Dealing with Partial Observations in Dynamic Spectrum Access: Deep Recurrent Q-Networks

Y. Xu, J. Yu, and R.M. Buehrer
Wireless @ Virginia Tech
email:xuyue24@vt.edu, jianyuan@vt.edu, buehrer@vt.edu*

***Abstract*—This paper investigates the use of deep reinforcement learning (DRL) to solve a fundamental problem in dynamic spectrum access. Specifically, we develop a technique to allow a secondary radio node to share the spectrum with multiple existing (primary) radio nodes in the presence of partial observations and no *a priori* knowledge of the primary nodes' behaviors. There are multiple discrete channels shared by the radio nodes, and the secondary node cannot communicate with the primary nodes, but instead must base its spectrum access decisions on *partial observations* of the spectrum at each time step. The secondary radio node's objective is to maximize its own long-term expected number of successful transmissions (i.e., probability of full spectrum utilization) while minimizing collisions using limited feedback. The problem is formulated as a Partially-Observable Markov Decision Process (POMDP) with unknown system dynamics. In order to overcome the challenge of an unknown environment combined with partial observations, we apply a specific DRL approach: the Deep Recurrent Q Network (DRQN). We examine two partial observation patterns along with full observations in order to gain insight into the impact partial observations. Further, we compare previously proposed deep Q-networks (DQN) with the DRQN approach and demonstrate that the proposed DRQN approach can overcome the limitations of DQN-based solutions. We examine four specific scenarios which involve different types of primary nodes and show that the proposed DRQN technique can learn to avoid collisions and achieve near optimal performance.**

***Index Terms*—dynamic spectrum access, Partial Observation, Deep Reinforcement Learning, Recurrent Neural Network**

I. Introduction

It is generally understood that the current static spectrum access mechanisms cannot effectively support the explosively increasing demands for spectrum from a growing number of wireless devices. Dynamic spectrum access (DSA) [1][2] is a key concept which has the potential to improve spectrum utilization efficiency in wireless networks and thus help meet the aforementioned need for more spectrum.

This work considers an uncoordinated multichannel access problem with discrete channels under partial observations. Each channel has three possible states (occupied, idle and unknown). The channels are also assumed to have different channel qualities due to independent link SNRs. Each primary radio node in the environment selects one channel at each time step to transmit a packet (primary nodes are assumed to avoid collisions with each other). The secondary node must observe the spectrum (channels) and choose a channel on which to transmit, or to delay its transmission until later. If the secondary node chooses to transmit and the selected channel is idle (and not simultaneously selected by any primary nodes), the transmission is successful. If the channel has been selected by another node, there is a collision. The secondary node is assumed to be able to detect the channel state and the channel quality using spectrum sensing and channel-probing. The primary nodes are assumed to be of various types based on their spectrum behavior which will be described in more detail shortly.

Generally speaking, in order to exploit idle channels and improve spectrum utilization efficiency without knowing the various node types, reinforcement learning (RL), especially MDPs, has emerged as a potential solution [3]. The advantage of RL is that exploration actions are selected in situations where knowledge about the environment is uncertain resulting in learning through experience. However, traditional RL methods typically require prohibitively large computational resources when applied to dynamic spectrum access due to the high-dimensional transition and reward matrices. In order to overcome this problem, Deep Reinforcement Learning [4], and in particular, Deep Q-Learning has been proposed to *learn* the optimal channel access policy via online learning[1]. These approaches are able to deal with large systems, and find a good suboptimal or even the optimal policy directly from historical experience without any prior information about the system dynamics [4][5].

During the learning process, the more information the secondary node obtains, the higher the probability that it will select the optimal action. However, in practice, the node cannot observe all channel states at each time step. Thus, the learning approach (e.g., a DQN) may not be able to completely solve the resulting Partially-Observable Markov Decision Process (POMDP). However, although the secondary node cannot observe all state information simultaneously, it can observe the full environment *sequentially*. Using these sequential observations, and by combining a recurrent neural network [6] with a DQN, we can form a DRQN which can estimate the Q values[2] using partial observations. We will show that this approach is effective for the examined environments.

The remainder of this paper is organized as follows. Section II describes the related previous work. In Section

[1]Online learning refers to learning in real-time without using prior training samples.

[2]Q values are the learned values used to make decisions in Q-learning and are related to the value of taking a particular action when in a particular state.

