Dynamic Spectrum Access with Deep Q-learning in Densely Occupied and Partially Observable Environments

Slavica Tomovic, *Member, IEEE*, and Igor Radusinovic, *Senior Member, IEEE*

***Abstract* — In this paper, we propose a new Dynamic Spectrum Access (DSA) method for multi-channel wireless networks. We assume that DSA nodes do not have prior knowledge of the system dynamics, and have only partial observability of the channels. Thus, the problem is formulated as a Partially Observable Markov Decision Process (POMDP) with exponential time complexity. We have developed a novel Deep Reinforcement Learning (DRL) based DSA method which combines a double deep Q-learning architecture with a recurrent neural network and takes advantage of a prioritized experience buffer. The simulation analysis shows that the proposed method accurately predicts a channel state based on the fixed-length history of partial observations. Compared with other DRL methods for DSA, the proposed solution can find a near-optimal policy in a smaller number of iterations and suits a wider range of communication environments, including dynamic ones, where channel occupancy pattern changes over time. The performance improvement increases with the number of channels and with a channel state transition uncertainty. To boost the performance of the algorithm in densely occupied environments, multiple DRL exploration strategies are examined and evaluation results are presented in the paper.**

***Keywords* — Cognitive radio, Reinforcement learning, Q-learning.**

I. Introduction

In traditional wireless networks, fixed radio spectrum bands are statically allocated to licensed users for exclusive access. The proliferation of mobile devices and Internet of Things services in the past decades has revealed many limitations of this policy [1]-[2]. According to some recent reports, most of the licensed spectrum bands are under 30% utilized [3]. The unused spectrum holes offer a great capacity opportunity for Cognitive Radio Networks (CRNs) [4]. CRN allows secondary users (SUs) to identify and temporarily exploit spectrum holes that are not occupied by the primary (licensed) users (PUs). This concept implies that SUs have Dynamic Spectrum Access (DSA) capability, which allows them to dynamically change the operating frequency channel over time, according to the indication of availability.

In general, DSA imposes a lot of challenges because channel occupancy patterns vary and are not known a priori. To avoid interference with PUs, SUs have to continuously monitor spectrum channels. However, in practice, DSA nodes cannot sense all the channels simultaneously [5]. Thus, unlike many decision-making problems in communications, such as handoff and power management, which can be modeled as the Markov Decision Processes (MDPs), the DSA problem is formulated as partially observable MDP (POMDP), which implies exponential time and space complexity [6].

Most previous research on DSA has focused on communication scenarios with multiple independent and identically distributed channels. If additionally, the channels are positively correlated, the Myopic policy [7] achieves an optimal performance. The same result also holds for scenarios with two or three negatively correlated channels. Besides these results, the Whittle Index policy [8] can achieve a near-optimal performance when the channels follow different Markov Chains (MCs). However, when the channels are correlated, the Myopic and Whittle Index approaches cannot be applied. Another limitation of these approaches is that the system transition matrix must be known in advance. Since channel statistics are generally unknown and vary over time, SUs have to estimate the transition matrix. The accuracy of the estimation drastically impacts performance. Thus, recent research in this field has focused on more realistic problems, where the channels are correlated and the system dynamics are not known a priori [3]-[6], [9].

Recent developments of Deep Reinforcement Learning (DRL) have motivated research on DRL-based DSA methods. Implementation of deep Q-network (DQN) for CRN has been proposed in [6]. The proposed DQN design enables SU to learn an efficient channel sensing policy from historical partial observations. A multi-user DSA scenario has been analyzed in [5], but without considering the impact of PUs. In each time slot, DQN is used to map an observation from the previous time step to channel selection action. A single DQN is trained for all users at the mobile edge computing server. In [3], collisions with both PUs and SUs have been considered, as well as channel sensing errors. However, the authors have adopted the impractical assumption that SUs can sense all channels at the beginning of each time slot. Unlike the previous works, in [9], DQN is

Paper received April 04, 2021; accepted June 13, 2021. Date of publication July 31, 2021. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Aleksandar Nešković.

This paper is revised and expanded version of the paper presented at the 28th Telecommunications Forum TELFOR 2020 [13].

