Deep-Q Reinforcement Learning for Fairness in Multiple-Access Cognitive Radio Networks

Zain Ali, Zouheir Rezki*, Hamid Sadjadpour
Electrical and Computer Engineering Department, Baskin School of Engineering,
University of California, Santa Cruz, USA
Emails: zali4@ucsc.edu, zrezki@ucsc.edu, hamid@soe.ucsc.edu

2022 IEEE Wireless Communications and Networking Conference (WCNC) | 978-1-6654-4266-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/WCNC51071.2022.9771661

***Abstract*—This work presents a deep-Q reinforcement learning (DQ-RL) framework to achieve fairness in multi-access cognitive radio (CR) systems. The proposed framework provides fast solution and is robust to channel dynamics. Further, to remove the computational overhead and the burden to feedback thousands of weights from the secondary receiver (SR), we propose a solution where the process of learning is carried out at the secondary transmitters (STs). The simulations show that by using the proposed technique, a good level of fairness is achievable with an outage probability of the primary system less than 0.04. We also provide the comparison of the proposed technique with a brute-forcing optimization method, and show the fairness gain of the proposed framework compared to the rate maximization model.**

I. Introduction

Cognitive radio (CR) enhances the spectral efficiency of the system by allowing secondary users (SUs) to share the spectral resources of the primary users (PUs) [1]. In simple CR networks, a single secondary transmitter (ST) shares the channel of a primary transmitter (PT). A system that can accommodate multiple STs on the same channel could offer better service. However, contrary to single SU case, when multiple STs share the same channel, the optimization problem becomes non-convex. For these problems, the conventional optimization frameworks can not be used since convergence is not guaranteed.

Machine-learning-based frameworks have been shown to provide fast solutions as compared to conventional techniques. However, the standard supervised machine learning techniques require a large set of data for training the model [2]. Further, if the parameters of the system change and re-training is required, we need to reproduce the training set for the new parameters [3]. Recently, different reinforcement learning (RL) frameworks have been shown to provide fast solutions and do not need any other techniques to provide the training data [4]. In RL, the training agents learn by taking actions and receiving a reward from the environment that quantifies the merit of the action.

Several works in the literature have proposed RL solutions for optimization problems. The authors in [5] designed a Q-learning framework for cooperative sensing in overlay CR networks. Then, to maximize the energy efficiency of the underlay CR system, a Q-learning framework was designed in [6]. The proposed technique provides good results, however the scheme only learns to optimize power allocation for the current channels and may need to retrain every time the channel gains change. The work in [7] proposed a robust DRL-based spectrum access framework that does not require retraining when the channel gains change. However, the solution in [7] requires centralized training for a distributed system, hence thousands of weights of the deep neural networks (DNNs) are transferred to all the users after the completion of the training phase.

The problem of achieving fairness in different communication systems has been considered broadly in the literature. The authors in [8] optimized resource allocations to achieve fairness in orthogonal frequency division multiple access systems. The problem of unfairness becomes more serious in interference limited systems where multiple STs are assigned the same channel. Thus, many STs may not be allowed to transmit to prevent outage and/or to maximize the system rate. Considering the problem of fairness in CR systems, the authors in [9] proposed a centralized resource allocation framework to enhance the fairness in rates of the STs. In the considered system, the STs are immune to interference from other STs. Thus, the problem of power allocation becomes convex. Further, the authors proposed a Lagrangian-dual-based solution which is not guaranteed to converge in the case when multiple STs are assigned the same channel because the problem becomes non-convex. The authors in [10] achieved fairness at the wireless powered STs subject to the interference threshold of the primary system. Considering multiple STs on the same channel, the authors in [11] optimized power allocation to achieve fairness in the rates of all the STs in the system. To solve the non-convex optimization problem, the authors employed a sequential-quadratic-programming-based iterative technique. The techniques in [8]- [11] are iterative in nature and require a lot of time to provide the solution. The issue with these approaches is that once the channel gains of the system change, we need to re-run the optimization.

