A Reinforcement Learning Approach to ARQ Feedback-based Multiple Access for Cognitive Radio Networks

Sara A. Attalla[‡], Karim G. Seddik[‡], Amr A. El-Sherif[+†], and Tamer ElBatt[⊥]

[‡]Electronics and Communications Engineering Department, American University in Cairo, New Cairo 11835, Egypt.
[+]Wireless Intelligent Networks Center (WINC), Nile University, Giza 12588, Egypt.
[†]Department of Electrical Engineering, Alexandria University, Alexandria 21544, Egypt.
[⊥] Computer Science and Engineering Department, American University in Cairo, New Cairo 11835, Egypt.
Email: saraattlla@aucegypt.edu, kseddik@aucegypt.edu, aelsherif@nu.edu.eg, tamer.elbatt@aucegypt.edu

Abstract—**In this paper, we propose a reinforcement learning (RL) approach to design an access scheme for secondary users (SUs) in a cognitive radio (CR) network. In the proposed scheme, we introduce a deep Q-network to enable SUs to access the primary user (PU) channel based on their past experience and the *history* of the PU network's automatic repeat request (ARQ) feedback. In essence, SUs cooperate to avoid collisions with other SUs and, more importantly, with the PU network. Since SUs cannot observe the state of the PUs queues, they partially observe the system's state by listening to the PUs' ARQ packets. To model this system, a Partially Observable Markov Decision Process (POMDP) is adopted, and an RL deep Q-network is employed for the SUs to learn the best actions. A comparative study between the proposed scheme with baseline schemes from the literature is presented. We also compare the proposed scheme with the perfect sensing system (which constitutes an upper bound on the performance) and the system exploiting only the last ARQ feedback. Our results show that the proposed RL based access scheme yields comparable performance to the baseline ARQ-based access schemes, yet, with minimal knowledge about the environment compared to the baseline which assumes perfect knowledge of key system parameters, e.g., PUs arrival rates. On the contrary, our proposed scheme autonomously learns these parameters and, hence, dynamically adapts to their variation.**

I. Introduction

The cognitive radio (CR) communication paradigm is a major solution to absorb the dramatic increase in wireless and mobile users. The number of Internet users in the United States is projected to increase from 272 million users in 2017 to 283.5 million users in 2022 [1]. Therefore, there is a growing interest in wireless research pertaining to opportunistic radio spectrum access and interference management.

A cognitive radio network typically follows the cognitive radio cycle shown in Fig. 1 [2]. In our prior research, we focused only on the perception and cognitive adaptation aspects of CR systems [3], [4]. We considered a CR systems in which SUs exploit the automatic repeat request feedback (ARQ), and/or the channel quality indicator (CQI), which are transmitted in the PU network. On another front, the research community has been also focusing on the learning and reasoning aspects of cognitive radio networks [2]. Learning in CR networks aims at collecting and analyzing data over the course of past experience of the CR user. It then utilizes the gathered data for the reasoning step. In [5], a supervised machine learning-based approach is proposed to regulate dynamic handover in CR networks. The proposed scheme is shown to result in better decisions than the traditional cooperative spectrum sensing (CSS) techniques. In [6], the authors proposed a Distributed Spectrum-Aware Clustering (DSAC) scheme for cognitive radio sensor networks (CRSN). The proposed scheme shows superior performance in terms of scalability and stability compared to other baseline schemes.

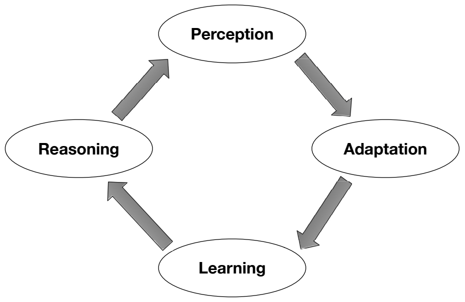

Fig. 1: The Cognitive Cycle [2]

In [7], the authors adopt a reinforcement learning (RL) approach for dynamic channel allocation and power control for spectrum management. The used RL approach achieves less interference to the PU, while keeping a high probability of successful transmissions. The authors in [8] employ Q-learning for channel allocation in a CR system. It is observed that Q-learning results in fast convergence to a near-optimal solution in small scale networks.