S. Tomovic, University of Montenegro, Faculty of Electrical Engineering, Dzordza Vasingtona bb, 81000 Podgorica, Montenegro (e-mail: slavicat@ucg.ac.me).

I. Radusinovic, University of Montenegro, Faculty of Electrical Engineering, Dzordza Vasingtona bb, 81000 Podgorica, Montenegro (e-mail: igorr@ucg.ac.me).

used to solve the problem of DSA and spectrum aggregation, under the assumption that nodes can synthesize multiple channels for successful transmission.

In this paper, the scenario from [6] was considered, where a SU selected a single channel to sense and potentially transmit over, in each time slot. SU cannot observe all the channels simultaneously, but it can observe the state of different channels in sequence. Thus, a fixed-length history of the node's partial observations is used as an input of the proposed decision-making algorithm. It has been shown that the DRL algorithm from [6] cannot scale to a large number of channels and hardly deals with the channel state transition uncertainty. To alleviate these issues, we designed a new DQN architecture with double Q-learning [10] and the prioritized experience buffer [11], which integrates a Long Short-Term Memory (LSTM) recurrent neural network [12]. The proposed DQN architecture is able to learn temporal correlation among the historical observations in a time-efficient manner and reduce the collision rate. The simulation analysis suggests that the proposed method can deal with a larger number of channels than the method from [6], and is more tolerant to uncertainty in channel switching patterns. The main idea of the proposed algorithm is presented in [13]. Here, the algorithm is extended to better deal with dynamic and densely occupied environments. The problem of efficiently balancing exploration and exploitation in the DRL process is emphasized. Thereby, the focus is on complex cognitive radio scenarios that are modeled with POMDPs with large state space and the sparse reward setting [14]. Several solutions for the exploration problem have been analyzed and conclusions on their applicability have been made based on the simulation results.

The rest of the paper is organized as follows. In Section II, the formulation of the DSA problem is presented. The proposed DQN-based DSA method is explained in Section III. The simulation results are presented in Section IV. The concluding remarks are given in Section V.

II. Network Model and the Problem Definition

We consider the DSA problem from [6], where a SU dynamically selects one out of K potentially correlated channels to sense and transmit a packet. In practice, the channels are highly correlated, as a result of external interference [9]. Each channel has two possible states: vacant (0) or occupied (1). Therefore, the whole system can be modeled by a Markov chain with 2^K states [6]. If a channel selected by the SU is vacant, a packet transmission will be successful, otherwise not.

In each time slot, SU senses a single channel and learns its state. Since the SU's observation of the environment is incomplete in each time slot, the DSA problem can be formulated as POMDP, where the goal is to predict channel states based on the previous decisions and observations. To solve this problem, we propose a new method based on DRL. In this method, a DRL agent interacts with the environment by choosing a channel for sensing and potential transmission. Then, it receives feedback on its action and updates the decision-making policy.

We denote the system state at time t as $s_t = [s_1, s_2, \ldots, s_K]$ where $s_t \in \{0,1\}$ is the state of the i-th channel. The system state is not completely observable. Given the sensing action $a_t \in \{1,2,\ldots, K\}$, which determines the channel to be sensed at time t, the user can observe only the state of the selected channel. We denote this local observation as $o_t \in \{0,1\}$. DRL agent learns based on the history $x_t = [a_{t-1}, o_{t-1}, \ldots, a_{t-N}, o_{t-N}]$ of N previous actions and observations. The reward function is defined as

$$r(x_t, a_t) = \begin{cases} 0, & if \quad o(t) = 1 \\ 1, & if \quad o(t) = 0 \end{cases} \tag{1}$$

Our objective is to find an optimal DSA policy π^* for mapping x_t to the next action a_{t+1}, such that the expected long-term reward is maximized. This is defined as:

$$\pi^* = \arg\max_{\pi} E_{\pi} \sum_{t=1}^{\infty} \gamma^{t-1} r(x_t, a_t), \tag{2}$$

where $0 \le \gamma < 1$ is a discounting factor that controls the impact of future rewards on the cumulative reward.

