Self-Optimized Agent for Load Balancing and Energy Efficiency: A Reinforcement Learning Framework with Hybrid Action Space

Bishoy Salama[1], Aamen Elgharably[2], Mariam Aboelwafa[3], Ghada Alsuhli[4], Karim Banawan[2] and Karim G. Seddik[1]

[1]The American University in Cairo
[2]Alexandria University
[3]NewGiza University
[4]Khalifa University

Corresponding author: Mariam Aboelwafa (email: mariam.nabil@ngu.edu.eg).

ABSTRACT We consider the problem of jointly enhancing the network throughput, minimizing energy consumption, and improving the network coverage of mobile networks. The problem is cast as a reinforcement learning (RL) problem. The reward function accounts for the joint optimization of throughput, energy consumption, and coverage (through the number of uncovered users); our RL framework allows the network operator to assign weights to each of these cost functions based on the operator's preferences. Moreover, the state is defined by key performance indicators (KPIs) that are readily available on the network operator side. Finally, the action space for the RL agent comprises a hybrid action space, where we have two continuous action elements, namely, cell individual offsets (CIOs) and transmission powers, and one discrete action element, which is switching MIMO ON and OFF. To that end, we propose a new *layered* RL agent structure to account for the agent hybrid space. We test our proposed RL agent over two scenarios: a simple (proof of concept) scenario and a realistic network scenario. Our results show significant performance gains of the proposed RL agent compared to baseline approaches, such as systems without optimization or RL agents that optimize only one or two parameters."

INDEX TERMS Reinforcement Learning, Cellular Networks, Self-Optimized Networks, Mobility Load Balancing, Energy Efficiency, Hybrid Action Space

I. INTRODUCTION

THE exponential growth of cellular data traffic in the last decade urges network providers to enact significant network optimization. To put that into perspective, mobile network data traffic grew 39% between the second quarter of 2021 and the second quarter of 2022 and reached 100 exabytes/month [1]. As a result, continuous configuration and management are necessary to sustain a balanced performance while facing such continued growth and endless changes. Nevertheless, network management needs to be *adaptive* and *self-optimized* due to the dynamic nature of the cellular network, heterogeneity of traffic loads, and the continuous adjustments of the operators' needs [2]. For instance, the number of users drastically fluctuates over time according to human activities. The coverage requirement varies as well due to fast urban changes. Moreover, energy conservation has become a universal need in the past decade [3], [4]. All this motivates the need for self-organizing networks (SONs) [5], which support autonomous planning, configuration, optimization, and healing.

Cellular network optimization is a challenging task in general. The problem becomes more burdensome if we require the network to self-optimize its operational parameters. This gives rise to the concept of AI-native networks. *AI-native* networks embrace artificial intelligence (AI) as an innate feature network functionality. In AI-native networks, the AI techniques are embedded in the design, deployment, operation, and maintenance of the network. AI solutions are appealing in situations where classical solutions are extremely complex (if they even exist), and the data is abundant to enable inference and learning. Specifically, an AI-native network utilizes system behavioral data (e.g., key performance indicators) to train an AI-based solution (e.g., neural networks). The AI solution aims to replace familiar

rule-based mechanisms with data-driven policies. AI-native networks need to detect and adapt to environmental variations. The learning modules include such new experiences to improve the learned policies over time. The AI-native concept is considered one of the disruptive technologies related to 6G cellular networks [6]–[8]. Recently, AI-enabled solutions have become more relevant with the introduction of open radio access networks (O-RAN). To break vendors' monopoly over the telecommunication market, O-RAN implements cellular networks as virtualized RANs. RANs, in this case, comprise disaggregated components, which are connected via open interfaces, and optimized by radio intelligent controllers (RIC). RICs are enabled by AI techniques, implemented in software, and can be developed as third-party applications (a.k.a., xApps, and rApps). Consequently, cellular network optimization can be implemented as an xApp/rApp, which is AI-native in nature to avoid the complexity of the problem [9]–[11].

To motivate the network self-optimization problem further from a technical perspective, consider the following concrete scenarios; first, consider the scenario of optimizing a cellular network with heterogeneous traffic loads to maximize the total throughput of the network. In this case, a controller of the base stations (a.k.a., eNBs in LTE) may opt to offload the traffic from heavily congested cells to less congested cells using Cell Individual Offsets (CIOs) as in [12]–[14]. The CIO is a power offset that controls the handover power level threshold without affecting the QoE of the mid-cell users. Using CIO, cell-edge users may be forced to leave congested cells such that they can enjoy a higher number of physical resource blocks (PRBs), which in turn may lead to higher total throughput. Nevertheless, in this case, the received power at cell-edge users is inferior to the received power if CIO is disabled. This reduces the Quality of Experience (QoE) of these users and prevents employing high modulation and coding schemes (MCS). Note that the CIO *fakes* the cell edges to enforce a handover decision that may not be optimal in terms of channel quality. This motivates the second scenario, where the controller has the same objective, but instead of using the CIO control, it controls the transmit power of the eNB as in [15]–[17]. Although this approach controls the actual boundaries of the cells in contrast to the CIO, it creates a new set of problems. More specifically, increasing the transmit power of one eNB may result in decreasing the signal-to-noise-and-interference (SINR) ratio of adjacent cells, which in turn results in lower QoE. Moreover, decreasing the transmit power may lead to having coverage holes. Consequently, tension arises between QoE and coverage if transmit power is controlled. Thus, joint optimization of the CIO and transmission power has proved its superiority in enhancing the performance of the network and the QoE of the users in [18], [19], compared to separate optimization of each parameter. In the final scenario, we consider the case where the controller aims to minimize the energy consumption of the cellular network. Disabling the MIMO features is an attractive solution, as it is known that the MIMO module is one of the most energy-hungry modules of the eNB. Switching the MIMO OFF consistently, however, leads to a significant reduction in the QoE. Consequently, a tradeoff between QoE and energy arises.

From the aforementioned scenarios, we conclude that optimizing cellular networks requires a joint and intricate choice of network parameters to satisfy a set of requirements that may be in tension. Due to the huge dimensionality of the aforementioned problem, it is challenging to devise a good analytical model to capture the system's dynamics and/or tackle the problem using classical optimization approaches. This motivates the adoption of Machine Learning (ML) techniques in this work.

We employ Reinforcement Learning (RL) in this work to implement our self-optimizing controller. RL is suitable for our work as we do not rely on labeled historical data as in supervised learning, but rather, we aim at maximizing a long-term average goal (e.g., throughput, coverage, or energy optimization). RL is beneficial in situations where the end goal is known, but the intermediate optimal steps are not learned in advance. However, RL depends mainly on trial and error, which is challenging to apply in a live cellular network since it directly impacts the QoE of the end-user. To tackle this issue, we use a simulated cellular network environment as will be explained later.

In this paper, we present a comprehensive framework for self-optimization of cellular networks using deep RL techniques[1]. The simulations primarily focus on LTE (4G) technology but remain relevant to 5G networks due to structural similarities. Technically, both LTE and 5G utilize Orthogonal Frequency Division Multiple Access (OFDMA) for downlink transmission. Dynamic resource allocation techniques, like Resource Block (RB) allocation and scheduling algorithms, optimize resource utilization and meet Quality of Service (QoS) targets based on user needs and network conditions. The Key Performance Indicators (KPIs) used in the work, such as Resource Block Utilization (RBU), Total Downlink Throughput of each cell, Number of active users per cell, and Modulation and Coding Scheme (MCS) Matrix remain relevant in 5G networks. Furthermore, 6G is anticipated to adopt Orthogonal Frequency Division Multiplexing (OFDM); this will, in turn, render our model as a potential contender for 6G load balancing with minimal adjustments required, subject to the specifics of the standard. Additionally, the vision for 6G entails it being an AI-native network standard, streamlining all manual tasks related to both the radio and core networks. This positions 6G as an ideal candidate for our model, facilitating the transition towards self-optimizing networks.

Our scheme exhibits a *self-optimizing* nature since it is capable of reaching a *stable state* of the cellular network

[1]Although our experiments are performed over an LTE network, our framework extends seamlessly to other cellular systems as long as the needed KPIs are reported to the central agent.

after any sudden variation in the surrounding environment. More specifically, we design a centralized deep RL agent that aims at 1) maximizing the sum throughput of the cellular network, 2) balancing the traffic load across the cells, 3) minimizing the number of uncovered users, 4) satisfying the QoE of individual users, and 5) minimizing the energy consumption of the eNBs. To that end, we propose the following controls to satisfy the aforementioned challenging (sometimes even conflicting) requirements: 1) CIOs, which affect load balancing and user blocking, 2) eNBs' power levels, which affect the handover decisions, inter-cell interference, in addition to the QoE/throughput of users, 3) switching the MIMO features of the eNB ON or OFF. By switching the MIMO feature ON, the users enjoy better QoE by either having a low probability of error via transmit diversity, or higher throughput via spatial multiplexing techniques. On the other hand, switching OFF the MIMO features significantly decreases the energy consumption of the eNB. This implies that selected actions contain actions that are picked from a *continuous* space, namely, CIOs and power levels, in addition to actions from a *discrete* space, which is enabling/disabling the MIMO features.

To that end, we design a novel RL agent that jointly optimizes the mentioned *hybrid* actions. To deal with the hybrid nature of the action space, we propose a *layered* agent structure. The decision-making process depends on a set of key performance indicators (KPIs) similar to [20], which are periodically reported to the network without incurring extra signaling costs. Moreover, we propose a versatile reward function (end-goal function) that incorporates the requirements under consideration and can be tweaked to the liking of the network operator. [2]

We tested our proposed scheme in a simulated, yet accurate settings. In this work, we simulate the cellular network using network simulator 3 (NS3). We extend the simulated cellular network in [19] to handle MIMO switching and scheduling. Our simulator takes into account actual eNB placements, and actual transmission parameters from a major cellular operator, along with actual mobility patterns using the simulation of urban mobility (SUMO) and the open street map (OSM). We provide extensive numerical results to assess the performance of our agent. Notably, our agent shows sum throughput enhancement of 18.5%, with switched OFF MIMO features for 34% of time, while maintaining a 65% covered users in all cells, in the actual scenario over the baseline system (fixed transmission power, zero CIO, and enabled MIMO). This implies that the operator can simultaneously reduce energy consumption while enhancing the QoE with reasonable coverage.

The main contribution of this paper can be summarized as:

- Providing a holistic self-optimizing framework for cellular networks, that can optimize throughput, coverage, energy, etc.
- Proposing a novel layered RL agent that deals with a mixture of discrete and continuous (hybrid) action spaces.
- Proposing a versatile penalized throughput reward function that takes into account the sum throughput, coverage, and energy consumption.
- Implementing an accurate NS3 simulator of a real cellular network in an urban Fifth Settlement neighborhood in Egypt along with realistic mobility patterns.

II. Literature Review

As mentioned in the previous section, the main objective of this work is to enable the cellular network to be a *self-optimizing* network. The network needs to autonomously alter its parameters to be more *immune* to urban changes or varying traffic demands. In this work, we train the cellular network to concurrently reach a maximized sum throughput, a balanced load among cells, a minimum number of uncovered users, and optimized energy consumption. As an extension to our work in [21], we simultaneously address three network management controls. These are CIOs, transmission Powers, and enabling the MIMO feature. Hence, the most closely related works to this paper are those concerned with energy saving and load balancing problems.

The energy-saving problem has been investigated in the literature. For instance, in [22], the authors discuss how cellular base stations can be dynamically operated by switching OFF redundant base stations during periods of low traffic. This proves to provide significant energy savings in cellular networks. Also, in [23], the authors design an efficient online scheduling algorithm to minimize energy consumption while optimizing the end-user experience. Moreover, in [24], the energy optimization problem in mobile networks is considered by using a neural network-based algorithm to enable the MIMO feature only when necessary to reach a satisfactory user QoE. The authors presented a different viewpoint in [25], where they use RL for power control in cellular networks. They propose a novel training strategy to accelerate learning to avoid performance degradation during state space exploration.