In this work, which is a shorter version of [12], we propose a deep-Q reinforcement learning (DQ-RL) framework to achieve fairness in the rates of all the STs sharing a channel in an underlay CR setting. The proposed technique does not require the generation of a large data-set for training and provides fast solution. Further, once the training is complete, the framework provides excellent results for new values of channel gains,

* This material is based upon work supported by the National Science Foundation (CAREER) under Grant No. 2114779.

because in DQ-RL a DNN is trained to predict the Q-values corresponding to each action when the state parameters are provided to the DNN. Hence, after the training, when the channel gains change, the trained DNN provides the estimated Q-value of each action for the new state of the system.[1]

II. System Model and Problem Formulation

We consider an underlay CR system where N STs share a frequency channel. The channel gains from the nth ST to the SR and from nth ST to primary receiver (PR) are denoted as h_n and f_n, respectively. Provided that the SR adopts single-user decoding and treats interference as noise, an achievable rate of the nth ST is: $R_n = \log_2\left(1+\frac{p_n h_n}{\sum_{\substack{i=1\\ i\neq n}}^{N} p_i h_i + I_p + \sigma^2}\right)$, where, p_n is the transmission power allocated by the nth ST, σ^2 is the variance of AWGN and I_p denotes the interference from the primary system to the SR. In this work, we aim to enhance the fairness in rates of all the STs in the system. To quantify the fairness in rates, we use the linear product-based fairness index θ, calculated as:

$$\theta = \prod_{n=1}^{N} \frac{R_n}{\max_i R_i}. \tag{1}$$

If the rate of one or more STs is zero then $\theta = 0$ which shows that the system is unfair, whereas $\theta = 1$ means that the rate of each user in the system is exactly the same. Thus, the closer the value of θ is to 1, the better is the fairness of the system.

A system where all STs are achieving very small values of rates may offer very high value of fairness. However, the system is not efficient because the resources are not fully utilized. Therefore, instead of only maximizing the fairness, a better objective is to jointly maximize the sum-rate and the fairness index. The sum-rate of the system is computed as follows:

$$\lambda = \sum_{n=1}^{N} R_n. \tag{2}$$

The problem of maximizing the sum-rate of the system in a fair manner is written as:

$$\max_{p_n} \lambda\,\theta, \qquad \text{s.t.:}\Big\{p_n \leq Pt_n, \forall n, \sum_{n=1}^{N} p_n f_n \leq I_{th}\Big\}, \tag{3}$$

where, Pt_n and I_{th} denotes the battery capacity of the nth ST and the interference threshold to prevent outage at the primary system, respectively. First constraint guarantees that the transmission power of each user is smaller than the battery capacity and the second constraint protects the primary system against harmful interference from the STs.

III. Proposed Deep reinforcement learning-based model

The above considered optimization problem (3) is non-convex in nature. Using conventional optimization frameworks may lead to unsatisfactory results as convergence is not guaranteed. For non-convex problems, RL has been shown to provide excellent solutions.

[1]Although the current paper has been developed to be self-contained, more details regarding the proposed framework is reported in [12].

In RL, the RL agent (ST in our case) observes the state of the environment (system parameters) and takes an action (allocates power for transmission). After an action is taken, the environment returns a reward to the agent. The aim of the RL agent is to maximize the reward value. In DRL, a DNN is used to learn the optimal state-to-action mapping, where the DNN is trained in real-time from the feedback of the environment. Directly learning the state-to-action mapping is a hectic process because the state transition probability after taking an action is not known. Thus, we employ Q-value-based RL also called Q-learning which provides an alternate learning approach by giving the expected reward (Q-value) of taking an action in a particular state and then following the policy thereafter. For a given state-action pair, the Q-value is computed as:

$$Q(s,a) = r(s,a) + \gamma \max_{\bar{a}} Q(\bar{s},\bar{a}), \tag{4}$$

where $r(s,a)$ represents the immediate reward of taking action a, when the sate of environment is s and $\gamma \max_{\bar{a}} Q(\bar{s},\bar{a})$ is the discounted long term reward of taking the action a, where $\bar{s}$ represents the next state after the agent takes action a in state s, γ is called the discount factor and $0 \leq \gamma \leq 1$. The $\bar{a}$ represents the action that gives maximum value of reward. The deep Q-leaning uses two separate DNNs to predict the values of $Q(s,a)$ and $Q(\bar{s},\bar{a})$ called the policy DNN and target DNN, respectively [13].