III. The Algorithm

The proposed DSA solution is based on deep Q-learning algorithm. Q-learning is a powerful RL method that aims at finding an optimal policy π^* that maximizes the expected cumulative reward. This is done by assessing the values of different actions under various circumstances, i.e. in various states. As explained before, we use x_t to define the state observable by SU. The Q-value $Q^{\pi}(x_t, a_t)$ of the state-action pair (x_t, a_t) represents the expectation of the cumulative reward when taking action a_t in an initial state x_t and then following the policy π afterward [6]. If Q-values are well estimated, the optimal policy π^* is:

$$\pi^* = \arg\max_{a} Q^{\pi}(x, a), \tag{3}$$

i.e. an action with the largest Q-value should be chosen.

The Q-values of state-action pairs are estimated via online learning. If RL agent is in state x_t at time t, it will take an action that maximizes $Q^{\pi}(x_t, a_t)$, according to the current policy π. Then, after receiving reward r_{t+1}, the Q-value of (x_t, a_t) is updated as follows:

$$Q(x_t, a_t) = Q(x_t, a_t) + \\ + \lambda \left[r_{t+1} + \gamma \max_{a_{t+1}} Q(x_{t+1}, a_{t+1}) - Q(x_t, a_t) \right], \tag{4}$$

where $\lambda \in \{0,1\}$ is the learning rate, and $r_{t+1} + \max_{a_{t+1}} Q(x_{t+1}, a_{t+1})$ is the target Q-value learned from the interaction with the environment.

In traditional Q-learning, Q-values of all possible state-action pairs are stored in a lookup table. This approach becomes impractical when the set of observable states is large. In particular, large state space entails large memory requirements. Also, some states are very rarely visited, which makes the algorithm difficult, or even impossible, to converge. To address these limitations, we adopt DRL approach, where a deep neural network (i.e. DQN) is used to map state-action pairs to Q values.

A. DQN architecture

Input Layer: The input of DQN is *NxK* matrix which represents the history of *N* last actions and observations. The matrix rows are provided sequentially to the input layer of DQN, which has *K* nodes. The performed actions and the related observations are encoded using the one-hot encoding scheme. Specifically, if a node has chosen channel *k* for transmission in the previous time slot, the first row of the input matrix will be a vector of size *K*, where the k^{th} entry is 1 or -1, and the rest of the entries are 0. The value of 1 is an indicator that the transmission on that channel was successful, while -1 indicates a transmission failure.

LSTM Layer: Since a DSA node has just a partial observation of the system state at each time slot, it is necessary to efficiently utilize previous observations to solve the POMDP. The more previous observations we use, the more optimal a solution we get. Concatenating previous observations in a single vector and providing it as an input of Feed-Forward Neural Network (FFNN) is problematic for several reasons. First, this approach increases the resources for computation. On the other hand, not all observations from the past are useful for the prediction of a future event, and FFNN is not able to "compress" the memory. Therefore, we propose the integration of DQN and Recurrent Neural Network (RNN) to capture the dependencies of sequential observations from the past. The RNNs use feedback connections between outputs and inputs, that are particularly useful for time series prediction. They are provided with a sequence of inputs, and each element of the sequence is processed in the same way, with the current output being dependent on the previous output. To avoid exploding and vanishing gradients problems [10], we use LSTM RNN architecture. A structure of a single LSTM cell for sequential processing is shown in Fig 1. In addition to hidden states h_t, which are common in conventional (Vanilla) RNNs, LSTM introduces the concepts of a cell state c_t, and the „gates" which regulate the flow of activations: input gate I_t, output gate O_t, and forget gate F_t. LSTM block outputs a hidden state h_t for a given input x_t by filtering the cell state information. The cell state carries relevant information from the previous time steps. The gates add and remove information from/to the cell state. They can be considered as separate NNs trained to learn what information from the past is relevant to keep and what should be forgotten.

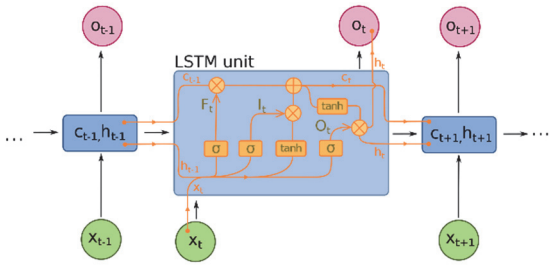

Fig. 1. LSTM block.