Load-balancing techniques are ubiquitous in the literature as well. In [12] and [13], the authors design an RL framework for optimizing cell parameters to balance traffic load across the cells. Their target is controlling the CIO of neighboring cells to force cell-edge users to hand over from congested cells into lighter-load cells. In [19], and [18], the authors propose an RL agent that controls both eNB transmitted power and CIOs to achieve traffic load balancing. The goal of the RL controller in these works is to maximize the DL sum throughput while minimizing the number of uncovered users.

[2] This work is extension of our published conference paper [21].

In this work, we adopt RL for realizing the cellular network controller. RL has been used to train cellular networks in many previous works. [26] presents a survey of the applications of deep RL in cellular networks.

The authors in [27] present an RL-based scheduler that aims to dynamically adapt to traffic variation by optimally scheduling Internet of Things (IoT) traffic. [28] considers a deep RL approach for coordinating the inter-cell interference. More specifically, the work [28] controls power in cellular networks aiming at maximizing the sum rate of the network. A downlink scheduling approach based on RL is presented in [29] to autonomously schedule the active traffic flows. RL can also be used to learn the optimal caching policy as in [30]. In [30], the authors formulate the content caching problem as a Markov decision process (MDP). They define the system reward as the delay reduction after performing a caching action. In [12], [13], [18], [19], [21], RL was adopted aiming to reach load balancing in the network with an end goal of maximizing the sum throughput.

Furthermore, as mentioned earlier, the action space in this work is a mix of continuous and discrete sets. Handling a hybrid action space exists in the literature as well. In [31], the authors focus on the hybrid action space of video games. They adopt the Soft-Actor-Critic (SAC) algorithm in a parameterized manner. They duplicate the continuous components that depend on other discrete components (one for each discrete component) as long as the model dimension remains reasonable. This does not apply to a cellular network with a large number of cells, as it suffers from the curse of dimensionality. Another example is in [32], where hybrid control in Robotics is the main concern. The authors propose a new scheme on top of the Maximum a posteriori Policy Optimization (MPO) algorithm. Although this scheme is considered applicable to relatively large problems, we preferred to adopt a layered RL agent that utilizes DDQN and TD3. This layered structure constitutes a natural generalization for our previous DDQN and TD3 agents in [13], [18], [19], [21]. The authors in [33] deploy a multi-agent algorithm with a hybrid action space. In this work, we focus on cellular networks with a central agent. Hence, this multi-agent approach is not suitable in our setting. Moreover, in [34], a mixed deep RL algorithm is presented to handle hybrid discrete-continuous action spaces in smart homes' energy management.

Another related work is presented in [35], in which the authors discuss the deployment of heterogeneous wireless networks within the 5G framework, focusing on Non-Orthogonal Multiple Access (NOMA) technology to manage spectral efficiency and system complexity. The paper addresses user association and uplink power allocation issues in heterogeneous 5G networks using NOMA, utilizing a Contract Theory (CT) approach to handle incomplete Channel State Information (CSI) and Reinforcement Learning (RL) for iterative user-to-base station (BS) association. While [35] concentrates on NOMA-based solutions and CT/RL techniques for user association and power allocation, this work focuses on optimizing mobile networks using model-free RL methods to improve throughput, energy efficiency, and coverage, without addressing NOMA but adopting OFDMA instead.

Additionally, the work presented in [36] is a related approach. The research in [36] primarily centers on RL-based performance tuning and fault management. It presents a framework utilizing RL to automatically adjust cellular network parameters, tackling issues such as fault management and enhancing downlink performance. The study highlights RL-based indoor voice-over-LTE power control algorithms and outdoor fault management. In contrast, our work adopts a broader approach involving self-optimizing networks. Our paper explores the concept of AI-native networks, where AI techniques are integrated into network functionalities to facilitate autonomous planning, configuration, and optimization.

The *novelty* of this work lies in two main points. First, we aim to jointly control three different parameters/features to reach a stable state in the network. Those three features, as mentioned above, are the CIOs, transmission power, and MIMO status. As far as it came to our knowledge, no other works in literature address the same problem. Most work in literature aims to control one or two features at most. We, on the other hand, aim to have more holistic control over the network.

The second novel point is how we deal with the hybrid nature of the action space. The controls over CIOs and transmission power belong to a continuous action space. However, enabling/disabling MIMO belongs to a discrete action space. In this work, we handle the hybrid action space in a hierarchical approach. We propose a *novel* RL-based scheme that allows the agent to make its decision in two successive stages in a simple, yet efficient approach (details are to be presented later in Section IV).

To prove the merit of the proposed hybrid approach, we also considered having one agent with a continuous action space instead of the hybrid agent. The action includes CIOs, transmission power, and a float in the interval $[-1, 1]$. The float value is then compared to a threshold of 0 to decide whether or not to turn the MIMO feature ON or OFF. The comparison between both agents is shown in Section V.

It is worth mentioning that, as mentioned earlier, this work is an extension of our previously published conference paper [21]. In comparing the conference version and this version of the paper, several key distinctions emerge. Firstly, while the conference paper explores two simple scenarios, the current submission ventures into more realistic settings and diverse mobility patterns, incorporating actual site locations from an Egyptian operator. Secondly, the performance evaluation in the current submission surpasses the conference paper by introducing a comparison to the continuous agent only (TD3), revealing the superior performance of the proposed hybrid agent. Furthermore, a notable difference lies in the more extensive simulation results and analysis present in the

current submission, offering a richer and more comprehensive exploration.

III. Technical Background: Reinforcement Learning

RL is a process in which an agent learns to achieve a certain goal in the long run (maximize a certain specified reward) [26]. To that end, an RL agent makes decisions (applies actions), observes the impact of its decisions on the surrounding environment, and adjusts its strategy based on its observation. RL is an efficient technique for solving problems that can be modeled as an MDP [37, Chapter 3]. This MDP describes an interaction between the environment, which is the cellular network in our problem, and the agent we need to design. In MDPs, at each time step t, the agent receives some representation of the environment (a.k.a, a state, $s(t)$, that belongs to a state space, $\mathcal{S}$). Depending on this state, the agent selects an action, $a(t)$ from a predetermined set of actions, $\mathcal{A}$). Next, the agent receives the consequence of its action (feedback signal), which is a numerical reward $r(t+1)$ and a new state $s(t+1)$ [37, Chapter 3].

The objective of the decision-maker (a.k.a., the agent) is to reach the sequence of actions (a.k.a., the policy) that eventually maximizes the expected reward function [18], [37, Chapter 3]. This *objective* can be characterized as:

$$\max_{\pi} \lim_{L\to\infty} \sum_{t=0}^{L} E[\lambda^t r(t)], \tag{1}$$

where $r(t)$ is the reward function, λ is the discount factor that determines the significance of the reward's future expected values, and π is the policy to be learned.

The nature of the action space can be discrete, continuous, or hybrid. In a discrete action space, the agent selects the actions a finite set of possible actions. This is in contrast to the continuous-action-space case, where the agent picks its action from a bounded interval. A hybrid action space implies that the agent samples some actions from a finite set, while others are taken from bounded intervals.

To solve the aforementioned MDP using RL, there are several variants of RL techniques [37]. The basic Q-learning technique works with a discrete set of actions [37, Chapter 6]. The Deep Q-Network (DQN) technique employs a deep neural network to approximate the output of the Q-function (see the definition of Q-function in Section A). The Double-DQN (DDQN) technique is an extension of DQN. In DDQN, the agent implements two different neural networks for action selection and action evaluation [38]. More techniques exist in literature as well [39]. For a continuous set of actions, various RL techniques can be used. One way is to use SAC methods, whose details can be found in [40]. Another family of techniques is Policy Gradient methods (and their extensions like TD3), which can be understood from [41]. For hybrid action spaces, there are some parameterized approaches in the literature to handle this mixed type of action spaces [31]–[33]. In this work, we present a novel approach to handling hybrid action spaces.

We focus on two specific RL techniques, which are DDQN and TD3. Next, we provide a brief description of each of them.

A. Double Deep Q-Learning (DDQN)

Q-Learning is an effective algorithm for learning the decision-making policy for MDP. To that end, the agent learns the Q-function, which is the value of an action in a particular state. The Q-function of policy π is defined by [37, Chapter 3]

$$Q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \lambda^k r(t+k) | s(t)=s, a(t)=a \right]. \tag{2}$$

In the traditional Q-learning, the agent needs to build a Q-table that contains $Q_\pi(s,a)$ for all (s,a) pairs. The optimal Q-function $Q_{\pi^*}(s,a)$ is computed by solving the Bellman optimality equation as in (3), where (s',a') is any possible next state-action pair. The optimal policy π^* in (4) is obtained greedily with respect to $Q_{\pi^*}(s,a)$. The Bellman equation can be written as [37, Chapter 3]:

$$Q_{\pi^*}(s,a) = \mathbb{E}_{s'} \left[r(t) + \lambda \max_{a'} Q_{\pi^*}(s',a') | s(t)=s, a(t)=a \right] \tag{3}$$

which leads to the optimal policy,

$$\pi^*(s) = \arg\max_{a\in\mathcal{A}} Q_{\pi^*}(s,a). \tag{4}$$

Because of the curse of dimensionality problem associated with constructing the Q-table, the Deep Q-network (DQN) [42] has been developed. DQN relies on approximating the Q-function $Q(s,a;\mathbf{w})$ using a deep neural network with weights $\mathbf{w}$ [43, Chapter 4], i.e., the output of DQN generates the Q-values of all possible actions $a \in \mathcal{A}$ at state s. This fact makes the DDQN works only for a limited number of actions, i.e., a discrete action space.

At each time step t, the agent applies an action $a(t)$ to the environment and thus a sample $(s(t), a(t), s(t+1), r(t+1))$ is collected and used to update the Q-network. To explore as much as possible of (s,a) pairs, the agent selects random actions with probability $\varepsilon(t) \in [0,1]$, equation (8). The value of $\varepsilon(t)$ decays gradually during the learning process to allow for more exploitation of the gained experience by the agent [44]:

$$a(t) = \begin{cases} \arg\max_{a'\in\mathcal{A}} Q(s(t),a';\mathbf{w}_t) & \text{w.p. } 1-\varepsilon(t) \\ \text{random } a \in \mathcal{A} & \text{w.p. } \varepsilon(t) \end{cases} \tag{5}$$

For the training of the Q-network, the agent periodically updates the weights $\mathbf{w}$ such that the difference between the target Q-value $y(t)$ and the predicted Q-value is minimized, i.e., the training procedure minimizes the loss function $\mathcal{L}(\mathbf{w})$ [43, Chapter 4],

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}\left[(y(t) - Q(s(t),s(t);\mathbf{w}_t))^2 \right], \tag{6}$$

where the target Q-value is given by,

$$y(t) = r(t+1) + \lambda \max_{a'} Q(s(t+1), a'; \mathbf{w}_t). \quad (7)$$

Despite its efficiency, DQN suffers from unstable learning and unguaranteed convergence [45]. For this reason, in this work, we employ a new variant of DQN. In this version, we equip the DQN with an experience replay buffer and the target network [44]. We stabilize the learning process using an experience replay buffer $\mathcal{D}_F$ with maximum size F. This buffer is used to store the samples collected by the agent during the learning process, i.e., [44]

$$\mathcal{D}_F = \{(s(t), a(t), s(t+1), r(t+1)) \text{ for all } t\}. \quad (8)$$

Instead of using the correlated samples collected by the agent with the same collection order to update the weights of the DQN, a mini-batch of size B_s is randomly picked from $\mathcal{D}_F$ and used altogether for the update process. This random sampling breaks the correlation between the samples and prevents the DQN from forgetting the rare samples [45].

