we adopt and implement a deep deterministic policy gradient (DDPG) algorithm to our system model. DDPG uses neural networks to simulate both the policy (action selection) and the value function (anticipated cumulative reward), further details regarding the DDPG algorithm shall be presented in section IV.

Similarly, cognitive-inspired non-orthogonal multiple access (CR-NOMA) emerges as a promising solution to facilitate unlicensed users in the network while guaranteeing the quality of service (QoS) of the licensed users, thereby enhancing spectral efficiency [3]. This innovative approach holds immense significance in addressing the escalating demand for connectivity and data rates in contemporary networks, offering a promising solution to optimize resource utilization and meet the evolving needs of modern wireless networks.

Therefore, the integration of CR-NOMA, MEC, and DRL emerges as a novel paradigm, promising to redefine the landscape of connected systems with more responsive and efficient computation offloading strategies. Building on this, this paper extends the exploration to encompass DRL within CR-NOMA-assisted MEC for computation offloading. To this end, we adopt DRL with CR-NOMA-assisted MEC, which unfolds as a potent combination that effectively addresses the challenges associated with computation offloading in modern communication systems. This hybrid approach introduces adaptability to decision-making in real-time interactions, which becomes particularly crucial when computation offloading decisions hinge on rapidly changing channel gains, rendering traditional optimization methods less practical.

Recently, many efforts have been made to revolutionize MEC. The authors in [4] explored ML-based approaches such as reinforcement learning (RL), and deep learning, in which how different ML-based strategies enhance MEC was highlighted.

In [5], authors investigated a green MEC system with energy harvesting and proposed an algorithm to optimize offloading decisions and transmit power ensuring efficient computation with minimal complexity. Similarly, the work in [6] introduced a DRL-based method for optimizing computation offloading and resource allocation in a multi-access MEC network. The authors in [7] focused on a wireless-powered MEC network with edge devices (EDs) and an edge computing server (ECs). The study aimed to jointly optimize wireless power transmission (WPT) duration, transmission time allocation for each ED, and partial offloading decisions to maximize the sum computation rate. Another study, [8], addressed the resource allocation problem in NOMA-enabled MEC for 5G networks using a decentralized multi-agent reinforcement learning scheme. The authors in [9] integrate MEC into IoT, using NOMA for improved connectivity and energy-efficient MEC. They have addressed joint radio and computation resource allocation, significantly enhancing energy efficiency in IoT networks with NOMA. However, in prior studies, the problem of tackling computation offloading (or MEC) in a resource-constrained secondary device (RCSD) operating in the vicinity of prescheduled primary devices (PDs) has not been addressed. Therefore, we propose a DRL-enhanced strategy to minimize the total service delay of an EH-enabled RCSD, which harvests energy from the ambient RF signals of the PDs and employs CR-NOMA to transmit data while offloading computations to the MEC server. Accordingly, our key contributions are listed as follows

- We develop and formulate the minimization problem to minimize the total service delay of the RCSD operating in a CR-NOMA-assisted IoT network.
- We employ the DDPG algorithm to minimize the overall service delay of the RCSD by optimizing the transmit power of the RCSD and timesharing coefficient for energy harvesting from the ambient RF signal of the PDs.
- We present the performance analysis of the proposed strategy and benchmark the proposed scheme with the existing models such as never offloading, always offloading, and random offloading.

This paper is organized as follows: In Sec. II we introduce the system model under consideration. In Sec. III we discuss problem formulation, followed by Sec. IV which highlights the fundamentals of the DRL and its implementation to the problem. Discussions on the simulation results are provided in Sec. V, followed by Sec. VI which concludes the paper.