III, the system model including the problem formulation and the communication environment assumed in this paper is presented. Section IV describes the details of the DRQN. This is followed by the simulation and numerical results in Section V. Finally, we conclude this paper in Section VI.

II. Related Work

Dynamic spectrum access (DSA) has been widely studied [1][7]. When channels are independent and identically distributed (i.i.d.), there are many algorithms, such as vertical handoff [8], power allocation [9] and the restless multi-armed bandit (RMAB) approach [10], that have been shown to be near-optimal under certain conditions. However, these methods are based on prior knowledge of the systems' protocols, have large computational complexity and are not adaptable in complex communication environments.

In recent years, the development of machine learning has attracted the attention of researchers in the area of wireless communications, as it is applicable to the many of the challenges facing wireless systems. Many works have begun to focus on the more practical and complex DSA problem where the system dynamics are unknown. The most popular algorithm for accomplishing this is Q-learning since it is a model-free method and can learn the policy directly via online learning [11]. In 2013, Google DeepMind used a deep neural network, called a DQN, to approximate Q-values in Q-learning in order to overcome the limitations of Q-learning [4][12].

Real-world tasks often feature incomplete and noisy state information resulting from partial observability. Unfortunately, DQNs are limited in the sense that they learn a mapping from a limited number of past states. Thus in general a DQN is unable to solve POMDPs. In [13] and [14] Recurrent Neural Networks were combined with a DQN to solve POMDPs in computer games. However, to the best of our knowledge no one has applied this approach to DSA. Thus, in this paper, we will attempt to combine these DRL methods to solve the partial observation problem.

III. System Model

A. Problem Formulation

In this paper, we consider a dynamic spectrum access scenario with $2N$ primary radio nodes including N transmitters and N receivers. The N primary transmitters choose to transmit on one of K channels based on their node type (see section III-C) at each time step. The secondary node senses some fraction of the channels including both channel occupancy and channel quality (e.g., SNR) using a specific observation pattern. In the next time step, the secondary node chooses to either transmit on one of the K channels or not to transmit at all. If the node transmits and the corresponding channel is idle in that time step, the transmission succeeds and the user receives a positive reward. Success or failure of a transmission is assumed to be known using standard ACK/NACK messages. Because of channel noise (or possibly fading), each channel may have a different quality and resulting throughput. Thus, the rewards range from 0 to 100 for a successful transmission, depending on the quality. If the channel is occupied at the time of secondary user transmission, the transmission fails and there is a negative reward (-10). As mentioned, nodes can also choose to wait (i.e., not transmit) in order to receive a higher future reward. Because we do not want these nodes to wait too often and incur significant delay, the reward for waiting is set to -1. Note that the reward structure can be varied to meet different link requirements. The goal is to learn a policy that maximizes the expected long-term reward.

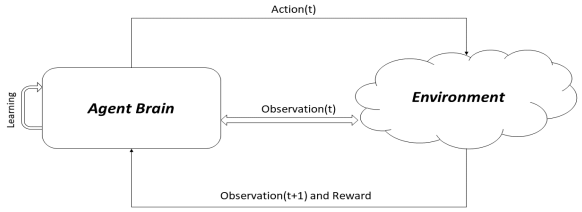


Fig. 1: The agent-environment interaction process

B. Dynamic Spectrum Access using Reinforcement Learning

As shown in Figure 1, an intelligent agent (our secondary node) interacts with an environment over a sequence of discrete times to accomplish a task. At time t, the agent observes the environment to obtain $o_t \in O$, where O is the set of possible observations (spectrum occupancy in our case). It then takes an action, $a_t \in A_{o_t}$ where A_{o_t} is the set of possible actions (transmission in a particular band or waiting in our case) given observation o_t. As a result of the observation-action pair, (o_t, a_t), the agent receives a reward r_{t+1} (based on successful/unsuccessful packets), and the environment provides a new observation o_{t+1} at time $t+1$. The goal of the agent is to maximize some function of that reward. For example, the performance criterion to be maximized at time t could be $R_t \triangleq \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_{\tau+1}$, where $\gamma \in (0,1]$ is a discount factor for weighting future rewards. In general, the agent takes actions according to some decision policy π and, with sufficient experience, the agent can learn an optimal decision policy π^* which will maximize its performance criterion (typically long-term reward).