To enhance the convergence of the DQN, another deep neural, called the target network, is added to the agent. The target network has weights $\hat{\mathbf{w}}$ and is added beside the base network to obtain Double DQN (DDQN) [46]. DDQN aims to reduce the overestimation of Q-values in the learning process by separating the selection of the action from its evaluation. Thus, the base network with weights $\mathbf{w}$ selects the best action in the next state, while the target network evaluates the action by estimating its Q-value. Thus, the target Q-value of the DDQN is defined as follows [46]:

$$y(t) = r(t+1) + \lambda \times \underbrace{Q(s(t+1), \underbrace{\arg\max_{a'} Q(s(t+1), a'; \mathbf{w}_t)}_{\text{from base NN}}; \hat{\mathbf{w}}_t)}_{\text{from target NN}}. \quad (9)$$

The base network is updated in the same way as DQN. Whilst the target network is a delayed version of the base network, i.e., the target network is updated with period T_u by copying the weights of the base network.

B. Twin Delayed Deep Deterministic Policy Gradient (TD3)

To deal with continuous action spaces, we move away from the value-based RL (e.g., Q-learning-based methods) toward the policy gradient-based RL (e.g., actor-critic-based methods). The main difference between Q-learning and actor-critic methods is the separation between policy selection and policy evaluation. Specifically, in actor-critic methods, the optimal policy is learned directly and separately without the need to predict the Q-value of all state-action pairs in a single model. This separation relaxes the requirement of tabulating all possible action values and allows for considering continuous actions. Hence, the actor-critic methods have two distinct models. The first is the actor function $\mu(s)$, which is responsible for learning the best action for a state s. The second is the critic function $Q(s, a)$, which estimates the Q-value of the (s, a) pair.

Without loss of generality, in this work, we employ TD3 algorithm [47], which has shown great success over the other actor-critic methods in several recent optimization problems [48]–[52]. TD3 is a variant of the deep deterministic policy gradient (DDPG) [53] with improved performance. As in the case of DDPG, TD3 shares several DDQN enhancement techniques, such as experience replay and target network, which we discussed in the previous section. However, instead of using one critic model as in DDPG, TD3 trains two independent critic functions (thus, "twin"). The agent uses the *smaller* of the two Q-values to calculate the target and update the critic functions. Since underestimation errors do not propagate, this limits the overestimation problem known in both Q-learning and actor-critic methods. In addition, the policy and the target networks are updated less frequently than the Q-function in TD3, every T_u iterations. This delayed update results in more stable learning and less training time. Moreover, TD3 uses a regularization strategy by adding clipped noise to the target action. This regularization smooths out the Q-value and mitigates the impact of Q-function errors on the policy. Next, we describe our implementation of the TD3 algorithm in more detail.

In our implementation, the TD3 uses six NNs; two NNs as critics functions, $Q(s, a; \mathbf{w}_t^{c_1})$ and $Q(s, a; \mathbf{w}_t^{c_2})$; one NN for the actor function $\mu(s; \mathbf{w}_t^a)$; three NNs represents the critics and the actor targets with weight vectors $\hat{\mathbf{w}}_t^{c_1}$, $\hat{\mathbf{w}}_t^{c_2}$, $\hat{\mathbf{w}}_t^a$, respectively, where $\mathbf{w}_t^{c_i}$, $i = 1, 2$ is the ith critic weight vector, and $\mathbf{w}_t^a$ is the weight vector of the actor. We randomly initialize the weights of the base NNs whereas the weights of the target networks are set to their respective base weights: $\hat{\mathbf{w}}_t^{c_i} \leftarrow \mathbf{w}_t^{c_i}$ for $i = 1, 2$ and $\hat{\mathbf{w}}_t^a \leftarrow \mathbf{w}_t^a$.

Initially, the experience buffer $\mathcal{D}_F$ is empty. Hence, the agent starts to explore the action space $\mathcal{A}$ by interacting with the environment and applying actions extracted from the output of the actor added to uncorrelated Gaussian noise $\mathcal{N}(0, \sigma^2)$. For a time step $t = 0, 1, \cdots, T$, where T is the total number of steps per episode, if the environment is at state $s(t)$, the selected action $a(t)$ by the agent is [47]

$$a(t) = \text{clip}(\mu(s(t); \mathbf{w}_t^a) + e, \theta_L, \theta_U) \quad (10)$$

where e is a noise vector, whose ith component $e(i) \sim \mathcal{N}(0, \sigma_n^2)$; $\text{clip}(x, l, u) = l$ if $x < l$, $\text{clip}(x, l, u) = u$ if $x > u$, and $\text{clip}(x, l, u) = x$ if $l \leq x \leq u$. The clip function aims to keep the selected actions in the action space range (θ_L, θ_U), where θ_L, θ_U are the lower and upper bounds of the action space interval, respectively. After applying the action $a(t)$, the agent receives feedback from the environment that includes the next state $s(t+1)$ and the reward function $r(t+1)$. Then, a sample $(s(t), a(t), s(t+1), r(t+1))$ is stored in the experience buffer $\mathcal{D}_F$.

Once the buffer contains more than B_s samples, we randomly select a batch of size B_s from $\mathcal{D}_F$ for each iteration. Let $(s_i, a_i, r_{i+1}, s_{i+1})$ be the ith sample of the batch, where $i = 1, 2, \cdots, B_s$, the target action is computed

based on the output of the target actor-network [47],

$$a_{i+1} = \mu(s_{i+1}; \hat{\mathbf{w}}_t^a) \tag{11}$$

Then, a clipped noise $\tilde{e} = \text{clip}(\mathcal{N}(0, \tilde{\sigma}_n^2), -c, c)$ is added to compute the smoothed target action [47] ,

$$\tilde{a}_{i+1} = \text{clip}(a_{i+1} + \tilde{e}, \theta_L, \theta_U), \;\; i = 1, \cdots, B_s \tag{12}$$

where $c > 0$ is the maximum absolute value of the clipped noise. Afterward, a single target of both critics' Q-functions is calculated using the smaller Q-value [47] , i.e.,

$$y_i = r_{i+1} + \lambda \min_{j=1,2} Q(s_{i+1}, \tilde{a}_{i+1}; \hat{\mathbf{w}}_t^{c_j}) \tag{13}$$

We update the weights of the two critics by minimizing the mean square error along the mini-batch [47] , i.e., for $j = 1, 2$,

$$\mathbf{w}_{t+1}^{c_j} = \arg\min_{\mathbf{w}_t^{c_j}} \frac{1}{B_s} \sum_{i=1}^{B_s} (y_i - Q(s_i, a_i; \mathbf{w}_t^{c_j}))^2 \tag{14}$$

On the other hand, we update the actor function less frequently every T_u iterations. This function is updated such that the expected Q-value function is maximized. Accordingly, we calculate the gradient ascent of the expected Q-value with respect to $\mathbf{w}_t^a$ as [47]

$$J(\mathbf{w}_t^a) = \mathbb{E}[Q(s, a; \mathbf{w}_t^c) | s = s_i, a = \mu(s_i; \mathbf{w}_t^a)] \tag{15}$$

Using the chain rule, we can compute the gradient $\nabla_{\mathbf{w}_t^a} J(\mathbf{w}_t^a)$

$$\approx \frac{1}{B_s} \sum_{i=1}^{B_s} \nabla_a Q(s, a; \mathbf{w}_t^c)|_{s=s_i, a=\mu(s_i; \mathbf{w}_t^a)} \nabla_{\mathbf{w}_t^a} \mu(s; \mathbf{w}_t^a)|_{s=s_i} \tag{16}$$

Regarding target networks update, the soft update is used in TD3 to stabilize the learning. The three target NNs are updated as a linear combination of old target weights and newly learned weights [47],

$$\hat{\mathbf{w}}_{t+1}^c \leftarrow \beta \mathbf{w}_{t+1}^c + (1 - \beta) \hat{\mathbf{w}}_t^c \tag{17}$$

$$\hat{\mathbf{w}}_{t+1}^a \leftarrow \beta \mathbf{w}_{t+1}^a + (1 - \beta) \hat{\mathbf{w}}_t^a \tag{18}$$

where $\beta \in [0, 1]$ is the soft update coefficient.

IV. Problem Description and Proposed Approach

In this section, we present a comprehensive description of the problem at hand and the proposed algorithm. First, we discuss the system model and the presented framework in detail.

A. System Model

We consider an LTE cellular network that consists of N eNBs and U User Equipment (UEs).

1) eNodeBs

Each eNB sends its transmission in the DownLink (DL) with a power level $P_n \in [P_{\min}, P_{\max}]$ dBm. At times $t = 0, 1, 2, 3, \cdots$, each UE measures the Signal-to-Interference-plus-Noise-Ratio (SINR) of near eNBs and attaches to the cell that results in the highest SINR.

An eNB can be over-utilized or under-utilized. This is determined according to the value of the eNB utilization ρ_n:

$$\rho_n = \frac{\sum_{i=1}^{U_n} K_{i,n}}{B_n / B_{\text{PRB}}}, \tag{19}$$

where U_n is the number of UEs served by the nth eNB, $K_{i,n}$ is the number of Physical Resource Blocks (PRBs) that serve the ith user in the nth eNB, B_n is the bandwidth of the nth eNB and B_{PRB} is the bandwidth of one PRB (=180 KHz in LTE). Note that ρ_n is the ratio of the total number of required PRBs of the nth eNB (to serve the attached users) to the maximum number of PRBs it can offer. Therefore, $\rho_n < 1$ means that the eNB is underutilized, while $\rho_n > 1$ means that the eNB is overutilized. Underutilization allows the eNB to serve all its attached users with satisfactory rates, which is not the case with over-utilization.

Every eNB can have the MIMO feature turned ON or OFF (depending on the decision of the network manager). Turning the MIMO feature ON has a considerable effect on the rate received at the receiving end. Specifically, switching the MIMO feature ON results either in a higher data rate (e.g., using spatial multiplexing modes) or a lower BER (e.g., using spatial diversity mode). Consequently, UEs can enjoy a better QoE. Nevertheless, MIMO is one of the most energy-consuming features in the eNB. When the MIMO feature is turned ON, a *scheduler* decides whether to use the multiple antennas to apply Spatial Multiplexing (SMux) or Transmit Diversity (TxD) transmission modes. The decision depends on the channel quality of the UE. Nodes with high channel quality are assigned the SMux transmission mode to achieve higher data rates. On the other hand, nodes with lower channel quality (e.g., cell-edge nodes) are assigned the TxD transmission mode to improve their received SINR and combat the effect of channel fading.

2) UEs

Each UE is assumed to have a random motion. It regularly searches for a better cell (according to the higher SINR) and attaches to the better cell if found. Moreover, The channel quality indicator (CQI) ϕ_u of the uth UE is reported to the associated cell periodically. The CQI is a discrete measure that represents the quality of the channel. According to the standard, $\phi_u \in \{0, 1, \cdots, 15\}$. When $\phi_u = 0$, the uth UE is out of coverage. A higher CQI value corresponds to higher channel quality [54], [55].

When a certain UE is attached to cell i, it might require handover to another neighboring cell j if [54]:

$$\text{RSRP}_j + \theta_{j-i} > \text{Hys} + \text{RSRP}_i + \theta_{i-j}, \tag{20}$$

where RSRP_i and RSRP_j are the measured Reference Signal Received Power from eNBs i and j, respectively. θ_{i-j} is the CIO value of eNB i with respect to eNB j and θ_{j-i} is the CIO value of eNB j with respect to eNB i. Hys is a hysteresis value to minimize repeated handover requests that might occur due to minor signal quality fluctuations.

It is worth noting that the aforementioned system model can represent 5G network as well. Both systems have an OFDM-based air interface. The definition of the RBU in (19) is consistent with the 5G air interface. The A3 handover procedure in (20) is readily available in the 5G NR specifications [56].

