Deep Q-Learning with Multiband Sensing for Dynamic Spectrum Access

Ha Q. Nguyen, Binh T. Nguyen, Trung Q. Dong, Dat T. Ngo and Tuan A. Nguyen
Viettel Research and Development Institute
Hoa Lac High-tech Park, Hanoi, Vietnam
Email: {hanq8, binhnt27, trungdq8, datnt72, tuanna118}@viettel.com.vn

***Abstract*—We study a dynamic spectrum access situation where, in each time slot, a single cognitive agent decides to either stay idle or access one of the N frequency channels based on its sensing of the whole spectrum. The channels are occupied or vacant according to N independent nonidentical 2-state Markov chains. We prove that the optimal access policy can easily be found if the state transition probabilities of all channels are known. When the agent has no knowledge about the channel model, we propose to use the deep Q-learning method to learn a state-action value function that determines an access policy from the observed states of all channels. In this method, the optimal Q-function is approximated with a neural network of all dense layers that is trained via experience replay. We demonstrate through experiments that the learning-based policies consistently achieve performances that are close to the optimal ones.**

***Index Terms*—cognitive radio, dynamic spectrum access, reinforcement learning, wide-band spectrum sensing, Q-learning, deep Q-network, Markov decision process**

I. Introduction

Cognitive radio networks [1]–[3] are believed to be the next big step of wireless communication, in which the dynamic spectrum access (DSA) plays a central role. In the DSA model, the frequency subbands (channels) of the radio spectrum are assigned to primary users as usual, while the secondary users (a.k.a. cognitive radios) keep sensing and monitoring the radio environment to access the spectrum holes for communication. This paradigm will help improve the spectrum utilization and ease the spectrum scarcity—a problem that has become more and more severe in the boom of wireless devices and services. To realize the DSA, the secondary users (SUs) must learn the usage behaviors of the primary users (PUs) by interacting with the radio environment in a trial-and-error manner, a mechanism that perfectly fits into the reinforcement learning framework [4]. Especially, the huge success of deep reinforcement learning in game play [5], [6] has provided a new set of powerful tools to solve the DSA problem.

Most of existing works on DSA [7]–[14] has adopted the deep reinforcement learning approach in which the Q-function that measures the value of each state-action pair is approximated by a neural network. However, all of these methods only dealt with narrowband spectrum sensing. This means that each SU can only sense a single frequency subband at a time, which make it difficult to learn the occupancy pattern of the spectrum. To the best of our knowledge, we are the first to consider a DSA model where the SU is capable of sensing multiple channels in every time slot. This assumption can be realized via wideband spectrum sensing techniques (see [15] and the references therein).

To simplify the problem while still illustrating our main ideas, we model total N channels as N independent Markov chains whose states can either be vacant or occupied. Based on the *current* occupancy pattern of the *whole* spectrum, a single SU takes one of the $N + 1$ actions in the next time slot: access one of the N channels or just wait. The goal of the SU is to minimize the access collisions with the PUs. We emphasize that the SU actually consists of a transmitter and a receiver; the synchronization between them, however, is beyond the scope of this paper. We always assume that the spectrum access decision made in every time slot is shared between the SU transmitter and receiver. The contributions of the paper are summarized as follows:

- Formalism of the DSA problem with multiband sensing as a Markov Decision Process (MDP).
- An analytical solution for the optimal access policy when the MDP are fully known.
- A neural network based algorithm to learn the Q-function that defines an access policy in case the state transition probabilities of the MDP are unknown to the SU.
- Comparison of the learned policy with the optimal policy. Simulations suggest that the gap between these policies are relatively small.

The rest of the paper is outlined as follows: In Sec. II, we provide a formalism of the DSA problem and some mathematical tools. In Sec. III, we describe and prove the optimal policy for the DSA. In Sec. IV, we discuss the model-free deep Q-learning algorithm. Simulation results are provided in Sec. V. Finally, Sec. VI draws some conclusion.

II. Problem Formulation

The proposed DSA model consists of N PUs operating over N channels in the spectrum of interest and a single SU who opportunistically searches for idle spectrum blocks to access. In every time slot, the SU is aware of the states of *all* N channels and chooses one channel to access or decides *not* to access any of them. The task of the SU is to act intelligently in order to maximize its access throughput while minimizing the collisions with the PUs. In what follows, we describe the channel model, formulate the DSA problem as a Markov decision process, and discuss the important Bellman equations.