Deep Q-learning (or Deep Q Network, DQN) is a recently developed technique within Reinforcement Learning[12]. It overcomes the main defect of traditional Q-Learning [3]: the requirement of a large amount of resources to compute and store the state-action value function, i.e., the Q-values. These requirements result in an unacceptable amount of necessary computation and memory resources, especially when the number of channels is large. In DQN, deep neural networks are used to approximate the state-action value function, $q(s,a;\theta) \approx Q^*(s,a)$, where (s,a) is the state-action pair

and the parameter vector θ, is the vector of weights in the deep neural network. Thus, the DQN can be trained by minimizing the prediction error of $q(s,a;\theta)$. Meanwhile, during online learning, we must balance exploitation (i.e, using the best known action) and exploration (learning new, possibly better actions). The ε-greedy policy is often adopted to accomplish this tradeoff. For the ε-greedy policy, the agent selects the greedy action $a_t = \arg\max_a q(s,a;\theta)$ with probability $1-\varepsilon$, and selects a random action with probability ε. The reason for randomly selecting an action some percentage of the time is to avoid getting stuck with a $q(s,a;\theta)$ function that has not yet converged to $q^*(s,a;\theta)$ (i.e., learning better actions).

However, a DQN cannot completely solve the DSA problems under *partial* observations. This is because, in a DQN, the Q-value should be calculated based on the current state which requires a full (and perfect) channel state observation. Thus, if secondary nodes are unable to observe the full state, a DQN-based approach may perform poorly. Therefore, we need to introduce another approach, viz., a recurrent neural network, to solve the partial observation problem. We will introduce a deep recurrent Q network (DRQN) in the following section.

C. Primary Radio Nodes

In the assumed communication system, there are several types of primary nodes assumed:

- Legacy node: Occupies one fixed channel at all times
- Hopping node: Dynamically occupies the channels under a regular pattern
- Intermittent node: Periodically occupies one fixed channel (this could be a TDMA node or a bursty node)
- Greedy node: Occupies the first available channel it observes. It can be regarded as a type of DSA node

There are two types of secondary nodes considered: DQN-based nodes and DRQN-based nodes. Note that we assume that secondary nodes can exploit ACK/NACK information to determine whether a transmission was successful or not.

IV. Deep Recurrent Q-Learning

A. RNN and LSTM

The recurrent neural network (RNN) is one class of artificial neural network which has the capability of modeling the dynamic temporal behavior of a time sequence. The idea behind an RNN is to exploit sequential information. In a traditional neural network, it is assumed that all inputs (and outputs) are independent of each other. However, if the network wants to predict the next state, it needs to know which observations came before it. Another way to think about an RNN is that it has "memory" which captures information about what has been calculated so far. In theory an RNN can make use of information in an arbitrarily long sequence, but in practice they are limited to looking back only a few steps. An RNN is called *recurrent* because it performs the same task for every element of a sequence, with the output being dependent on the previous computations.

In a traditional neural network, the basic computing component is termed a "unit", but in an RNN, the basic unit is a termed a "cell". In this work, Long Short-Term Memory (LSTM) [6], which is one of the most effective types of cell structure, is applied to an RNN. Its structure is described in the following.

At time t, the input observation is o_t. First, we compute the input gate values I_t and the candidate value $\tilde{C}_t$ for the states of the memory cells at time t:

$$I_t = sigmoid\left(W_i o_t + U_i h_{t-1} + b_i\right) \tag{1}$$

$$\tilde{C}_t = \tanh\left(W_c o_t + U_c h_{t-1} + b_c\right) \tag{2}$$

Here, W_i, U_i and b_i are parameters of the input gate, W_c, U_c and b_c are parameters of the self-recurrent connection and h_{t-1} is information from the memory input. Second, we compute the forget gate value f_t:

$$f_t = sigmoid\left(W_f o_t + U_f h_{t-1} + b_f\right) \tag{3}$$

Here, W_f, U_f and b_f are parameters of the forget gate. Based on the values of the input gate, the forget gate and the candidate state, the new candidate state value is:

$$C_t = I_t \times \tilde{C}_t + f_t \times C_{t-1} \tag{4}$$

Finally, we compute the output value O_t and upgrade the memory information h_t:

$$O_t = sigmoid\left(W_o o_t + U_o h_{t-1} + V_o C_t + b_o\right) \tag{5}$$

$$h_t = O_t \times \tanh\left(C_t\right) \tag{6}$$

where, W_o, U_o and b_o are parameters of the output gate.

