Dynamic Spectrum Access in Non-stationary Environments: A DRL-LSTM Integrated Approach

Mingjie Feng[1], Wenhan Zhang[2], and Marwan Krunz[2]
[1]Wuhan National Laboratory for Optoelectronics, Huazhong University of Science & Technology, Wuhan, 430074 China
[2]Dept. Electrical & Computer Engineering, The University of Arizona, Tucson, AZ 58721 USA
Email: mingjiefeng@hust.edu.cn, wenhanzhang@email.arizona.edu, krunz@email.arizona.edu

***Abstract*—In this paper, we investigate the problem of dynamic spectrum access (DSA) in non-stationary environments, where secondary users (SUs) and primary users (PUs) operate over a shared set of orthogonal channels. The non-stationarity is caused by the time-varying PU activity and the coupled channel access strategies of different SUs. Considering such non-stationarity and the channel dynamics, the DSA problem is formulated as a hidden-mode Markov Decision Process (HMMDP), which can be decomposed into multiple MDPs under different modes. At each time, one of the modes is active, each mode corresponds to a unique MDP. The HMMDP is solved when the *active* mode is determined and the MDP under this mode is solved. We first propose a deep reinforcement learning (DRL) framework for solving the MDP under a given mode. We then propose a long short-term memory (LSTM)-based approach to predict the active mode at each time slot. Simulation results show that the proposed scheme outperforms benchmark schemes by achieving significantly fewer collisions and improved spectrum utilization.**

***Index Terms*—Dynamic spectrum access; non-stationary environment; hidden-mode Markov Decision Process; deep reinforcement learning; long short-term memory.**

I. Introduction

The proliferation of wireless applications has triggered unprecedented growth of mobile traffic. It is projected that the sixth-generation (6G) wireless networks will serve 10^7 mobile devices per km^2 [1], [2], which represents a 10x growth from the fifth-generation (5G) networks. Meanwhile, with the emergence of data-intensive applications, mobile devices need to be supported by wireless links with much higher data rates. Given the limited spectrum, providing high data rate services to a large number of devices is challenging. Utilizing new spectrum bands, licensed or unlicensed, is a promising approach to deal with such a challenge. However, the expansion of licensed bands is limited by the high cost of the spectrum and regulatory constraints. In contrast, unlicensed bands are free to use, but are often overcrowded and are subject to strict operational rules due to interference concerns. Due to these limitations, there is a consistent scarcity of spectrum bands, which necessitates achieving higher spectrum utilization in future wireless networks.

A popular approach to increase spectrum utilization is to support dynamic spectrum access (DSA) [3]. DSA allows secondary users (SUs) to opportunistically access the channels of primary users (PUs) without causing interference to these PUs. DSA has been applied to the Advanced Wireless Services-3 (AWS-3) band, as well as the Citizens Broadband Radio Service (CBRS) band. Furthermore, DSA-based Wi-Fi operation in TV white space has been standardized in IEEE 802.11af. However, due to practical concerns and various challenges that remain unsolved, current DSA systems operate in a conservative way. Thus, the full potential of DSA has not yet been harnessed.

A key design issue in DSA is to enable the SUs to accurately detect idle channels. Most existing works adopt a listen-before-talk (LBT) approach, where an SU senses the channels and makes its channel access decision purely based on the sensing result [5]. However, this approach can only be applied in environments with slowly varying spectrum availability, since it is based on the assumption that the sensing results at a given instant remain valid for a sufficiently long time. To enable DSA under fast-varying spectrum dynamics, an SU needs to *predict* the spectrum usage of PU(s) and access the channels accordingly. As indicated in existing works, spectrum usage exhibits a Markovian behavior in fast-varying environments [4]. Thus, the DSA problem was formulated as Markov Decision Process (MDP) and solved with reinforcement learning (RL) algorithms [6]–[12]. In particular, due to the ability to solve large-scale MDPs, deep reinforcement learning (DRL) was recently employed as a powerful tool for capturing the temporal variations in spectrum usage make optimized DSA decisions (e.g., [9]–[12]). The key idea of DRL is to train a deep neural network (DNN) to approximate the state-action value functions (a.k.a. Q-functions). With these estimated Q-values, a near-optimal policy (i.e., action selection strategy under a given state) can be obtained.