B. Reinforcement Learning Framework

The mapping of our joint optimization problem to the RL algorithm can be explained briefly as follows: The *agent* is a central network manager that exerts action on eNBs to control different parameters. The *environment* is the cellular network under consideration, which we aim to deliver to a self-optimized state. The *state* is selected to be a subset of the network KPIs, which are readily available to the network operator in practice. These KPIs are:

- Resource Block Utilization (RBU) $(B(t))$: The fraction of used PRB blocks that serve the users of each cell. It is an N-length vector. It is a representation of how congested each cell is.
- Total DL throughput of each cell $(R_n(t))$: It represents the eNB performance. It can be expressed as $R_n(t) = \sum_{u_n=1}^{U_n} R_{u_n}(t)$, where $R_{u_n}(t)$ is the measured throughput of user u_n in the nth eNB.
- Number of active users in each cell $(C(t))$: This measures the number of users that are not idle in a certain time step.
- Modulation and Coding Scheme (MCS) Matrix $(M(t))$: It is a matrix that gives the fraction of users with a certain MCS. The MCS matrix represents the quality of the channels.

The state is the concatenation of the above vectors (after reshaping $M(t)$):

$$s(t) = [B(t)^T \quad R(t)^T \quad C(t)^T \quad \text{vec}(M(t))^T]. \tag{21}$$

where $\text{vec}(\cdot)$ represents matrix vectorization process.

We note that we chose these KPIs as they are already in use by current network operators for monitoring cellular network performance. Furthermore, the chosen KPIs accurately reflect the state of the network. To see this, we argue that cell congestion implies whether the agent should apply a load-balancing action or not. The agent monitors the congestion event through RBU and the number of active users. Additionally, any control action must not hurt the total throughput of the network. That is why the agent monitors the network's throughput and deals with it as a part of the state fed to the agent. Finally, the agent needs to monitor the channel qualities to decide on actions that affect the QoE of the users. For example, Transmit Diversity is more suitable than Spatial Multiplexing in case of low channel quality.

The **action** contains the features that the agent has control over. These features are:

- Relative CIO values between every two neighboring cells. $\theta_{ij} = -\theta_{ji} = \theta_{i-j} - \theta_{j-i}$. This action belongs to a continuous actions space $[-\theta_{\max}, \theta_{\max}]$.
- Transmission Power of each eNB P_n. It belongs to a continuous action space. The agent chooses a value from the set $[P_{n_0} - P_{\max}, P_{n_0} + P_{\max}]$ for some constant value P_{n_0}.
- Turning MIMO feature ON/OFF for nth eNB m_n. The whole MIMO action vector is $[m_1 \quad m_2 \quad \cdots \quad m_N]^T$. This action is selected from a discrete set of size 2^N binary vectors since each eNB has a decision of $m_n = 0$ (MIMO OFF) or $m_n = 1$ (MIMO ON).

The main target of the RL agent is to reach the policy that maximizes the expected reward function, in the long run, [18]. That is:

$$\max_{\pi} \lim_{L\to\infty} \sum_{t=0}^{L} E[\lambda^t r(t)], \tag{22}$$

where λ is the discount factor that determines the significance of the reward's future expected values. The above formulations allow for the flexibility of selecting an operator-preferred reward function. In our paper, we focus on the following **reward** function[3]:

$$r(t) = \sum_{n=1}^{N} R(t) - \eta \bar{R}(t) \sum_{u=1}^{U} \mathbf{1}(\phi_u = 0) - \mu \sum_{n=1}^{N} m_n, \tag{23}$$

where η and μ are hyper-parameters, which are selected to meet the operator's requirements, and $\bar{R}(t)$ is the average user throughput at time instant t. This reward function consists of a linear combination of three terms. The first term $(\sum_{n=1}^{N} R(t))$ is the total network throughput (to be maximized). The second term $(\eta \bar{R}(t) \sum_{u=1}^{U} \mathbf{1}(\phi_u = 0))$ is the sum throughput of the uncovered users[4] scaled by a hyper-parameter η. In this work, we define the uncovered users as those with a throughput less than or equal to a certain threshold. Since one of the objectives is to minimize the uncovered users (a penalty); therefore it is subtracted, and we scale this penalty by η to control how significant it is to the agent. The last term $(\mu \sum_{n=1}^{N} m_n)$ is also a penalty. It is the number of eNBs that have the MIMO feature turned ON. We scale the MIMO enabling penalty by another hyper-parameter μ. We note that the choice of the hyper-parameters is determined based on the operator's

[3]It should be noted that our proposed framework can readily be extended for any other reward function.

[4]In this work, we use the number of uncovered users as a *proxy* for the effective coverage of the cell. Specifically, having $\phi_u = 0$ implies that the UE has not received any usable LTE signals and that the channel is inoperable. This, in effect, restricts the coverage of the cell.

interests, which in turn are based on the strategic objectives of the service provider (e.g., coverage/capacity tradeoff), the channel conditions, and the network settings.

Thus, the agent's objective is to reach the policy (sequence of action) that maximizes the total network throughput, minimizes the number of out-of-coverage users, and minimizes the consumed energy due to turning MIMO ON. Note that choosing the reward function as the total throughput with no penalties may not reflect the operator's objective. Without the uncovered users' penalty, the agent may choose to keep only the users with high rates. Thus, the agent may alter CIOs or reduce power levels to force edge users to hand over to a poorer-performance cell. The other penalty limits the consumed energy due to turning MIMO ON. Without this energy penalty, the agent may always opt to turn MIMO ON with no regard to energy consumption.

C. Problem Formulation

In this section, we describe the problem formulation. In our setting, the control agent aims to simultaneously maximize the long-term average of the sum throughput, $\sum_{i=1}^{N} R_n(t)$, minimize the probability of the uncovered users, $\mathbb{P}(\phi_u = 0) = \mathbb{E}[\mathbf{1}(\phi_u = 0)]$, and minimize the energy consumption of the network, which can be measured by the number of the cells, where the MIMO feature is ON, $\sum_{n=1}^{N} m_n$. To that end, at time t, the central agent needs to jointly optimize the CIOs, $\boldsymbol{\theta}(t) = [\theta_{ij}(t) : i \neq j, i, j = 1, 2, \cdots, N]$, the transmit power of the eNBs, $\mathbf{P}(t) = [P_n(t) : n = 1, 2, \cdots, N]$, and the MIMO feature, $\mathbf{m}(t) = [m_n(t) : n = 1, 2, \cdots, N]$ subject to the (stochastic) dynamics of the cellular network. This formulation can be written as the following optimization framework in (24).

where $\tilde{s}(t)$ denotes the *full* cellular network state at time t, $f(\cdot|\cdot)$ denotes the stochastic dynamics of the cellular network, i.e., the conditional distribution of the next state of the network given the old state and the actions taken at time t, $g_1(\cdot,\cdot)$, $g_2(\cdot,\cdot)$ denote mapping functions from the network state to the throughput of the cell, and the CQI of the users, respectively. We note that, in our problem, the stochastic dynamics $f(\cdot|\cdot)$, and the mapping functions $g_1(\cdot,\cdot)$, $g_2(\cdot,\cdot)$ are unknown to the agent in advance and cannot be represented by specific model. The agent needs to *learn* these functions from experience. Furthermore, our agent does not have access to the full cellular state but a partial state $s(\cdot)$ represented by a subset of network readily available KPIs.

As depicted in the problem formulation, it becomes evident that the relationship between penalized throughput and transmitted power, CIOs, and MIMO functionality is characterized by an unknown function. When faced with such instances of uncertainty in mathematical modeling, Machine Learning emerges as the optimal choice for decision-making. Particularly, Reinforcement Learning stands out as the solution for problems afflicted by data scarcity, mirroring the challenges encountered in our case.

D. Algorithm

The main issue in formulating the proposed cellular network optimization as an MDP is having a hybrid action space. To solve this issue, we present a scheme that adopts two existing RL algorithms in a layered fashion. The first one is the Double Deep Q-Network (DDQN) [38], which is used for discrete action spaces (MIMO ON/OFF in our case). The second one is Twin Delayed Deep Deterministic Policy Gradient (TD3) [41], which is used for continuous action space (transmission power and CIOs in our case). The presented scheme is simple, yet efficient. One of its main advantages is that it requires no modification in the core of the used techniques (DDQN and TD3), i.e., the proposed scheme deals with DDQN and TD3 techniques as black boxes.

The agent takes its decision in two stages.

- **First Stage:** The agent observes the state and takes the action of MIMO ON/OFF based on the DDQN technique [38]. The action is taken from the discrete set $\{0, 1\}$ (for each eNB). Note that the chosen action *is not applied* to the environment until the end of the second stage.
- **Second Stage:** We augment the first-stage action with the observed state. The second stage decides the CIO and the variation in power level actions based on the TD3 technique. They are selected from the continuous intervals $[-\theta_{\max}, \theta_{\max}]$ and $[-P_{\max}, P_{\max}]$ respectively.

After the two stages, the augmented action is given by:

$$\begin{aligned} a(t) =&[(\theta_{ij} : i \neq j,\ i, j \in \{1, \cdots, N\}, \\ &(P_n : n \in \{1, 2, \cdots, N\}), \\ &(m_n : n \in \{1, 2, \cdots, N\})] \end{aligned} \tag{25}$$

$a(t)$ is then applied to the environment. As the agent explores the whole action space, the agent learns the effect of the different combinations of the first-stage action (MIMO ON/OFF) and second-stage action (Relative CIOs and Power Levels).

An overview of the proposed scheme can be seen in Fig. 1. We describe the scheme in algorithmic steps in Algorithm 1 , where $s(t)$ is the observed state at time t, $a_M(t)$ is the MIMO enabling action vector, $a_C(t)$ is the CIO values action vector and $a_P(t)$ is the transmitted powers action vector. Moreover, we demonstrate DDQN and TD3 schemes in Fig. 2, and Fig. 3.

It is worth mentioning that turning ON the MIMO feature for a certain eNB does not mean that SMux can be applied for all users attached to this eNB. There is a *scheduler* applied for each eNB, which is responsible for deciding which users to apply SMux and which to use TxD. This decision depends on the channel quality of each user. NS3 has SISO as the default running scheme. NS3 leaves absolute liberty to the user to turn MIMO modes (SMux or TxD) ON or OFF (i.e., The scheduler does not have a role in

$$
\begin{aligned}
\max_{\boldsymbol{\theta}(t),\mathbf{P}(t),\mathbf{m}(t)} \quad & \lim_{L\to\infty} \mathbb{E}\left[\sum_{t=1}^{L}\lambda^t\left(\sum_{n=1}^{N}R_n(t)-\eta\bar{R}(t)\sum_{u=1}^{U}\mathbf{1}(\phi_u(t)=0)-\mu\sum_{n=1}^{N}m_n(t)\right)\right] \\
\text{s.t.} \quad & \tilde{s}(t+1)\sim f\left(\tilde{s}(t+1)|\tilde{s}(t),\boldsymbol{\theta}(t),\mathbf{P}(t),\mathbf{m}(t)\right) \\
& R_n(t)=g_1(\tilde{s}(t+1),\tilde{s}(t)), \quad n=1,2,\cdots,N \\
& \phi_u(t)=g_2(\tilde{s}(t+1),\tilde{s}(t)), \quad u=1,2,\cdots,U \\
& \theta_{ij}(t)\in[-\theta_{\max},\theta_{\max}], \quad i\neq j,\ i,j=1,2,\cdots,N \\
& P_n(t)\in[P_{n_0}-P_{\max},P_{n_0}+P_{\max}], \quad n=1,2,\cdots,N \\
& m_n(t)\in\{0,1\}
\end{aligned} \tag{24}
$$

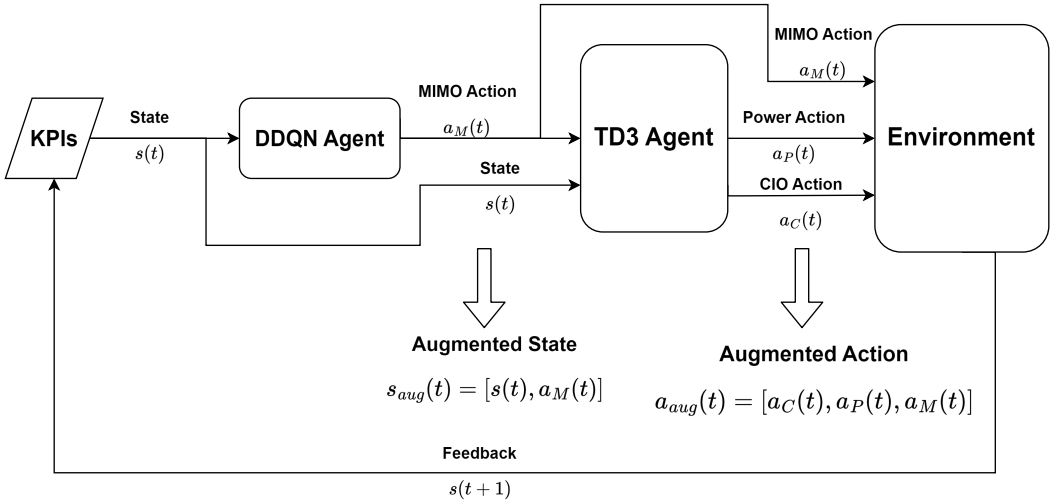


FIGURE 1. An Overview of the Decision-Making Process.