978-1-5386-5191-9/18/$31.00 ©2018 IEEE

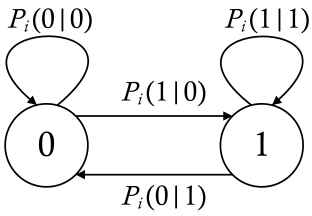


Fig. 1: State transition diagram for each channel i.

A. Channel Model

We model the status of each channel as either 0 (VACANT) or 1 (OCCUPIED). Also, assume that the vacancy/occupancy pattern of the N channels evolves according to N independent 2-state Markov chains. In particular, for $1 \leq i \leq N$, let $s_{t,i}$ denote the state of channel i at time slot t. Each channel i is then characterized by a 2×2 transition probability matrix

$$\mathbf{P}_i = \begin{bmatrix} P_i(0|0) & P_i(1|0) \\ P_i(0|1) & P_i(1|1) \end{bmatrix}, \tag{1}$$

where $P_i(u|v)$, for $(u, v) \in \{0, 1\}^2$, is defined as

$$P_i(u|v) = \text{Prob}\left(s_{t+1,i} = u | s_{t,i} = v\right), \forall t \geq 0. \tag{2}$$

Note that all of these transition matrices are independent of the time slot t and *different* across N channels. Fig. 1 illustrates the state transition of the Markov chain for each channel.

B. Markov Decision Process

As in the reinforcement learning literature [4], we formulate the DSA problem through a Markov Decision Process (MDP) that is characterized by a tuple $(\mathcal{S}, \mathcal{A}, \{\mathbf{P}_i\}, r, \gamma)$. Each component of the MDP is specified below.

1) State space

$$\mathcal{S} = \{\boldsymbol{s} = (s_1, \ldots, s_N) \,|\, s_i \in \{0, 1\}\}. \tag{3}$$

2) Action space

$$\mathcal{A} = \{0, 1, 2, \ldots, N\}. \tag{4}$$

For an action $a \in \mathcal{A}$, the SU accesses channel a if $a \geq 1$ or waits if $a = 0$.

3) State transition probabilities

$$P\left(\boldsymbol{s}'|\boldsymbol{s}\right) = \prod_{i=1}^{N} P_i(s_i'|s_i), \quad \text{for } \boldsymbol{s}', \boldsymbol{s} \in \mathcal{S}. \tag{5}$$

4) Reward function $r : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is given by

$$r(a, \boldsymbol{s}') = \begin{cases} 1 - 2s_a', & \text{if } 1 \leq a \leq N \\ 0, & \text{if } a = 0. \end{cases} \tag{6}$$

The reward at time slot t is defined as

$$R_t = r(a_t, \boldsymbol{s}_{t+1}). \tag{7}$$

This means that, at time t, the SU gets a score of $+1$ if it accesses a channel that will be vacant at time $t+1$, a score of -1 if it accesses a channel that will be occupied at time $t+1$, or a score of 0 if it decides to wait.

5) Discount factor γ is a real number in the interval $(0, 1)$.

Our goal is to learn a (deterministic) policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the total expected reward

$$V_\pi(\boldsymbol{s}) = \mathrm{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t \,|\, \boldsymbol{s}_0 = \boldsymbol{s}\right], \tag{8}$$

for all initial state $s \in \mathcal{S}$. In (8), E_π denotes the expectation under the condition that policy π is carried. We refer to $V_\pi(\boldsymbol{s})$ as the *state value function* (or V-function) associated with policy π. The optimal state value function V^* is defined as

$$V^*(\boldsymbol{s}) = \max_\pi V_\pi(\boldsymbol{s}), \quad \text{for } \boldsymbol{s} \in \mathcal{S}. \tag{9}$$