In this work, we build upon the work in [9] by applying reinforcement learning to the spectrum access problem. In [9], the authors proposed a protocol based on PU ARQ feedback. The SU observes the history of the PU ARQ feedback. A POMDP models the SU access policy. However, an exact solution for the POMPD was not derived, and the authors proposed a greedy algorithm that enables the SU to use the PU feedback history to decide its channel access actions.

In this work, we adopt reinforcement learning to enable the SU to explore the environment and adapt its actions to best exploit the PU's ARQ feedback. By observing the *history* of the PU ARQ feedback messages, the SU can estimate the number of packets in the PU queue and, consequently, decide on the access probabilities to the PU channel. It is noted that RL enables the SU to achieve this goal without prior knowledge about the PU network, the system model or parameters. Since the SU does not directly observe the state of the PU's queue, the system is modelled using a POMDP, and we adopt a deep Q-network (DQN) to learn the best action for each PU feedback state. The system is then generalized to the multiple SUs case by a cooperative multi-agent RL algorithm. Moreover, our results show that we do not have to exploit the full history of the ARQ feedback bits, and that using only the last ARQ bit to make the SUs' access decision does not cause any significant performance loss.

The rest of the paper is organized as follows. Section II presents the system model along with the POMDP and MAC access policy. The proposed deep RL algorithm and its implementation are presented in section III. Finally, numerical results and conclusions are drawn in section IV and section V, respectively.

II. System Model

In our cognitive radio model, we consider M_s SUs and M_p PUs. Time is slotted, and PUs employ a classic TDMA access scheme such that only one PU accesses the channel in any given time slot. PUs have infinite buffers for storing incoming packets. The packet arrival process is a Bernoulli process with independent and identically distribute arrivals and average arrival rate λ_p packets per time slot. It is assumed that a slot duration is equal to the packet transmission time. Therefore, $0 \leq \lambda_p \leq 1$, otherwise, the PUs queues will not be stable.

SUs access the PUs channel using the slotted ALOHA random access scheme with access probability $a_s()$. The access probability is selected based on the SUs estimate of the state of PUs queues. In our proposed system presented in the next section, the access probabilities will be determined using the DQN. The SUs are assumed to be fully backlogged, i.e., they always have packets to transmit. Finally, we assume an ideal communication channel and adopt the collision channel model. In the collision model, if the PU and SU transmit concurrently, all involved packets will be lost and the PU will re-transmit its lost packet in the next time-slot. At the end of each transmission attempt, the PU receiver sends an ACK or NACK packet, declaring a successful or failed transmission, respectively.

A. The POMDP Model

In an ideal cognitive radio system, the SU would know the state of the PU. The SU would transmit if the PU queue is empty, and remain silent otherwise. However, in a real system, SUs do not have access and cannot observe the exact state of the PU queue. SUs resort to spectrum sensing to detect when the PU is not transmitting. In this work, SUs listen to the PUs ARQ feedback and use this feedback's history to estimate the state of the PUs queues. Since the actual state of the PUs queues cannot be observed, we use a partially observable Markov decision process (POMPD) model to model the system dynamics. Furthermore, since our network consists of M_s SUs, our systems is considered as a cooperative multi-agent system.

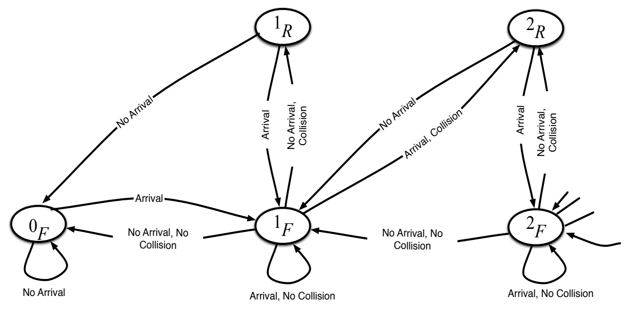

Fig. 2: The PU queue Markov chain model

In our ARQ based system, there are three types of PUs feedback states, namely, ACKs, NACKs, and No-Feedback (No-FB). An ACK denotes a successful transmission, a NACK denotes a failed transmission, and No-FB means that the PU did not attempt any transmission in the last time slot. Based on the observed feedback packets, each SU constructs a belief vector representing the Markov chain states of the PU queue.

