IEEE ComSoc IEEE Communications Society | IEEE COMPUTER SOCIETY | IEEE Signal Processing Society | VTS | IEEE Transactions on Machine Learning in Communications and Networking

Received 17 October 2023; revised 11 March 2024; accepted 15 April 2024.
Date of publication 18 April 2024; date of current version 2 May 2024.

The associate editor coordinating the review of this article and approving it for publication was A. Savard.

Digital Object Identifier 10.1109/TMLCN.2024.3391216

Multi-Agent Double Deep Q-Learning for Fairness in Multiple-Access Underlay Cognitive Radio Networks

ZAIN ALI, ZOUHEIR REZKI (Senior Member, IEEE), AND HAMID SADJADPOUR (Senior Member, IEEE)

Electrical and Computer Engineering Department, Baskin School of Engineering, University of California at Santa Cruz, Santa Cruz, CA 95064 USA

CORRESPONDING AUTHOR: Z. ALI (zainalihanan1@gmail.com)

The work of Zouheir Rezki was supported by the National Science Foundation, [Faculty Early Career Development Program (CAREER)] under Grant 2114779.

ABSTRACT Underlay Cognitive Radio (CR) systems were introduced to resolve the issue of spectrum scarcity in wireless communication. In CR systems, an unlicensed Secondary Transmitter (ST) shares the channel with a licensed Primary Transmitter (PT). Spectral efficiency of the CR systems can be further increased if multiple STs share the same channel. In underlay CR systems, the STs are required to keep interference at a low level to avoid outage at the primary system. The restriction on interference in underlay CR prevents some STs from transmitting while other STs may achieve high data rates, thus making the underlay CR network unfair. In this work, we consider the problem of achieving fairness in the rates of the STs. The considered optimization problem is non-convex in nature. The conventional iteration-based optimizers are time-consuming and may not converge when the considered problem is non-convex. To deal with the problem, we propose a deep-Q reinforcement learning (DQ-RL) framework that employs two separate deep neural networks for the computation and estimation of the Q-values which provides a fast solution and is robust to channel dynamic. The proposed technique achieves near optimal values of fairness while offering primary outage probability of less than 4%. Further, increasing the number of STs results in a linear increase in the computational complexity of the proposed framework. A comparison of several variants of the proposed scheme with the optimal solution is also presented. Finally, we present a novel cumulative reward framework and discuss how the combined-reward approach improves the performance of the communication system.

INDEX TERMS Cognitive radio, distributed double deep-Q learning, fair rate maximization, multi-agent training, resource optimization.

I. INTRODUCTION

COGNITIVE Radio (CR) technology is a powerful tool that improves the spectral efficiency of the communication networks by enabling the Secondary Users (SUs) to share the spectrum of the Primary Users (PUs) [1]. In these CR systems, the channel of a Primary Transmitter (PT) is shared by a single Secondary Transmitter (ST). In traditional CR systems when the number of SUs increases, many STs could be denied service due to unavailability of frequency channels. Therefore, a CR system that can serve many SUs per channel could improve the quality of service (QoS). However, unlike the single SU case where the problem of power allocation is convex and can be solved using conventional optimization frameworks [2], when multiple SUs per channel are considered, the problem of optimizing resource allocation becomes non-convex in nature. Classical optimization models cannot be applied to such problems, as convergence cannot be guaranteed. Moreover, the conventional iteration-based optimizers take a long time to compute the solutions [3]. The problem of optimizing power allocation at the STs when several STs utilize the same channel is of dire importance to ripe the full benefits of CR systems.

A. MOTIVATION

Research has shown that machine learning-based frameworks can provide faster solutions than conventional techniques [4]. Although machine learning-based schemes provide excellent

performance [5]. Nevertheless, classical supervised machine learning techniques need a substantial volume of training data [6]. The problem of generating training data becomes more severe in the case of non-convex problems, as no framework is guaranteed to provide the optimal solution within reasonable time, and usually a lot of time is required to generate the training data. In addition, if the system parameters change, re-training the model for the new parameters is necessary, and this requires reproducing the training data set [7]. To regenerate the training data, a lot of processing power is required, to the point that the solution becomes impractical, specially in the case of distributed systems. Recently, various reinforcement learning (RL) schemes have been demonstrated to offer quick results without requiring additional methods for producing the training data [8]. The RL techniques involve agents that are trained by taking actions and getting rewards as a result of those actions. The reward serves as a measure of how effective the action was in achieving a particular goal. The RL agent learns the relationship between the input state and the optimal action. Hence, the overhead of generating large training data-set is removed. Different improvements to simple RL models have been proposed. For instance: deep reinforcement learning (DRL) where a Deep Neural Network (DNN) is used to provide the estimated state-to-action mapping [9], or deep Q-learning where a Q-value function is employed for learning [10], have been shown to further enhance the performance of RL models.

B. RELATED WORK

Many works have considered RL solutions for solving optimization problems. The work in [12] proposed DRL framework for interference control in ultra-dense small cells. In [13], the authors proposed a DRL-based scheme for edge computing against jamming attacks. The authors in [11] proposed DRL-based solution for power allocation to maximize the sum-rate of the cellular networks. In [14], the problem of optimizing power allocation for rate maximization in heterogeneous networks was considered. Therein, the authors provided a distributed DRL framework for the solution where a centralized node is required for the training. For overlay CR system, the work in [15] proposed a deep Q-learning-based solution where the STs opportunistically access the spectrum if the PT is absent. The optimization was performed to minimize the detection error. Later, taking into account the problem of cooperative sensing in CR networks, the authors in [16] proposed a Q-learning framework. Considering an underlay CR system, the work in [17] proposed a Q-learning-based framework for power allocation at ST such that the ST can simultaneously transmit with the PT while avoiding the outage at the primary system. The results showed that the proposed framework provides a good performance, however the solution requires multiple sensors scattered over the network to feedback the received power levels to the ST. Further, the proposed solution only considers a simple scenario where only one ST can transmit at a channel. To maximize the energy efficiency of CR network the authors in [18] designed a Q-learning-based solution technique. The proposed scheme provided good performance for the channel gains used in training; however, when the channels change retraining of the model may be required to achieve good performance. The solution proposed in [19] for spectrum access is based on a DRL framework that is robust to changes in channel gains. Nevertheless, the proposed approach mandates centralized training for a distributed system, necessitating the transmission of thousands of DNN weights to all users once the training phase is finished.

The issue of attaining fairness across various communication systems has been widely explored in the literature. The work in [7] optimized power allocation for achieving fairness, the authors maximize the sum of log-average rates which is a loose fairness objective and many times does not provide perfect fairness even in the systems where perfect fairness is achievable. Further, the performance of the proposed scheme suffers a lot when tested for random channels. In [20], considering the problem of achieving fairness in orthogonal frequency division multiple access networks the authors proposed an iteration based optimization framework. To achieve fairness in non-orthogonal multiple access systems, the authors in [21] designed an alternate-optimization algorithm for power allocation. In situations where various STs share the same channel in interference limited systems, the issue of unfairness becomes more severe. In such systems, in order to maximize the system rate and/or to avoid outage, it may become necessary to prevent many STs from transmitting. A user that is allocated a channel but is not allowed to transmit gains no benefit from the channel which makes a multi-user CR system unfair in terms of achievable rates of the STs. In order to address the issue of fairness in CR systems, a centralized resource allocation framework was proposed by the authors in [22]. The framework aimed to improve the fairness in rates of the STs. In the system considered in [22], the STs are not subject to interference from other STs, resulting in a problem convex in nature. The authors proposed a Lagrangian-dual-based solution, but it is not guaranteed to converge when multiple STs are assigned to the same channel since the problem becomes non-convex. Another approach to achieve fairness in wireless powered CR systems was presented by the authors in [23]. Similar to [22], the authors considered a system where each SU is assigned a separate channel for transmission. Thus, the SUs do not face any interference from the other SUs. The authors in [24] aimed to achieve rate fairness among multiple STs sharing the same channel by optimizing power allocation. For solving the non-convex problem, the authors presented a sequential-quadratic-programming-based (SQP) approach. However, SQP methods have a slow convergence rate. Although iterative techniques such as those proposed in [20], [21], [22], [23], and [24] offer potential solutions, they are time-consuming and require re-optimization when

the channel gains change. In the cases where the coherence time of the channels is small, these iterative frameworks may become useless.

C. CONTRIBUTIONS

In this work which is a substantial extension of [25],[1] we present a deep-Q reinforcement learning (DQ-RL) framework to address the issue of fairness in the rates of multiple STs sharing a channel in an underlay CR scenario. The proposed method provides a quick solution without requiring the generation of an extensive dataset for training. Additionally, the framework performs well for new channel gain values after training is completed since a DNN is used in DQ-RL to predict the Q-values for each action based on the provided state parameters. Consequently, when channel gains change after training, the DQ-RL agent can use the trained DNN to identify the action that results in the highest reward. The main contributions of the work can be summarized as follows:

- A DQ-RL model is designed to ensure fairness in the system. The proposed framework ensures that fairness in rates of all the STs is achieved while protecting the primary system from harmful interference.
- Two reward functions are proposed to improve the performance of a system. The first one enables simultaneous transmission by every ST in each time slot to ensure low latency. It is designed to maximize the fairness index and the sum-rate of the secondary system for latency sensitive communication scenarios. The second reward function uses a cumulation-based approach that allows STs to alternate between high and low transmission power or take turns to transmit, ensuring high fairness in average ST rates while reducing overall interference.
- In order to enable fast training and reduce correlation between Q-value calculation and estimation, two separate DNNs are used at each agent. For the instantaneous reward framework, fully connected DNNs are utilized. However, for the cumulation-based reward function, LSTM units replace the nodes in the first hidden layers of the DNNs, which helps in handling sequential inputs.
- We propose a multi-agent distributed training model that carries out training at the STs to eliminate the overhead of training and transferring all agents from the SR. This model is robust to channel dynamics and provides fast solutions compared to existing solutions. Different variations of the proposed model are presented, considering various numbers of input features, sequential inputs, and resolution of action space. These variations provide different levels of fairness, system rewards, and computational complexities while requiring varying levels of feedback from the SR.