Additional Hidden Layers: Perform additional non-linear operations to better approximate Q-value function.

Output layer: A vector with *K* elements, where *k*-th element corresponds to the estimated Q-value for selecting a channel *k* in the given state.

Double DQN: The basic DQN architecture often overestimates true Q-values. This happens due to the *max* operator in (4). A double Q-learning algorithm [10] is applied to overcome this problem. This algorithm uses two DQNs: the main DQN $Q(x,a;\theta)$ and the target DQN $Q'(x,a;\theta')$. The first is used to select the best action, while the second is used to evaluate the Q-value of the selected action. The corresponding learned Q-value is given by:

$$Q^{\tau}(x_t,a_t) = r_{t+1} + \gamma \max_a Q'(x_{t+1}, \arg\max_a Q(x_{t+1},a)) \; . \quad (5)$$

Both DQNs have an identical structure. The main DQN is trained in the usual way. The weights of the target DQN (θ') are a delayed copy of the main network's weights.

The main steps of the DQN-based DSA algorithm are described in Algorithm I. During the training phase, the algorithm uses ε-greedy strategy to choose a channel. To explore the solution space, a node randomly chooses an action with a probability ε. With the probability 1-ε, an action with the largest Q-value is chosen. Then, the node gets reward r_{t+1}, and observes the next state x_{t+1}. A tuple (x_t, a_t, r_{t+1}, x_{t+1}) is stored in the experience buffer. A mini-batch of tuples from the experience buffer is used to train the network in each iteration. This is done by providing the learned Q-values (5) and the estimated Q-values $Q(x,a)$ to the mean squared error (MSE) loss function.

Algorithm 1: *The proposed DQN-based DSA algorithm*

1: **for** *t=1,2,…* **do**:
2: **if Train**:
3: *Use ε-greedy policy to select action a_t*
4: **else:**
5: *Choose the action a_t, as* $\arg\max_a Q(x_t,a)$
6: Take action a_t, obtain the reward r_{t+1}, and observe x_{t+1}
7: **if Train**:
8: *Store (x_t, a_t, r_{t+1}, x_{t+1}) in the experience buffer*
9: *Sample a **mini-batch** of experiences from the prioritized experience buffer*
10: ***for each** sample **in** mini-batch:*
11: *Compute the target value $Q^{\tau}(x,a)$ based on (5)*
12: *Compute MSE for $Q^{\tau}(x,a)$ and $Q(x,a)$*
13: *Train the main DQN by backpropagating the loss error*
14: *Update the priorities in the experience buffer*
15: **if** *t* mod *T* == 0:
16: *Copy θ to θ'*

To efficiently handle complex communication scenarios with sparse (positive) rewards, the elements of the experience buffer are prioritized, as proposed [11]. The experiences with the highest priority form a mini-batch for training in each iteration. More precisely, each mini-batch sample is chosen proportionally to the P_i^{ζ} value, where P_i is the priority of the *i-th* experience and $0 \leq \zeta \leq 1$ is a constant that controls the impact of the priority. When a new experience is added to the buffer, it is initially assigned a maximum priority value to ensure it will be sampled soon. The priorities of the samples are updated proportionally to the individual training loss multiplied with a weight factor $w_i=(MP_i)^{-\beta}$, where *M* is the number of experiences stored in buffer and $0 \leq \beta \leq 1$ is a constant which handles the bias of the priority-based sampling. The MSE loss is multiplied by the corresponding weight factor before backpropagation in NN training.

B. Exploration strategy

One of the key challenges in reinforcement learning is an efficient exploration of the environment. In an environment with sparse and/or deceptive rewards, DRL agent can get stuck in local optima, and might never approach an optimal policy. In our case, this is particularly problematic because the environment is partially observable, and the state space is very large. Also, it should be noted that traditional RL techniques are designed to work in static environments [14], while in many real-world problems environment is dynamic. In the example of cognitive radio, it is hard for a DRL agent to distinguish if a reward is a consequence of the taken action or it is due to a change in the channel occupancy pattern. In case of larger changes, it is necessary to explore the environment efficiently and to forget experiences that are not related to a current occupancy pattern as quickly as possible.