The PU queue is modeled using the Markov chain shown in Fig. 2. This MC the state space given by $\mathcal{S} = \{K_D : K = 0, 1, 2, \cdots \text{ and } D \in \{F, R\}\}$, where K denotes the number of packets in the PU queue, and the subscript D refers to the state of the packet at the head of the queue; F refers to the first transmission, while R means that the packet is being retransmitted, after receiving a NACK feedback.

The POMDP is characterized by the tuple $\big(\mathcal{S},\ \mathcal{A},\ P(s'|s,a), R(s,a),\ \Omega,\ O\big)$; the set $\mathcal{S}$ denotes the states of the PU queue's MC, $S = \{\{i_F\}, \{j_R\}\}$, $i = 0, 1, \cdots$ and $j = 1, 2, \cdots$. The set $\mathcal{A}$ denotes the set of each SU actions (which correspond to different SU access probabilities). $P(s'|s,a)$ is the state transition probability function, it denotes the probability to go from state s to state s' given the action a. Below we give some examples for $P(s'|s,a)$ for several values of s, s' and a.

$$\begin{aligned}
&P(i_R|j_F,\ \text{no access}) = 0,\ \forall i,\ j\\
&P(i_F|j_F,\ \text{access}) \quad = 0,\ \forall i,\ j \neq 0\\
&P(1_F|0_F,\ \text{access}) \quad = \lambda_p,\\
&P(1_F|0_F,\ \text{no access}) \quad = \lambda_p,\\
&P(i_R|j_F,\ \text{access}) \quad = \begin{cases} 1-\lambda_p & \text{if } i = j,\ j \neq 0 \\ \lambda_p & \text{if } i = j+1,\ j \neq 0 \\ 0 & \text{otherwise,} \end{cases}\\
&P(i_F|j_R,\ \text{no access}) = \begin{cases} 1-\lambda_p & \text{if } i = j-1 \\ \lambda_p & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}
\end{aligned} \tag{1}$$

The reward function for each SU is defined as follows,

$$R(s,a) = \begin{cases} r_1 & a = \text{access},\ s = 0_F \\ 0 & a = \text{no access},\ \forall s \\ -1 & a = \text{access},\ s \neq 0_F \end{cases}. \tag{2}$$

It is the immediate reward that the SU earns for taking a specific action in a given state. If the queue of the PU is empty, i.e., $s = 0_F$ and the SU accessed the channel, it will gain a positive reward r_1[1]. However, If the queue is not empty and the SU accessed, it receives a penalty for causing a collision. No reward or penalty is received in the case of not accessing the channel.

The set Ω is the set of observations and is given by $\Omega = \{\text{ACK, NACK, No-FB}\}$.

The function $O(o|s', a)$ defines the conditional probability of observing o when action a is applied and the system makes a transition to state s'. It can be calculated as follows [2]:

$$O(o|s' = i_F,\ \text{no access})$$
$$= \begin{cases} 0, & o = \text{NACK and } \forall i_F \\ P_{\text{ACK}}(0_F), & o = \text{ACK},\ i_F = 0 \\ P_{\text{No-FB}}(0_F) = 1 - P_{\text{ACK}}(0_F), & o = \text{No-FB},\ i_F = 0 \\ P_{\text{ACK}}(1_F), & o = \text{ACK},\ i_F = 1 \\ P_{\text{No-FB}}(1_F) = 1 - P_{\text{ACK}}(1_F), & o = \text{No-FB},\ i_F = 1 \\ 1, & o = \text{ACK},\ i_F \geq 2 \\ 0, & o = \text{No-FB},\ i_F \geq 2 \end{cases}. \tag{3}$$

Note that the values of $P_{\text{ACK}}(0_F)$, $P_{\text{No-FB}}(0_F)$, $P_{\text{ACK}}(1_F)$ and $P_{\text{No-FB}}(1_F)$ will not affect our formulation and are mentioned here for the completeness of the POMPD analysis.

$$O(o|i_F,\ \text{access}) = \begin{cases} 0, & \forall o \text{ and } i_F \geq 2 \\ 1, & o = \text{No-FB},\ i_F = 0,1 \\ 0, & o = \text{ACK or NACK},\ i_F = 0,1 \end{cases} \tag{4}$$

$$O(o|i_R,\ \text{no access}) = 0, \quad \forall o \tag{5}$$

$$O(o|i_R, \text{access}) = \begin{cases} 1, & o = \text{NACK} \\ 0, & \text{otherwise} \end{cases}. \tag{6}$$

The belief vector is given by $\mathbf{b}(s_t) = [b(0_F)_t,\ b(1_F)_t,\ b(1_R)_t,\ \cdots]$, where t is the time index. After taking an action a_t and observing o_{t+1}, the new belief at time $(t+1)$ can be calculated by

$$b(s_{t+1}) = \eta O(o_{t+1}|s_{t+1}, a_t) \sum_{s_t \in \mathcal{S}} P(s_{t+1}|s_t, a_t) b(s_t), \tag{7}$$

[1]The value of r_1 is a control parameter in our system that controls the access decisions of the SU and how aggressive these access decisions could be.