In the training phase, the agent first takes the state of the environment as input. Then, the agent uses a predefined strategy to take an action. The training of DQ-RL agent contains two major phases: *exploration* and *exploitation*. During *exploration*, the agent takes an action randomly and obtains a reward from the environment. In the *exploitation* phase, the agent tries to take the best action according to the past experience learned by the DNN. The rewards obtained in both phases are used for training the agent. The *exploration* phase compels the agent to try new actions, while in the *exploitation* phase the agent obtains feedback about its learned behaviour from the environment. Many strategies are proposed in the literature to switch between *exploration* and *exploitation*. In this work, we employ the ϵ-greedy strategy because it has been shown to provide fast results and a good level of DQ-RL training for a broad spectrum of problems. After taking the action, the agent receives a reward. The agent then saves the state, action taken and the received reward in a tuple. The experience is saved to a buffer and after a certain number of iterations ('G'), the experiences in the buffer are sampled, the Q-values corresponding to these samples are computed and used to train DNN_1. The steps involved in the training of an agent are also shown in Fig. 1.

The last two steps of training a DQ-RL agent shown as Block A in Fig. 1, are the most complex. Hence, we provide a dummy example to expand the Block A in Fig. 2 in order to explain the steps in details. In Fig. 2, the Block **1** shows a single experience of the agent. Block **2** shows multiple experiences stored in the buffer. A single training sample contains two consecutive experiences. The preceding experience contains the current state, the action, and the reward

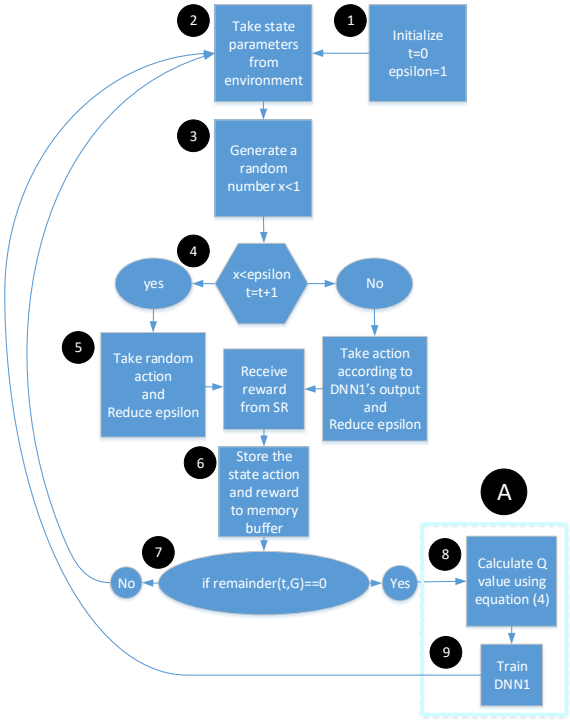

Fig. 1. Flowchart of DQ-RL training: First, the parameters of the algorithm are initialized in step (1). In step (2) the agent receives the state parameters. In step (3) a random value is produced for the ϵ-greedy action strategy, the action policy is determined by the ϵ-greedy method in step (4), in step (5) the agent takes action and receives the reward. The experience (state, action, and reward) of the agent is saved in the memory buffer in (6). In step (7), we check if the buffer-size is completely divisible with a variable 'y' (value of y is a parameter of the algorithm, and 'y' can be set to have any positive value, smaller values result in more frequent training but also uses the processing resources more often.). If the training requirements are satisfied, the Q-values of the experiences are calculated in step (8), otherwise, we go back to step (2). Then, the DNN is trained in step (9) and the execution returns to step (2).