Another useful function is the *state-action value function*

$$Q_\pi(\boldsymbol{s}, a) = \mathrm{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t \,|\, \boldsymbol{s}_0 = \boldsymbol{s}, a_0 = a\right], \tag{10}$$

which measures the goodness of every state-action pair. We also refer to this function as Q-function. The optimal state-action value function Q^* is defined according to

$$Q^*(\boldsymbol{s}, a) = \max_\pi Q_\pi(\boldsymbol{s}, a), \quad \text{for } \boldsymbol{s} \in \mathcal{S}, a \in \mathcal{A}. \tag{11}$$

The main advantage of using the state-action value function over the state value function is that the optimal policy π^* can simply be found via the optimal Q-function as

$$\pi^*(\boldsymbol{s}) = \arg\max_{a \in \mathcal{A}} Q^*(\boldsymbol{s}, a). \tag{12}$$

Note that, once the optimal Q-function (or an approximate of it) is known, (12) does not require any knowledge about the transition probabilities of the MDP. Thus, the Q-function is commonly used in model-free reinforcement learning methods.

While sampling the MDP, for each time slot t, we refer to the tuple $(\boldsymbol{s}_t, a_t, r_t, \boldsymbol{s}_{t+1})$ as an *experience*. Note that r_t here is the actual reward (not the random variable R_t) collected by running the MDP.

C. The Bellman Equations

Most reinforcement learning algorithms for solving the MDP rely on recursive estimation of the optimal value functions. This can be done via the Bellman equation

$$V^*(\boldsymbol{s}) = \max_a \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s})\left(r(a, \boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')\right) \tag{13}$$

and its counterpart for the optimal Q-function

$$Q^*(\boldsymbol{s}, a) = \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s})(r(a, \boldsymbol{s}') + \gamma \max_{a' \in \mathcal{A}} Q^*(\boldsymbol{s}', a')). \tag{14}$$

It is noteworthy that these equations are modified for the particular MDP described in Sec. II-B and, thus, slightly different from the standard Bellman equations in the literature [4].

III. Optimal Policy

In this section, we provide an analytical solution for the MDP when all the state transition probabilities are known to the SU. This is also called the *model-based* planning. We start by proving that the Bellman equation for optimal V-functions has a unique solution and then point out the optimal policy whose V-function satisfy the Bellman equation. This optimal policy can be obtained by a simple comparison of the transition probabilities at each time slot.

Proposition 1. *For $0 < \gamma < 1$, the Bellman equation* (13) *has a unique solution.*

Proof: We show by contradiction. Suppose there exist 2 different state value functions V_1 and V_2 that both satisfy the Bellman equation, i.e.

$$V_1(\boldsymbol{s}) = \max_a \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s}) r(a, \boldsymbol{s}') + \gamma \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s}) V_1(\boldsymbol{s}')$$
$$V_2(\boldsymbol{s}) = \max_a \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s}) r(a, \boldsymbol{s}') + \gamma \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s}) V_2(\boldsymbol{s}').$$

Subtracting side by side of the above equations, we obtain

$$D(\boldsymbol{s}) = \gamma \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s}) D(\boldsymbol{s}'), \quad \forall \boldsymbol{s} \in \mathcal{S}, \tag{15}$$

where $D(\boldsymbol{s}) = V_1(\boldsymbol{s}) - V_2(\boldsymbol{s})$. Let $\boldsymbol{s}^*$ be a state such that

$$\boldsymbol{s}^* = \arg\max_{\boldsymbol{s} \in \mathcal{S}} |D(\boldsymbol{s})|. \tag{16}$$

Since $D(\boldsymbol{s})$ is not identical to the zero-function, it must be that $D(\boldsymbol{s}^*) \neq 0$. Then, applying (15) to $\boldsymbol{s}^*$ yields

$$D(\boldsymbol{s}^*) = \gamma \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s}^*) D(\boldsymbol{s}'). \tag{17}$$

Combining (17) with (16) and that $0 < \gamma < 1$ leads to

$$\begin{aligned} |D(\boldsymbol{s}^*)| &\leq \gamma \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s}^*) \, |D(\boldsymbol{s}')| \\ &\leq \gamma |D(\boldsymbol{s}^*)| \cdot \underbrace{\sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s}^*)}_{1} \\ &< |D(\boldsymbol{s}^*)|, \end{aligned}$$

which is contradictory. This means that the Bellman equation (13) has at most one solution. ∎

We are now ready to state the main result of this section.