[2]By abuse of notations, we set $\Pr(A|B) = 0$ if $\Pr(B) = 0$ (for example we set $\emptyset(o|i_R,\ \text{no access}) = 0$ since $\Pr(i_R,\ \text{no access}) = 0$ under our collision channel model assumption).

where η is a probability normalization factor given by

$$\eta = \frac{1}{\sum_{s_{t+1} \in \mathcal{S}} O(o_{t+1}|s_{t+1}, a_t) \sum_{s_t \in \mathcal{S}} P(s_{t+1}|s_t, a_t) b(s_t)}.$$

B. POMDP MAC Policy

In this section, we describe the MAC policy in which the belief vector at the SU is used to define its action. The mapping between the belief vector and the actions is affected by the current state's reward as well as the expected reward in the subsequent states, which is governed by the dynamics of the Markov chain. Based on a belief vector $\mathbf{b}$ and an action a, the expected reward is given by

$$r(\mathbf{b}, a) = \sum_{s \in \mathcal{S}} b(s) R(s, a). \tag{8}$$

For any belief vector $\mathbf{b}$, the SU access policy π is defined by an action $a_\pi = \pi(\mathbf{b})$, where the policy defines the probability for each action under a certain belief vector. The target is to maximize the accumulated reward over an infinite horizon. Starting with a belief vector $\mathbf{b}_0$, the estimated reward for policy π is given by

$$J^\pi(\mathbf{b}_0) = \sum_{t=0}^{\infty} \gamma^t r(\mathbf{b}_t, a_t) = \sum_{t=0}^{\infty} \gamma^t E\Big[R(s_t, a_t) \mid \mathbf{b}_0, \pi\Big] \tag{9}$$

where $0 \leq \gamma < 1$ is a discount factor. The optimal policy π^* is given by

$$\pi^* = \underset{\pi}{\text{argmax}}\ \ J^\pi(\mathbf{b}_0) \tag{10}$$

where $\mathbf{b}_0$ is the initial belief vector as defined above.

For each belief state, the maximum expected reward value specifies the optimal policy, π^*. It is closely modeled by the best value function V^*, which is the solution for the following Bellman equation

$$V^*(\mathbf{b}) = \max_{a \in \mathcal{A}} \Big[r(\mathbf{b}, a) + \gamma \sum_{o \in \mathcal{O}} O(o \mid \mathbf{b}, a) V^*(\tau(\mathbf{b}, a, o)) \Big], \tag{11}$$

where $\tau(\mathbf{b}, a, o)$ is the belief state transition function.

It should be noted that one might not always be able to construct the belief vector, for instance, if the SU does not know λ_p. Therefore, and unlike the work in [9], we propose to implement an RL based MAC that can efficiently learn model-free systems in which the underlying dynamics are not fully characterized.

III. Deep Q-Network Architecture and Implementation

A. ϵ-Greedy Q-Learning Algorithm

Q-learning [10] is one of the most commonly used RL algorithms. It aims at finding the optimal policy that maximizes the expected reward over a finite or an infinite time horizon. The Q-function (or table) is a representation of the quality of all state-action combinations. The Q-table follows the shape (state, action), and its entries are the Q-values for the corresponding state-action combination. The Q-learning algorithm can be summarized as follows:

1) Initialize all entries in the Q-table to zero.
2) The agent starts at state $s_t \in S$.
3) ϵ-greedy policy is used to select the agent's action. It selects the action with the highest Q-value with probability $1-\epsilon$ and a random action (for exploration) with probability ϵ.
4) The agent applies the selected action $a_t \in A$ to the environment. The instantaneous reward $r_t \sim R(s_t, a_t)$ is determined and the environment moves to the next state $s_{t+1} \sim P(s'|a, s)$.
5) The Q-table is updated using the Bellman equation:

$$\begin{aligned} Q(s_t, a_t) =& (1-\alpha)Q(s_t, a_t) \\ &+ \alpha(r_t + \gamma \max_a Q(s_{t+1}, a)), \end{aligned} \tag{12}$$

where $\alpha \in (0, 1)$ is the learning rate, $\gamma \in (0, 1)$ is the discount factor, and $\max_a Q(s_{t+1}, a)$ is the maximum expected reward for all possible actions at the next state $s_t + 1$

It was proven that the Q-learning algorithm converges to the optimal Q-function if all state-action pairs can be visited infinitely often [10], [11].

B. Deep Q-Learning (DQN) Algorithm

The size of the Q-table constructed by the Q-learning algorithm described above can be very large depending on the number of states of the environment and the number of possible actions per state. Furthermore, it can be seen that the Q-value for a non-visited state cannot be inferred from already visited states. To alleviate these shortcomings, the use of deep Q-learning has been proposed. In deep Q-learning, a neural network is used to approximate the Q-function. The input to the neural network is the state of the environment and the outputs are the Q-values for all possible actions.

In Deep Q-learning, two neural networks are used, namely, the predicted network and the target network. Both networks are parameterized by θ to represent $Q(s, a; \theta)$. The input to the predicted network is the observation s and the output are the Q-values $Q(s, a, \theta_i)$ for each action $a \in A$, where θ_i are the parameters of the predicted network. ϵ-greedy policy is used to select the action that maximizes $Q(s, a, \theta_i)$ with probability $1-\epsilon$ and a random action for exploration with probability ϵ. The target network aims to minimize the loss:

$$L_i(\theta_i) = E[(r_t + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, u, \theta_i))^2], \tag{13}$$

where θ_i^- are the target network parameters that are used to update the parameters for the predicted network [12].

In this paper, we consider the case of multiple agents representing the multiple SUs in the network. In a multi-agent environment, each agent independently learns its own Q-function and selects its action. Agents can cooperate by sharing information needed to select the optimum action for each agent. In this paper, we use the cooperative Q-learning algorithm described as follows [13]:

1) Agent n observes the state s_t of the environment.

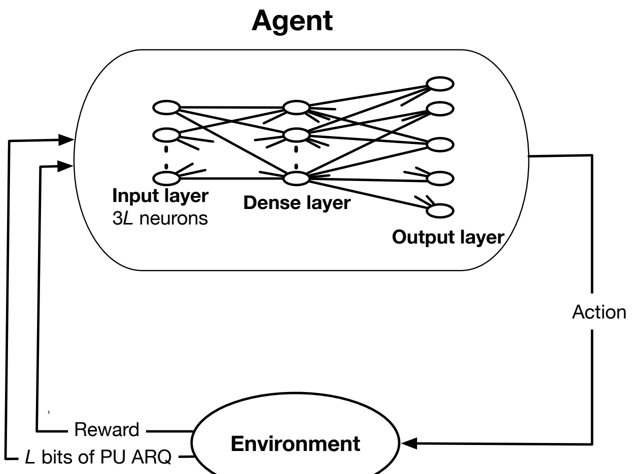

Fig. 3: The SU Deep RL Architecture.

2) Agent n shares the row of its Q-table that corresponds to the current state s_t, $Q_n(s_t)$, with all other cooperating agents.
3) Agent n determines its action $\hat{a}_n$ based on

$$\hat{a}_n = \arg\max_{\hat{a}_{n'}}\left(\sum_{1<n'<N} Q_{n'}^t(s_t)\right) \tag{14}$$

4) Agent n receives a reward r_n corresponding to action $\hat{a}_n$ and state s_t.
5) Agent n updates the Q-value ($Q(s_t, \hat{a}_n)$) using (12), and the process is then repeated

Fig. 3 depicts the DQN model used in this paper. The SU (agent) is a deep neural network that has the PU's ARQ feedback (observation) as input. The output of the deep RL model is the Q-value for each action (an action is the probability with which the SU decides to access the PU channel in the current time slot). The SU then chooses the action with the highest Q-value. In the case of the ϵ-greedy policy, the SU chooses the action with the highest Q-value or a random action with probability $1-\epsilon$ and ϵ, respectively.