II. System Model

We examine a wireless IoT network consisting of a base station (BS) equipped with an MEC server. The MEC server is equipped with $\mathbb{N}_c$ cores that are evenly divided into partitions, with each partition capable of performing computations at a rate of F cycles per second. The task related to the RCSD can be described using two parameters: the task size, and the task complexity. There are N prescheduled primary devices (PDs) alongside a single resource-constrained secondary device (RCSD) as shown in Fig 1. Each PD employs a time division multiple access (TDMA) scheme, utilizing dedicated timeslots

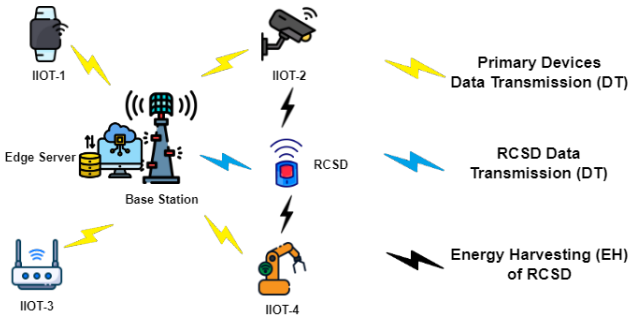

Fig. 1. Illustration of the considered IoT network.

for seamless transmission. In contrast to the PDs, the RCSD lacks a dedicated timeslot and instead seeks opportunistic transmissions. Additionally, the RCSD is considered to be energy-constrained, relying on the harvesting of energy from the radio frequency (RF) transmissions of the PDs. For instance, in the m-th time slot, the RCSD allocates the initial $\alpha_k T$ seconds for data transmission, and the remaining $(1-\alpha_k)T$ seconds for energy harvesting. Here, α_k is a time-sharing coefficient between 0 and 1.

Our goal is to minimize the overall service delay metric of the RCSD represented by, $\mathbf{S}_{T,rs}$. The total service delay of the RCSD for uplink communication encompasses both propagation delay and computational delay and is given by

$$\mathbf{S}_{T,rs} = \mathbf{T}_{p,rs} + \mathbf{T}_{c,rs}, \tag{1}$$

where $\mathbf{T}_{p,rs}$ and $\mathbf{T}_{c,rs}$ are the propagation and computational delays, respectively. These propagation and computation delays are given by

$$\mathbf{T}_{p,rs} = \frac{\mathrm{D}_{rs}}{\log\left(1+\frac{\Theta_{rs,k}h_{rs}}{1+\Theta_{p,k}h_p}\right)}, \tag{2}$$

and

$$\mathbf{T}_{c,rs} = \frac{\mathrm{D}_{rs}\cdot\omega_{rs}}{\mathbb{N}_c\cdot\mho_{rs}}, \tag{3}$$

respectively. In (2) and (3), D_{rs} represents the data size. In (2), Θ_{rs} denotes the transmit power of the RCSD, Θ_p, is the transmit power of the respective primary device, h_p represents the channel gain of between the PD and BS, and, $h_{r,s}$ is the channel gain between RCSD and BS. In (3), D_{rs} represents the data size, $\mathbb{N}_c$ is the number of cores, ω_{rs} denotes the computational complexity, and $\mho_{rs}$ represents the CPU cycles. We assume that RCSD is always required to offload its computations to the MEC server. Consequently, upon receiving the task, the RCSD will employ DRL to intelligently switch between energy harvesting and data transmission to keep a balance between the two, thereby achieving an efficient task offloading strategy while optimizing its transmit power and timesharing coefficient for harvesting energy from ambient RF signals of the PDs. In the following section, we formulate the service delay minimization problem into the DRL framework and address it by the DDPG algorithm.

III. Problem Formulation

This section formulates the service delay minimization problem into a DRL framework. First, we define ϕ_k as the amount of energy available in the battery of RCSD at time k, consequently, the RCSD's total transmission energy is upper bounded by ϕ_k.

$$\alpha_k T(\Theta_{rs,k}) \leq \phi_k, \tag{4}$$

consequently, the harvested energy by the RCSD at time $k+1$ is represented as

$$\phi_{k+1} = \min\left\{\phi_k + M, \phi_{\max}\right\}, \tag{5}$$

where $M = (1-\alpha_k)\, T\eta\Theta_{p,k}\left|h_{rs,k}\right|^2 - \alpha_k T(\Theta_{rs,k})$ and $h_{rs,k}$ denote the channel gain between the p-th primary device and the RCSD, respectively, at time k. Further, η denotes the energy harvesting efficiency coefficient, whereas $\phi_{\max}$ represents the RCSD's maximum battery capacity.