value. The succeeding experience contains the next state of the system after taking the action, shown in Block **3** of the Fig. 2. Then, to calculate the Q-value as shown in Block **4**, we need the current reward and the maximum Q value of the next state ($\max_{\overline{a}} Q(\overline{s},\overline{a})$), where $\overline{s}$ denotes the next state and $\overline{a}$ is the action taken in the next state. For calculating $\max_{\overline{a}} Q(\overline{s},\overline{a})$ we use a separate DNN (DNN_2) to reduce the correlation, as was discussed before. The DNN used to compute $\max_{\overline{a}} Q(\overline{s},\overline{a})$ is named DNN_2 and is shown in Block **5**. Then, the next state parameters are provided to DNN_2, which gives the estimated Q-value of each action at the output. The maximum of these Q-values is the solution of $\max_{\overline{a}} Q(\overline{s},\overline{a})$. For the value of $r(s,a)$ in Block **4** of figure 2, we choose the reward value corresponding to the current state. After calculating the Q-values, the current state is given as input to the main DNN (DNN_1) that is used for the computation of Q-values by the DQ-RL agent in the *exploitation* phase. The DNN_1 gives us the estimated Q-value of each action corresponding to the current state. Then, the loss in the estimation is computed as shown in Block **7**. In this work, we have employed the mean-square error function to compute the error in the estimations of DNN_1. After computing the error/loss in the estimation of each output node of DNN_1, the error value is multiplied with the values of the corresponding actions. In DQ-RL, one-hot encoding is used to represent the actions of the agents. Thus, in the string representing the action, only one bit can have a value of 1, and all other bits are set to zero. The multiplication of the loss with the corresponding action value is very important because it allows us to correct the error in the estimated Q-value of the action taken by the agent, while leaving the parameters of other actions unaffected. Then, the error is back-propagated to update the weights of the DNN_1. After every 'F' iterations the weights of DNN_2 are replaced by the weights of DNN_1 to improve the estimations of DNN_2.

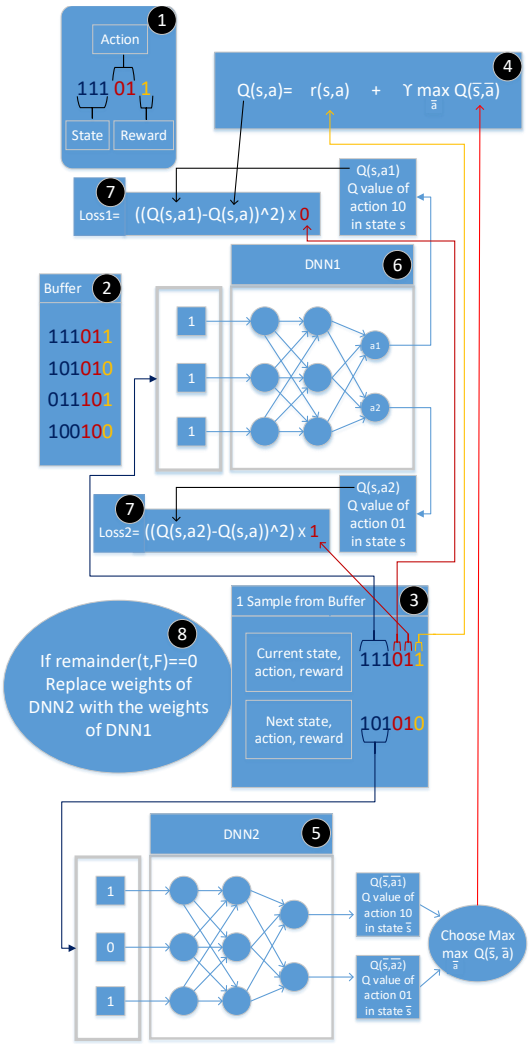

Fig. 2. Computation of Q-value and DNN training: (1) shows a single experience of the agent. (2) is the experiences saved in the buffer of the agent. (3) is the example of a single training sample from the buffer. (4) shows how the Q-value is computed. (5) is the DNN_2 that is used to compute the value of $\max_{\overline{a}} Q(\overline{s},\overline{a})$ for the computation of Q-value in equation (4). (6) shows DNN_1 that provides the estimated Q-values for the current state. In (7), the errors in the estimated Q-values of DNN_1 are computed and back-propagated for correction of the weights. In step (8) we check if the condition for replacing DNN_2 with DNN_1 is satisfied, if the condition is satisfied, the weights of DNN_2 are replaced by the weights of DNN_1.

IV. Proposed Solution

For the solution, we propose a multi-agent DQ-based RL framework. In DQ-RL, the action space contains the set of

possible action values. For example, in a power allocation problem, if the value of battery capacity is 1W and the action space has a resolution of 3. Then, the output-layer of the DNN will have 3 nodes, and the agent can have 3 solution values: $\{1,0,0\}$, $\{0,1,0\}$ and $\{0,0,1\}$ representing 0 W, 0.5 W and 1 W, respectively[2]. We consider that the state-set of the system for each agent is defined as: $\mathcal{S}$=$\{h_1, h_2, ..., h_N, \sum_{n=1}^{N} p_n f_n, \text{SINR}_1, \text{SINR}_2, ..., \text{SINR}_N\}$[3]. The parameters of the state-set are fed back to each ST by the SR where the value of the true interference is provided by the PR to the SR. Including some other parameters like the interference channel gains (f_n) to the state-set may improve the performance of DQ-RL agents. However, it would increase the feedback load on the PR, and thus, is not practical.