Theorem 1. *The policy π^* defined by*

$$\pi^*(\boldsymbol{s}) = \begin{cases} \arg\min_{a \geq 1} P_a(1|s_a), & \text{if } \min_{a \geq 1} P_a(1|s_a) < \frac{1}{2} \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

is optimal for the MDP described in Sec. II-B.

Proof: For the sake of convenience, let us put

$$\begin{aligned} \mathcal{S}_1 &= \left\{ \boldsymbol{s} \in \mathcal{S} : \min_{a \geq 1} P_a(1|s_a) < 1/2 \right\}, \\ \mathcal{S}_2 &= \mathcal{S} \setminus \mathcal{S}_1, \\ \check{\mathcal{S}} &= \{0,1\}^{N-1}, \\ \mathcal{N}_{-a} &= \{1, 2, \ldots, N\} \setminus \{a\}, \\ s_{-a} &= (s_i)_{i \in \mathcal{N}_{-a}}. \end{aligned}$$

We need to prove that the state value function V_{π^*} associated with the policy π^* is optimal. From Proposition 1, it suffices to show that V_{π^*} satisfies the Bellman equation (13). We first rewrite (13) as

$$V^*(\boldsymbol{s}) = G^*(\boldsymbol{s}) + \gamma \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s}) V^*(\boldsymbol{s}'), \tag{19}$$

where

$$\begin{aligned} G^*(\boldsymbol{s}) &= \max \left\{ 0, \max_{a \geq 1} \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s})(1 - 2s'_a) \right\} \\ &= \max \left\{ 0, 1 - 2 \min_{a \geq 1} \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s}) \, s'_a \right\}. \end{aligned} \tag{20}$$

We now perform the following manipulations:

$$\begin{aligned} \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s}) \, s'_a &= \sum_{\boldsymbol{s}' \in \mathcal{S}} \prod_{i=1}^{N} P_i(s'_i|s_i) \cdot s'_a && (21) \\ &= \sum_{\boldsymbol{s}' \in \mathcal{S}: s'_a = 1} \prod_{i=1}^{N} P_i(s'_i|s_i) \cdot 1 && (22) \\ &= P_a(1|s_a) \sum_{\boldsymbol{s}'_{-a} \in \check{\mathcal{S}}} \prod_{i \in \mathcal{N}_{-a}} P_i(s'_i|s_i) && (23) \\ &= P_a(1|s_a) \sum_{\boldsymbol{s}'_{-a} \in \check{\mathcal{S}}} \Pr\left(\boldsymbol{s}'_{-a}|\boldsymbol{s}_{-a}\right) && (24) \\ &= P_a(1|s_a), \quad \forall a \in \mathcal{A}. && (25) \end{aligned}$$

where (21) follows from the independence of the channels; (22) is due to the fact that s'_a can only takes value in $\{0, 1\}$; (25) is a consequence of the law of total probability. By plugging (25) into (20), we get

$$G^*(\boldsymbol{s}) = \max \left\{ 0, 1 - 2 \min_{a \geq 1} P_a(1|s_a) \right\}. \tag{26}$$

Next, let $G_{\pi^*}(\boldsymbol{s})$ be the expected immediate reward obtained by carrying the policy π^* at state $\boldsymbol{s}$. Put $a^* = \pi^*(\boldsymbol{s})$. We are left to show that $G_{\pi^*}(\boldsymbol{s}) = G^*(\boldsymbol{s})$ By the definition of π^*, for $\boldsymbol{s} \in \mathcal{S}_1$, we have that

$$\begin{aligned} G_{\pi^*}(\boldsymbol{s}) &= \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s}) \; r(a^*, \boldsymbol{s}') && (27) \\ &= \sum_{\boldsymbol{s}' \in \mathcal{S}} P(\boldsymbol{s}'|\boldsymbol{s}) \, (1 - 2s'_{a^*}) && (28) \\ &= 1 - 2P_{a^*}(1|s_{a^*}) && (29) \\ &= 1 - 2 \min_{a \geq 1} P_a(1|s_a), \quad \forall \boldsymbol{s} \in \mathcal{S}_1 && (30) \end{aligned}$$

where (28) follows from definition (6) of the reward function; (29) is a consequence of (25); and (30) follows from the definition of a^*. On the other hand, it is clear that

$$G_{\pi^*}(\boldsymbol{s}) = 0, \quad \forall s \in \mathcal{S}_2. \tag{31}$$

Combining (30) with (31) gives

$$G_{\pi^*}(\boldsymbol{s}) = \max\left\{0, 1 - 2\min_{a\geq 1} P_a\left(1|s_a\right)\right\}, \forall \boldsymbol{s} \in \mathcal{S}. \tag{32}$$