The main contributions of this work as compared to the present literature are summarized in the table 1.

[1]A summary of the major differences between this work and the conference version is provided in table 2.

D. OUTLINE

The remaining of the paper is organized as follows: section II presents the system model and the mathematical formulation of the problem. The proposed DRL model is discussed in section III, then the solution framework is explained in detail in section IV. Section V presents different variants of the proposed framework. A detailed discussion on the performance and comparison of the different variants based on the simulation results and computational complexities is provided in section VI. Finally, the work is concluded in section VII.

II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider an underlay CR system where a frequency channel is shared by K STs, simultaneously. The gains of the channel from kth ST to primary receiver (PR) and from the kth ST to the SR are represented as f_k and h_k, respectively. Considering that the SR uses single-user decoding and treats interference as noise, the rate of the kth ST is written as:

$$R_k = \log_2\left(1 + \frac{p_k h_k}{\sum_{\substack{j=1 \\ k\neq k}}^{K} p_j h_j + I_{pn} + \sigma^2}\right), \tag{1}$$

where, p_k is the transmission power of the kth ST, σ^2 denotes the variance of additive white Gaussian noise (AWGN) and I_{pn} represents the interference from the primary network to the SR. The objective of our work is to improve the rate fairness among all the STs in the system by optimizing their power allocation. To measure the degree of fairness in rates, we utilize the linear product-based fairness index θ, calculated as [26]:

$$\theta = \prod_{k=1}^{K} \frac{R_k}{\max_j R_j}. \tag{2}$$

Note that the value of θ is always bounded between 0 and 1. If the rate of one or more STs is zero, the value of θ is also zero, indicating that the system is unfair. Conversely, when $\theta = 1$, it implies that each user in the system has an equal rate. Thus, higher value of θ signifies better fairness in the rates among all STs.

Now consider a scenario where no user is transmitting/communicating, although we know that the system has no practical usage, the system is fair in terms of achievable rate because the rate of each user is exactly the same (i.e., zero). Thus, maximizing the fairness in rates of all the users has an intrinsic issue. Therefore, we have the condition that $\theta = 0$ if $\max_j R_j = 0$. Similarly, if all STs are achieving very low rates, the value of fairness index can still be high. However, such a system is very inefficient due to the under-utilization of resources and because a greater value of the sum-rate could have been achieved. Thus, only maximizing the fairness of the system can lead to some problems, a more effective objective is to jointly maximize the fairness index and the sum-rate of the system (the objective considered

TABLE 1. Summary of the literature review. Our contribution is highlighted and distinguished from existing work in the last row.

	Considered Fairness	Supports Variable Length Input Sequence	Fast Solution	Robust to Channel Dynamic	Multiple STs on Channel	Distributed Training/Solution	Robust to Change in System Parameters
[7]	✓	✗	✗	✓	Non CR system	✗	✗
[11], [14], [15]	✗	✗	✓	✓	Non CR system	✗	✗
[16], [19]	✗	✗	✓	✓	✗	✓	✗
[17]	✗	✗	✓	✓	✗	✓	✗
[18]	✗	✗	✗	✗	✓	✓	✗
[20], [21]	✓	✗	✗	✗	Non CR system	✗	✗
[22], [23]	✓	✗	✗	✗	✗	✗	✗
[24]	✓	✗	✗	✗	✓	✗	✗
[25]	✓	✗	✓	✓	✓	✓	✗
This work	✓	✓	✓	✓	✓	✓	✓

TABLE 2. Summary of the major differences between this work and the conference version.

	Supports flexible length of input sequence	Supports variable interference threshold of primary system without retraining	Applicable when limited state information is available	Study the impact of varying system parameters
[25]	✗	✗	✗	✗
This work	✓	✓	✓	✓

in this work). The sum-rate of the system is computed as follows:

$$\kappa = \sum_{k=1}^{K} R_k. \tag{3}$$

The problem to achieve fairness in the system while maximizing the sum-rate can be expressed mathematically as:

$$\max_{p_k} \kappa\, \theta \tag{4}$$

$$\text{s.t.: } p_k \leq P_k^{th}, \quad \forall k, \tag{5}$$

$$\sum_{k=1}^{K} p_k f_k \leq I_{th}, \tag{6}$$

Here, P_k^{th} and I_{th} represent the battery capacity of the kth ST and the interference threshold required to prevent primary system outage, respectively. The constraint in (5) ensures that each user follows its battery capacity, while (6) safeguards the primary network against STs' interference.

III. DEEP Q-LEARNING-BASED OPTIMIZATION

The optimization problem (4)-(6) discussed above is inherently non-convex. Using conventional optimization frameworks to solve it may result in abysmal performance since convergence is not guaranteed. However, for non-convex problems, it has been demonstrated that RL can offer excellent results [27], [28].

The RL agent (ST) observes the system parameters, takes actions (power allocation), and receives rewards from the environment. Its goal is to maximize rewards, learning optimal actions for diverse states. Vanilla RL excels in finite state-sets, but our problem involves real-number parameters (h_i,f_i, interference). Creating a state-to-action table is impractical, leading to the use of a DRL-based framework. DRL employs a DNN for state-to-action mapping, eliminating the need for pre-training data [4]. Real-time generation of training data by RL agents is termed online learning [29].

Learning the state-action mapping is challenging due to the unknown probability of state transitions after an action. Addressing this, we adopt Q-learning, estimating the expected reward (Q-value) for taking an action in a state. In our DRL framework, the DNN provides Q-value estimates for all possible actions. The agent selects the action with the maximum Q-value as the solution. The Q-value for a state-action pair is computed as:

$$Q(s, a) = \Gamma(s, a) + \gamma \max_{\bar{a}} Q(\bar{s}, \bar{a}), \tag{7}$$

here, $\Gamma(s, a)$ is the immediate reward obtained by taking action a in state s, and $\gamma \max_{\bar{a}} Q(\bar{s}, \bar{a})$ represents the discounted long-term reward of taking action a, where $\bar{s}$ is the next state after the agent takes action a in state s, γ is the discount factor (with $0 \leq \gamma \leq 1$), and $\bar{a}$ is the action that leads to the maximum reward value. To perform Q-learning efficiently, we use deep Q-learning, which utilizes two distinct deep neural networks (DNNs) to predict the values of $Q(s, a)$ and $Q(\bar{s}, \bar{a})$. These are known as the policy DNN and the target DNN, respectively[2] [30]. In this work,

[2]Using two separate DNNs reduces the correlation and results in a fast convergence [31].

we designate the policy DNN as DNN_1 and the target DNN as DNN_2. The DNN_1 is used to choose an action and DNN_2 provides the estimated Q-value associated to the action. Then, after some iterations, the weights of the DNN_2 are replaced by the weights of DNN_1. Once the training phase is finished, we do not need the values of $Q(\bar{s}, \bar{a})$ as the agent only requires DNN_1 to make the decisions.

The training of a DQ-RL agent involves two main phases: *exploration* and *exploitation*. In the *exploration* phase, the agent randomly selects actions, receiving rewards from the environment. During *exploitation*, the agent aims to take optimal actions based on learned experiences. We adopt the ϵ-greedy strategy, initializing epsilon to 1. The agent randomly selects actions if a random value is less than epsilon (*exploration*); otherwise, it selects actions based on the DNN's output (*exploitation*). In each iteration, epsilon decreases by a fixed factor, shifting from *exploration* to *exploitation*. After taking an action, the agent stores the reward, state, and action in a tuple. Experiences are added to a buffer, and after 'G' iterations, a sample of experiences is randomly selected. Q-values for these experiences train DNN (DNN_1).

To compute the Q-value we need the current reward and the maximum Q-value of the next state. To calculate $\max_{\bar{a}} Q(\bar{s}, \bar{a})$, we use a separate DNN (DNN_2), as discussed earlier. The next state parameters are provided to DNN2, which produces the estimated Q-value of each action at the output. The maximum Q-value is then selected as the value of $\max_{\bar{a}} Q(\bar{s}, \bar{a})$. Once Q-values are computed, the current state serves as input to the main DNN_1 during the *exploitation* phase. DNN_1 generates Q-values, and the estimation error is calculated using the mean-square error function. Then, DNN_1 weights are updated through back-propagation of the error. After 'F' iterations, DNN_2 weights are replaced by DNN_1 weights to enhance DNN_2 estimations. Using different DNNs and weight replacement has demonstrated improved training and faster convergence [30], [31].

IV. PROPOSED SOLUTION

We propose a multi-agent DQ-RL-based technique. We provide a solution technique where a separate agent is trained at each ST. The distributed learning method has the advantage of removing the overhead of transferring the values of weights from the SR to all the ST in each time slot (in the case of centralized learning). In DRL, the action space comprises the possible set of actions that the agent can take. For instance, in the case of power allocation, considering the battery capacity of a ST is 2W, and that the resolution of the action space is 5, the agent can choose from 5 discrete power values to allocate. Then, the action set is given by, e.g., $\mathcal{A} = \{0, 0.5, 1, 1.5, 2\}$. Since there are 5 possible actions, the output-layer of the DNN will have 5 nodes, therefore, the possible solution values are: {1, 0, 0, 0, 0}, {0, 1, 0, 0, 0}, {0, 0, 1, 0, 0}, {0, 0, 0, 1, 0} and, {0, 0, 0, 0, 1} referring to 0, 0.5, 1, 1.5 and 2 W power allocation, respectively. Increasing the resolution of the action space may result in enhancing the performance, but at the expense of more training time. Thus, there is clearly a trade-off between the training time and the system performance. Further, a high number of possible actions makes it unlikely that the best action is visited by the agent during the *exploration* phase especially if the number of *exploration* iterations is small. Similarly, the parameters that define the state of the environment also impact the performance. Using more system parameters results in slower training, but could provide better results. In theory, if we consider all the system parameters i.e., all channel gains, all SINRs, threshold values of interference temperature and battery capacity, then the DNN should be able to identify the important features and assign more weight to them, while assigning zero weights to the features that have no impact. Nevertheless, it should be noted that if the process of feature selection is not left to the DNN, the training phase could be completed in far less time.