Our objective is to minimize total service delay, therefore, the minimization problem can be formulated as

$$\begin{aligned}
\underset{\alpha_k,\Theta_{rs,k}}{} \quad & \mathbf{S}_{T,rs}(\alpha_k,\Theta_{rs,k}) && \text{(P1)}\\
\text{s.t.} \quad & \mathbf{S}_{T,rs}(\alpha_k,\Theta_{rs,k}) \geq 0, && \text{(P1a)}\\
& 0 \leq \Theta_{rs,k} \leq \Theta_{\max}, && \text{(P1b)}\\
& 0 \leq \alpha_k \leq 1, && \text{(P1c)}\\
& \mathbb{R}_{p,k} \geq \log\left(1+\Theta_{\mathrm{p,k}}|h_p|^2\right), && \text{(P1d)}\\
& \text{(4) and (5).} && \text{(P1e)}
\end{aligned}$$

In Problem (P1), constraint (P1a) ensures that the overall service delay $\mathbf{S}_{T,rs}$ for RCSD remains non-negative. Constraint (P1b) restricts the transmit power $\Theta_{rs,k}$ for the RCSD, ensuring it adheres to the specified maximum transmit power, i.e., $\Theta_{\max}$. Constraint (P1c) limits the time-sharing coefficient α_k within the range of [0, 1], and (P1d) guarantees the QoS requirement of the corresponding PD.

Problem (P1) is non-convex due to the non-affine nature of constraint (5) in (P1e), and both optimization variables appear in multiplication in constraint (4) in (P1e). Nevertheless, given the values of the optimization variables are continuous, we can address this non-convex problem with the deep deterministic policy gradients (DDPG) algorithm. To this end, first, we split Problem (P1) into two sub-problems, because the range of values for the optimization variables makes direct DDPG implementation challenging. Therefore, the first sub-problem is defined as

$$\begin{aligned}
\underset{\alpha_k,\Theta_{rs,k}}{} \quad & \mathbf{S}_{T,rs}(\alpha_k,\Theta_{rs,k}) && \text{(P2)}\\
\text{s.t.} \quad & \bar{\phi}_k = (1-\alpha_k)T\eta\Theta_{p,k}|h_{rs,k}|^2 - \\
& \qquad (1-\alpha_k)T\Theta_{rs,k}, && \text{(P2a)}\\
& \text{(P1a), (P1b), (P1c), (P1d) and (4)} && \text{(P2b)}
\end{aligned}$$

where $\bar{\phi}_k$ denotes the energy fluctuation parameter. Problem (P2) is solved by convex optimization, where the closed-form expressions for $\Theta^*_{rs,k}$ and α^*_k are obtained and given by

$$\Theta^*_{rs,k}(\bar{\phi}_k) = \frac{(1-\alpha^*_k)\eta\Theta_{p,k}|h_{rs,k}|^2}{\alpha^*_k} - \frac{\bar{\phi}_k}{\alpha^*_k T}, \tag{6}$$

and

$$\alpha^*_k(\bar{\phi}_k) = \min\{1, \max\{x^*, \Omega_0\}\}, \tag{7}$$

where $\Omega_0 = \max\left\{1 - \frac{\phi_k+\bar{\phi}_k}{T\eta\Theta_{p,k}|h_{rs,k}|^2}, \frac{T\eta\Theta_{p,k}|h_{rs,k}|^2-\bar{\phi}_k}{T\eta\Theta_{p,k}|h_{rs,k}|^2+TP_m}\right\}$, $x^* = \frac{x_1-x_2}{e^{w_0(e^{-1}(x_1-1))+1}-1+x_1}$, and $x_1 = \frac{\eta\Theta_{p,k}|h_{rs,k,}|^2|h_{rs}|^2}{1+\Theta_{p,k}|h_p|^2}$, $x_2 = \frac{\bar{\phi}_k|h_{rs}|^2}{T(1+\Theta_{p,k}|h_p|^2)}$. Also, $W_0(\cdot)$ represents the Lambart-W-Function.