By comparing (32) with (26), we deduce that $G_{\pi^*}(\boldsymbol{s}) = G^*(\boldsymbol{s})$, and so the value function V_{π^*} associated with policy π^* satisfies the Bellman equation (19). By Proposition 1, V_{π^*} must be identical to the optimal value function V^* defined in (9), which finally shows the optimality of the policy π^*. ■

IV. Deep Q-Learning Algorithm

This section deals with the *model-free* learning. We propose a neural network based reinforcement learning algorithm to solve the MDP under the practical assumption that the state transitions probabilities are unknown. One of the most successful reinforcement learning algorithms is the Q-learning [16] where the estimation of the optimal Q-function is based on 1-step look-ahead (bootstrapping). The classical Q-learning is table-based, i.e. the values of the Q-function are stored in a table of size $|\mathcal{S}| \times |\mathcal{A}|$. However, when the size of the state space gets large, the table-based Q-learning is quickly infeasible. For example, with $N = 20$ channels, the Q-table will be of size 104857×21, which requires a giant number of iterations to converge. In these cases, we adapt the deep Q-learning approach in [5] to approximate the Q-function by a neural network $Q_{\boldsymbol{\theta}}$ called Deep Q-Network (DQN) and train its weights $\boldsymbol{\theta}$ using experience replay. From experiments, we propose to use a feed-forward network that consists of 3 fully connected layers, each of which has N neurons. The input to the DQN is a state $\boldsymbol{s}$ of size $1 \times N$. The output of the network is a vector of size $1 \times (N + 1)$ that contains the values of the Q-function with respect to state $\boldsymbol{s}$ and each of the $N + 1$ actions. In all hidden layers, we use the hyperbolic tangent activation function

$$\tanh(x) = \left(1 - e^{-2x}\right) / \left(1 + e^{-2x}\right). \tag{33}$$

The architecture of the proposed DQN is described in Fig. 2.

To train the DQN, experiences are first accumulated in a memory using ϵ-greedy policies: for a state $\boldsymbol{s}_t$, an action a_t is taken randomly with probability ϵ_t, and greedily with respect to the current DQN with probability $1 - \epsilon_t$. Then, when the memory is long enough, at every time slot t, a minibatch of B experiences $\{(\boldsymbol{s}_i, a_i, r_i, \boldsymbol{s}'_i)\}_{i\in\mathcal{B}_t}$ is randomly sampled from the memory. Here, $\mathcal{B}_t$ is a random subset of the experience indices currently in available in the memory. We then update the weights $\boldsymbol{\theta}$ of the DQN $Q_{\boldsymbol{\theta}}$ to minimize the loss function

$$L_t(\boldsymbol{\theta}) = \frac{1}{B}\sum_{i\in\mathcal{B}_t}\left(r_i + \gamma \max_a Q_{\boldsymbol{\theta}}(\boldsymbol{s}'_i, a) - Q_{\boldsymbol{\theta}}(\boldsymbol{s}_i, a_i)\right)^2. \tag{34}$$

Note that this loss function is based on the Bellman equation for the optimal Q-function in (14). In the training phase, the exploration rate ϵ_t will be linearly decreased after each time slot to balance the exploration-exploitation trade-off. The above training procedure is detailed in Algorithm 1.

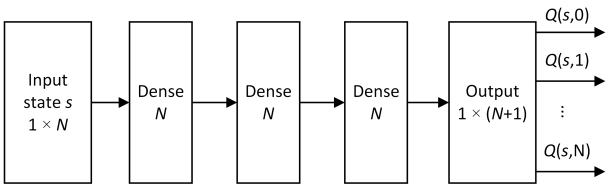


Fig. 2: Diagram of the Deep Q-network for DSA.