TABLE 3. Number of layers and nodes in the proposed DNN.

Number of hidden layers	3	Nodes in 1st hidden layer	40
Nodes in 2nd hidden layer	100	Nodes in 3rd hidden layer	50

For the solution, we consider that the system's state-set contains all h_n, the value of interference at the PR, and all SINRs of the STs. Thus, each agent needs to know all the values in the state where the state-set is: $\mathcal{S} = \{h_1, h_2, \ldots, h_K, \sum_{k=1}^{K} p_k f_k, \text{SINR}_1, \text{SINR}_2, \ldots, \text{SINR}_K\}$.[3] The SR provides feedback to each ST, which includes the parameters of the state-set along with the true interference value, supplied by the PR. Adding more parameters such as the gains of the interference channels (f_k), to the state-set could potentially enhance the performance of DQ-RL, however, it is not feasible because the feedback load on the PR would increase to impractical levels. To reduce the feedback bandwidth requirement, we also propose DQ-RL models that require smaller state-sets than $\mathcal{S}$. Further, we also consider a case where the secondary system receives the reward value from SR and a single bit interference feedback from the PR which has the advantage of reducing the bandwidth requirement compared to the case when the actual value of interference is provided to the secondary system. Moreover, when a single bit interference feedback is considered, the secondary system may not need to engage the PR as the SR can listen for the negative acknowledgement (NACK) signal from the PR to PT.

A. DNN MODEL

For DQ-RL, DNN is necessary to estimate the Q-values. The count of nodes in the input and output layers of the DNN always match the number of input features and the resolution of the action space, respectively. The capacity of a

[3]Each DNN receives an input consisting of the state-set $\mathcal{S}$ and p_k, where p_k represents the ST's transmission power in the previous time slot. It is important to note that each ST already knows its own transmission power in the previous time slot, so the value of p_k is already available at each STs.

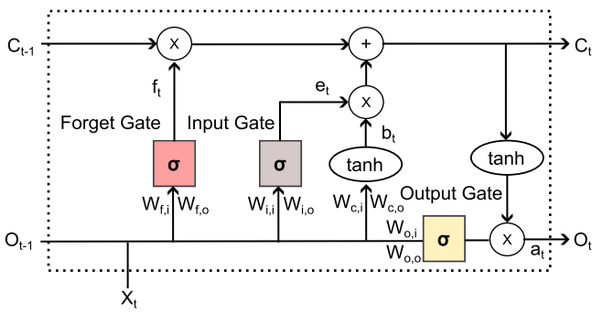

FIGURE 1. **Detailed architecture of the LSTM unit.**

DNN depends on the number of layers it has and the number of nodes present in each layer. A DNN's capacity must be large enough to fit the function of a specific complexity to achieve a decent approximation.[4] The proposed framework employs DNN models where the number of hidden layers and the number of nodes in each layer are provided in table 3. We have considered dense fully connected layers for the DNNs. However, for the case where we consider cumulative reward of more than 1 previous state (discussed in the next section) we replace the neurons in the input layer with LSTM units because LSTM units allow to make decisions based on a sequence of inputs, which the simple dense layers are unable to do.

Recall that LSTM is a type of recurrent neural network (RNN) architecture designed to provide a more robust performance compared to the simpler RNNs, and capture sequential dependencies in data. The detailed architecture of the LSTM unit is shown in Fig. 1, where C_t denotes the cell state, O_t is the output of LSTM unit and X_t is the input to the unit, at time t. Then, $W_{f,i}$, $W_{f,o}$, $W_{i,i}$, $W_{i,o}$, $W_{c,i}$, $W_{c,o}$, $W_{o,i}$ and $W_{o,o}$ are the weights of the LSTM unit. LSTMs consist of memory cells with three gates: input gate, forget gate, and output gate. The forget gate controls the retention or removal of information from the cell's memory, allowing the network to discard irrelevant data. The output of the forget gate is computed as [5], [32]:

$$f_t = \sigma(W_{f,i}X_t + W_{f,o}O_{t-1} + b_f), \tag{8}$$

where $\sigma(\cdot)$ denotes the sigmoid activation function and b_f is a vector of biases. The input gate regulates the information flow into the cell, determining which values to update. At input gate we have the following operations:

$$e_t = \sigma(W_{i,i}X_t + W_{i,o}O_{t-1} + b_i), \tag{9}$$

$$z_t = \tanh(W_{c,i}X_t + W_{c,o}O_{t-1} + b_z), \tag{10}$$

[4]Computing the hyper-parameters of a DNN-based on the complexity of function is still an open research topic. The theory mostly relies on the observations that when a more complex function is considered, the number of hidden layers and/or the number of nodes needs to be increased to achieve a good performance. In this work, we were able to reduce the required number of nodes while achieving good performance through trial and error.

where b_i and b_z denote the vectors of biases. Using these values, the cell state is computed as:

$$C_t = f_t \odot C_{t-1} + e_t \odot z_t, \tag{11}$$

where $\odot$ denotes Hadamard product.

The output gate manages the information output from the cell, determining the final hidden state. The output of the gate also known as hidden state is computed as:

$$a_t = \sigma(W_{o,i}X_t + W_{o,o}O_{t-1} + b_o), \tag{12}$$

where b_o denotes the bias value. Then the output of the LSTM unit is computed as:

$$O_t = a_t \odot \tanh(C_t). \tag{13}$$

These gates collectively enable LSTMs to selectively store, update, and retrieve information from the sequences, making them well-suited for tasks involving temporal dependencies and memory retention in sequential data processing. While simple RNNs and GRUs may offer comparable performance for short sequential inputs, the proposed framework prioritizes flexibility to efficiently handle longer input sequences. Therefore, we have opted for the utilization of LSTMs.

There are two phases in the training of DNN: forward propagation and backward propagation. During forward propagation, the output of a layer is calculated as:

$$\psi^x_{1\times y} = f(\psi^{x-1}_{1\times z}\, W_{z\times y}), \tag{14}$$

the output of layer x with y neurons, denoted as $\psi^x_{1\times y}$, is obtained using the activation function $f(\cdot)$, and the weight matrix $W_{z\times y}$ connecting layer $x-1$, with z neurons. The order of the vector is also specified as $(1 \times y)$. The estimated Q-value of each action for the given state $(Q(s, a))$ is obtained from the output layer. In the proposed DNN model, the ReLU activation function is employed in all the layers of the system, which is given by:

$$f(b) = \begin{cases} b & \text{if } b \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

During the backward propagation, the network calculates the error and then propagates it backward in order to optimize the weight values. We have considered mean-square error which is calculated as:

$$L = \frac{1}{\omega}\sum_{i=1}^{\omega}(Q(s, a)_i - \overline{Q(s, a)_i})^2, \tag{16}$$

where the training batch size is denoted by ω, while $Q(s, a)_i$ represents the Q-value of the ith sample calculated using (7). Additionally, $\overline{Q(s, a)_i}$ refers to the estimated Q-value of the ith training sample, which is provided by the DNN. We use gradient descent optimization to update the weights as:

$$W_{z\times y} = W_{z\times y} - \Delta\left(\frac{\delta L}{\delta W_{z\times y}}\right), \tag{17}$$

where Δ is the learning rate of the framework. $\left(\frac{\delta L}{\delta W_{z\times y}}\right)$ is the gradient of error (L) with respect to the weights $(W_{z\times y})$

that we are updating. During the testing phase of DQ-RL, the DNN does not need to be trained. Therefore, by inputting the state parameters into the DNN, we can obtain the estimated Q-values for the actions at the output.

B. SOLUTION TO MAXIMIZE THE INSTANTANEOUS REWARD

In many RL solutions, the reward function for the agent is based on the objective function of the problem, with the goal of selecting actions that maximize the objective values. In the context of the power allocation problem explored in this paper, constraint (5) is always satisfied as the possible actions simply represent the fractions of available power used for transmission. However, constraint (6), which places an interference constraint on the problem, can be violated if the agent does not account for it. Thus, rather than using the objective function value as the reward in this scenario, we define the reward as follows:

$$\Gamma = \kappa\theta\beta, \tag{18}$$

where β is the indicator function of the primary network outage and is defined as:

$$\beta = \begin{cases} 1 & \text{if } \sum_{k=1}^{K} p_k f_k \leq I_{th} \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

Training the agent to maximize the reward function defined in (18) enables the agent to allocate power in a way that maximizes the objective and satisfies the interference constraint.

C. SOLUTION TO MAXIMIZE THE CUMULATIVE REWARD

In the considered problem, we aim to jointly maximize the fairness and sum-rate of the system. Achieving fairness in an interference limited system is very complicated as it requires that each ST be transmitting (if the transmission power of any user is equal to zero then the fairness factor $\theta = 0$ and also the objective function in (4) is 0). With the fairness requirement, each user is assigned some transmission power irrespective of its interference channel gain, which results in producing high values of interference, therefore, the outage probability at the primary receiver defined as the probability that ($\sum_{k=1}^{K} p_k f_k > I_{th}$) also increases. Whereas, for the considered interference-limited system, in the case of other problems where energy efficiency [33] or sum-rate [34] is maximized, some STs may not be allowed to transmit at all, in order to reducing the overall interference temperature. So that, the STs with best channels can transmit with little interference, resulting in increasing the sum-rate and the energy efficiency of the system.

In RL, agents are provided with the knowledge of the current state of the environment, and through training, they learn which actions are likely to yield a high reward value. As a result, it becomes possible to maximize the cumulative reward value over multiple time slots. Instead of allocating power to maximize fairness in each individual time slot, it is possible to assign power in a way that maximizes the average fairness of two or more consecutive time slots. To achieve this, we define the average sum-rate as follows:

$$\kappa = \sum_{k=1}^{K} \tau_k, \tag{20}$$

where

$$\tau_k = \frac{\sum_{t=1}^{T} R_k(t-T)}{T}, \tag{21}$$

the parameter T represents the number of time slots used in calculating the cumulative reward. The rate of the kth ST in the current time slot is ($R_k(0)$), its rate in the previous time slot is denoted as ($R_k(-1)$), and so on. The average fairness index θ is computed by:

$$\theta = \prod_{k=1}^{K} \frac{\tau_k}{\max_j \tau_j}. \tag{22}$$

The cumulative reward approach becomes very beneficial in the considered system because the system is interference-limited. Therefore, when all the STs are transmitting on the same channel, each ST faces interference due to the transmissions of the other STs. Hence, the overall interference that each ST faces is high, resulting in poor performance for each ST.