A. DNN model

In DQ-RL, we need DNN to provide the estimated Q-values. The number of nodes in the input and output layers of the DNN are always equal to the number of input features ($\mathcal{S}$ and p_n) and the resolution of the action space, respectively. The number of hidden layers, and nodes in each hidden layer dictates the capacity of the DNN. In theory, for a good approximation, the capacity of DNN should be sufficient to fit the function of a particular complexity. In the proposed framework, we consider DNN models with 3 hidden layers and 200, 100 and 50 nodes in the first, second and third hidden layer, respectively. The training of DNN contains two phases: forward propagation and backward propagation. In the forward propagation phase, the output of a layer is calculated as: $\psi^x_{1\times y} = f(\psi^{x-1}_{1\times z} W_{z\times y})$, where $\psi^x_{1\times y}$ is the output of layer number x containing y neurons, $(1 \times y)$ denotes the order of the vector and $f(\cdot)$ is the activation function of the layer, and $W_{z\times y}$ denotes the weight matrix connecting layer $x-1$ having z neurons to layer x containing y neurons. Then, the output layer gives us the estimated Q-value of each action for the given state ($Q(s,a)$). In the considered DNN model, we have used ReLU activation function in the first hidden layer while a linear activation function is employed for all other layers in the system. The linear activation function is defined as $f(\eta) = \eta, \forall \eta$. The ReLU function is given by: $f(\eta) = \eta$, if $\eta \geq 0$, and $f(\eta) = 0$, otherwise.

In the backward propagation phase, the error is calculated and is propagated backward through the network to optimize the values of weights. The error is calculated as: $L = \frac{1}{\omega} \sum_{i=1}^{\omega} (Q(s,a)_i - \overline{Q(s,a)_i})^2$, where ω is the mini-batch size used for training, $\underline{Q(s,a)_i}$ is the Q-value of the ith sample computed in (4), and $\overline{Q(s,a)_i}$ is the estimated Q-value of the ith training sample provided by the DNN_1 (Block 6 in Fig. 2). Then, to update the weights, we employ the gradient descent optimizer.

[2]The outputs/solutions are represented as one-hot encoded bits.

[3]The input of each DNN contains the state-set $\mathcal{S}$ and p_n. The value of p_n denotes the transmission power of the ST in the previous time slot and each ST already has the knowledge of its own transmission power in the previous time slot. Thus, the value of p_n is not fed back to the STs and is not included in the state-set in this work.

TABLE I
EXAMPLES OF POSSIBLE ACTIONS

	ST-1	ST-2	ST-3	Reward
Channel Gains	h_1=0.3325 f_1=0.6663	h_2=0.9763 f_2=0.8032	h_3=0.9142 f_3=0.4360	
Actions-1	0.9W	0.3W	0.3W	0.8404
Actions-2	0.3W	0.3W	0.9W	0.0329
Actions-3	0.9W	0.3W	0W	0

B. Instantaneous reward-based solution

In most RL solutions for power allocation, the objective function of the problem is used as the reward function. For the power allocation problem, as considered in this paper, the battery capacity constraint is always satisfied as the action space represents the fractions of available power. However, in the considered problem we have an interference constraint given in (3), which can be violated if it is not accounted for by the agent. Hence, in this work, we define the reward to be:

$$\Gamma = \lambda \theta \alpha, \tag{5}$$

where α is the interference indicator function and is defined as:

$$\alpha = \begin{cases} 1 & \text{if } \sum_{n=1}^{N} p_n f_n \leq I_{th} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Maximizing the reward in (5) trains the agent to allocate power such that the objective/reward is maximized and the interference constraint is also satisfied. Table 1 shows some examples of possible actions for different channel gain values. For instance, Actions-1 are the best actions as they return the maximum reward, whereas, Actions-3 result in a zero reward. Table 1 also highlights a limitation of distributed learning, as an ST has no information about the actions of the other STs in the system, a wrong action by a single ST would affect the learning of all the STs (in the case of Actions-3 in Table 1, ST-1 and ST-2 may learn that the actions taken by them are bad, which is not true). However, because of *exploration* while training the agents, the users check for each action multiple times and try to learn the effectiveness of a particular action based on the combined knowledge of all the experiences. Thus, *exploration* helps the agents to learn the impact of their own actions by reducing the effects of the actions of other STs.