In Algorithm 1, we assumed ε-greedy exploration strategy, as the simplest approach. However, unlike the related works [3]-[6], [9], that use a constant value of ε, we decrease it gradually over time from ε_{start} to ε_{stop}. This is the so-called decayed ε-greedy algorithm [15], which avoids performance losses due to random actions once the environment is explored "enough". This leads to higher throughput when the environment is stationary, but if the channel occupancy pattern changes after some time, the algorithm takes a lot of time to discover a new optimal channel allocation scheme. Therefore, when the ε reaches ε_{stop} value, our agent begins to track changes in the received reward. In particular, every 100 time slots the agent measures and stores the mean reward score over that period. The measured score is compared with the previous one, and if a performance drop over 30% is detected, the DQN parameters are reset, as well as ε and β values. In addition, all the experiences older than 100 slots are deleted from the experience buffer. These steps enable faster convergence to the optimal policy when the channel occupancy pattern drastically changes.

In Section IV we analyze the applicability of other, more advanced, exploration strategies as well.

IV. Simulation Results

We implemented simulations in Python, using PyTorch, to evaluate the performance of the proposed DSA method. In simulations, we considered a scenario with a single DSA node and K channels, where K was varied between 2 and 20. The proposed approach, which we refer to as Deep Double Recurrent Q-Network with Prioritized Buffer (DDRQN-PB), has been compared with the DQN method from [6], Optimal algorithm, and Random channel selection method. The summary of the hyperparameters for our method is provided in Table I. For the DQN, we use the hyperparameters provided by the authors in [6], with the only difference in ε-greedy strategy. Namely, instead of a constant ε value, we use a gradually decreasing ε, because we find that leads to improved performance. DDRQN-PB includes one LSTM layer with 30 hidden units and one hidden layer with 200 nodes. Unless otherwise stated, a sequence of last K observations is used as input.

Table I: The hyperparameters settings.

Hyperparameter	Value
ε	0.1-0.01
Minibatch size	32
Optimizer	Adam
Learning rate	10^{-3}
γ	0.9
ζ	0.6
β	0.4-1 (gradually increasing)
Experience buffer size	10^6
Activation function	ReLU

We divide simulation analysis into two scenarios. We first consider the scenario where the channels follow independent MCs with the transition probabilities: p_{00}=0.8 and p_{11}=0.9. The DQN and DDRQN-PB algorithms have been trained for 10000 iterations. The obtained results in terms of mean reward per slot are shown in Fig. 2. For this scenario, it has been proved that Myopic policy achieves an optimal performance when the system dynamics are known in advance. Thus, from the obtained results it can be concluded that the proposed algorithm converges to the near-optimal solution without initial knowledge of the channel statistics. Further on, it outperforms the DQN and the Random method. Performance improvement over the DQN approach increases with the number of channels.

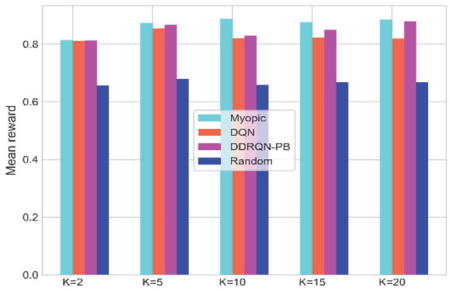

Fig. 2. Mean reward per slot in the first scenario.

Next, we analyze a scenario where 20 channels are divided into non-overlapping clusters of the same size. In each time slot, only a single cluster is activated. The clusters are being activated according to a fixed pattern. With the probability p=0.7, the next cluster in the pattern becomes activated in the following time slot. With the probability 1-p, the cluster from the previous slot remains activated. Fig. 3 shows the obtained results for the experiments with 1 and 2 channels per cluster (c_1 and c_2). The Myopic policy performs poorly in this scenario because it is not able to recognize and exploit the correlation patters. Thus, the Optimal algorithm is now defined as in [6], assuming that the correlation pattern is known in advance, as well as the transition probability p. From Fig. 3 it can be noticed that DDRQN-PB method converges to the near-optimal solution even in very complex communication scenarios where only a single channel is vacant in each time slot. That is not the case with the DQN algorithm. It should be noted that the performance improvement increases with the transition uncertainty, i.e. with 1-p. For the larger cluster sizes, we have found that the performance gap between DQN and DDRQN-PB decreases, but remains significant when transition uncertainty is relatively high. Thereby, the complexity of the DDRQN-PB in terms of the number of

parameters required is smaller.