With the cumulative reward strategy, the STs that transmit with less power in one time slot may transmit with more power in the next slot; i.e., users can transmit in a sequential manner. An ST that has transmitted with less power in a time slot to keep the interference at a low level will transmit with more power in the next time slot, whereas the STs that transmitted with more power would choose to transmit with less power. The cooperation among the STs ensures that the average rates of all the STs are comparable (providing fairness in rates) while reducing the interference faced by each ST, resulting in better performance for each ST.

Thus, we also consider a variation of the proposed DRL model where the agents are trained to maximize the cumulative reward instead of the instantaneous reward. The value of the reward, denoted as Γ, is computed as $\Gamma = \kappa\theta\beta$, where β is defined as in (19).

D. DQ-RL ALGORITHM

The steps involved in the training of each agent are given in Algorithm 1. First we initialize, F that denotes the number of iterations after which the weights of the DNN are replaced in DQ-RL to make training relatively fast. Since, we employ replay memory based training where in each training round a min-batch of samples is chosen from the memory buffer to train DNN_1, a memory buffer of size D is initialized. Further, instead of training the agent in each iteration, after every training round, we wait for G number of iterations for a reasonable amount of new samples to be stored into the memory buffer before training DNN_1, hence, we initialize G at the start of the algorithm. Also, we initialize both the

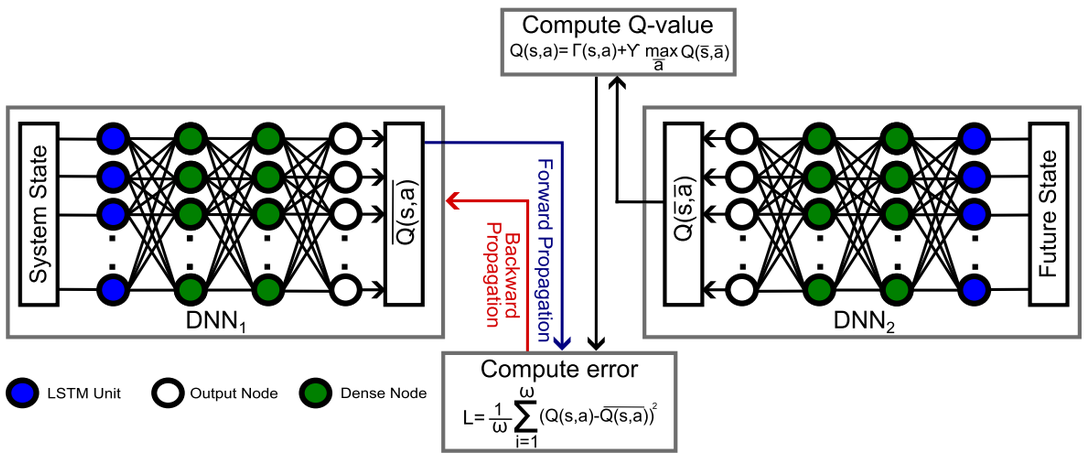

FIGURE 2. **Training of the proposed DNN framework.**

DNNs with random weights. The DNN used to estimate the values of $Q(s, a)$ is represented as DNN_1, and the DNN that estimates $Q(\overline{s}, \overline{a})$ is denoted as DNN_2. Note that the hyper-parameters (number of layers, nodes per layer) of both DNNs are the same, which is necessary because the weights of DNN_2 are updated from the weights of DNN_1. In the algorithm, the weights of DNN_1 are denoted as W, while $\overline{W}$ are the weights of DNN_2. Then, in each time slot/iteration, the state of the system is provided to the agents after which each agent takes an action according to the ϵ-greedy algorithm. After taking the action, the agent receives a reward in step 7 of the algorithm. In step 8, the agent saves the experience (state, action taken and reward received) in the memory buffer. After every G time slots, a mini-batch of training samples are selected at random from the buffer in step 10. Then the Q-value for each selected sample is calculated in step 11 using the reward value and the output of DNN_2, as shown in (7). In step 12, the estimated Q-values are obtained from DNN_1, after which the error in estimation is computed as the mean squared error of the estimated Q-value and the value obtained from (7). After which, DNN_1 is trained using back propagation (the steps involved in computing the loss function are shown in Fig. 2, where the detailed architecture of the LSTM unit is presented in Fig. 1). For optimizing the weights of DNN_1, we have used gradient descent optimizer. After every F iterations the weights of DNN_2 are replaced by those of DNN_1 to improve the future reward estimates of DNN_2.

In the testing phase, the steps of *exploration* and training are removed and in each time slot, the agent takes actions according to the output of the DNN_1. Note that, in the proposed framework, the training of the DNNs can be continued in the testing phase as well. The continuous training will have the benefit of making the system more flexible, and robust to change in the number of STs, the value of I_{th}, or the number of possible actions (resolution of the action space). However, the continuous training would require the agents to perform periodic *exploration* and would demand continuous availability of processing resources.

Algorithm 1 DQ-RL Agent Training

1) Initialize memory buffer with size D, F, G and hyper-parameters of the DNNs
2) Initialize DNN_1 weights for $Q(s, a)$ as W
3) Initialize DNN_2 weights for $Q(\overline{s}, \overline{a})$ as $\overline{W}$
4) **for** t in time-slots
5) Observe state at time t, (s)
6) Take action (a) according to ϵ-greedy algorithm
7) Receive reward Γ
8) Store s, a, and Γ in the memory buffer
9) **if** remainder(t/G)==0
10) Randomly sample training examples from memory buffer
11) Calculate $Q(s, a)$ for each training example
12) Compute the loss and train DNN_1 to predict the values of $Q(s, a)$, $\forall s, a$
13) **end if**
14) **if** remainder(t/F)==0
15) update $\overline{W}$ as $\overline{W} = W$
16) **end if**
17) **end for**
18) **Return** trained agent

V. SOLUTION SCHEMES

This section discusses different variants of the proposed framework and the optimal strategy, and also provides complexity comparison of all the techniques. Note that, in every variant, each agent takes the state of the system and its own transmission power in the previous time slot as the input of the DNN.

- **Optimal:** The **Optimal** algorithm employs a brute-force search over all the potential power allocation combinations to maximize the instantaneous reward.

This approach yields the best possible solution for the given resolution of the action space. However, the **Optimal** technique necessitates a high degree of cooperation among all the STs, as examining all feasible permutations of the action space necessitates coordination between the STs. Specifically, each ST must possess accurate information about the actions of the other STs in the succeeding time slot. It is important to note that the possible permutations of the action space differ from the resolution of the action space of the STs. For example, suppose the system has two STs, and each ST can take three possible actions. In that case, there are only three possible actions at the ST's end. However, from the perspective of the system, there are 3^2 potential actions.

- **Maximize Sum-Rate:** In this scheme, the agents are trained to maximize the sum-rate of the system. In interference-limited systems, if the objective is to maximize the sum-rate, many users are not allowed to transmit, consequently, the system is unfair. Note that, for the **Maximize Sum-Rate** scheme, the state space is defined by $2K + 1$ parameters (i.e., $\mathcal{S}$=$\{h_1, h_2, \ldots, h_K, \sum_{k=1}^{K} p_k f_k, \text{SINR}_1, \text{SINR}_2, \ldots, \text{SINR}_K\}$).
- **Instantaneous Reward:** Here we train the agents to maximize the reward in (18), in each time slot. The parameters that define the state of the environment are the same as in the case of **Maximize Sum-Rate** technique (i.e., $\mathcal{S}$=$\{h_1, h_2, \ldots, h_K, \sum_{k=1}^{K} p_k f_k, \text{SINR}_1, \text{SINR}_2, \ldots, \text{SINR}_K\}$).
- **Cumulative Reward, 1 Previous State:** *OR* **Feedback interference, all h_k, all SINRs:** In this scheme, we train the agents to maximize the cumulative reward of every two consecutive time slots, the reward value is computed as in (18) and κ and θ are calculated as given in (20) and (22). Note that, the scheme is exactly the same as **Instantaneous Reward**, except, instead of considering the reward of the current slot, the agents maximize the average cumulative reward of the current and previous time slots. As each agent already has the knowledge of the current state and with training learns the best action to maximize the average combined reward of the current and previous state, the scheme is expected to outperform the technique where we maximize the instantaneous reward in each time slot (**Instantaneous Reward**). The state-set for each time slot is defined as $\mathcal{S}$=$\{h_1, \ldots, h_K, \sum_{k=1}^{K} p_k f_k, \text{SINR}_1, \ldots, \text{SINR}_K\}$.
- **Cumulative Reward, 2 Previous States:** If maximizing the rewards of two consecutive time slots improves the average performance of the system, it might be interesting to consider even more previous time slots before making the decision on the best action. Hence, we train the agents to maximize the average reward of every three consecutive time slots. The considered multi-layer DNNs take the decision based on the current inputs and have no memory of the previous inputs. If we want to maximize the average cumulative reward of three time slots, the DNN needs to account for the previous two time slots. Therefore, the nodes in the first hidden layer of the DNNs are replaced by LSTM units. The LSTMs are memory-full units and are used when we need the DNNs to provide the solution while considering more than 1 previous inputs (sequence of inputs). To provide the solution, the **Cumulative Reward, 2 Previous States** scheme requires the state parameters of 2 previous time slots where the state-set of a time slot is given as: $\mathcal{S}$=$\{h_1, h_2, \ldots, h_K, \sum_{k=1}^{K} p_k f_k, \text{SINR}_1, \text{SINR}_2, \ldots, \text{SINR}_K\}$.
- **Cumulative Reward, 3 Previous States:** In this framework, the DNNs of each agent takes the states of 3 successive time slots as input and maximize the average reward of 4 consecutive time slots.
- **Feedback interference, all h_k and reward:** In this case, each agent is provided with the true value of the interference produced, the reward obtained in the previous time slot, and the current values of the channel gains of all the STs. Thus, the scheme requires $N + 3$ features to define the state (i.e., $\mathcal{S}$=$\{h_1, h_2, \ldots, h_K, \sum_{n=1}^{K} p_k f_k, \Gamma\}$).
- **Feedback binary interference and reward:** Here we reduce the feedback burden on the PR and SR. In this scheme the state is defined by two features, binary feedback of interference (0 if the interference threshold was violated, otherwise 1), and the reward value. Thus, each agent makes the decision based on the binary interference variable and the reward generated in the previous time slot. That is, the state-set is $\mathcal{S}$=$\{\beta, \Gamma\}$, where β and Γ are given by (19) and (18), respectively. The binary feedback of interference just requires a single bit, thus, the approach may become more suitable in some scenarios where the feedback channel has very limited capacity. Further, when a binary interference variable is considered, the secondary system may not need to engage the PR as the SR can listen for the negative acknowledgement (NACK) signal from the PR to PT.