B. Deep Recurrent Q Network

Unlike the DRQN structures in [13] and [14], our work focuses on the use of a DRQN in a communication system. The proposed DRQN structure is shown in Figure 2. In the DRQN brain, the input layer is ignored and the output layer is shown in the last layer. More hyper-parameters and structure details will be illustrated in the following section.

V. Simulation results

This section investigates the performance of the proposed DRQN nodes via simulation. In the communication environment, we assume that there are $K = 10$ discrete frequency channels. For our experiments, the neural network structure has 3 hidden layers: 2 fully connected layers (each layer containing 40 and 25 units respectively) and one RNN (40 hidden cells) layer. These values are typical for the number of inputs and outputs. The activation function used for the fully connected layers is the ReLU (Rectified Linear

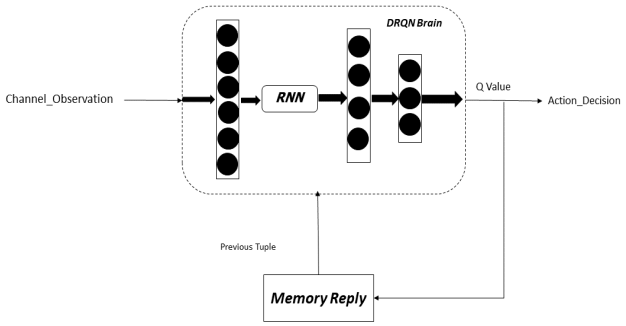


Fig. 2: The proposed DRQN structure

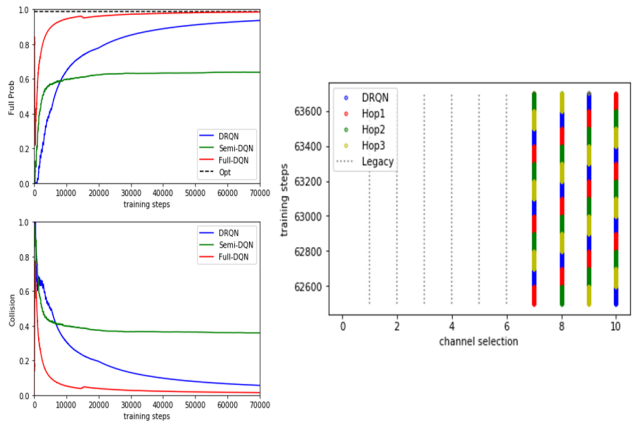


Fig. 3: Performance in Communication Environment 1: Plots on the left show the probability of full spectrum utilization (top) and collision rate (bottom) versus time, while plots on the right show channel occupancy at a specific subset of times.

Unit) function. When updating the weights of the network, a minibatch of experience-tuples are randomly selected from a memory box of 1200 prior experiences for the computation of the loss function. The Adam algorithm [15] is used to conduct a Stochastic Gradient Descent for updating. In order to avoid getting stuck with a suboptimal decision policy before sufficient learning experience, we apply an adaptive ε greedy algorithm which is initially set to 0.1 and is decreased by 0.00005 every time step until reaching 0.01.

A. *Half-Observation Pattern*

In the first set of simulations, the observation pattern of the DRQN toggles between two sets of channels. This means that the secondary node will alternately observe half of channels each time step. Additionally, we consider four different scenarios to show that the proposed DRQN node can learn the primary nodes' spectrum usage patterns without prior knowledge. For the sake of comparison, a DQN-based node with full observations and partial observations is also examined and plotted.

In the first environment, 6 of 10 channels (#1 to #6) are occupied by legacy (i.e., fixed) nodes. Further, three hopping nodes alternately occupy channels #7 - #10. The occupancy pattern for the three nodes at each time step is: (1) $\#7\#8\#9 \rightarrow$, (2) $\#8\#9\#10 \rightarrow$, (3) $\#9\#10\#7 \rightarrow$, (4) $\#10\#7\#8 \rightarrow$, and (6) $\#7\#8\#9 \rightarrow$..... Further time steps continue this pattern. As can be seen, in this system there is always one available channel that can be used by the DRQN node (assuming that it can learn to predict the open channel).