C. Cumulative reward-based solution

In RL, the agents have the knowledge of current state of the environment, and with training, the agents learn what actions would produce a high value of reward, which gives us the opportunity to maximize the cumulative reward of two consecutive time slots. Hence, we can assign power such that the average fairness of two consecutive time slots is maximized. That is, we first define the average sum-rate as:

$$\lambda = \sum_{n=1}^{N} \tau_n, \tag{7}$$

where

$$\tau_n = \frac{\sum_{t=1}^{2} R_n(t-2)}{2}, \tag{8}$$

where $R_n(0)$ denotes the rate of nth ST in the current time slot, and $R_n(-1)$ is its rate in the previous time slot. Then,

the average fairness index θ is computed as:

$$\theta = \prod_{n=1}^{N} \frac{\tau_n}{\max_i \tau_i}. \quad (9)$$

The value of reward (Γ) is computed as $\Gamma = \lambda\theta\alpha$, where α is defined as in (6).

V. Solution Schemes

This section discusses the schemes proposed and compared in this work, and also provides the complexity comparison of all the techniques:

- **Optimal:** The **Optimal** algorithm is a brute-force search over all the possible combinations of power allocation of the STs. The algorithm provides the best possible solution. However, for the **Optimal** technique to work, a high level of cooperation is required between all the STs since checking all possible permutations of the action space, requires coordination between the STs.
- **Maximize Sum-Rate:** In this scheme, the agents are trained to maximize the sum-rate of the system. When we increase the fairness in users' rates, we lose some performance in terms of sum-rate. Thus, a comparison of the proposed fair rate maximization scheme with the sum-rate maximization scenario is important.
- **Instantaneous Reward:** Here we train the agents to maximize the reward in (5), in each time slot.
- **Cumulative Reward:** In this scheme, the agents are trained to maximize the reward in (5) for every two consecutive time slots, where the values of λ and θ are computed as in (7) and (9), respectively.

Discussion on Complexity

The computation complexity of the **Optimal** scheme depends on the number of STs and the resolution of the action space. If the action space contains Y values, each time a new user is added to the system the complexity increases Y times. Thus, the complexity of the **Optimal** scheme can be written as $O(Y^N)$.

In the case of DQ-RL frameworks, the major part of computational complexity is induced by the DNNs. We have considered DNNs with three hidden layers, thus, the computational complexity of the testing phase is $O(IJ + JK + KL + LY)$, where I is the number of nodes in the input layer and is equal to the number of input features[4], J, K, L are the number of nodes in the first, second and third hidden layer, respectively, and Y is the number of nodes in the output layer and is equal to the resolution of the action space. As such, the computational complexity of the **Maximize Sum-Rate**, **Instantaneous Reward** and **Cumulative Reward** schemes is given by $O(IJ + JK + KL + LY)$. That is, in regard of our state space and the power used by each agent in the previous time slot, $I = 2N + 2$ and the latter complexity becomes $O((2+2N)J + JK + KL + LY)$. Hence, in DQ-RL models, the complexity increases linearly with the number of users, whereas, the complexity of the **Optimal** scheme scales exponentially with the number of users N. The convergence time of all the schemes is given in Table III, it can be seen that the DQ-RL-based solution converges in far less time as compared to the **Optimal** framework.[5]

[4] N values of the channel gains (h_n), N values of SINRs, 1 value for the interference, and 1 value of the transmission power of the agent in the previous time slot (not a part of the state-set).

TABLE II
Simulation Parameters

Parameters	Pt_n	I_{th}	I_p	σ^2	N
Values	1 W	1 W	0.5 W	0.1	3

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

TABLE III
Convergence Time

Scheme	Convergence Time	Scheme	Convergence Time
Optimal	0.5381s	**Maximize Sum-Rate**	0.0140s
Instantaneous Reward	0.0168s	**Cumulative Reward**	0.0315s

VI. Simulation Results

For our simulation setting, the system parameters are depicted in Table II. All the channels were taken from independently and identically distributed Gaussian random variables with zero mean and variance 1. The results shown in this section are averaged over 5000 testing samples.