VI. SIMULATION RESULTS

In this section, first we discuss the computational complexities of all the solution schemes and then we compare the performance of all the frameworks.

A. DISCUSSION ON COMPLEXITY AND COMMUNICATION OVERHEAD

Here we compare the Big-O complexities of the proposed schemes, which provide an insight on how the worst case complexity of the frameworks increase with the number of users in the system. The computational complexity of the **Optimal** approach is directly proportional to the number of STs and the resolution of the action space. When a new user is added to the system, the complexity increases Y times, where Y is the number of values in the action space. Thus,

the computational complexity of the **Optimal** scheme can be expressed as $O(Y^K)$.

On the other hand, in DQ-RL frameworks, the primary source of computational complexity arises from the DNNs. In this study, when the number of users is increased, the size of the input laeyr of the DNN increases. Consequently, the computational complexity of the testing phase can be expressed as $O(IE)$. Here, I represents the number of nodes in the input layer and equals the number of features in the state. Meanwhile, E, represent the number of nodes in the first hidden layers, respectively.

The computational complexity of **Maximize Sum-Rate**, **Instantaneous Reward** and **Cumulative Reward, 1 Previous State** schemes is $O(IE)$. When the number of input features is reduced as in the case of **Feedback binary interference and reward** and **Feedback interference, all h_k and reward** the value of I reduces and therefore the complexity also decreases. Then, for the cases where more than 1 previous state is considered, the complexity of the LSTM layer increases linearly with the size of sequential inputs and the number of features per input. Therefore, for **Cumulative Reward, 2 Previous States** and **Cumulative Reward, 3 Previous States**, the computational complexity is $O(XIE)$, where X denotes the number of previous states being considered for making the decision.

When the number of STs is increased, the complexity of **Feedback binary interference and reward** remains unchanged. However, the complexity of other frameworks changes. The solutions of all other schemes depend on the SINRs and/or channel gains of all the STs. Therefore, the number of nodes in the input layer I also depends on the number of STs. The relation is given as $I = 2 + 2K$ for all the frameworks with $\mathcal{S} = \{h_1, h_2, \ldots, h_K, \sum_{k=1}^{K} p_k f_k, \text{SINR}_1, \text{SINR}_2, \ldots, \text{SINR}_K\}$.[5] The complexity of these techniques have an order of $O((2 + 2K)E)$. Likewise, the complexity in the case of **Feedback interference, all h_k and reward** is given as $O((3 + K)E)$ because the number of channels increases with the STs. Therefore, in DQ-RL models, the complexity grows linearly with the users. In contrast, the complexity of the **Optimal** scheme increases exponentially with the number of users (K). It should be noted that in the testing phase, the DNN only performs forward propagation of the data. Whereas, in the training phase the network first performs the forward propagation and then backward-propagation. Hence, the complexity of the single pass of the training phase is approximately double that of the testing phase.[6] Computational complexity of all the frameworks is summarized in table 2, where **Cumulative Reward, X Previous States** represents the framework where '**X**' previous states are required by the DNN to provide the solution. A single iteration of a DQ-RL-based solution takes approximately 3.3 ms where consideration of cumulative or instantaneous reward has negligible impact on the duration of an iteration. A single iteration of the optimal scheme lasts for 0.41 ms. However, the optimal scheme requires much more iterations to provide the solution as compared to the DQ-RL schemes (as shown in the results (Fig. 5) the DQ-RL schemes converge within 15 iterations, however, the optimal scheme requires $11^3 = 1331$ iterations to give the solution). Hence, the overall time of DQ-RL schemes is far less than the optimal scheme. The Big-O complexity of the schemes for the specified simulation parameters is depicted in Fig. 3. Notably, the **Optimal** scheme exhibits the highest worst-case complexity. To complement this analysis, Fig. 4 illustrates the computational time required for convergence by the proposed schemes. It is evident that the *Optimal* scheme demands a significantly longer time to converge. The huge difference in the convergence time is due to the fact that **Optimal** scheme takes 1331 iterations to give the solution whereas the DQ-RL framework converges within 15 iterations.

[5]These include the K channel gain values (h_k), K SINR values, a single value for interference, and one value for the transmission power used by the agent in the previous time slot, which is not included in the state-set.

[6]Note that the frameworks that make the decision with less features or require fewer consecutive inputs need smaller data-sets (less number of samples) for training. However, finding the relationship of the optimal number of samples with the number of features is still an open research topic. For fairness in comparison, we have trained the DNNs in every scheme with the same number of samples.

TABLE 4. Table of computational complexities.

Scheme	Computational Complexity
Optimal	$O(Y^K)$
Maximize Sum-Rate	$O((2+2K)E)$
Instantaneous Reward	$O((2+2K)E)$
Cumulative Reward, X Previous States, Feedback interference, all h_n, all SINRs	$O(X(2+2K)E)$
Feedback interference, all h_k, reward	$O((3+K)E)$
Feedback binary interference and reward	$O(3E)$

We now examine the communication overhead for each framework. In DQ-RL schemes, requiring SINR values and channel gains, the SR broadcasts $(1 + 2K)$ floating-point values to each ST. Considering 32 bits per floating-point value, the communication overhead becomes $32(1 + 2K)$ bits. For the solution scheme not relying on SINR values, the overhead decreases to $32(2 + K)$ bits. The scheme using only the binary interference indicator function and reward incurs a communication overhead of 33 bits (32 bits for reward and 1 bit for the binary indicator function). Notably, in the scheme employing the binary interference indicator function and reward, the communication overhead becomes independent of the number of users in the system.

B. COMPARISON OF PERFORMANCE

For the simulations, we have used a Core-i5 computer with a 2.50GHz processor, 4GB RAM, and no dedicated GPU. For our simulation setting, the system parameters are depicted in Table 5. We consider Rayleigh fading channels, where the real and imaginary part of the channel are drawn from independently and identically distributed Gaussian distribution with zero mean and variance 1 [35].

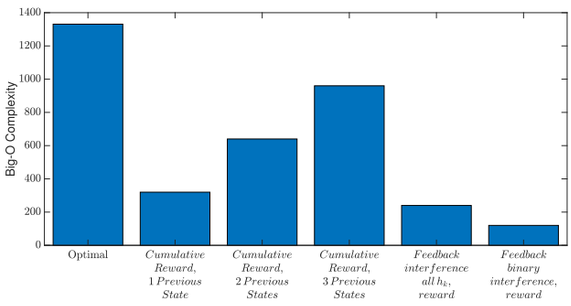

FIGURE 3. **The figure compares the worst case Big-O complexity of the proposed frameworks for the considered simulation parameters.**

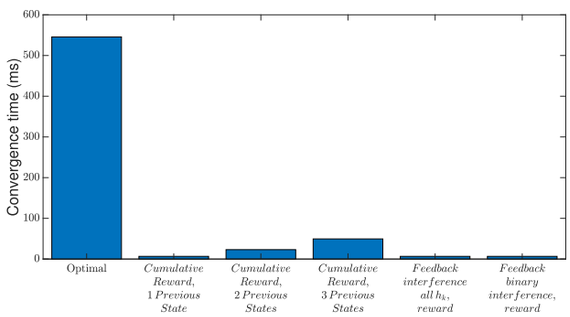

FIGURE 4. **The figure compares the convergence time that each framework takes to provide the solution.**

TABLE 5. **Simulation parameters.**

Parameters	P_k^{th}	I_{th}	I_{pn}	σ^2	K	F
Values	1 W	1 W	0.5 W	0.1	3	30

Parameters	G	D	Training Samples	Testing Samples	Y	
Values	10	1000	30,000	5000	11	

Rayleigh fading models the effect of multipath reception in dense environments where non-line of sight components become prominent. Further, the considered system model is interference-limited because all the STs in the system share the same frequency channel, where the STs are assumed to be distributed uniformly. The agents were trained with cumulative reward function considering one previous state, with the state-set $\mathcal{S} = \{h_1, h_2, \ldots, h_K, \sum_{k=1}^{K} p_k f_k, \text{SINR}_1, \text{SINR}_2, \ldots, \text{SINR}_K\}$, unless stated otherwise.

As discussed earlier, when we aim to enhance the fairness of the system, training the agents to maximize the cumulative rewards instead of the instantaneous rewards may improve the performance. First, we present results comparing performances of the instantaneous and the cumulative reward policies. Figure 5 shows the impact of increasing the time slots for which the cumulative reward is calculated. It can be seen that the least value of the average reward is provided when the instantaneous reward is maximized. As expected, the optimal scheme provides the highest reward. When the features of more previous time slots are used as input, the value of reward also increases. However, the increase is not uniform as when we maximize the combined reward of two consecutive time slots (**Cumulative Reward, 1 Previous State**), we observe an increase of **27.26%**, as compared to the case where we maximize the instantaneous reward. Increasing the number of previous states from 1 to 2 results in an average increase of **7.51%**. Similarly, when the number of previous states is increased to 3, the increase in average reward is **5.50%**. Although increasing the number of previous states increases the average reward, there are many time-slots where one or more users are not transmitting (to reduce the interference faced by PR and other STs). Therefore, in a latency-sensitive system, the **Instantaneous Reward** scheme may be preferred.