The second sub-problem is formulated below. Since our objective is to minimize service delay, according to Problem (P2), it is evident that the service delay, denoted as $\mathbf{S}_{T,rs}$, at time k, remains unaffected by $\hat{\alpha}_k$ and $\hat{\Theta}_{rs,k}$ for $k \neq \hat{k}$. Consequently, the optimization problem (P1) can be reformulated as a function of $\bar{\phi}_k$ within the DDPG framework as follows

$$\begin{aligned}
\bar{\phi}_k \quad & \gamma^{k-1}\mathbf{S}_{T,rs}(\bar{\phi}_k|\alpha^*_k,\Theta^*_{rs,k}) && \text{(P3)}\\
\phi_{k+1} &= \min\left\{\phi_{\max}, \phi_k + \bar{\phi}_k\right\}, && \text{(P3a)}
\end{aligned}$$

where $\gamma \in [0,1]$ represents the discounted factor. Problem (P3) ensures that the action of the RCSD is to select $\bar{\phi}_k$ for a given α^*_k and $\Theta^*_{rs,k}$. By substituting the expression of $\mathbf{S}_{T,rs}$ in (P3), we get the minimization problem as

$$\begin{aligned}
\bar{\phi}_k \quad & \sum_{k=1}^{M}\gamma^{k-1}\alpha^*_k(\bar{\phi}_k) * \left[\frac{\mathrm{D}_{rs}}{\log_2\left(1+\frac{\Theta_{rs,k}h_{rs}}{1+\Theta_{p,k}h_p}\right)} + \frac{\mathrm{D}_{rs}*\omega_{rs}}{\mathbb{N}_c\cdot\mho_{rs}}\right] && \text{(P4)}\\
\phi_{k+1} &= \min\left\{\phi_{\max}, \phi_k + \bar{\phi}_k\right\}. && \text{(P4a)}
\end{aligned}$$

Therefore Problem (P4) being a univariate, and continuous action-spaced problem is well-suited to be addressed by the DDPG algorithm.

IV. Implementation Of the DRL Algorithm

This section presents the fundamentals of the DRL algorithm, specifically the DDPG algorithm. Moreover, we articulate our problem within the framework of the DDPG algorithm. DRL is an advanced machine learning paradigm that combines deep learning techniques with reinforcement learning principles. In DRL, an agent learns to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. Deep Q-networks (DQNs) (or policy networks) are employed to approximate complex decision-making functions. The key idea is to enable the agent to discover optimal strategies through trial and error. The agent explores the environment, takes actions, observes the outcomes, and adjusts its behavior based on the received feedback.

A. Deep Deterministic Policy Gradient (DDPG)

As a hybrid actor-critic RL algorithm, DDPG is based on the deterministic policy gradient and Deep Q-Network (DQN) [10]. The inefficiency of DQ-Learning (DQL) [11] becomes apparent when confronted with continuous and high-dimensional action spaces, making DDPG particularly well-suited for scenarios with continuous action spaces.

DDPG leverages neural networks to model both the policy (action selection) and the value function (expected cumulative reward). In the DDPG algorithm, at a given time step k, the objective of an agent is to determine an action $\mathbb{A}_k$ for an observation $\mathbb{S}_k$, yielding a reward r_k, to maximize the action value function, denoted as $A(\mathbb{S}_k, \mathbb{A}_k)$. Thus, the maximization problem can be expressed as follows

$$\mathbb{A}^* k(\mathbb{S}_k) = \arg\max A(\mathbb{S}_k, \mathbb{A}_k), \tag{8}$$

where $A(\mathbb{S}_k, \mathbb{A}_k)$ represents the expected return. The actor network (or A network) takes the action, whereas the critic network (or C network) acts as an evaluator, which evaluates how well the action taken by the actor network is. The parameter for the policy network is ν^{λ}, which takes $\mathbb{S}_k$ as an input and produces an action, represented by $\lambda(\mathbb{S}_k|\nu^{\lambda})$. The corresponding actor target network is parameterized by ν^{λ_k} and outputs $\lambda_k(\mathbb{S}_k|\nu^{\lambda_k})$. The critic network is parameterized by ν^{C}, which takes $\mathbb{S}_k$ and $\mathbb{A}_t$ as inputs and produces the state value function, represented by $C(\mathbb{S}_k, \mathbb{A}_k|\nu^{C})$. The corresponding critic target network is parameterized by ν^{C_k} and outputs $C_k(\mathbb{S}_k, \mathbb{A}_k|\nu^{C_k})$.