Algorithm 1 Deep Q-Learning for DSA

Input: transition matrices $\{\mathbf{P}_i\}_{i=1}^N$, number of time slots T, minibatch size B, discount rate γ, learning rate α.
Output: weights $\boldsymbol{\theta}$ of the DQN.

Initialize channel states $\boldsymbol{s} \in \{0, 1\}^N$
Initialize replay memory $\mathcal{M}$
Initialize network $Q_{\boldsymbol{\theta}}$ with random weights $\boldsymbol{\theta}$
Initialize exploration rate $\epsilon_0 \in (0, 1)$
for $t = 0, T$ **do**
  $\epsilon = \epsilon_0(T - t)/T$
  With probability ϵ randomly choose $a \in \{0, \ldots, N\}$, otherwise choose $a = \arg\max_{a'} Q_{\boldsymbol{\theta}}(\boldsymbol{s}, a')$
  Generate next states $\boldsymbol{s}'$ randomly from $\boldsymbol{s}$ and $\{\mathbf{P}_i\}_{i=1}^N$
  Collect reward $r = 1 - 2s'_a$ if $a \geq 1$, otherwise $r = 0$
  Store experience $(\boldsymbol{s}, a, r, \boldsymbol{s}')$ in $\mathcal{M}$
  Set $\boldsymbol{s} = \boldsymbol{s}'$
  if $t \geq B$ **then**
    Sample from $\mathcal{M}$ a minibatch $\{(\boldsymbol{s}_i, a_i, r_i, \boldsymbol{s}'_i)\}_{i\in\mathcal{B}_t}$
    Form the loss function $L_t(\boldsymbol{\theta})$ as in (34)
    Update $\boldsymbol{\theta}$ with learning rate α to minimize $L_t(\boldsymbol{\theta})$
  end if
end for
return $\boldsymbol{\theta}$

V. Simulations

We implemented Algorithm 1 in Python with Keras library and Tensor Flow backend. The simulations were performed for the DSA with $N = 20$ channels and for 5 scenarios with 5 different sets of transition probability matrices $\{\mathbf{P}_i\}_{i=1}^{20}$, which are randomly generated. In all scenarios, the same set of hyperparameters were used in training as described in Table I. The training was performed in $T = 200,000$ time slots. In the testing phase, we deployed the greedy policy associated with the learned DQN and compared it with the optimal policy found in Sec. III. Both policies were carried in the same environment for total 50,000 time slots. Their performances are measured in terms of the SU access throughput, which is

$$\text{throughput} = \frac{\#\text{successes} - \#\text{failures}}{\#\text{timeslots}}, \tag{35}$$

where $\#\text{successes}$ is the number of time the SU accesses a vacant channel and $\#\text{failures}$ is the number of collisions.

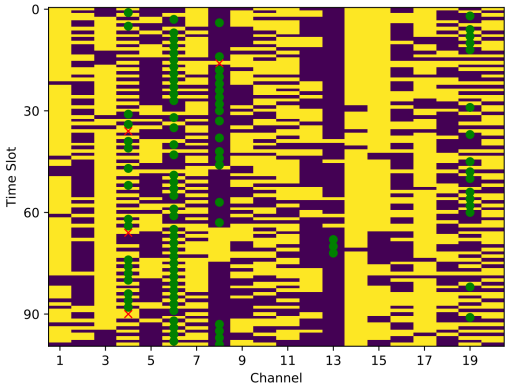
TABLE I: Hyper-parameters used in training the DQN.

Hyper-parameters	Value
Number of channels	20
Number of time slots	200,000
Memory size	2000
Batch size B	32
Exploration rate ϵ	$0.8 \rightarrow 0$
Discount rate γ	0.95
Learning rate α	0.001
Optimizer	Adam