The convergence behavior of the DQ-RL schemes can be observed in Fig. 5. Specifically, for the case where $K = 3$ and the action space consists of 11 possible actions, the **Optimal** scheme is able to converge after evaluating $11^3 = 1331$ different combinations. Once convergence is achieved, the reward values of the **Optimal** scheme are displayed in Fig. 5. It should be noted that the time taken by all DQ-RL techniques to converge in the testing phase is far less than that of the **Optimal** scheme. Each DQ-RL framework converges within 15 iterations, as shown in Fig. 5. The reason behind the fast convergence of the DQ-RL frameworks is that once the training of DQ-RL schemes is completed, the agents no longer perform exploration. Hence, in each time slot, the agent observes the state, their previous action, and the reward obtained to estimate what action could improve the reward compared to the previous time slot. The reward obtained by the agents increases with each time slot, as a result of improved actions. Until a maximum value of reward is achieved, at which point no better action is possible, and the system converges. When we increase the number of previous states required to make the decision, the time to convergence increases as a result of increased complexity. However, it can be concluded that the DQ-RL schemes provide much faster convergence than the **Optimal** scheme.

The problem of maximizing the sum-rate has been considered extensively in the literature for different networks. However, the objective of rate maximization is unfair as the users with better channels are allocated more power, whereas the users with comparatively bad channels are allocated less power for communication. In interference-limited systems as considered in this work, if the sum-rate is maximized the unfairness increases because the users with relatively bad channels are not be allowed to communicate at all. Now, we provide the comparison of a scheme where the sum-rate is maximized (**Maximize Sum-Rate**) with the proposed frameworks in Fig. 6. It is apparent that the fairness offered by the **Maximize Sum-Rate** scheme is very low, indicating that the scheme is highly unfair. However, when we consider the objective of maximizing the instantaneous reward with the **Instantaneous Reward** scheme, the fairness of the

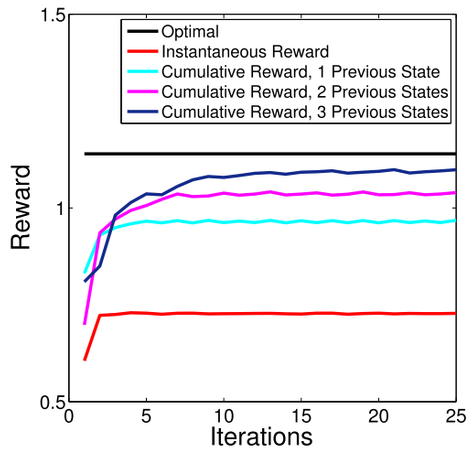

FIGURE 5. **A comparison of the average received reward of the proposed frameworks with the optimal scheme and the impact of increasing the number of sequential inputs on the average reward.**

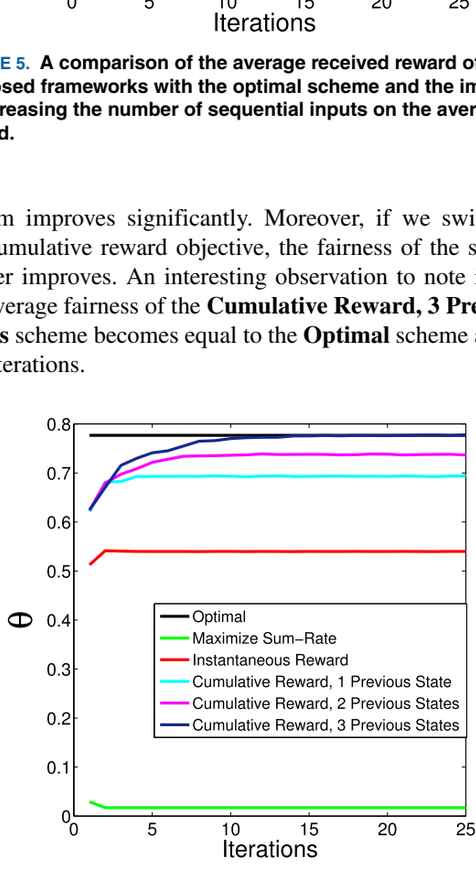

FIGURE 6. **The figure shows average fairness provided by the proposed frameworks versus the fairness of sum-rate maximization technique.**

To ensure fairness in a wireless communication system, users with weaker channel gains are allotted more transmission power than those with better channel conditions. However, this strategy typically results in a decrease in the system's overall sum-rate when fairness is increased. Fig. 7 compares the sum-rate of each scheme, with the **Maximize sum-rate** scheme exhibiting the highest rate while the fairness-based schemes offer lower average sum-rates.

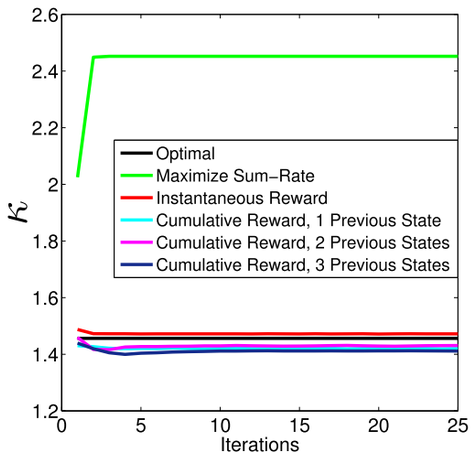

FIGURE 7. **Average sum-rate provided by each scheme and the drop in the sum-rate of the system when fairness in rates of the STs and the sum-rate of the system are jointly maximized.**

It is also interesting to note that the sum-rates offered by the fairness based schemes are comparable to each other. Furthermore, it should be noted that in the scheme where the sum rate is maximized, mostly, only the user with the best channel condition is allowed to transmit, while the other users in the system remain idle. It was observed during the simulations that for the **Maximize sum-rate** scheme on average 1.025 users were allowed to transmit on the channel. Therefore, although the sum rate maximization scheme offers a higher rate, it is highly unfair to the users in the system.

Figure 8 presents the outage probability results of each proposed framework. In the case of the **Optimal** scheme, the outage probability is zero, as the system considers all possible power allocation combinations and selects the best solution that satisfies the constraint ($\sum_{k=1}^{K} p_k h_k < I_{th}$). We would like to emphasize that the figure shows the outage probability of the **Optimal** scheme after it has searched for the best solution. In contrast, the **Maximize Sum-Rate** scheme has a lower outage probability compared to the fairness-based schemes because it does not allow users with poor channel conditions to transmit, thereby minimizing overall system interference. However, with the **Instantaneous Reward** scheme, each user is required to transmit in every time slot. As a result, in a distributed solution framework, where the STs cannot be certain about the decisions of other STs and must base their decisions on previous observations, there is a higher chance of taking incorrect action and transmitting with excessive power, leading to higher outage probability at the primary end. Therefore, the outage probability for the **Instantaneous Reward** scheme is the highest among the proposed frameworks. However, if we consider cumulative reward over multiple time slots, the STs can take turns transmitting or adjust their transmission power levels based on previous observations. Hence, the cumulative reward based approaches result in higher average rewards values

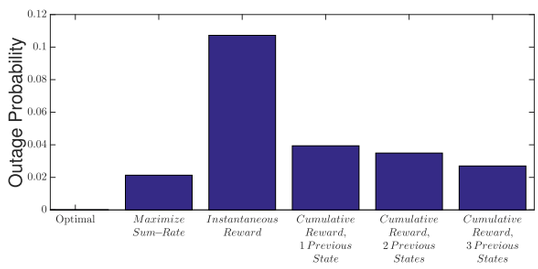

FIGURE 8. **The figure compares the outage probability of the primary network for the different schemes.**

while maintaining lower interference and outage probability levels.

The performance of DQ-RL frameworks heavily relies on the number of features defining the state of the environment. More features may enhance performance but would increase the feedback load and necessitate longer training times. Conversely, considering only a few features reduces feedback load and computational complexity but might result in suboptimal performance. In this comparison, we assess the performance of three frameworks utilizing different numbers of state features. As depicted in Fig. 9, the highest reward is achieved by the **Feedback interference, all h_n, all SINRs** framework. When SINRs are excluded from the state-set (**Feedback interference, all h_n, reward**), the average reward decreases. The **Feedback binary interference, reward** scheme, which relies solely on the feedback of a binary indicator of interference and the reward function value, exhibits the poorest performance. The system's performance with imperfect channel state information (CSI) is contingent on the error in the CSI. In the best-case scenario with perfect CSI, the performance matches that of **Feedback interference, all h_n, all SINRs** or **Feedback interference, all h_n, reward**, depending on the available state features. Conversely, in the worst-case scenario with maximum error, the performance aligns with **Feedback binary interference, reward**. For all other cases (partial error in CSI), the proposed frameworks consistently perform between these two extremes, depending on the value of error.

In CR systems, if the value of I_{th} is increased, the performance of the system improves because the power allocation at the STs becomes more flexible. Thus, the users can increase the transmission power (if it would result in a higher reward). Now we present the results that show the impact of I_{th} on the performance of the proposed frameworks. For the results in Fig. 10, the agents were trained for different values of I_{th}. Figure 10 shows that increasing I_{th} results in increasing the average reward. When I_{th} is increased from 0.6 W to 0.8 W, the marginal reward (increase in reward per unit increase in the value of the parameter) is greater than when we increase I_{th} from 0.8 W to 1 W. If the value of I_{th} is further increased, the marginal reward keeps on decreasing. There are two reasons for this marginal decrease: first, the

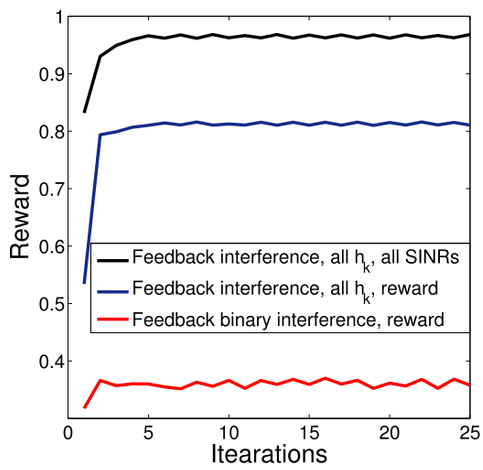

FIGURE 9. **The average reward values of the secondary system under different variants of the state-set.**

sum-rate is a logarithmic function of allocated power, thus, when the value of I_{th} is increased linearly, a logarithmic increase in the value of rate is observed, second, when I_{th} is increased, all STs may increase their transmit power initially, however, after a certain point when some STs are already transmitting with full available power, the transmission power can not be further increased.