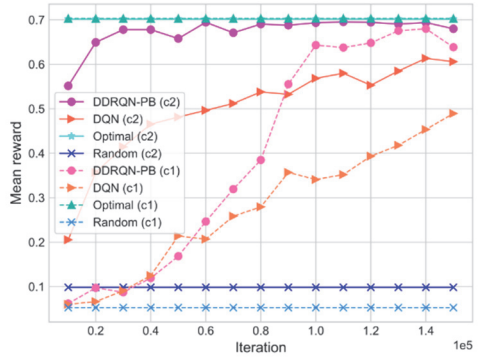

Fig. 3. Mean reward per slot in the second scenario.

Fig. 4 illustrates the contribution of different features of DDRQN-PB method during the training process, in the scenario with clusters of size 2 (K=20). It can be concluded that DRQN (DQN enhanced with the LSTM layer only) brings significant performance improvement. The priority buffer and the double Q-learning extensions further increase the convergence rate.

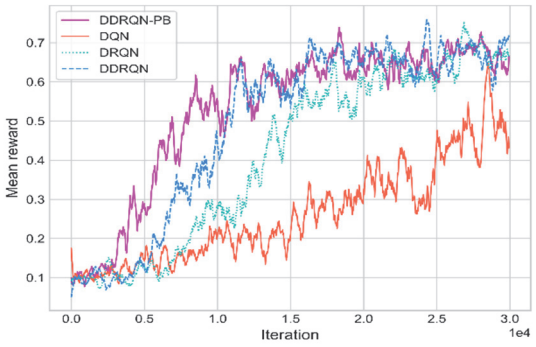

Fig. 4. The comparison of different DQN-based DSA algorithms.

Fig. 5 shows how the length of history of observations used in state definition (N) affects the training process of DDRQN-PB algorithm. It can be concluded that the training process is more stable for larger N in the considered scenario. Although the optimal value of N depends on the channel occupancy pattern that should be learned by the algorithm, it is important to notice that in the case of DDRQN-PB N does not need to be adjusted accurately. If changes of channel occupancy pattern are expected, then N could be set to a larger value and that will not increase the convergence time. The impact of N on the training process of DQN algorithm is shown in Fig. 6. It can be noticed that performance significantly deteriorates with the increase of N. This suggests that the DQN algorithm would not be able to learn the behavior of more complex environments within an acceptable time.

Next, we analyzed the ability of the proposed algorithm to adapt to a dynamic environment by changing the channel occupancy pattern over time. Channels are divided into non-overlapping clusters of size 1, and an initial cluster switching pattern is generated in the same way as in the previous simulations. As the algorithm takes about 100000 iterations to find a near-optimal policy in such a scenario (Fig. 3), we introduce a new random cluster switching pattern in the 100000th iteration. Fig. 7 shows the mean reward results achieved during training. Thereby, we have investigated how the performance of the algorithm is affected by different exploration strategies:

- the decayed ε -greedy with ε reset in case of a larger performance drop (explained in Section III),
- the ordinary ε -greedy with $\varepsilon = \varepsilon_{start} = 0.1$,
- Random Network Distillation (RND) [16] – In this approach, the agent receives a small intrinsic reward in addition to the regular reward. The intrinsic reward motivates the agent to explore unfamiliar states. To compute the intrinsic reward two extra neural networks are introduced, which take the current observation as input and output a single number. The first network is randomly initialized, and it is not trained. The second network is trained to minimize MSE between its prediction and the output of the first network. The absolute difference between the outputs of the first and the second NN is used as the intrinsic reward. The more the agent has explored some state, the smaller the difference (i.e. intrinsic reward) will be. In simulations, we formed RND NNs by using two hidden layers of dimension 64. The intrinsic reward is normalized to 0-1 range.
- NoisyNets [17] - The Gaussian noise is added to the fully-connected layers of the DRQN, and the parameters of the noise are adjusted during training. In this way, very complex and state-dependent changes in policy can be introduced.