The structure of the agent's deep neural network model is shown in Fig. 3. The agent's model is implemented using the Keras functional API [14] and using TensorFlow back-end [15]. It consists of an input layer and two fully connected layers. The inner layers use the sigmoid activation function, and the final output layer uses a linear activation function.

The overall RL environment is implemented using the OpenAI Gym toolkit [16]. Table I states the values of the parameters used in our implementation. It is worth noting that the values are selected based on trial and error to yield the best performance possible.

IV. Numerical Results

In this section, we present the performance results of our proposed feedback-based deep RL channel access scheme. We consider the SUs throughput as the performance metric. For our RL based scheme, we consider the case where the system

TABLE I: List of network parameters.

Parameter	Value
The number of episodes (M)	10000
The discount factor (γ)	0.85
Initial value in ϵ -greedy exploration($\epsilon_{initial}$)	1
Final value in ϵ -greedy exploration (ϵ_{min})	0.01
Decay factor of ϵ -greedy exploration(ϵ_{decay})	0.995
Learning rate (α)	0.001
Agent history length (L)	1-5

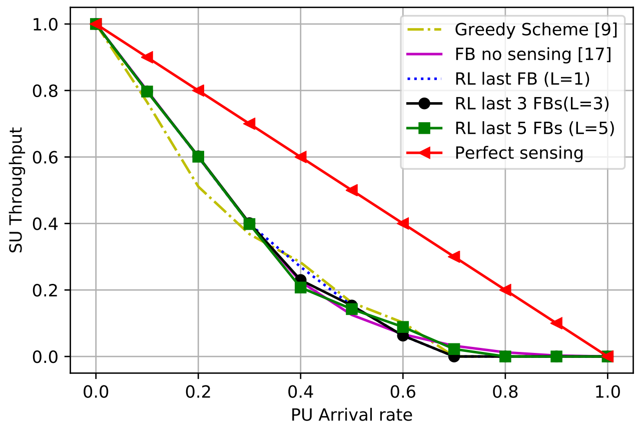

Fig. 4: SU throughput for different PU feedback history length (L) without spectrum sensing.
$(M_s = 1, M_p = 1)$

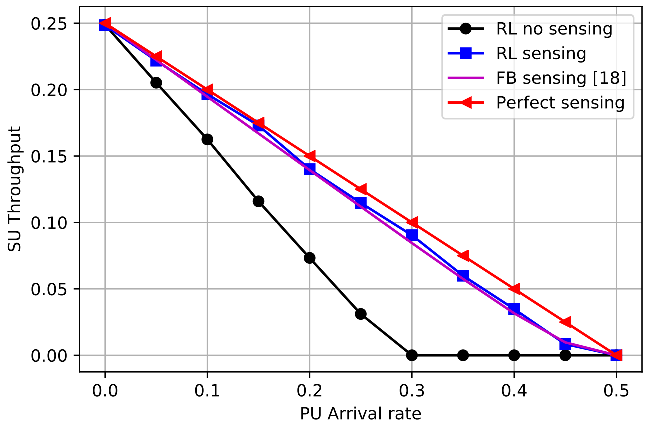

Fig. 5: SU throughput (per user) for different spectrum access schemes
$(M_s = 2, M_p = 2, p_d = 0.9, p_f = 0.1, L = 1)$.

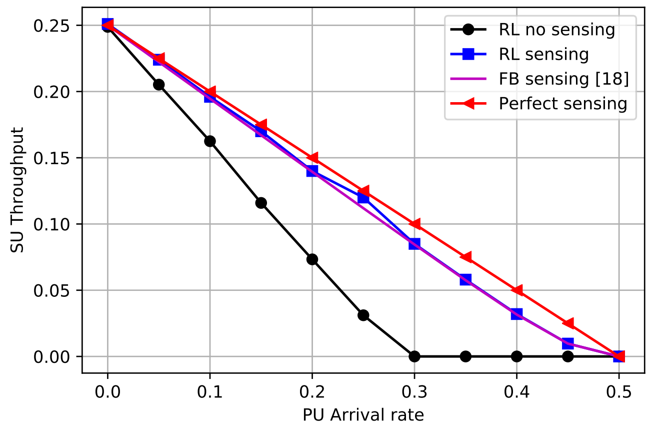

Fig. 6: SU throughput (per user) for different spectrum access schemes
$(M_s = 2, M_p = 2, p_d = 0.6, p_f = 0.3, L = 1)$.