In Figure 3, we present the results of the first set of simulations. Specifically, we plot the probability of full spectrum utilization and the collisions over time (left plots). On the right, we plot the channels occupied by the DRQN node as well as the channels occupied by the hopping nodes at a specific set of times. Note that for clarity, we do not plot the channels occupied by the legacy nodes (they occupy channels $\#1 - \#6$ at all times). Because ε is a non-zero parameter, DQN and DRQN nodes still have some probability to select an occupied channel due to exploration. Thus, the probability of full spectrum utilization is upper-bounded by 0.99. Comparing a DQN with full observations to one with partial observations, we see that the DQN can reach near optimal performance with full observations, but has severely limited performance with partial observations. On the other hand, the DRQN node can learn the pattern of the legacy nodes and hopping nodes even in the presence of partial observations. Specifically, the DRQN node can reach a probability of full spectrum utilization of $\gamma = 0.955$ which is nearly what a DQN-based approach with full observations can achieve. Note that the collision rate is not equal to $1 - \gamma$, since the nodes also can choose to wait which reduces spectrum utilization but does not result in a collision.

In the second environment, 3 of 10 channels (#1 to #3) are always occupied by legacy (fixed) nodes. Further, there are three hopping nodes with a dynamic channel occupancy pattern of (1) #5#6#7 $\rightarrow$ (2) #6#7#8 $\rightarrow$ (3) #7#8#5 $\rightarrow$ (4) #8#5#6 $\rightarrow$ (5) #5#6#7. There are also intermittent nodes on three channels: #4 (a primary node occupies this channel for 250 consecutive steps and is idle 5 consecutive steps), #9 (a primary node occupies the channel for 250 consecutive steps and is idle for 10 consecutive steps) and #10 (a primary node occupies the channel for 250 steps and is idle for 3 steps). In this system, there is always at least one available channel that can be used by the DRQN node and at times there is more than one available channel.

From Figure 4, it can be seen that in general the results are very similar to the previous case. Additionally, due to the different qualities of the channels, when the system has more than one idle channel, the DRQN will chose the channel with the higher quality (e.g., see channel #9 on right which has very high quality but is rarely available). As before, because ε is a non-zero parameter, DQN nodes still have some probability to select an occupied channel (due to exploration). Thus, the optimal probability of full spectrum

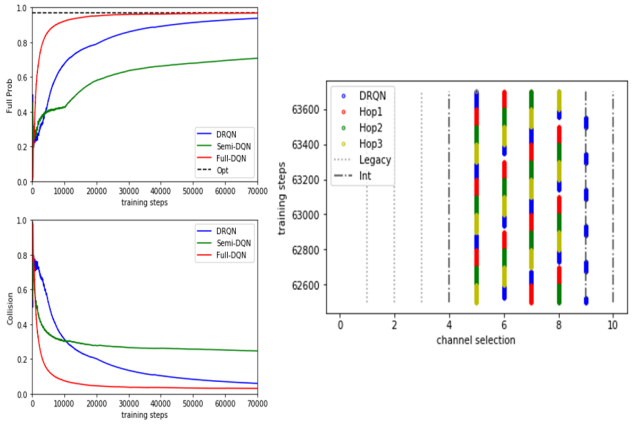


Fig. 4: Performance in Communication Environment 2: Plots on the left show the probability of full spectrum utilization(top) and collision rate (bottom). Plots on the right show channel occupancy versus time.

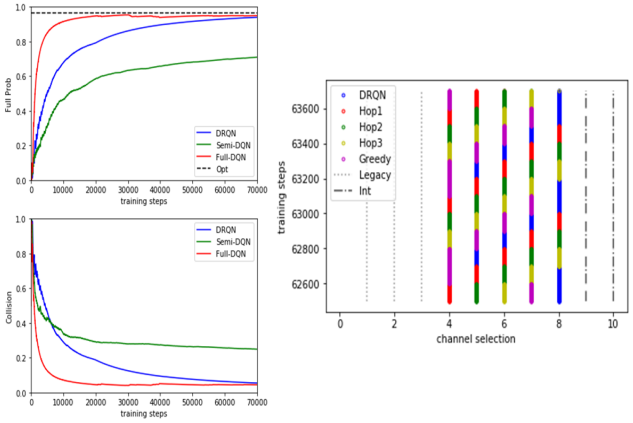


Fig. 5: Performance in Communication Environment 3: Plots on the left show probability of full spectrum utilization (top) and collision rate (bottom). while plots on the right show channel occupancy versus time.

utilization should be approximately 0.96 (Assume DRQN can always make right decision, $Opt = 1 - 10/260$). Compared to a DQN with full observations or partial observations, we see that once again the DRQN node shows substantially better performance than the DQN under partial observations and nearly as good as a DQN-based node with full observations. Once again the DRQN-based secondary node can learn the spectrum occupancy pattern of the primary nodes.