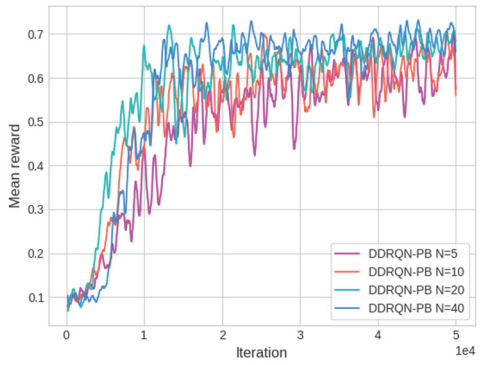

Fig. 5. The impact of N of DDRQN-PB performance.

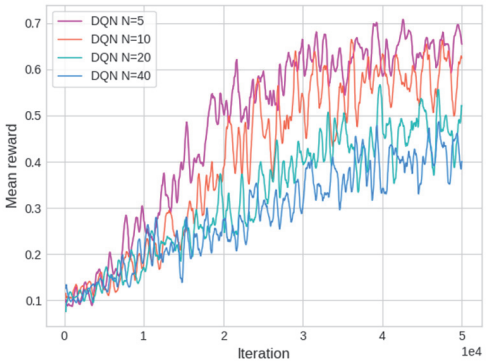

Fig. 6. The impact of N of DQN performance.

Since our analysis has shown that resetting the experience buffer and network parameters is always beneficial in case of significant environmental changes, we have done this in each simulation scenario to ensure a fair comparison. Also, it is important to note that at the beginning of each simulation an agent makes random actions consecutively for some time to generate sufficient

experiences to start the training process. Thus, as can be seen from Fig. 7, the convergence rate differs substantially for the first and the second cluster switching pattern. The conclusions below are derived by observing the training process after the change of the environment.

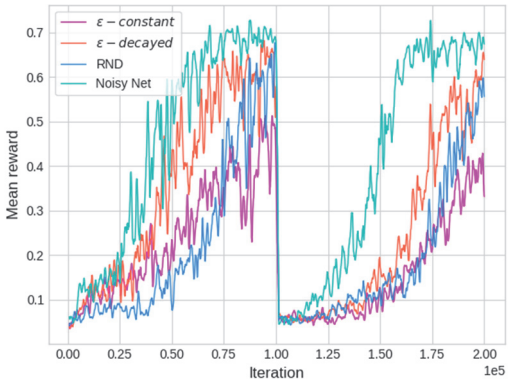

Fig. 7. The performance of DDRQN-PB with different exploration methods.

From Fig. 7, it can be concluded that the *decayed-* ε *-greedy* method outperforms the *constant-* ε *-greedy* method significantly. This is so because *decayed-* ε *-greedy* reduces performance loss due to random actions as the algorithm converges to the optimal policy. On the other side, it is interesting to note that *decayed-* ε *-greedy* outperforms RND, which is a significantly more complex exploration method. This could be so because of the large state space and inappropriate balance between the regular and the intrinsic reward, which has to be carefully tuned. The presented results for RND correspond to the case where the regular and intrinsic rewards are equally weighted. The best performance by far is achieved with the NoisyNet variant of the algorithm. Compared to *decayed-* ε *-greedy*, the NoisyNets exploration doubles the number of parameters in fully connected layers. On the other hand, the performance gain is exceptional, and it is worthy to implement it in a sparse-reward environment if the increase in computational delay is not significant.

V. Conclusion

In this paper, we have proposed a new DSA method that enables SU to make near-optimal channel selection decisions based on the fixed-length history of partial observations regarding the spectrum. The proposed method uses a deep Q-learning neural network with the LSTM layer to efficiently capture stochastic channel switching patterns. Additionally, we adopt a double Q-learning architecture with the prioritized experience buffer to increase the convergence rate. The simulation results have confirmed that the proposed DDRQN-PB outperforms the DQN approach [6] in scenarios with both correlated and uncorrelated communication channels. Thereby, the performance gap increases with the number of channels and channel transition uncertainty. The algorithm version proposed in [13] is augmented with several features that enable faster adaptation to dynamic environments. The performance of different DRL exploration strategies has been evaluated in sparse-reward environments. It has been concluded that decayed ε-greedy strategy, with the proposed modifications, achieves promising results. On the other hand, exploration with Noisy Networks [17] brings a substantial performance gain at the expense of somewhat greater complexity.