Figure 3 shows the mean fairness provided by each scheme along with the 99% confidence level. It can be seen that the fairness offered by the **Maximize Sum-Rate** scheme is close to zero, which confirms that this scheme is very unfair. When the objective of maximizing the instantaneous reward is considered (**Instantaneous Reward**), the fairness of the system increases significantly. If we consider the cumulative reward instead, the fairness of the system improves further. As expected, the **Optimal** scheme provides the largest value of fairness in the system. Figure 3 also shows the convergence behaviour of the DQ-RL schemes. For the considered case of $N = 3$ and action space containing 11 actions, the **Optimal** scheme converges after a search over $11^3 = 1331$ possible combinations. Figure 3 shows the values of the **Optimal** technique after the convergence. It can be seen from Fig. 3 that the DQ-RL techniques converge within a few (around 6) iterations.

To achieve fairness in a wireless communication system, the user with a relatively lower value of the channel gain transmits with more power than the users with better channel conditions. Thus, when the fairness of the system is increased, generally the sum-rate of the system decreases. For completion of the analysis, the comparison of the sum-rate offered by each scheme and the 99% confidence level is provided in Fig. 4. As expected, the figure shows that the **Maximize sum-rate** scheme offers the maximum sum-rate, whereas, the average

[5] The simulations were carried out on a CORE i5 machine with 2.50 GHz processor and 4 GB RAM.

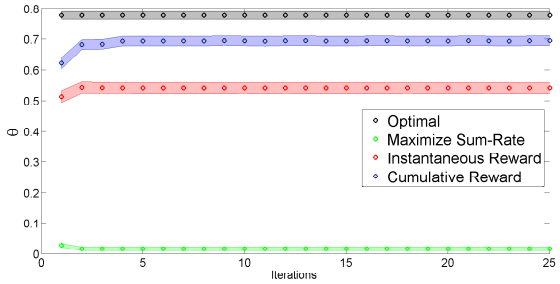

Fig. 3. The average gain in fairness of the proposed schemes as compared to the sum-rate maximization scheme and the effect of increasing the number of sequential inputs on the value of fairness index. The bands around the points represent the 99% confidence level.

sum-rates of the fairness-based schemes are less, and are comparable to each other.

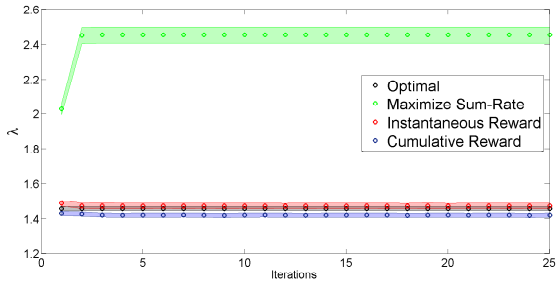

Fig. 4. Average sum-rate provided by each scheme. The 99% confidence levels are shown as bands around the points.

The outage probability of each proposed framework is provided in Fig. 5. In the case of **Optimal** scheme, the outage probability is equal to zero since the system checks for all possible combinations of power levels and chooses the best solution (any solution where $\sum_{n=1}^{N} p_n h_n > I_{th}$ returns a zero reward, and hence can not be selected). The outage probability in the case of the **Maximize Sum-Rate** scheme is less than that of the fairness-based schemes because when we maximize the sum-rate, the users with bad channel conditions are not allowed to transmit. Hence, the overall interference of the system remains at a low level. If the **Instantaneous Reward** scheme is considered, each user is required to transmit in every time slot. Consequently, more interference is produced as compared to other frameworks, thus, the outage probability in the case of the **Instantaneous Reward** scheme is the highest. If we consider the **Cumulative Rewards** scheme, the STs can take turns to transmit, or, an ST that transmits with low power in one time slot can transmit with a higher power in the next time slot and vice versa. Thus, the **Cumulative Reward** scheme achieves higher values of fairness while keeping the interference at a lower level. We expect that increasing the number of time slots over which the average reward is computed, would provide lower outage levels, although not shown in Fig. 5.