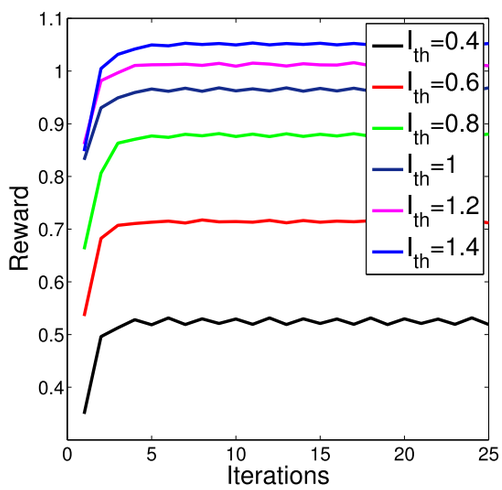

FIGURE 10. **The average rewards in the case of different systems with values of I_{th} ranging from 0.4 W to 1.4 W. The case is homogeneous to the scenario where the value of I_{th} of a system changes and we retrain the agents for the new value of I_{th}.**

The results in Fig. 10 show the effect of increasing I_{th} on the performance of DQ-RL agents trained separately for each value of I_{th}. The results showed that the DQ-RL-based technique provides good performance for any value of I_{th}. However, the approach to train the agents for each value of I_{th} might not be practical because it is time-consuming.

Now, we present the impact of varying I_{th} on the performance of a DQ-RL agent trained for $I_{th} = 1$ W. As the model is trained to keep interference below 1 W. We need to modify the interference value before it is used by the agent to provide a viable solution. Therefore, we modify the value of interference as: interference $= (\sum_{k=1}^{K} p_k f_k) + (1 - I_{th})$. The modification allows us to use the agent trained for $I_{th} = 1$ for any other value of I_{th}.[7]

Figure 11 shows the impact of I_{th} on the reward value. As seen before, the reward increases with I_{th}. The proposed model provides smaller values of reward as compared to the previous case where a different DQ-RL model was trained for each value of I_{th}. The performance at lower I_{th} is worse, however, when I_{th} increases the rewards become comparable to those shown in Fig. 10. Further, when I_{th} is increased from 1.2 W to 1.4 W the average reward remains approximately the same. Whereas, in the case of Fig. 10 a small increase in reward was observed at these points.

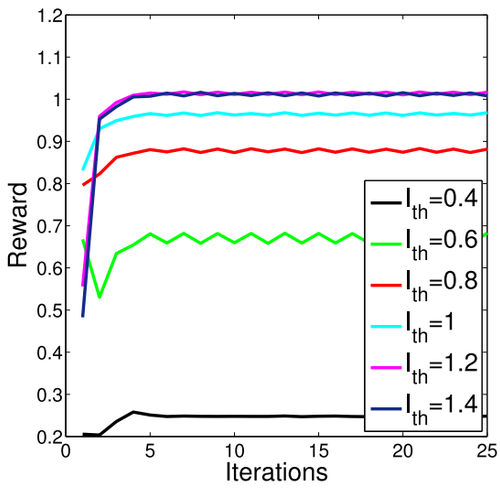

FIGURE 11. The average reward of the secondary systems when the modified interference-value is provided to the agents such that the same agents can be used for any value of I_{th} without any retraining.

Increasing the resolution of the action space improves the system's performance, as depicted in Fig. 12. The heightened resolution of the action space provides more precise options for power allocation. For instance, when considering 6 actions with $P_k^{th} = 1$W, the smallest power allocation step an agent can take is 0.2 W. However, in the case of 11 possible actions, the smallest step is reduced to 0.1 W. For 6 actions, the maximum gap in power allocation from the optimal point would be 0.1 W (10%). Conversely, with 11 actions, the maximum gap from the optimal point is reduced to 0.05 W (5%). Increasing the resolution from 6 to 11 actions results in a reduction of the maximum error by 0.05 W. Further increasing the number of actions to 16 decreases the maximum error to 0.0333 W (3.33%), and expanding the action space from 16 to 21 actions further reduces the maximum error to 0.025 W (2.5%). Hence, for a linear increase in the number of actions, the error decreases in a non-linear fashion.[8] Therefore, as we continue to increase the number of actions, the framework progressively approaches optimal power allocation, and the sub-optimality introduced due to discretization diminishes. However, the marginal reward keeps decreasing, as it is observed that there is only a very small gain in the reward when increasing the number of actions from 16 to 21.

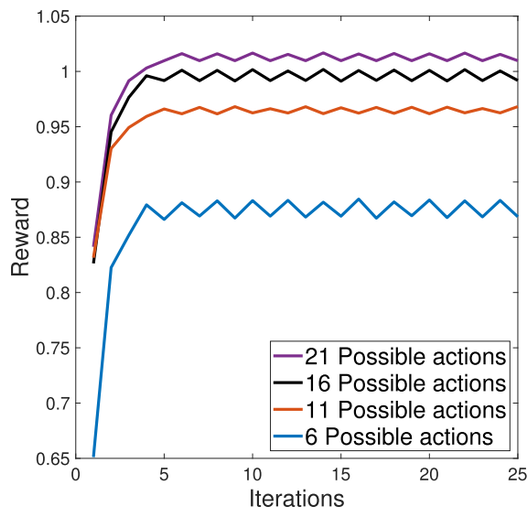

FIGURE 12. Average reward provided by DQ-RL frameworks trained for different resolutions of the action space, and each agent in the system has the same resolution of action space.

Next, in Fig. 13 we compare the average reward provided by the proposed framework when the number of STs in the system is changed. We consider the scenarios of 2, 3, and 4 STs. Increasing K decreases the average reward because it becomes hard to achieve a high value of fairness index while satisfying the interference constraint. Furthermore, as all the STs in the system share the same channel, with an increase in the number of STs, the interference of the system also increases, resulting in reduced per-user SINR and high outage probability of the primary network. Therefore, the average reward value decreases when K is increased.

In Fig. 14, we compare the performance of an LSTM-based framework with a counterpart using simple RNNs. While maintaining consistency in other aspects of the proposed scheme, we replaced the LSTM nodes in the first layer of the DNN model with simple RNN nodes. The results indicate that both frameworks exhibit good

[7]Note that, it is not necessary to train the agent for $I_{th} = 1W$. The proposed solution could work for the agents trained for any other value of I_{th}.

[8]For fairness in comparison, we have trained all the considered frameworks with an equal number of samples. However, DNNs with more output nodes usually require more training to reach their maximum performance.

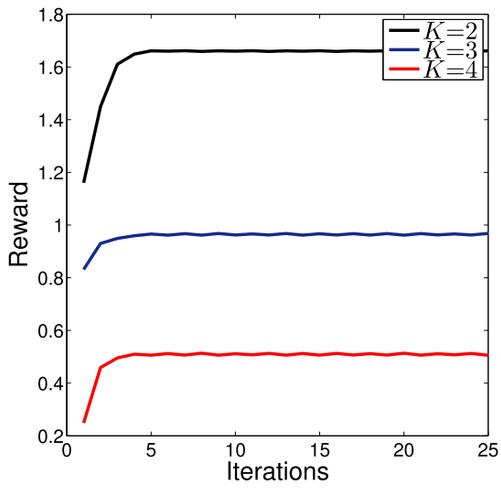

FIGURE 13. **Average reward of the secondary systems containing different number of STs communicating simultaneously.**

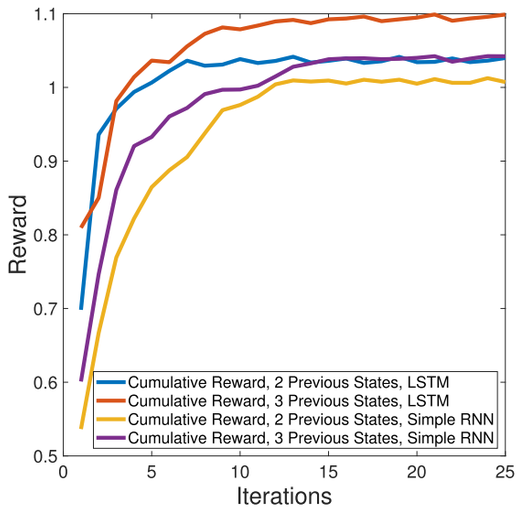

FIGURE 14. **Average reward comparison of the proposed LSTM model with a simple-RNN model [32].**

performance, but the LSTM-based scheme achieves superior reward values. Additionally, the figure illustrates that the simple RNN-based framework requires more iterations to converge. It is worth noting that although a simple RNN provides inferior performance compared to LSTM, the computational complexity differs. A simple RNN requires $(2 + 2K)X + 2X^2$ multiplication operations and $(2 + 2K)X + 2X^2$ addition operations, whereas LSTM units demand $3(2 + 2K)X + 3X^2 + 3X$ multiplications and $3(2 + 2K)X + 3X^2 + 2X$ additions [32]. While LSTM units are computationally more complex, modern programming languages like Python support multi-thread processing, allowing parallel execution of independent operations and minimizing the impact on execution time. For devices with limited computational capabilities where multi-thread processing is not feasible, employing simple RNN-based models becomes more practical.