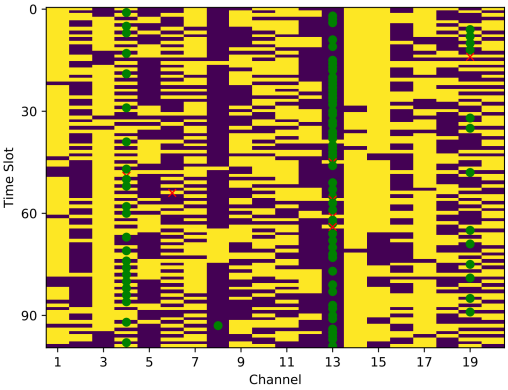
In Table II, the throughputs obtained by using the DQN for 5 scenarios are reported against those obtained by using the optimal policies. The throughput gaps between these schemes are highlighted in the last row of the table. It can be seen that the DQN method achieves nearly optimal throughputs in all of the testing scenarios. Fig. 3 visualizes the 2 access policies running for 100 time slots in Scenario 3.

TABLE II: Access throughputs obtained by using the optimal and DQN-based policies for 5 scenarios with different sets of transition probabilities.

Scenario	1	2	3	4	5
Optimal	92.52%	88.81%	93.11%	95.35%	78.64%
DQN	86.21%	84.02%	89.27%	89.79%	73.86%
Gap	**6.31%**	**4.79%**	**3.84%**	**5.56%**	**4.78%**



(a) Optimal access policy: throughput 93.11%.



(b) DQN-based access policy: throughput 89.27%.

Fig. 3: Visualization of DSA in Scenario 3 over 100 time slots using the optimal policy (a) and DQN policy (b). Occupied spectrum blocks are in yellow. Successful and colliding accesses are indicated by green dots and red crosses, respectively.

VI. Conclusion

We have studied in this paper the Dynamic Spectrum Access problem in which the secondary user is equipped with a wideband spectrum sensing mechanism. With the Markovian assumption on the channel states, the optimal policy can be obtained via a simple comparison of the state transition probabilities. When these probabilities are unknown, a deep Q-learning algorithm is proposed to learn an access policy that is comparable to the optimal one. For future research, it is worth investigating more complex variations of this DSA model such as non-Markovian channels, antijamming, multiple cognitive agents, and contiguous multiband access. Furthermore, future DSA models need to take into account the noisy sensing, which was omitted in this paper.

References

[1] J. Mitola and G. Q. Maguire, "Cognitive radio: Making software radios more personal," *IEEE Pers. Commun.*, vol. 6, no. 4, pp. 13–18, 1999.

[2] Y.-C. Liang, K.-C. Chen, G. Y. Li, and P. Mahonen, "Cognitive radio networking and communications: an overview," *IEEE Trans. Vehicular Technology*, vol. 60, no. 7, pp. 3386–3407, Sep. 2011.

[3] E. Hossain and V. K. Bhargava, *Cognitive Wireless Communication Networks*. New York, NY: Springer, 2007.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[5] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[6] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, Jan. 2016.

[7] 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," arXiv:1712.07365 [cs.IT], 2018.

[8] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," arXiv:1802.06958 [cs.NI], 2018.

[9] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," arXiv:1704.02613 [cs.NI], 2017.

[10] M. A. Aref and S. K. Jayaweera, "A cognitive anti-jamming and interference-avoidance stochastic game," in *IEEE Int. Conf. Cognitive Inform. Cognitive Comput. (ICCI*CC)*, Oxford, UK, 2017, pp. 1–8.

[11] M. A. Aref, S. K. Jayaweera, and S. Machuzak, "Multi-agent reinforcement learning based cognitive anti-jamming," in *IEEE Wireless Commun. Network. Conf. (WCNC)*, San Francisco, CA, USA, Mar. 19-22, 2017, pp. 1–6.

[12] J. Lundén, S. R. Kulkarni, V. Koivunen, and H. V. Poor, "Multiagent reinforcement learning based spectrum sensing policies for cognitive radio networks," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 5, pp. 858–868, Oct. 2013.

[13] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," arXiv:1712.00162 [cs.NI], 2017.

[14] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access," in *Proc. Int. Conf. Comput. Network. Commun. (ICNC)*, Jan. 26-29, 2017, pp. 1–5.

[15] H. Sun, A. Nallanathan, and C.-X. Wang, "Wideband spectrum sensing for cognitive radio networks: a survey," *IEEE Wireless Commun.*, vol. 20, no. 2, pp. 74–81, Apr. 2013.

[16] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, May 1992.