VII. Conclusion

In this work, we proposed a DQ-RL-based model for power allocation in an underlay cognitive radio network where multiple STs share the same channel. Considering the unfairness issue in interference-limited systems, we proposed a novel

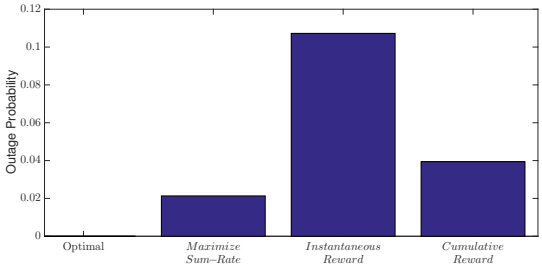

Fig. 5. The outage probability of the primary system offered by the proposed frameworks as compared to the outage probability in the case of optimal solution and the rate maximization objective.

distributed framework to achieve fairness in the rates of all the STs, while satisfying the interference threshold of the primary network. One of the most interesting results highlighted by the current framework is how considering the cumulative reward can enhance the fairness of the network and lower the primary outage probability as compared to the instantaneous reward scheme.

References

[1] W. Lee, "Resource Allocation for Multi-Channel Underlay Cognitive Radio Network Based on Deep Neural Network," in *IEEE Commun. Letters,* vol. 22, no. 9, pp. 1942-1945, Sept. 2018.

[2] G. Qian, Z. Li, C. He, X. Li and X. Ding, "Power Allocation Schemes Based on Deep Learning for Distributed Antenna Systems," in *IEEE Access*, vol. 8, pp. 31245-31253, 2020.

[3] Y. S. Nasir and D. Guo, "Multi-Agent Deep Reinforcement Learning for Dynamic Power Allocation in Wireless Networks," in *IEEE Journal on Selected Areas in Commun.,* vol. 37, no. 10, pp. 2239-2250, 2019.

[4] M. Botvinick, S. Ritter, J. X. Wang, Z. K. Nelson, C. Blundell and D. Hassabis, "Reinforcement Learning, Fast and Slow," in *Trends in cognitive sciences*, vol. 23 , no. 5, pp. 408-422, 2020.

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

[6] I. AlQerm and B. Shihada, "Enhanced Online Q-Learning Scheme for Energy Efficient Power Allocation in Cognitive Radio Networks," *IEEE Wireless Commun. and Networking Conf. (WCNC),* 2019, pp. 1-6.

[7] O. Naparstek and K. Cohen, "Deep Multi-User Reinforcement Learning for Distributed Dynamic Spectrum Access," in *IEEE Trans. on Wireless Commun.,* vol. 18, no. 1, pp. 310-323, Jan. 2019.

[8] X. Zhang, T. Chang, Y. Liu, C. Shen and G. Zhu, "Max-Min Fairness User Scheduling and Power Allocation in Full-Duplex OFDMA Systems," in *IEEE Trans. on Wireless Commun.,* vol. 18, no. 6, pp. 3078-3092, June 2019.

[9] F. Zhou, Z. Li, N. C. Beaulieuz, J. Cheng and Y. Wang, "Resource Allocation in Wideband Cognitive Radio with SWIPT: Max-Min Fairness Guarantees,"*IEEE Global Commun. Conf. (GLOBECOM),* 2016, pp. 1-6.

[10] L. Yuan, S. Bi, X. Lin and H. Wang, "Optimizing throughput fairness of cluster-based cooperation in underlay cognitive WPCNs," in *Computer Networks,* vol. 166, pp. 106853, Jan. 2020.

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

[12] Z. Ali, Z. Rezki and H. Sadjadpour, "Deep-Q Reinforcement Learning for Fairness in Multiple-Access Underlay Cognitive Radio Networks", submitted to the *2022 IEEE Wireless Commun. and Networking Conf. (WCNC'2022)*, October 1st, 2022. Available: https://drive.google.com/file/d/1YQQoIDqhSPAIkAwbkIaTufbmf8ePxMhy/view

[13] S. Rezwan and W. Choi, "Priority-Based Joint Resource Allocation With Deep Q-Learning for Heterogeneous NOMA Systems," in *IEEE Access,* vol. 9, pp. 41468-41481, 2021.