VII. CONCLUSION

We proposed a DQ-RL-based approach for optimizing power allocation in an underlay cognitive radio network where multiple STs share a channel. To address the issue of unfairness in interference-limited systems, a novel distributed framework is proposed to ensure fairness in rates among all STs while satisfying the primary network's interference threshold. Several modifications to the DQ-RL model are also proposed to enhance its effectiveness. One of the most interesting outcome of the framework is that the proposed cumulative reward based techniques result in greater network fairness and lower primary outage probability compared to the instantaneous reward technique.

In the future, we will work on improving the developed model such that it could also provide the solution for channel allocation in a multi-channel scenario. Improving the scalability of the proposed scheme such that they are insensitive to the number of secondary transmitters, is of paramount importance. Furthermore, it would be interesting to evaluate the impact of a regularization parameter for penalizing primary outage, where instead of returning a 0 reward in the case of a primary outage the reward value is decreased by a certain factor. In this work, we have considered linear quantization of the available power, hence, the number of actions increases linearly with P_k^{th}. In the future, we will try to find a better quantization technique that provides better performance with less number of actions.

REFERENCES

[1] S. D. Okegbile, B. T. Maharaj, and A. S. Alfa, "Interference characterization in underlay cognitive networks with intra-network and inter-network dependence," *IEEE Trans. Mobile Comput.*, vol. 20, no. 10, pp. 2977–2991, Oct. 2021.

[2] Z. Ali, G. A. S. Sidhu, M. Waqas, L. Xing, and F. Gao, "A joint optimization framework for energy harvesting based cooperative CR networks," *IEEE Trans. Cognit. Commun. Netw.*, vol. 5, no. 2, pp. 452–462, Jun. 2019.

[3] M. Zhu and S. Martínez, "An approximate dual subgradient algorithm for multi-agent non-convex optimization," in *Proc. 49th IEEE Conf. Decis. Control (CDC)*, Dec. 2010, pp. 7487–7492.

[4] Z. Ali, G. A. S. Sidhu, F. Gao, J. Jiang, and X. Wang, "Deep learning based power optimizing for NOMA based relay aided D2D transmissions," *IEEE Trans. Cognit. Commun. Netw.*, vol. 7, no. 3, pp. 917–928, Sep. 2021.

[5] A. K. Gizzini and M. Chafii, "A survey on deep learning based channel estimation in doubly dispersive environments," *IEEE Access*, vol. 10, pp. 70595–70619, 2022.

[6] M. Alrabeiah, Y. Zhang, and A. Alkhateeb, "Neural networks based beam codebooks: Learning mmWave massive MIMO beams that adapt to deployment and hardware," *IEEE Trans. Commun.*, vol. 70, no. 6, pp. 3818–3833, Jun. 2022.

[7] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2239–2250, Oct. 2019.

[8] D. Ebrahimi, S. Sharafeddine, P.-H. Ho, and C. Assi, "Autonomous UAV trajectory for localizing ground objects: A reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1312–1324, Apr. 2021.

[9] J. Zhang, S. Chen, X. Wang, and Y. Zhu, "Dynamic reservation of edge servers via deep reinforcement learning for connected vehicles," *IEEE Trans. Mobile Comput.*, vol. 22, no. 5, pp. 2661–2674, May 2023.

[10] G. Alsuhli et al., "Mobility load management in cellular networks: A deep reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 22, no. 3, pp. 1581–1598, Mar. 2023.

[11] F. Meng et al., "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6255–6267, Oct. 2020.

[12] L. Xiao et al., "Reinforcement learning-based downlink interference control for ultra-dense small cells," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 423–434, Jan. 2020.

[13] L. Xiao, X. Lu, T. Xu, X. Wan, W. Ji, and Y. Zhang, "Reinforcement learning-based mobile offloading for edge computing against jamming and interference," *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 6114–6126, Oct. 2020.

[14] D. Guo, L. Tang, X. Zhang, and Y. Liang, "Joint optimization of handover control and power allocation based on multi-agent deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13124–13138, Nov. 2020.

[15] X. Meng, H. Inaltekin, and B. Krongold, "Deep reinforcement learning-based power control in full-duplex cognitive radio networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–7.

[16] W. Ning, X. Huang, K. Yang, F. Wu, and S. Leng, "Reinforcement learning enabled cooperative spectrum sensing in cognitive radio networks," *J. Commun. Netw.*, vol. 22, no. 1, pp. 12–22, Feb. 2020.

[17] X. Li, J. Fang, W. Cheng, H. Duan, Z. Chen, and H. Li, "Intelligent power control for spectrum sharing in cognitive radios: A deep reinforcement learning approach," *IEEE Access*, vol. 6, pp. 25463–25473, 2018.

[18] I. AlQerm and B. Shihada, "Enhanced online Q-learning scheme for energy efficient power allocation in cognitive radio networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2019, pp. 1–6.

[19] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 310–323, Jan. 2019.

[20] X. Zhang, T. Chang, Y. Liu, C. Shen, and G. Zhu, "Max-min fairness user scheduling and power allocation in full-duplex OFDMA systems," *IEEE Trans. Wireless Commun.*, vol. 18, no. 6, pp. 3078–3092, Jun. 2019.

[21] P. Xu and K. Cumanan, "Optimal power allocation scheme for non-orthogonal multiple access with α-fairness," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 10, pp. 2357–2369, Oct. 2017.

[22] F. Zhou, Z. Li, N. C. Beaulieuz, J. Cheng, and Y. Wang, "Resource allocation in wideband cognitive radio with SWIPT: Max-min fairness guarantees," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[23] L. Yuan, S. Bi, X. Lin, and H. Wang, "Optimizing throughput fairness of cluster-based cooperation in underlay cognitive WPCNs," *Comput. Netw.*, vol. 166, Jan. 2020, Art. no. 106853.

[24] A. Attar, O. Holland, M. R. Nakhai, and A. H. Aghvami, "Interference-limited resource allocation for cognitive radio in orthogonal frequency division multiplexing networks," *IET Commun.*, vol. 2, no. 6, pp. 806–814, Jul. 2008.

[25] Z. Ali, Z. Rezki, and H. Sadjadpour, "Deep-Q reinforcement learning for fairness in multiple-access cognitive radio networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Austin, TX, USA, Apr. 2022, pp. 2023–2028.

[26] R. Mehta, "Recursive quadratic programming for constrained nonlinear optimization of session throughput in multiple-flow network topologies," *Eng. Rep.*, vol. 2, no. 6, Jun. 2020, Art. no. e12169.

[27] L. Xiao, Y. Li, C. Dai, H. Dai, and H. V. Poor, "Reinforcement learning-based NOMA power allocation in the presence of smart jamming," *IEEE Trans. Veh. Technol.*, vol. 67, no. 4, pp. 3377–3389, Apr. 2018.

[28] Q. Zhai, M. Bolic, Y. Li, W. Cheng, and C. Liu, "A Q-learning-based resource allocation for downlink non-orthogonal multiple access systems considering QoS," *IEEE Access*, vol. 9, pp. 72702–72711, 2021.

[29] J. Si and Y.-T. Wang, "Online learning control by association and reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 264–276, Mar. 2001.

[30] S. Rezwan and W. Choi, "Priority-based joint resource allocation with deep Q-learning for heterogeneous NOMA systems," *IEEE Access*, vol. 9, pp. 41468–41481, 2021.

[31] T. Li, X. Zhu, and X. Liu, "An end-to-end network slicing algorithm based on deep Q-learning for 5G network," *IEEE Access*, vol. 8, pp. 122229–122240, 2020.

[32] A. K. Gizzini and M. Chafii, "RNN based channel estimation in doubly selective environments," *IEEE Trans. Mach. Learn. Commun. Netw.*, vol. 2, pp. 1–18, 2024.

[33] F. Zhou, X. Zhang, R. Q. Hu, A. Papathanassiou, and W. Meng, "Resource allocation based on deep neural networks for cognitive radio networks," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Aug. 2018, pp. 40–45.

[34] W. Lee, "Resource allocation for multi-channel underlay cognitive radio network based on deep neural network," *IEEE Commun. Lett.*, vol. 22, no. 9, pp. 1942–1945, Sep. 2018.

[35] R. Sun, Y. Zhang, H. Zheng, J. Guo, J. Sun, and J. Xue, "A Douglas-Rachford splitting approach based deep network for MIMO signal detection," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Glasgow, U.K., Mar. 2023, pp. 1–6.

ZAIN ALI received the Ph.D. degree in electrical engineering from COMSATS University, Islamabad, Pakistan, in 2021. Currently, he is a Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, University of California at Santa Cruz, USA. He received the HECs Indigenous Scholarship for the M.S. and Ph.D. studies. His research interests include cognitive radio networks, energy harvesting, multihop relay networks, orthogonal frequency division multiple access (OFDMA), nonorthogonal multiple access (NOMA), machine learning, and engineering optimization.

ZOUHEIR REZKI (Senior Member, IEEE) was born in Casablanca, Morocco. He received the Diplôme d'Ingénieur degree from École Nationale de l'Industrie Minérale (ENIM), Rabat, Morocco, in 1994, the M.Eng. degree from École de Technologie Supérieure, Montreal, QC, Canada, in 2003, and the Ph.D. degree in electrical engineering from École Polytechnique, Montreal, in 2008. He is currently an Assistant Professor with the Electrical and Computer Engineering Department, University of California at Santa Cruz.

HAMID SADJADPOUR (Senior Member, IEEE) received the B.S. and M.S. degrees from the Sharif University of Technology and the Ph.D. degree from the University of Southern California (USC). In December 1995, he joined the ATT Shannon Research Laboratory, as a Technical Staff Member and later as a Principal Member of Technical Staff. In 2001, he joined the University of California at Santa Cruz, where he is currently a Professor. He has authored over 200 publications and holds 25 patents. His research interests are in the general areas of wireless communications, security, and networks. He was a co-recipient of the Best Paper Awards at the 2007 International Symposium on Performance Evaluation of Computer and Telecommunication Systems, the 2008 IEEE Fred W. Ellersick Award in Military Communications conference, the 2010 European Wireless Conference, and the 2017 Conference on Cloud and Big Data Computing. He has served as a technical program committee member and the chair for numerous conferences.