The actor network executes the action, and concurrent networks verify that the actor network has undergone thorough training to evaluate its output (action) effectively. Consider a tuple $(\mathbb{S}_k, \mathbb{A}_k, r_k, \mathbb{S}_{k+1})$, where $\mathbb{S}_k$ denotes the current state, $\mathbb{A}_k$ signifies the action taken by the agent based on the observed state, r_k represents the reward for the action taken, and $\mathbb{S}_{k+1}$ denotes the subsequent state. Derived from the aforementioned tuple, the update process for the networks is outlined as follows

1) The actor network undergoes training by optimizing, identified as the state value function. By utilizing the parameters of both the actor and critic networks, it can be rephrased as

$$K(\nu^{\lambda}) = A(\mathbb{S}_k, \mathbb{A}_k = \lambda(\mathbb{S}_k|\nu^{\lambda})|\nu^{A}). \tag{9}$$

Upon calculating the gradient of (9) with respect to ν^{λ}, we obtain

$$\Delta_{\nu^{\lambda}} K(\nu^{\lambda}) = \Delta\mathbb{A}_k A(\mathbb{S}_k, \mathbb{A}_k|\nu^{A})\Delta\nu^{\lambda}\mu(\mathbb{S}_k|\nu^{\lambda}). \tag{10}$$

2) The critic network undergoes modification through the involvement of two actor networks. Initially, this involves providing the target actor network's output to the target critic network, yielding the target value as a state value function expressed as

$$Y_k = r_k + \gamma C_k(\mathbb{S}_{k+}, \lambda_k(\mathbb{S}_{k+}|\nu^{\lambda_k})|\nu^{C_k}). \tag{11}$$

The secondary assessment for the state value function can be acquired by minimizing the loss function specified as

$$L_f(\nu^{C}) = \left| Y_k - C(\mathbb{S}_k, \mathbb{A}_k|\nu^{C}) \right|^2. \tag{12}$$

3) Employing a soft target that assumes a smaller value, leads to a gradual adjustment of parameters for both the critic target network and the actor target network, which occurs less frequently in comparison to their respective counterparts. The associated parameters undergo updating in the following manner

$$\nu^{\lambda_k} \rightarrow \beta\nu^{\lambda} + (1-\beta)\nu^{\lambda_k} \tag{13}$$

and

$$\nu^{Q_k} \rightarrow \beta\nu^{Q} + (1-\beta)\nu^{Q_k}, \tag{14}$$

where β represents the soft updating parameter.

The DDPG algorithm incorporates two additional crucial components: the replay buffer and exploration. In the context of DDPG, the replay buffer involves storing past tuples $(\mathbb{S}_k, \mathbb{A}_k, r_k, \mathbb{S}_{k+1})$ in a buffer. These tuples play a vital role in enhancing the agent's learning process. After the network updating phase, random batches of tuples are selected from the stored experiences and used for training the algorithm. Regarding the exploration, the actor network is compelled to explore its environment thoroughly. To achieve this, a noise factor is introduced to the actor network's output, expressed as

$$\mathbb{A}(\mathbb{S}_k) = \lambda(\mathbb{S}_k|\nu^{\lambda}) + \Psi, \tag{15}$$

where Ψ is the introduced noise.

B. Problem Formulation into the DDPG Framework

In this subsection, we frame our optimization problem into the framework of the DDPG algorithm. Accordingly, we define the state space, action space, and reward as follows.