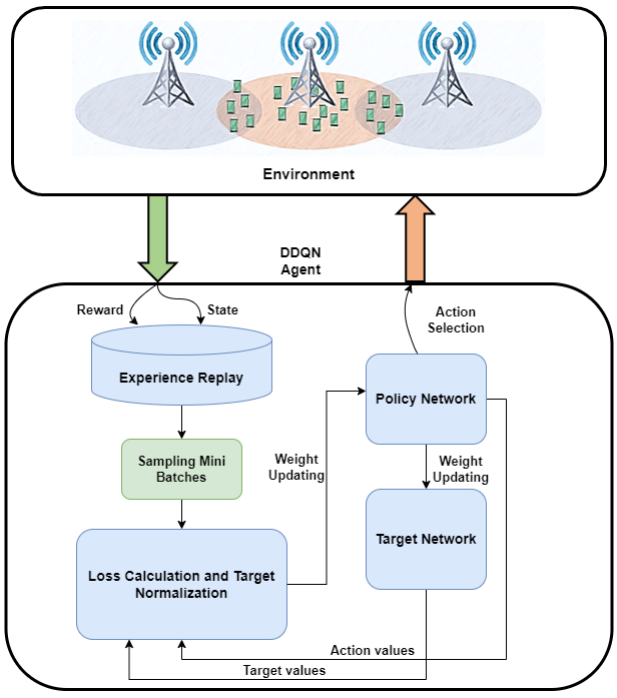


FIGURE 2. DDQN-based Optimization [13]

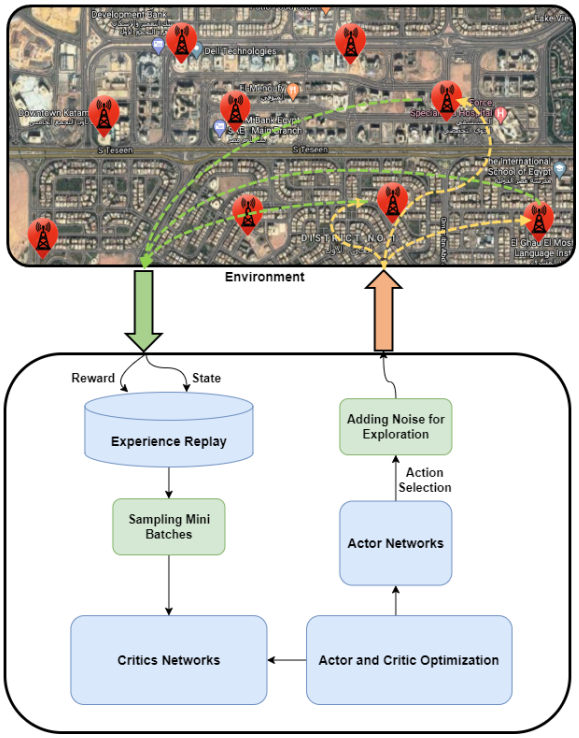


FIGURE 3. TD3-based Optimization [13]

Algorithm 1 Proposed RL framework

1: Determine Reward Function.
2: Reset all values.
3: **repeat**
4: **procedure** STAGE ONE
5: Observe State $(s(t))$.
6: Select MIMO feature decision (DDQN) $(a_M(t))$.
7: Create a new augmented state $(s_{aug}(t) = [s(t), a_M(t)])$.
8: **end procedure**
9: **procedure** STAGE TWO
10: Observe state $(s_{aug}(t))$.
11: Select relative CIO and power level actions (TD3) $([a_C(t), a_P(t)])$
12: Apply augmented action to the network $a_{aug} = [a_C(t), a_P(t), a_M(t)]$.
13: **end procedure**
14: Calculate Reward.
15: Calculate the next state.
16: **until** Reward Function Converges

selecting the suitable MIMO mode). The main concern is that turning SMux for users with low CQI will only make things worse. TxD is more suitable here to enhance the channel of less fortunate users. To solve this problem, we developed a simple scheduler that applies SISO for all users when the agent chooses $m_n = 0$. On the other hand, when the agent chooses $m_n = 1$, the CQI of each attached user is tested. The SMux is selected for users having $\phi_u \geq 7$, and TxD is selected otherwise. The CQI threshold is determined such that TxD is applied for users that use QPSK and SMux for users with higher MCS.

V. Performance Evaluation

A. Network Simulator

The proposed approach relies on the fact that the central agents can interact and make changes to the cellular network. This is because our RL agents have no prior knowledge of the optimal policy and need to learn this policy from experience. Consequently, using historical records of cellular operators is not applicable in our case. To that end, we need to utilize a realistic simulator that can mimic the existing cellular network. Our proposed simulator builds on the NS3 network simulator. NS3 is a discrete-event simulator that contains a dedicated module to simulate the LTE system. This LTE module is a realistic and a highly-accurate simulator that emulates the entire LTE system. More specifically, using the LTE module, we can simulate the complete protocol stack of the eNodeBs and the UEs. We have modified several NS3 built-in protocols to (i) allow the agent to control the relative CIOs for neighboring cells, (ii) and to allow the designed scheduler mentioned in D to select the appropriate MIMO mode of operation. The connection between the NS3 simulator and the agent was realized using the NS3gym interface. NS3gym framework allows for seamless integration of OpenAI Gym and NS3 to enable RL techniques for networking optimization. NS3gym interface is responsible for delivering the agents' actions to the environment and relaying back the states and rewards calculated by NS3 simulation to the gent. Next, we describe the details of the environments simulated using NS3.

TABLE 1. Simulation Parameters

Parameter	Value
System Bandwidth (B_n)	5 MHz
Basic eNB transmission power (P_{n_0})	30 dBm
eNB antenna height	30 m
eNB antenna pattern	Omni
UE antenna height	$1.5 - 2m$
Path loss model	COST Hata
Penalty on uncovered users (η)	$\in [0, 2]$
Penalty on applying MIMO (μ)	$\in [0, 5]$
Blockage Threshold (ρ)	0.5 Mbps
Training steps (Steps)	50,000
Steps per episode (T)	250
Step time	0.2 seconds
Handover	Hysteresis = 3 dB Time to Trigger = 40 ms

B. Simulation Setup:

In this work, we simulate two scenarios [5]. The first one is the *simple scenario*. We design this scenario to show the gain that can be obtained by applying the proposed approach to the LTE network. The second setup is the *real scenario*, which a cellular operator should consider before adopting any approach for practical network implementation.

We simulate the environment for both scenarios using the aforementioned NS3 simulator (LENA module) [57]. We implement the agent using Python. We realize the interface between the agent and the environment using the NS3gym interface [58]. This interface is responsible for applying agent actions to the environment and feeding back the reward and new environment state to the agent. After applying the agent action, the environment is updated using the control parameters sent by the agent action. The reward is calculated using (23). We summarize our simulation parameters in Table 1.

The used RL algorithms in this work (DDQN and TD3) are simulated using the stable implementation of *Open AI Baselines* [59]. The simulation parameters of these algorithms are listed in Table 2.

1) Simple Scenario

In this scenario, we consider a cellular network consisting of 3 eNBs, placed on the vertices of an equilateral triangle, and

[5]The Codes for this work are readily available at the Github repository: https://github.com/AamenElgharably/Self-Optimized-Agent-for-Load-Balancing-and-Energy-Efficiency

TABLE 2. Simulation parameters of different RL agents

DDQN Parameter	Value
Memory size	20000
Discount factor (λ)	0.95
Exploration rate (ϵ)	1.0
Exploration rate multiplier	0.9995
Minimum ϵ (Final value)	0.01
Learning Rate	0.001
Batch Size	32
Hidden Layers ($N_h, n_l, Activation$)	(2,(24,24),ReLU)
Output Layer Activation	Linear
Optimizer	Adam
Loss	Huber loss
TD3 Parameter	**Value**
Batch size (B_m)	128
Soft update coefficient (γ)	0.005
Policy delay (T_u)	2
Exploration noise	Normal ($\sigma = 0.1$)
Hidden Layers ($N_h, n_l, Activation$)	(2,(64,64),ReLU)
Discount factor (λ)	0.99
Number of steps per episode	250

42 UEs. Each cell has 10 users centralized around the eNB. On the edges between every two nodes, there are 4 edge users. We set the inter-eNB distance to be 500 meters. UEs have random mobility patterns in boxes around their starting points with a constant velocity v_u. UEs are assumed to be active all the time. UE traffic model is the Constant Bit Rate (CBR) model at 1 Mbps. UE mobility model is "Random Walk" (speed $= 3$ m/s).

2) Real Scenario

In this setup, we select an area of size 900m$\times$1800m from the urban Fifth Settlement neighborhood in Egypt to perform the simulation using a realistic placement of eNodeBs. The resulting network represents a cluster of six eNBs, whose locations are provided by the 4G network operator.

For more realistic users' mobility in our environment, we use Simulation of Urban Mobility (SUMO) [60], due to its simplicity and efficiency in producing realistic mobility patterns. In addition, SUMO can import accurately-emulated environments from factual maps such as Open Street Map (OSM). This imported environment considers the existing road structure, number of lanes, traffic light rules, buildings, $\cdots$, etc. After extracting the map from OSM, we use the SUMO simulator to introduce realistic mobility models of the UEs. The mobile UEs in this scenario are either vehicles or pedestrians. The pedestrians walk at a speed range between 0-3 m/s. Whereas the vehicles' mobility characteristics, i.e., acceleration, deceleration, speed factor, and speed deviation, are taken from [61] to emulate the realistic behavior of the vehicles. We randomly distribute those UEs on the available streets and pedestrian lanes at the beginning of the simulation. Afterward, each UE has a random trip from a source to a destination street during the simulation time. UEs are assumed to have a full buffer traffic model, i.e., the users are always active. The unlimited demand of the users allows reaching the congestion with a lower number of UEs, which reduces the NS3 environment simulation time, which is the bottleneck in our simulations.