In the third environment, we assume three legacy (fixed) nodes (channels #1 to #3), three hopping nodes, one greedy node, and two intermittent nodes. Thus, there are nine primary nodes occupying the 10 channels. The dynamic pattern of the Hopping nodes is: (1) #4#5#6 $\rightarrow$ (2) #5#6#7 $\rightarrow$ (3) #6#7#8 $\rightarrow$ (4) #7#8#4 $\rightarrow$ (5) #8#4#5 $\rightarrow$ (6) #4#5#6. Intermittent nodes occupy the following channels: #9 (the primary node occupies the channel for 250 consecutive steps and is idle for 10 steps) and #10 (the primary node occupies the channel for 250 steps and is idle for 3 steps). Here, the greedy node can observe all channels and then take an action which has influence on the DRQN node's observation. Due to the facts that (a) the DRQN nodes have a 1% possibility of random selection (exploration) and (b) the intermittent nodes' don't always occupy their respective channels, the secondary node's optimal spectrum utilization is approximately 0.96. However, the presence of the greedy node does negatively impact the performance of the DRQN-based node.

From Figure 5, compared to DQN with full observations and partial observations, the DRQN node can again learn the spectrum occupancy pattern of the primary nodes. Although the DRQN can not reach the optimal performance, its performance (0.95) is still very close to the optimal performance of the DQN node with full observations.

In the last environment, we assume three legacy nodes (channels #1 to #3), three hopping nodes, one greedy nodes, and two intermittent nodes. The dynamic pattern of the hopping nodes is:#4#5#6 $\rightarrow$ #5#6#7 $\rightarrow$ #6#7#8 $\rightarrow$ #7#8#4 $\rightarrow$ #8#4#5 $\rightarrow$ #4#5#6. Intermittent nodes occupy channels: #9 (node occupies the channel for 250 consecutive steps and is idle for 10 steps), #10 (node occupies the channel for 250 steps and is idle for 3 steps). Here, the greedy node and the DRQN observe the channel state and take actions at same time. Based on the DRQN nodes' 1% exploration rate and the intermittent nodes' periodicity, the system's optimal probability of full spectrum utilization is 0.96.

From Figure 6, again compared to a DQN with full

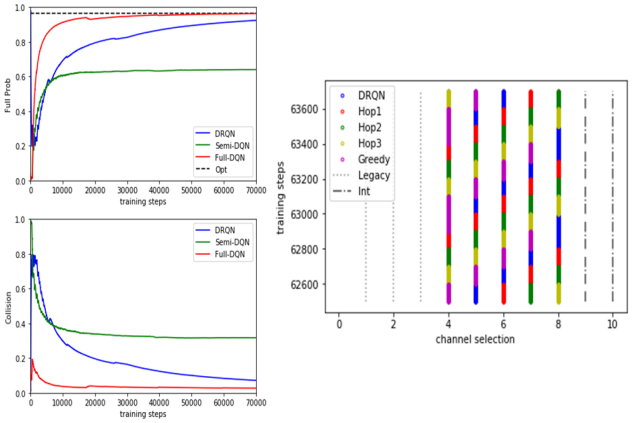
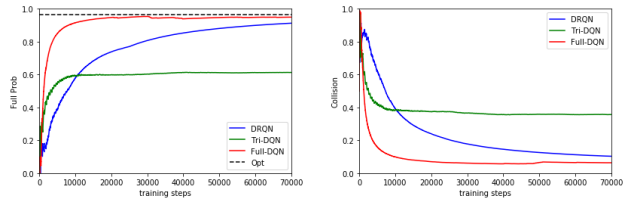
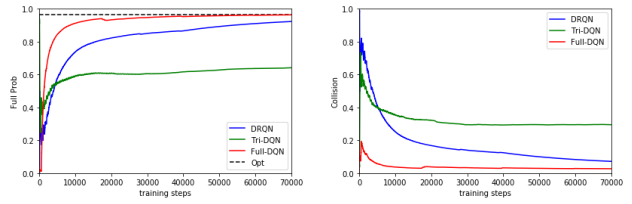