1) State Space: The state space is defined as a tuple that encompasses channel gains and the available energy in the battery of the RCSD.

$$\mathbb{S}_k = \left[\phi_k, |h_{rs,k}|^2, |h_r s|^2, |h_p|^2 \right]^T. \tag{16}$$

2) Action Space: According to Problem (P4), the action space consists of a single parameter, denoted as $\overline{\phi}_k$. The range of $\overline{\phi}_k$ is given by

$$-\min\left\{T\Theta_{\max}, \phi_k\right\} \leq \overline{\phi}_k \leq \min\left\{\phi_{\max} - \phi_k, T\eta\Theta_{p,k}|h_{rs,k}|^2\right\}. \tag{17}$$

The lower bound accounts for scenarios with $\alpha_k = 1$, implying no energy harvesting but only transmission. The upper bound on $\overline{\phi}_k$ is set because $\alpha_k = 0$, indicating no transmission but only energy harvesting, and acknowledges the finite amount of energy that can be harvested at time T_k. Given that equation (17) can take on values that are significantly larger or smaller, therefore, we normalize the range as follows

$$\begin{aligned} \overline{\phi}_k = \delta_k \min\left\{\phi_{\max} - \phi_k, T\eta\Theta_{p,k}|h_{rs,k}|^2\right\} - \\ (1-\delta_n)\min\left\{T\Theta_{\max}, \phi_k\right\}. \end{aligned} \tag{18}$$

Now according to (18), the action parameter for the DDPG algorithm is δ_k, where $\delta_k \in [0, 1]$.

TABLE I
SIMULATION PARAMETERS

Parameter Description	Symbol	Value
Learning rate for Actor Network	α_{actor}	0.002
Learning rate for Critic Network	α_{critic}	0.005
Data Batch Size	$\mathcal{B}$	64 Tuples
Memory Storage Capacity	$\mathcal{R}$	10000
Noise Spectral Density	σ_{noise}	-190 dBm
Maximum Battery Capacity	$\phi_{\max}$	0.2 J
Peak Transmit Power	$\Theta_{\max}$	23 dBm
Signal bandwidth	W_{signal}	10 MHz
Central Frequency	f_{center}	914 MHz
Energy Efficiency Factor	η	0.9
Duration of Time Slot	T	1 s
Discount Factor	γ	0.99
Soft Update Parameter	β	0.01

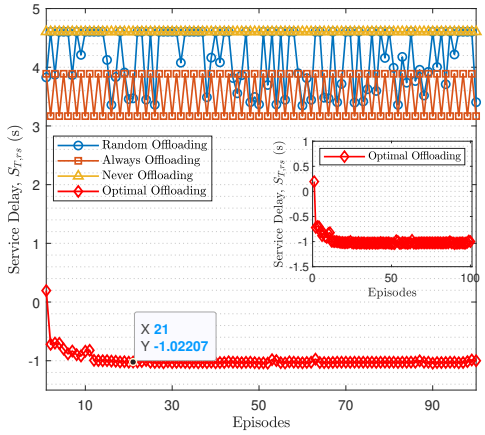

Fig. 2. Comparison of the episodic reward (service delay) convergence of the RCSD for the proposed scheme against the benchmark offloading schemes, and $N = 2$.

3) Reward: Intuitively the reward parameter is set as the service delay attained by the RCSD, i.e., $\mathbf{S}_{T,rs}$.

V. SIMULATION RESULTS AND ANALYSIS

In this section, we present the performance evaluation of the proposed methodology in comparison with the benchmark schemes. We assess the effectiveness of the DDPG algorithm in comparison to different offloading approaches such as, never offload, random offload, and always offload.

In our simulations, we consider the BS placed at the origin of the x-y plane. A large-scale path loss is being considered with random fading. The neural networks consist of two hidden layers each. The activation function employed for the two hidden layers is the rectified linear activation function (ReLU). Meanwhile, the output layer employs the hyperbolic tangent function as its activation function. Table I lists the other parameters fixed for the simulations.

Fig. 2 illustrates the comparison between episodic rewards, specifically in the context of service delay, for the proposed computation offloading strategy and the benchmark schemes. The results indicate that the DDPG-based computation offloading scheme outperformed the benchmark methods in terms of episodic rewards. Furthermore, the DDPG algorithm exhibits a near-convergence behavior after approximately 15 episodes,

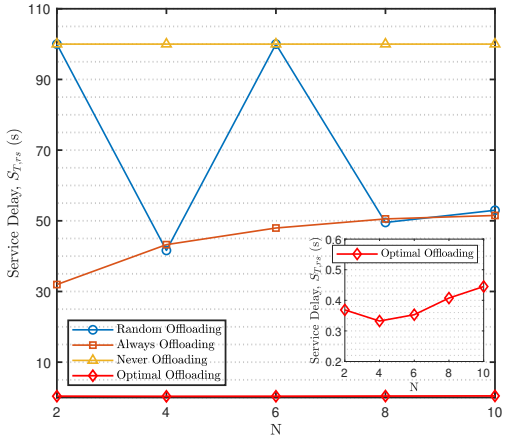

Fig. 3. Service delay comparison of the RCSD for various number of primary devices.