relies only on the information collected from the PUs' ARQ feedback without sensing the PUs' channel status and compare it to the works in [9] and [17]. We also consider the case where the system uses spectrum sensing in addition to the PUs' ARQ feedback and compare it with the scheme presented in [18]. Moreover, all systems are compared with the ideal case (which constitutes an upper bound on performance) where SUs can perfectly sense the PUs presence, hence, no collisions between PUs and SUs can take place, and SUs can exploit all idle time slots. We consider different number of PUs, M_p and SUs, M_s. Finally, it is assumed that the SUs action space has five actions. That is, a SU can access the PU's channel in a given time slot with an $a_s \in \{0, 0.25, 0.50, 0.75, 1\}$.

In Fig. 4, we investigate the effect of the length of the PU feedback (FB) history (L) collected by the SU on the achievable SU throughput. Here we consider only one SU and one PU with no spectrum sensing. The SU bases its access decision on the last $L \in 1, 3, 5$ ARQ feedback packets overheard from the PU. The results reveal that increasing the length of the feedback history used has a minor effect on the SU throughput. Therefore, it can be concluded that, based on the studied system setting, the last feedback packet overheard from the PU has the predominant effect on the SU throughput.

Fig. 4 also compares the performance of our RL-based scheme with the systems proposed in [17] and in [9] which exploits PUs' ARQ feedback for SUs channel access decisions. It can be seen that the performance of the three systems is comparable. However, it should be noted that in both [17] and [9] the SU is assumed to perfectly know the number of PUs and their arrival rates in addition to observing the ARQ feedback messages. On the contrary, our scheme assumes that SUs observe only the PUs ARQ feedback without any prior information about the number of PUs or any other system parameters.

In Fig. 5 and Fig. 6 we investigate the performance of our

RL-based access scheme using hard-decision energy detection for spectrum sensing. Our scheme is compared to the access scheme proposed in [18] which exploits the last PU's ARQ feedback and employs a more sophisticated soft-sensing energy detector. Since our system is based on hard-decision spectrum sensing, we had to rework the soft-sensing based SUs' throughput analysis of [18]. The hard-decision based SUs' throughput of the system proposed in [18] for the case of two SUs is given by

$$\mu_s = a_s(1-p_f)(p_f + (1-a_s)(1-p_f)) \\ (\lambda_p a_s^2 p_d^2 - 2\lambda_p a_s^2 p_d + \lambda_p a_s^2 + 2\lambda_p a_s p_d - 2\lambda_p a_s - \lambda_p + 1), \tag{15}$$

where p_d is the detection probability of the spectrum sensor and p_f is the probability of false alarm.
The access probability that maximizes the SU service rate a_s^* is obtained by differentiating the SU service rate in equation (15) with respect to a_s and equating the result to zero.

Fig. 5 and Fig. 6 depict the per SU throughput for our RL-based access scheme using the hard-decision spectrum sensing (RL sensing) using only the last PU's ARQ feedback ($L = 1$). The results are compared to the case of no spectrum sensing (RL no sensing), the system proposed in [18] (FB sensing), and the ideal system (perfect sensing). The system is composed of $M_s = 2$ SUs and $M_p = 2$ PUs. The effect of the performance of the spectrum sensing scheme on the SUs throughput is investigated by setting the probability of detection $p_d = 0.9$ and probability of false alarm $p_f = 0.1$ in Fig. 5, and setting $p_d = 0.6$ and $p_f = 0.3$ in Fig. 6.

From Fig. 5 and Fig. 6, it can be seen that there is a significant improvement in SUs throughput by adding the hard-decision spectrum sensing to our RL-based access scheme. It is evident by now that exploiting the information embedded in the PUs ARQ feedback alongside spectrum sensing, both, our RL-based scheme and the scheme in [18] have almost the same performance, which is very close to the ideal scheme of "perfect sensing". It is also noted that our RL based access scheme is robust to the changes in the spectrum sensing parameters. The achieved throughput is almost unaffected by the degradation of the detection and false alarm rates of the energy detector between Fig. Fig. 5 and Fig. 6.