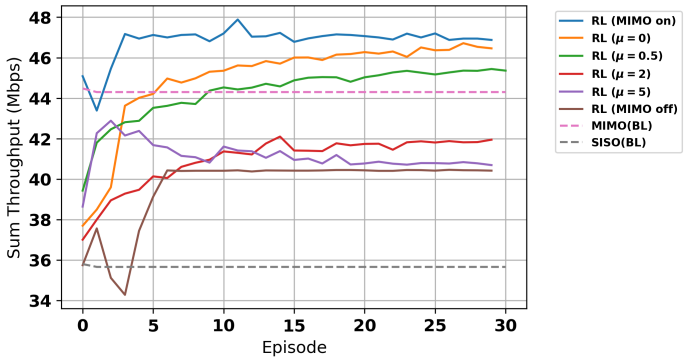


FIGURE 4. Effect of MIMO Energy Penalty μ on Sum Throughput (Simple Scenario)

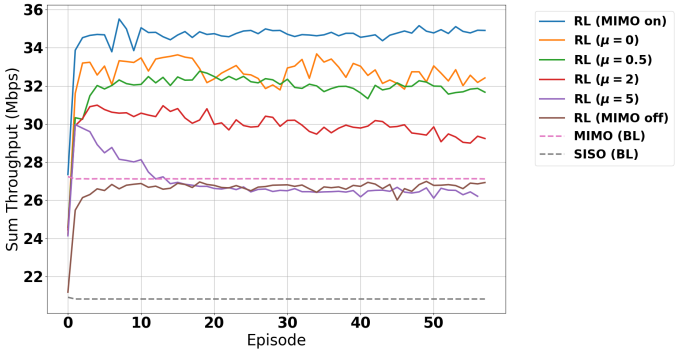


FIGURE 5. Effect of MIMO Energy Penalty μ on Sum Throughput (Real Scenario)

C. Results

In this section, we present the simulation results of both scenarios. In both setups, we test the effect of different hyperparameters on the sum throughput of the network, the percentage of uncovered users, and the amount of energy consumed by the MIMO feature. In this work, we assess the

performance of our proposed approach (i.e., the hybrid RL agent) against five benchmarks as follows[6]:

1) *SISO baseline (SISO BL):* In this benchmark, the agent has no control over any network feature. I.e., the MIMO feature is switched OFF, the CIO is set to zero for all cells, and the power levels of all eNBs are set to $P_{n_0} = 30$dBm.
2) *MIMO baseline (MIMO BL):* Similar to SISO BL, CIOs are set to zeros, and power levels are set to $P_{n_0} = 30$dBm. Different from SISO BL, the MIMO feature is switched ON at all times.
3) *RL with MIMO OFF:* In this benchmark, we employ an RL agent to optimize the CIOs and power levels only as in [18]. We switch OFF the MIMO feature at all times.
4) *RL with MIMO ON:* Similar to the previous benchmark with switching the MIMO feature ON at all times.
5) *RL with continuous action space (RL TD3):* In this approach, we utilize a TD3-based agent with a continuous action space only. The state that the TD3 agent observes is the same as which the hybrid agent observes in the first stage, i.e., $s(t)$ in equation (21). The TD3 agent jointly optimizes relative CIOs, power levels, and MIMO actions over continuous action space in one layer of optimization (in contrast to the proposed hybrid approach, which uses two layers). Specifically, the MIMO action, in this case, is a float in the range $[-1, 1]$. The MIMO action is then discretized into two levels $\{0, 1\}$ based on a threshold of 0. This dictates whether to activate or deactivate the MIMO feature for each eNB.

In Fig. 4, and Fig. 5, we plot the network sum throughput (in Mbps) versus the number of episodes of the training phase for the simple scenario and the real scenario, respectively. We compare the performance of the hybrid RL agent, RL agent that controls CIOs and power levels only with fixed MIMO feature (RL with MIMO ON/OFF agents), and the Baseline (BL) setting. By RL agent, we mean our proposed hybrid approach in this paper. We evaluate the hybrid RL agent for different values of the hyper-parameter μ, which scales the MIMO energy penalty. In all figures, we observe that the sum throughput increases during the training phase until it converges. We notice that as μ increases, the agent's tendency to turn MIMO ON decreases (since it negatively affects the reward function). Therefore, the sum throughput for smaller μ values is higher. However, reducing μ implies increasing the consumed energy due to turning the MIMO feature ON.

Next, we focus on investigating the significance of the *layered* agent design. To that end, we compare three different settings: Hybrid RL agent, TD3 agent, and baseline (SISO BL). By the *TD3 agent*, we refer to the RL with continuous action space (RL TD3) agent, which takes all continuous actions only and then discretizes the MIMO action by comparing to a threshold of 0. It is described in Algorithm 2, where $a_{M_c}(n,t)$ is the continuous MIMO action of the n^{th} eNB in the range $[-1, 1]$ at time t, $a_{M_c}(t)$ is the continuous MIMO action vector at time t, and $a_M(n,t)$ is the discrete (binary) MIMO action of the n^{th} eNB at time t.

In Fig. 6, we plot the network sum throughput (in Mbps) versus the number of episodes of the training phase for both the hybrid agent and the TD3-based agent in a real scenario. Fig 6 shows that the performance of the TD3-based agent (RL with continuous action space) is always lower than the proposed hybrid framework. At $\mu = 0$ throughput was about 7% lower than its hybrid counterpart. With increasing the MIMO penalty, the gap in throughput increases. Furthermore, when a penalty is applied to using MIMO, It is clear that hybrid agent training is more stable than the TD3 agent. This is evident from the erratic performance dip at the initial training episodes for the TD3 agent. That is due to the fact that, in the hybrid approach, the MIMO configuration is known before optimizing the CIO and Power actions. This, in turn, implies that the second layer of the hybrid scheme has the ability to take more informed actions. This is in contrast to the RL TD3 agent, which is agnostic to the actual MIMO decision while optimizing the CIO and power levels. Moreover, when using a TD3-based agent only, the penalty μ effect will be reflected in training, which leads to more unstable learning. In addition, the size of the action space for the TD3 agent is larger than its counterpart of the second layer of our proposed hybrid approach, which naturally leads to elongating the random exploration behavior at the beginning of the training, which may cause unexpected performance dips especially if the DNN weights are randomly initialized.

Algorithm 2 TD3 Agent

1: Determine Reward Function.
2: Reset all values
3: **repeat**
4: Observe State $(s(t))$
5: Select relative CIO and power level action and MIMO action$([a_C(t), a_P(t)], a_{M_c}(t))$
6: Discretize MIMO action
7: $a_M(n,t) = 1$ if $a_{M_c}(n,t) \geq 0$
8: $a_M(n,t) = 0$ if $a_{M_c}(n,t) < 0$
9: Apply action to the network $([a_C(t), a_P(t)], a_M(t))$
10: Calculate Reward
11: Calculate the next state
12: **until** Reward Function Converges

[6]Although there are other approaches to deal with hybrid action spaces, they are incompatible with our problem. For example: Discretization of the power/CIO would incur quantization errors and the action space would grow exponentially as the number of cells increases. Representing the discrete actions as one-hot vector encoding would face the same scalability issues. Finally, dealing with each action category via independent agent is suboptimal as the power/CIO decisions are linked to the MIMO feature activity as shown in Fig. 6.

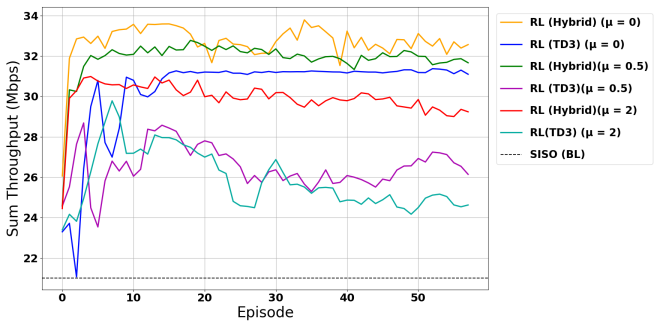


FIGURE 6. Effect on MIMO Energy Penalty μ on Sum Throughput Hybrid agent vs TD3 agent

We also observe that for all values of μ, the sum throughput is higher than the BL setting with MIMO OFF. In addition, the RL agent controlling the MIMO feature outperforms the RL agent with MIMO turned OFF by default (SISO). It also approaches a close performance to the RL agent that has MIMO turned ON by default (but with less consumed energy, as we can see later). Furthermore, we can see in Fig. 4 and Fig. 5 that turning ON MIMO with no other control on any network feature (BL) gives a higher sum throughput than turning MIMO OFF (with and without control over other features). Interestingly, our proposed scheme with small values of μ outperforms BL with MIMO ON besides consuming less MIMO energy.

It is worth mentioning that at $\mu = 5$, the sum throughput curve increases at first and decreases afterward. This behavior is because the first portion of the learning phase is mainly for action space exploration. I.e., the agent applies random actions to examine their effect on the environment. Turning MIMO ON in this case ($\mu = 5$) would potentially increase the sum throughput at the cost of increasing the penalty, which is scaled by 5. Thus, the combined reward decreases, and eventually, the agent learns that turning MIMO OFF is more beneficial (in this case) to maximizing the adopted reward (penalized sum throughput). Since we plot only part of the reward (sum throughput), the behavior shown in the curve is logical.

TABLE 3. Effect of MIMO Energy Penalty μ on Ratio of Time MIMO is Turned ON

μ	Simple Scenario	Real Scenario
0	97%	70%
0.5	73%	66%
2	32%	43%
5	3%	1%

Table. 3 illustrates the percentage of time the MIMO is turned ON for different values of μ after convergence. Please note that these percentages are considered a measure of the percentage of consumed energy compared to turning MIMO

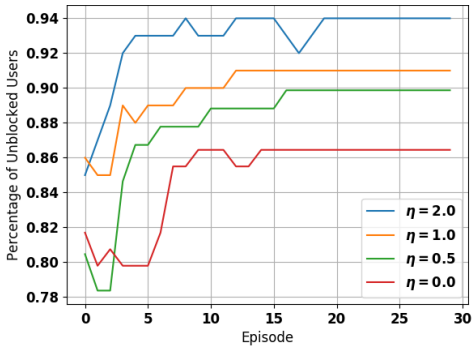


FIGURE 7. Effect of Uncovered Users Penalty (η) on Percentage of Covered Users (Simple Scenario)

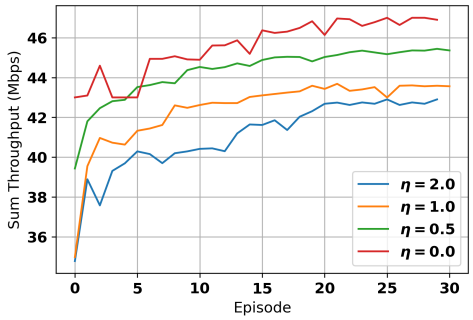


FIGURE 8. Effect of Uncovered Users Penalty (η) on Sum Throughput (Simple Scenario)

ON as a default setting. The presented values of μ include two extreme cases ($\mu = 0$) and ($\mu = 5$). When $\mu = 0$, the agent has no energy penalty. In this case, the sum throughput is close to the MIMO ON curve. The MIMO feature is turned ON 97% of time for the simple scenario and 70% for the real scenario. When $\mu = 5$, the sum throughput approaches the SISO curve. The MIMO feature is turned ON 3% and 1% of time for the simple and the real scenarios, respectively.

To study the effect of the uncovered users' penalty, we vary the hyper-parameter that scales it (η) and plot the percentage of covered users in the network and the sum throughput in Fig. 7 and Fig. 8 for the simple scenario. We obtain the same results for the real scenario in Fig. 9 and Fig. 10. We notice here that there is a trade-off between the sum throughput and the percentage of covered users. As (η) increases, the percentage of covered users increases while the sum throughput decreases. This is because achieving a higher sum throughput might lead to blocking users with low CQIs to other cells. More specifically, since low CQIs users consume cell resources and do not contribute much to the sum throughput, the agent may opt to block them and give more resources to users with high CQIs (which will benefit the sum throughput of the network).