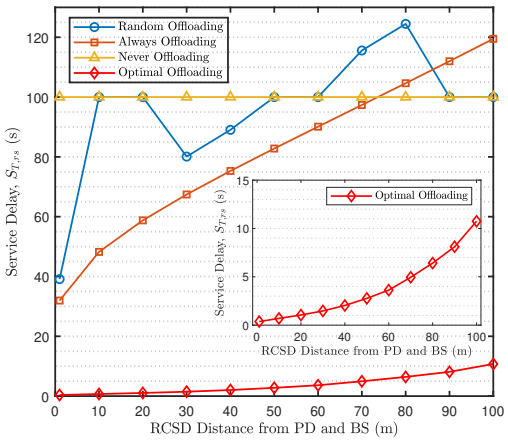

Fig. 4. Service delay comparison of the RCSD against the RCSD's distance from PD and BS and $N = 2$.

with only marginal improvements in episodic rewards thereafter. For a clearer understanding, and a more detailed insight into the performance of the proposed scheme, a magnified view of its performance is presented in Fig. 2. To assess the effectiveness of the DDPG-based computation offloading scheme, the service delay of the RCSD for all offloading schemes is presented in Fig. 3, considering the different numbers of the primary devices (PDs) in the network. We can observe that the proposed computation offloading strategy offers relatively less service delay compared to the benchmark methods. Additionally, we can observe that by increasing the number of PDs in the network the overall service delay of the RCSD increases, this is because the environment becomes complex and the agent faces a more complicated environment, thereby affecting the learning process of the agent and consequently an increase in the service delay is observed.

The impact of increasing the distance of RCSD from BS and PDs on the service delay is depicted in Fig. 4. It is evident from the figure that by moving the RCSD farther away from the PD and BS in the x-plane, the service delay increases. This increase in service delay is due to the weak channel between the PD and

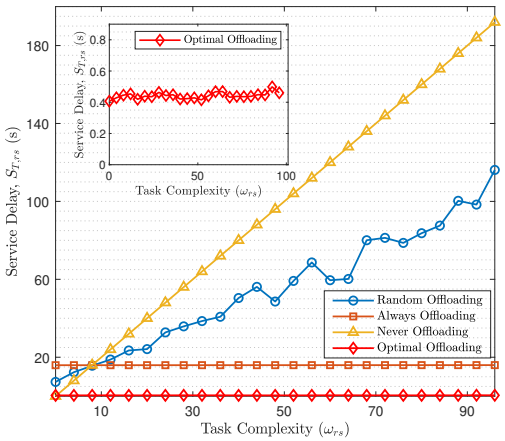

Fig. 5. Service delay of the RCSD against varying task complexity, and $N = 2$.

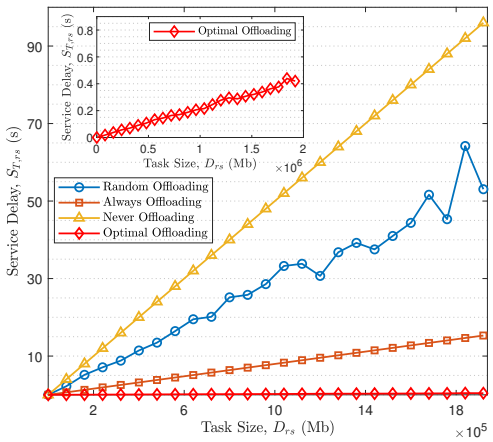

Fig. 6. Service delay of the RCSD against different task sizes, and $N = 2$.

RCSD, due to which the RCSD is unable to make frequent transmissions and harvest energy, thereby increasing the overall service delay. Nevertheless, we can see that the proposed scheme performs better in achieving lesser service delay compared to the benchmark schemes.