It is important to note that our proposed RL-based access scheme does not have any information about the PUs, except for the overheard ARQ feedback. However, in [18] it is assumed that the SUs know the PUs arrival rates in addition to overhearing the ARQ feedback messages. The information about the PUs arrival rate is not expected to be always available to the SUs. Additionally, this arrival rate can vary over time, which will force the system in [18] to recalculate its access probabilities while our RL-based scheme will be able to learn from the environment and adapt seamlessly to any change in PUs arrival rates.

V. Conclusions

In this paper, the design of a deep Q-learning based spectrum access scheme for cognitive radio networks is presented. In the proposed scheme, SUs overhear the ARQ feedback available in the PUs' network and exploit it to learn the PUs behavior. Since the SUs observe only the PUs' ARQ feedback and have no information about the PUs packet arrival rates or the states of their queues, the system is modeled as a partially observable Markov decision process (POMDP). The proposed deep Q-learning access scheme is used to solve this POMDP and find the best SUs' actions (channel access probabilities) based on the observed PUs' ARQ feedback and past experiences. The performance of the proposed scheme is shown to be on par with that of other feedback-based access schemes with more sophisticated channel sensing algorithms, with the added strength of only having partial information about the PUs and the primary network.

References

[1] Statista, "United states: number of internet users 2017-2023," 2019.

[2] L. Gavrilovska, V. Atanasovski, I. Macaluso, and L. DaSilva, "Learning and reasoning in cognitive radio networks," vol. 15, no. 4, pp. 1761 – 1777, March 2013.

[3] S.A. Attalla, K.G. Seddik, A.A. El-Sherif, and S.I Rabia, "Soft-sensing cqi feedback-based access scheme in cognitive radio networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 486 – 499, April. 2018.

[4] S.A. Attalla, K.G. Seddik, A.A. El-Sherif, and S.I. Rabia, "Hybrid feedback-based access scheme for cognitive radio systems," in *IEEE Global Communications Conference*, Singapore, Dec. 2017.

[5] H. Anandakumar and K. Umamaheswari, "Supervised machine learning techniques in cognitive radio networks during cooperative spectrum handovers," vol. 20, no. 2, pp. 1505–1515, June. 2017.

[6] H. Zhang, Z. Zhang, H. Dai, R. Yin, and X. Chen, "Distributed spectrum-aware clustering in cognitive radio sensor networks," in *Proc. IEEE Global Telecommun. Conf.*, Dec. 2011, pp. 1–6.

[7] C. Wu, K. Chowdhury, M. Di Felice, and W. Meleis, "Spectrum management of cognitive radio using multi-agent reinforcement learning," in *9th International Conference on Autonomous Agents and Multiagent Systems*, 2010, pp. 1705–1712.

[8] N. Hosey, S. Bergin, and I.D. Macaluso, ," in *Q-Learning for Cognitive Radios*, 2009.

[9] K.G. Seddik and A.A. El-Sherif, "A pomdp framework for cognitive mac based on primary feedback exploitation," in *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Atlanta, GA, USA, Dec. 2014.

[10] J. C. H. Christopher and P. Dayan, "Technical note q-learning," *Journal of Machine Learning*, vol. 8, no. 3-4, pp. 279– 292, 1992.

[11] F. S. Melo, "Convergence of q-learning: a simple proof," , Institute for Systems and Robotics, Instituto Superior Técnico, Lisboa, PORTUGAL, 2001.

[12] J.N. Foerster, Y.M. Assael, N. Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *NIPS'16 Proceedings of the 30th International Conference on Neural Information Processing Systems*, Barcelona, Spain, Dec. 2016, pp. 2145–2153.

[13] Hussein Saad, Amr Mohamed, and Tamer Elbatt, "Cooperative q-learning techniques for distributed online power allocation in femtocell networks," *Wiley Online Library*, vol. 15, no. 15, 2014.

[14] Keras Documentation, "https://keras.io/," .

[15] TensorFlow, "https://www.tensorflow.org/guide/keras," .

[16] OpenAI Gym, "https://gym.openai.com/," .

[17] K.G. Seddik, A.K. Sultan, A.A. El-Sherif, and A.M. Arafa, "A feedback-based access scheme for cognitive radio systems," in *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, San Francisco, CA, June 2011.

[18] A.M. Arafa, K.G. Seddik, A.K. Sultan, T. ElBatt, and A.A. El-Sherif, "A feedback-soft sensing-based access scheme for cognitive radio networks," *IEEE Transactions on Wireless Communication*, vol. 12, no. 7, pp. 3226–3237, July 2013.