For a more elaborate understanding of this work, we plot the reward function that the RL agent tries to maximize (which we call the penalized sum throughput) in Fig. 11 for the real scenario with $\mu = 2$ and $\eta = 2$. We compare the penalized throughput reward with the RL agent and BL setting. We observe that the reward function of the proposed RL algorithm is the highest, which is the main target of this

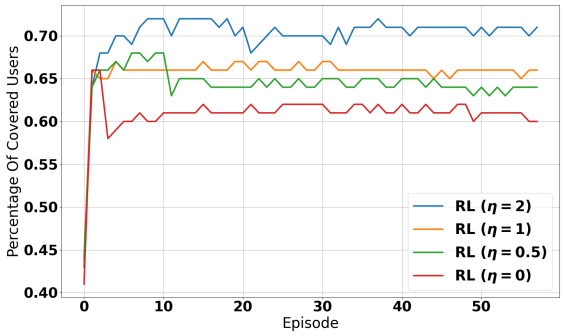


FIGURE 9. Effect of Uncovered Users Penalty (η) on Percentage of Covered Users (Real Scenario)

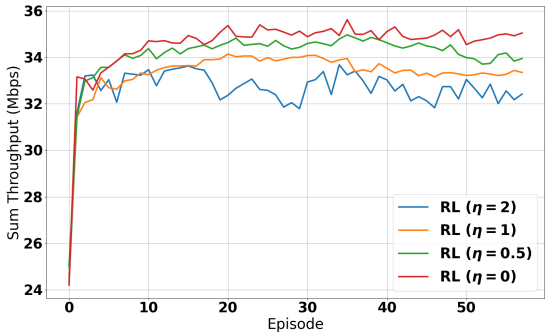


FIGURE 10. Effect of Uncovered Users Penalty (η) on Sum Throughput (Real Scenario)

work. Nevertheless, we see that, although the sum throughput of the BL with MIMO ON is naturally higher than that with MIMO OFF as seen in Fig. 5, the reward follows a contradictory behavior. This reward behavior is because the MIMO penalty, which is subtracted, affects the overall reward in case MIMO is turned ON by default, while it is zero in case MIMO is turned OFF.

After training, we tested the fully trained model for one episode (250 steps) to evaluate its performance. In Fig. C, we show the network sum throughput versus the number of steps for $\eta = 2$ and $\mu = 0, 2$ and 5, respectively.

As expected, for smaller μ's, the agent tends to achieve higher throughput by having MIMO ON with less regard to the consumed energy. On the other hand, as μ increases, the agent starts to turn MIMO OFF for energy conservation to maximize the overall reward; this generally results in a decrease in the achieved network throughput. Nevertheless, in the three cases reported in Fig. Fig. C, we observe that the average sum throughput achieved by the RL agent is higher than the BL model (with MIMO ON or OFF); however, the throughput gain achieved is affected by the value of μ as expected.

From another perspective, we conducted additional experiments to assess how well the model performs under varying environmental conditions, such as changes in the number of users. In Fig. 13, we compare the performance

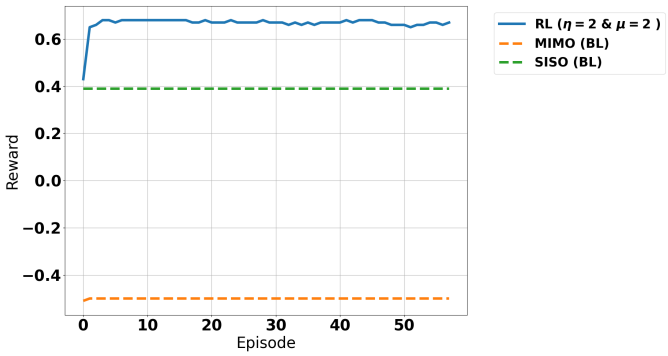


FIGURE 11. Penalized Sum Throughput

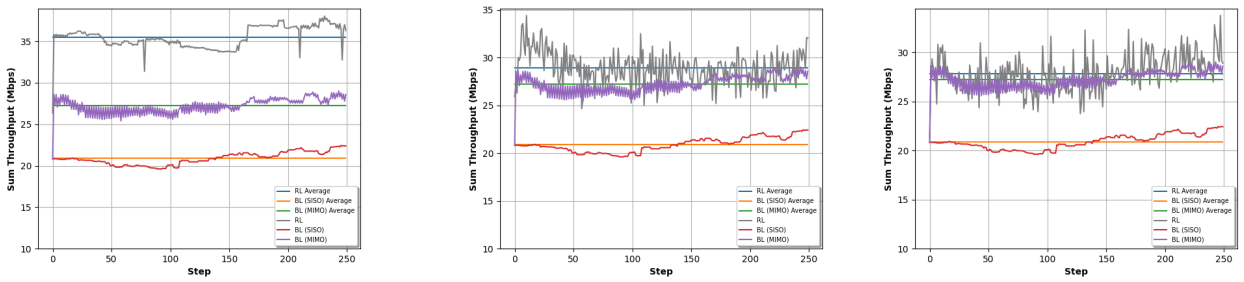
of our hybrid agent, trained with 40 users, when tested in a setting with 40 users against its performance when tested with 50 users. The MIMO Energy penalty (μ) is set to 0, and the uncovered users' penalty (η) is set to 2. Fig. 13 shows that the overall sum throughput increases with the higher user count. Notably, our agent consistently achieves higher throughput compared to the baseline scenario despite environmental changes. Additionally, Fig. 14 illustrates the penalized throughput, or reward, for the same scenario.

It is important to note that each step represents 0.2 seconds, meaning that even with significant environmental shifts requiring the agent to explore new actions, convergence may occur within approximately 5000 steps, equivalent to about 16 minutes—a relatively short duration in the context of cellular networks.

Moreover, we are exploring the potential of incorporating online learning into our future research. This approach would allow the agent to continue learning beyond the initial offline training phase, enhancing confidence in its decision-making over time. However, considerations regarding memory and computational resources must be carefully addressed.

VI. Conclusion

In this paper, we propose a novel *layered* RL agent to attain the aim of having a self-optimizing mobile network. The agent aims to strike a balance between maximizing the network throughput, minimizing energy consumption, and enhancing the network coverage. Our proposed framework is general enough to allow mobile operators to select their preferred network optimization cost. Moreover, the layered architecture of our novel RL agent addresses the fact that some of our control parameters are discrete, and others are continuous. The proposed architecture provides an efficient, yet effective means of addressing the hybrid nature of our action space. We tested our proposed RL agent over two scenarios; a simple (proof of concept) scenario and a realistic network scenario that matches the configuration of one of the biggest mobile operators in Egypt in the Fifth Settlement neighborhood in Cairo. Our results show some



(a) $\mu = 0$ (b) $\mu = 2$ (c) $\mu = 5$

FIGURE 12. **Sum throughput for a testing episode with $\eta = 2$.**

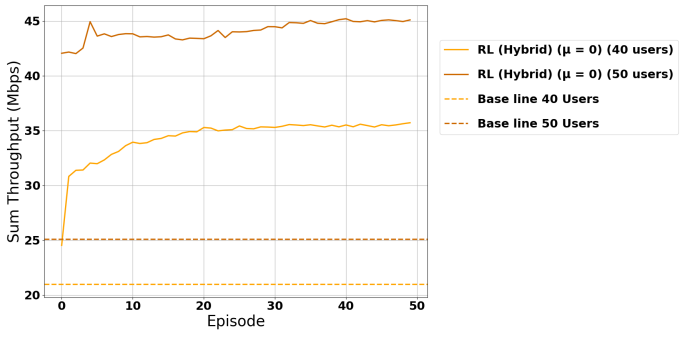


FIGURE 13. **Sum Throughput in the testing phase with environment change**

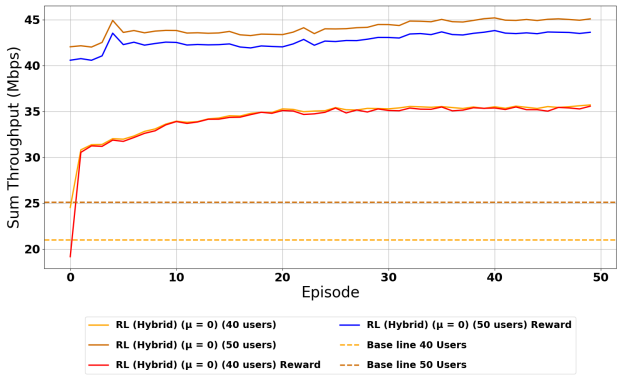


FIGURE 14. **Reward in the testing phase with environment change**

significant performance gains over the baseline approaches. Also, we showed that by controlling the parameters reward function, mobile operators could optimize diverse network performance metrics (e.g., throughput, energy efficiency, and coverage). As a future direction of this work, we aim to investigate replacing the binary vector $\mathbf{m}$, which represents the MIMO activity vector, with an integer vector that represents the number of active antennas in a certain cell. This will give a more flexible tradeoff between the performance and the energy consumption as we will not be forced to switch ON or switch OFF all antennas of the base stations completely.

REFERENCES

[1] Ericsson. Ericsson mobility report q2 2022. [Online]. Available: https://www.ericsson.com/4a4be7/assets/local/reports-papers/mobility-report/documents/2022/ericsson-mobility-report-q2-2022.pdf

[2] S. Mishra and N. Mathur, "Load balancing optimization in lte/lte-a cellular networks: a review," *arXiv preprint arXiv:1412.7273*, 2014.

[3] S. Buzzi, I. Chih-Lin, T. E. Klein, H. V. Poor, C. Yang, and A. Zappone, "A survey of energy-efficient techniques for 5g networks and challenges ahead," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 4, pp. 697–709, 2016.

[4] J. A. Khan, H. K. Qureshi, and A. Iqbal, "Energy management in wireless sensor networks: A survey," *Computers & Electrical Engineering*, vol. 41, pp. 159–176, 2015.

[5] 3GPP ETSI TS 36.902 V9.3.1, *3GPP, Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Self-Configuring and Self-Optimizing Network (SON) Use Cases and Solutions*, 2011.

[6] Ericsson. Defining ai native: A key enabler for advanced intelligent telecom networks. [Online]. Available: https://www.ericsson.com/4a4be7/assets/local/reports-papers/mobility-report/documents/2022/ericsson-mobility-report-q2-2022.pdf

[7] J. Hoydis, F. A. Aoudia, A. A. Valcarce, and H. Viswanathan, "Toward a 6g ai-native air interface," *IEEE Communications Magazine*, vol. 59, no. 5, pp. 76–81, 2021.

[8] J. Guo, C.-K. Wen, and S. Jin, *AI-Native Air Interface*. Cham: Springer International Publishing, 2024, pp. 143–163.

[9] M. Polese, L. Bonati, S. D'oro, S. Basagni, and T. Melodia, "Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[10] A. Garcia-Saavedra and X. Costa-Pérez, "O-ran: Disrupting the virtualized ran ecosystem," *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 96–103, 2021.

[11] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and learning in o-ran for data-driven nextg cellular networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 21–27, 2021.

[12] K. Attiah, K. Banawan, A. Gaber, A. Elezabi, K. Seddik, Y. Gadallah, and K. Abdullah, "Load balancing in cellular networks: A reinforcement learning approach," in *Consumer Communications & Networking Conference (CCNC)*. IEEE, 2020, pp. 1–6.

[13] G. Alsuhli, K. Banawan, K. Attiah, A. Elezabi, K. Seddik, A. Gaber, M. Zaki, and Y. Gadallah, "Mobility load management in cellular networks: A deep reinforcement learning approach," *Accepted for publication in IEEE Transactions on Mobile Computing*, 2021.

[14] Y. Xu, W. Xu, Z. Wang, J. Lin, and S. Cui, "Load balancing for ultradense networks: A deep reinforcement learning-based approach," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9399–9412, 2019.