In Fig. 5, a comparison of the service delay of the RCSD against the task complexity is presented. It can be seen that the DDPG-based offloading strategy outperforms the benchmark schemes in terms of service delay. Additionally, we can observe that, for the proposed method, the RCSD is still able to maintain a low and stable service delay as the task gets complex. This consistency in service delay is valuable in real-world applications where tasks can vary in complexity, ensuring reliable and predictable service delivery.

In Fig. 6, we compare the service delay against task size for the proposed and benchmark schemes. It can be witnessed that the DDPG-based offloading scheme consistently outperforms the benchmark schemes by demonstrating lower service delay as the task size increases. Furthermore, we can see that as tasks become more substantial and demanding, the RCSD needs more time to perform computations, thereby depicting increased service delay. This information is valuable for assessing the scalability and efficiency of the proposed approach in handling tasks of different sizes.

VI. Conclusion

This paper explored the service delay minimization problem of a resource-constrained secondary device (RCSD) in a CR-NOMA-assisted MEC-enabled IoT network. We proposed a novel computation offloading scheme that minimizes the total service delay using a deep reinforcement learning (DRL) framework, specifically the deep deterministic policy gradient (DDPG) algorithm. The performance of the proposed scheme was compared with benchmark algorithms, such as the random, always, and never offloading methods. Simulation results revealed that the employed DDPG algorithm outperformed the benchmark schemes in terms of minimized service delay metric. Notably, the service delay curve for the proposed scheme demonstrated quick convergence and achieved lesser service delay, indicating robust performance in diverse environmental conditions. In the future, the system model could be enhanced to accommodate multiple RCSDs in the network and minimize the overall service delay of the network, which shall eventually come up with a multi-agent DRL optimization problem, where each RCSD would be tackled as an agent. It is worth mentioning that the proposed framework may not be appropriate in a delay-sensitive setting where quick transmission of information is critical.

References

[1] S. Munawar, Z. Ali, M. Waqas, S. Tu, S. A. Hassan, and G. Abbas, "Cooperative computational offloading in mobile edge computing for vehicles: a model-based dnn approach," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 3376–3391, 2022.

[2] K. Wang, L. Wang, C. Pan, and H. Ren, "Deep reinforcement learning-based resource management for flexible mobile edge computing: Architectures, applications, and research issues," *IEEE Vehicular Technology Magazine*, vol. 17, no. 2, pp. 85–93, 2022.

[3] Z. Ding, R. Schober, and H. V. Poor, "A new QoS-guarantee strategy for NOMA assisted semi-grant-free transmission," *IEEE Transactions on Communications*, vol. 69, no. 11, pp. 7489–7503, 2021.

[4] B. Cao, L. Zhang, Y. Li, D. Feng, and W. Cao, "Intelligent offloading in multi-access edge computing: A state-of-the-art review and framework," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 56–62, 2019.

[5] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[6] H. Gao and X. Guo, "Deep reinforcement learning-based computation offloading and optimal resource allocation in industrial Internet of things with NOMA," pp. 198–203, 2022.

[7] S. Zhang, H. Gu, K. Chi, L. Huang, K. Yu, and S. Mumtaz, "DRL-based partial offloading for maximizing sum computation rate of wireless powered mobile edge computing network," *IEEE Transactions on Wireless Communications*, vol. 21, no. 12, pp. 10934–10948, 2022.

[8] N. Waqar, S. A. Hassan, H. Pervaiz, H. Jung, and K. Dev, "Deep multi-agent reinforcement learning for resource allocation in NOMA-enabled MEC," *Computer Communications*, vol. 196, pp. 1–8, 2022.

[9] B. Liu, C. Liu, and M. Peng, "Resource allocation for energy-efficient mec in noma-enabled massive iot networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 4, pp. 1015–1027, 2020.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[11] C. E. Mariano and E. F. Morales, "DQL: A new updating strategy for reinforcement learning based on q-learning," in *Machine Learning: ECML 2001: 12th European Conference on Machine Learning Freiburg, Germany, September 5–7, 2001 Proceedings 12*, pp. 324–335, Springer, 2001.