The research presented in this paper is only an initial step towards an applicable DRL solution for DSA. Further steps should involve work on adaptive exploration strategies with low computational overhead as well as analysis of multi-node environments. Finally, we will consider the integration of the proposed approach in software-defined IoT environments [18]-[19].

References

[1] S. Tomovic, M. Pejanovic-Djurisic, and I. Radusinovic, “SDN based Mobile Networks: Concepts and Benefits,” *Wireless Personal Communications*, vol. 78, no. 3, pp. 1629-1644, 2014.

[2] G. Gardasevic *et al.*, “The IoT Architectural Framework, Design Issues and Application Domains,” *Wireless Personal Communications*, vol. 91, no. 1, pp. 127-148, 2017.

[3] H. Chang, *et al.*, “Distributive Dynamic Spectrum Access Through Deep Reinforcement Learning: A Reservoir Computing-Based Approach,” *IEEE IoT Journal*, vol. 6, no. 2, pp. 1938-1948, 2019.

[4] G. Hattab and M. Ibnkahla, “Multiband Spectrum Access: Great Promises for Future Cognitive Radio Networks,” *Proc. of the IEEE*, vol. 102, no. 3, pp. 282-306, March 2014.

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

[6] S. Wang, H. Liu, P. H. Gomes and B. Krishnamachari, “Deep Reinforcement Learning for Dynamic Multichannel Access in Wireless Networks,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 2, pp. 257-265, June 2018.

[7] S. H. A. Ahmad, M. Liu, T. Javidi, Q. Zhao and B. Krishnamachari, “Optimality of Myopic Sensing in Multichannel Opportunistic Access,” *IEEE Trans. Inf. Theory*, vol. 55, no. 9, pp. 4040-4050, Sept. 2009.

[8] K. Liu and Q. Zhao, “Indexability of restless bandit problems and optimality of whittle index for dynamic multichannel access,” *IEEE Trans. Inf. Theory*, vol. 56, no. 11, pp. 5547–5567, Nov 2010.

[9] Y. Li, W. Zhang, C. Wang, J. Sun and Y. Liu, “Deep Reinforcement Learning for Dynamic Spectrum Sensing and Aggregation in Multi-Channel Wireless Networks,” *IEEE Trans. on Cog. Comm. and Netw.*, vol. 6, no. 2, pp. 464-475, June 2020.

[10] H. van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-Learning,” *Proc. of the AAAI-16*, pp. 2094–2100, 2016.

[11] T. Schaul, J. Quan, I. Antonoglou and D. Silver, “Prioritized experience replay,” arXiv preprint arXiv:1511.05952 (2015).

[12] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink and J. Schmidhuber, “LSTM: A Search Space Odyssey,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222-2232, Oct. 2017.

[13] S. Tomovic and I. Radusinovic, “A Novel Deep Q-learning Method for Dynamic Spectrum Access,” *2020 28th Telecommunications Forum (TELFOR)*, 2020, pp. 1-4.

[14] S. Y. Chen, *et al.* “Stabilizing Reinforcement Learning in Dynamic Environment with Application to Online Recommendation”, *Proc. of the 24th ACM SIGKDD Intern. Conf. on Knowledge Discovery & Data Mining*, 2018, pp. 1187–1196.

[15] T. H. Teng, A. H. Tan, Y. S. Tan, “Self-Regulating Action Exploration in Reinforcement Learning”, *Procedia Computer Science*, vol. 13, pp. 18-30, 2012.

[16] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, “Exploration by random network distillation,” arXiv preprint arXiv:1810.12894 (2018).

[17] M. Fortunato *et al.*, “Noisy networks for exploration,” arXiv preprint arXiv:1706.10295 (2017).

[18] B. Skrbic *et. al.*, “A decentralized platform for heterogeneous IoT networks management,” *2018 23rd Int. Scientific-Professional Conference on Information Technology (IT)*, 2018, pp. 1-4, pp.

[19] S. Tomovic *et al.*, “An architecture for QoS-aware service deployment in software-defined IoT networks,” 2017 20th WPMC, 2017, pp. 561-567.