[15] S. Musleh, M. Ismail, and R. Nordin, "Load balancing models based on reinforcement learning for self-optimized macro-femto lte-advanced

heterogeneous network," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 1, pp. 47–54, 2017.
[16] H. Zhang, X.-s. Qiu, L.-m. Meng, and X.-d. Zhang, "Achieving distributed load balancing in self-organizing lte radio access network with autonomic network management," in *2010 IEEE Globecom Workshops*. IEEE, 2010, pp. 454–459.
[17] A. Mukherjee, D. De, and P. Deb, "Power consumption model of sector breathing based congestion control in mobile network," *Digital Communications and Networks*, vol. 4, no. 3, pp. 217–233, 2018.
[18] G. Alsuhli, K. Banawan, K. Seddik, and A. Elezabi, "Optimized power and cell individual offset for cellular load balancing via reinforcement learning," in *IEEE WCNC*, 2021, pp. 1–7.
[19] G. Alsuhli, H. A. Ismail, K. Alansary, M. Rumman, M. Mohamed, and K. G. Seddik, "Deep reinforcement learning-based cio and energy control for lte mobility load balancing," in *Consumer Communications & Networking Conference (CCNC)*. IEEE, 2021, pp. 1–6.
[20] G. Alsuhli, K. Banawan, K. Seddik, and A. Elezabi, "Optimized power and cell individual offset for cellular load balancing via reinforcement learning."
[21] M. Aboelwafa, G. Alsuhli, K. Banawan, and K. G. Seddik, "Self-optimization of cellular networks using deep reinforcement learning with hybrid action space," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2022, pp. 223–229.
[22] E. Oh, B. Krishnamachari, X. Liu, and Z. Niu, "Toward dynamic energy-efficient operation of cellular network infrastructure," *IEEE Communications Magazine*, vol. 49, no. 6, pp. 56–61, 2011.
[23] Y. Cui, S. Xiao, X. Wang, Z. Lai, Z. Yang, M. Li, and H. Wang, "Performance-aware energy optimization on mobile devices in cellular network," *IEEE Transactions on Mobile Computing*, vol. 16, no. 4, pp. 1073–1089, 2016.
[24] M. Aboelwafa, M. Zaki, A. Gaber, K. Seddik, Y. Gadallah, and A. Elezabi, "Machine learning-based mimo enabling techniques for energy optimization in cellular networks," in *Consumer Communications & Networking Conference (CCNC)*. IEEE, 2020, pp. 1–6.
[25] A. Anzaldo and A. G. Andrade, "Buffer transference strategy for power control in b5g-ultra-dense wireless cellular networks," *Wireless Networks*, pp. 1–8, 2022.
[26] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.
[27] S. Chinchali, P. Hu, T. Chu, M. Sharma, M. Bansal, R. Misra, M. Pavone, and S. Katti, "Cellular network traffic scheduling with deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
[28] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6255–6267, 2020.
[29] Y. Hao, F. Li, C. Zhao, and S. Yang, "Delay-oriented scheduling in 5g downlink wireless networks based on reinforcement learning with partial observations," *IEEE/ACM Transactions on Networking*, 2022.
[30] P. Lin, Q. Song, J. Song, A. Jamalipour, and F. R. Yu, "Cooperative caching and transmission in comp-integrated cellular networks using reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5508–5520, 2020.
[31] O. Delalleau, M. Peter, E. Alonso, and A. Logut, "Discrete and continuous action representation for practical rl in video games," *arXiv preprint arXiv:1912.11077*, 2019.
[32] M. Neunert, A. Abdolmaleki, M. Wulfmeier, T. Lampe, T. Springenberg, R. Hafner, F. Romano, J. Buchli, N. Heess, and M. Riedmiller, "Continuous-discrete reinforcement learning for hybrid control in robotics," in *Conference on Robot Learning*. PMLR, 2020, pp. 735–751.
[33] H. Fu, H. Tang, J. Hao, Z. Lei, Y. Chen, and C. Fan, "Deep multi-agent reinforcement learning with discrete-continuous hybrid action spaces," *arXiv preprint arXiv:1903.04959*, 2019.
[34] C. Huang, H. Zhang, L. Wang, X. Luo, and Y. Song, "Mixed deep reinforcement learning considering discrete-continuous hybrid action space for smart home energy management," *Journal of Modern Power Systems and Clean Energy*, vol. 10, no. 3, pp. 743–754, 2022.
[35] M. Diamanti, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Unified user association and contract-theoretic resource orchestration in noma heterogeneous wireless networks," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 1485–1502, 2020.
[36] F. B. Mismar, J. Choi, and B. L. Evans, "A framework for automated cellular network tuning with reinforcement learning," *IEEE Transactions on Communications*, vol. 67, no. 10, pp. 7152–7167, 2019.
[37] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[38] Q. Zhang, T. Du, and C. Tian, "A sim2real method based on ddqn for training a self-driving scale car," *Mathematical Foundations of Computing*, vol. 2, no. 4, p. 315, 2019.
[39] J. Zhu, F. Wu, and J. Zhao, "An overview of the action space for deep reinforcement learning," in *2021 4th International Conference on Algorithms, Computing and Artificial Intelligence*, 2021, pp. 1–10.
[40] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.
[41] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour *et al.*, "Policy gradient methods for reinforcement learning with function approximation." in *NIPs*, vol. 99. Citeseer, 1999, pp. 1057–1063.
[42] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[43] V. François-Lavet, P. Henderson, R. Islam, M. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018.
[44] S. Adam, L. Busoniu, and R. Babuska, "Experience replay for real-time reinforcement learning control," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 2, pp. 201–212, 2012.
[45] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE transactions on cybernetics*, vol. 50, no. 9, pp. 3826–3839, 2020.
[46] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.
[47] S. Fujimoto, H. V. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.
[48] R. Dubey, R. Loka, and A. M. Parimi, "Maintaining the frequency of ai-based power system model using twin delayed ddpg (td3) implementation," in *2022 2nd International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC)*. IEEE, 2022, pp. 1–4.
[49] N. Baard and T. L. van Zyl, "Twin-delayed deep deterministic policy gradient algorithm for portfolio selection," in *2022 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr)*. IEEE, 2022, pp. 1–8.
[50] M. Shehab, A. Zaghloul, and A. El-Badawy, "Low-level control of a quadrotor using twin delayed deep deterministic policy gradient (td3)," in *2021 18th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*. IEEE, 2021, pp. 1–6.
[51] S. Bai, S. Song, S. Liang, J. Wang, B. Li, and E. Neretin, "Uav maneuvering decision-making algorithm based on twin delayed deep deterministic policy gradient algorithm," *Journal of Artificial Intelligence and Technology*, vol. 2, no. 1, pp. 16–22, 2022.
[52] Z. E. Liu, Q. Zhou, Y. Li, and S. Shuai, "An intelligent energy management strategy for hybrid vehicle with irrational actions using twin delayed deep deterministic policy gradient," *IFAC-PapersOnLine*, vol. 54, no. 10, pp. 546–551, 2021.
[53] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
[54] E. LTE, "Evolved universal terrestrial radio access (e-utra); physical layer procedures," *ETSI TS*, pp. 136–213, 2017.
[55] E. TSGR, "Lte: Evolved universal terrestrial radio access (e-utra)," *Multiplexing and channel coding (3GPP TS 36.212 version 10.3. 0 Release 10) ETSI TS*, vol. 136, no. 212, p. V10, 2011.
[56] ETSI, "5G NR, radio resource control (RRC), protocol specification, 3GPP TS 38.331 version 15.3.0 release 15," 2018.

[57] N. Baldo, M. Miozzo, M. Requena-Esteso, and J. Nin-Guerrero, "An open source product-oriented lte network simulator based on ns-3," in *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, 2011, pp. 293–298.

[58] P. Gawłowicz and A. Zubow, "ns3-gym: Extending openai gym for networking research," *arXiv preprint arXiv:1810.03943*, 2018.

[59] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol *et al.*, "Stable baselines," 2018.

[60] D. Krajzewicz and C. Rossel, "Simulation of urban mobility (sumo)," *Centre for Applied Informatics (ZAIK) and the Institute of Transport Research at the German Aerospace Centre*, 2007.

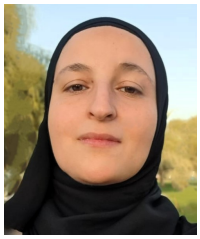
[61] A. Marella, A. Bonfanti, G. Bortolasor, and D. Herman, "Implementing innovative traffic simulation models with aerial traffic survey," *Transport infrastructure and systems*, pp. 571–577, 2017.

Authors

Bishoy S. Attia received the B.Sc. (Hons.) in Electronics and Communications Engineering from The American University in Cairo (AUC). Currently pursuing MASc. degree in Computer Science at Simon Fraser University (SFU). He is currently a Research Assistant and Teaching assistant at School of Computing science, SFU. Before joining SFU, he was research Assistant AUC. His research interests include applications of Reinforcement learning in Computer architecture, Memory architecture, processing in/near memory, secure cache and memory systems.

Aamen Elgharably received the B.Sc. in Electronics and Communications engineering from Alexandria University, Alexandria Egypt in 2023. He is currently working as RAN Planning Engineer at _VOIS, additionally he is a research assistant in American University in Cairo (AUC). His research interests include Open RAN, RAN optimization scalability and Reinforcement learning.

Mariam Nabil received the B.Sc. (Hons.) and M.Sc. degrees in electrical engineering from Alexandria University, Alexandria, Egypt, in 2010 and 2015, respectively, and the Ph.D. degree from the American University in Cairo, Cairo, Egypt, in 2022. She is currently an Assistant Professor in the Computer, Communications and Autonomous Systems Engineering program at NewGiza University (NGU). Before joining NGU, she was an Assistant Professor at Alexandria University and a Postdoctoral researcher at AUC. Her research interests include applications of machine learning in communication networks, autonomous vehicles and smart systems. She co-authored numerous papers in reputable journals and conferences. She has also reviewed for a number of esteemed journals.



Ghada Alsuhli earned her B.S. and M.S. in Electronics and Communication Engineering from Damascus University, Syria (2009, 2015), and completed her Ph.D. in Electronics and Communication Engineering at Cairo University, Egypt (2019). Her academic journey was enriched by roles at esteemed research centers including the National Research Center, The American University in Cairo, Egypt, and the Khalifa University SoC Center, UAE. Currently, she serves as a Post-Doctoral Researcher at Khalifa University, leading several projects focused on efficient hardware implementation for AI and post-quantum cryptography. Her research encompasses embedded systems, energy-efficient IoT solutions, edge computing, efficient hardware implementation, and AI applications for wireless communications and biomedical engineering. She is the primary author of numerous papers in reputable journals and conferences, she has also authored a book on efficient DNN hardware implementation. Her contributions extend to reviewing for esteemed journals and committee memberships for international conferences.

Karim Banawan (IEEE STM'13, M'18, SM'24) received the B.Sc. and M.Sc. degrees, with highest honors, in electrical engineering from Alexandria University, Alexandria, Egypt, in 2008, and 2012, respectively, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Maryland at College Park, MD, USA, in 2017 and 2018, respectively, with his Ph.D. thesis on private information retrieval and security in networks. He was the recipient of the Distinguished Dissertation Fellowship from the Department of Electrical and Computer Engineering, at the University of Maryland College Park, for his Ph.D. thesis work. In 2019, he joined the department of electrical engineering, Alexandria University, as an assistant professor. His research interests include information theory, wireless communications, physical layer security, and private information retrieval.

Karim G. Seddik (Senior Member, IEEE) received the B.Sc. (Hons.) and M.Sc. degrees in electrical engineering from Alexandria University, Alexandria, Egypt, in 2001 and 2004, respectively, and the Ph.D. degree from the University of Maryland, College Park, MD,

USA, in 2008. He is currently a Professor in the Electronics and Communications Engineering Department at the American University in Cairo (AUC), and an Associate Dean of graduate studies and research with the School of Sciences and Engineering (SSE), AUC. Before joining AUC, he was an Assistant Professor at Alexandria University. His research interests include applications of machine learning in communication networks, intelligent reflecting surfaces, age of information, cognitive radio communications, and layered channel coding. He currently serves as an Editor for the IEEE Transactions on Machine Learning in Communications and Networking journal. He has served on the technical program committees for numerous IEEE conferences in the areas of wireless networks and mobile computing. He was a recipient of the American University in Cairo Faculty Merit Award for Excellence in Research and Creative Endeavors in 2021. He was a recipient of the State Encouragement Award in 2016 and the State Medal of Excellence in 2017. He was a recipient of the Certificate of Honor from the Egyptian President for being ranked first among all departments in the College of Engineering, Alexandria University, in 2002. He received the Graduate School Fellowship in 2004 and 2005 and the Future Faculty Program Fellowship in 2017 from the University of Maryland. He also co-authored a conference paper that received the Best Conference Paper Award from the IEEE Communication Society Technical Committee on Green Communications and Computing in 2019.