Fig. 6: Performance in Communication Environment 4: Plots on the left show probability of full spectrum utilization (top) and collision rate (bottom), while plots on the right show channel occupancy versus time

observations and partial observations, the DRQN node successfully learns the spectrum occupancy pattern of the primary nodes nearly as well as the DQN with full observations and significantly better than a DQN with partial observations. The presence of the greedy node that acts simultaneously with the DRQN node demonstrates that the interaction of the two nodes does not negatively affect



(a) Full Spect. Util. Prob.- Comm3 (b) Collision Rate in Comm3



(c) Full Spect. Prob. Util. - Comm4 (d) Collision Rate in Comm4

Fig. 7: Impact of Observing 1/3 of the Channels Each Step

the secondary node's learning and resulting performance. Although the DRQN can not reach the optimal performance (0.96), its performance (0.94) is still good.

B. Impact of Fewer Observations

Based on the simulation results of the previous subsection, we conclude that DRQN nodes can learn the different nodes' spectral occupancy patterns/behaviors and even reach near optimal performance when observing half of the channels at each time step. We now examine a an observation pattern where only one third of the channels are observed at any time step. In practice, less observations means preservation of resources, thus it is meaningful to understand if fewer observations are acceptable.

We now assume that the DRQN node observes only one third of the channels at each time. Because our system has 10 channels, the order of observation is assumed to be: $111XXXXXXX \rightarrow XXX111XXXX \rightarrow XXXXXX1111$ in three consecutive time steps. In the simulation, we mainly focus on the collision rate and probability of full spectrum utilization of the node transmission in communication environments 3 and 4.

From the plots in Figure 7, we observe that under this more restricted observation pattern, although the DRQN's performance still has a gap as compared to the optimal performance and even compared to full-observation DQN, its performance (0.92) is stable and acceptable. The learning rate is slightly slower, but not dramatically slower. Thus, we conclude that the DRQN node can learn the different nodes' dynamic behavior without prior knowledge even when only observing one third of the channels at each time step.

VI. Conclusion

In this paper, we have considered a general and complex dynamic spectrum access problem where the system has several primary nodes dynamically accessing multiple channels and a single secondary node that has no *a priori* knowledge of the primary nodes' behaviors and bases its spectrum access decisions on *partial observations* of the spectrum. We have applied a DRQN approach to find the optimal access policy via online learning. Through simulations, we have shown that our proposed DRQN node is in fact able to learn the different nodes' patterns without any prior knowledge and spectrum utilization converges to near optimal performance. We have also shown that our DRQN can achieve near-optimal performance even in complex scenarios and with as few as one third of the channels being observed.

In future work, we will mainly focus on improving convergence speed and performance. We will also consider more realistic spectrum sharing scenarios.

References

[1] Qing Zhao and Brian M Sadler. A survey of dynamic spectrum access. *IEEE signal processing magazine*, 24(3):79–89, 2007.

[2] Ian F Akyildiz, Won-Yeol Lee, Mehmet C Vuran, and Shantidev Mohanty. Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *Computer networks*, 50(13):2127–2159, 2006.

[3] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[5] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 16, pages 2094–2100, 2016.

[6] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[7] Sudhir Srinivasa and Syed Ali Jafar. Cognitive radios for dynamic spectrum access-the throughput potential of cognitive radio: A theoretical perspective. *IEEE Communications Magazine*, 45(5), 2007.

[8] Enrique Stevens-Navarro, Yuxia Lin, and Vincent WS Wong. An mdp-based vertical handoff decision algorithm for heterogeneous wireless networks. *IEEE Transactions on Vehicular Technology*, 57(2):1243–1254, 2008.

[9] Pranav Sakulkar and Bhaskar Krishnamachari. Online learning of power allocation policies in energy harvesting communications. In *Signal Processing and Communications (SPCOM), 2016 International Conference on*, pages 1–5. IEEE, 2016.

[10] Keqin Liu and Qing Zhao. Indexability of restless bandit problems and optimality of whittle index for dynamic multichannel access. *IEEE Transactions on Information Theory*, 56(11):5547–5567, 2010.

[11] Mehdi Bennis and Dusit Niyato. A q-learning based approach to interference avoidance in self-organized femtocell networks. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 706–710. IEEE, 2010.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[13] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR, abs/1507.06527*, 7(1), 2015.

[14] Christopher Schulze and Marcus Schulze. Vizdoom: Drqn with prioritized experience replay, double-q learning, & snapshot ensembling. *arXiv preprint arXiv:1801.01000*, 2018.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.