Although DRL has been successfully applied to DSA systems in literature, most existing works consider a stationary environment with fixed spectrum dynamics. Spectrum availability patterns in actual DSA systems may non-stationary. On the one hand, the variations of PU activities (busy or idle) may change over time (e.g., a PU is more likely to be active during certain periods of a day). On the other hand, with potential mutual interference between SUs, the channel access behavior of an SU is part of the environment observed by other SUs. As each SU adapts its channel access strategy based on the strategies of other coexisting SUs, the statistics of spectrum dynamics observed by each SU change over time. Due to such non-stationarity, the MDP to be solved by each SU is time-dependent. Thus, standard DRL algorithms can only learn short-term variations in spectrum usage, while the long-

term trends caused by the time-varying PU activity patterns and the SU impact on such trends cannot be captured.

In this paper, we investigate the problem of DSA in non-stationary environments, aiming to maximize the number of collision-free transmissions for each SU. We propose an integrated approach that combines DRL and long short-term memory (LSTM) to derive efficient solutions for SU channel access. The main contributions of this paper are summarized as follows:

- We formulate the DSA problem as a hidden-mode MDP, which can be decomposed into multiple stationary MDPs, each corresponding to a specific mode. The MDPs under various modes share the same setup of state, action, and reward, but with different transition probabilities. At each time slot, the system is in one of the modes, which is determined by the PU activity pattern.
- We first propose a DRL-based solution to the MDP (i.e., SU channel access) under a given mode. In particular, we apply a mechanism called *importance sampling* to stabilize and accelerate the training process.
- We then propose an LSTM-based approach to capture the temporal pattern of spectrum usage and predict the active mode at each time slot. Each SU selects the piecewise-optimal policy under that predicted mode.
- We evaluate the performance of the proposed schemes with simulations. The results show that our approach can significantly improve spectrum utilization and reduce collision rates compared to benchmark schemes.

In the remainder of this paper, we first present the system model and problem formulation in Section II and III, respectively. The solution algorithms are introduced in Section IV. We then show the simulation results and conclude the paper in Section V and VI, respectively.

II. System Model

We consider a wireless system with K SU links, which coexist with M PU links. The two sets of links operate over a shared set of N orthogonal channels indexed by $n \in \{1, \ldots, N\}$. Time is slotted. At any time slot t $t = 1, 2, \ldots$, the activity of PU m ($m \in \{1, \ldots, M\}$) follows a two-state Markov chain with states 1 (active) and 0 (idle), and with transition probabilities given by:

$$\mathbf{P}_m(t) = \begin{bmatrix} p_m^{(00)}(t) & p_m^{(01)}(t) \\ p_m^{(10)}(t) & p_m^{(11)}(t) \end{bmatrix}. \tag{1}$$

We consider the case that each PU link can utilize one or more channels simultaneously, and the channel dynamics of each PU are time-invariant. The channel allocation for PUs is coordinated through a centralized entity, which ensures no collision between any two PUs. Let b_m be the number of channels used by PU m, we assume that $\sum_{m=1}^{M} b_m \leq N$.

At time t, the probability that SU k ($k \in \{1, \ldots, K\}$) has data to transmit is denoted by $q_k(t)$. Once SU k starts transmitting, it needs a random number of time slots (not necessarily contiguous) to complete its transmission. This number is dictated by the packet size at the SU as well as its modulation and coding scheme (MCS). At each time slot, an SU may access one of the N channels or not access any channel. Before accessing a channel, each SU senses all the N channels to determine which ones are idle. If a channel is sensed as busy by an SU, the SU also identifies the ID of the PU that is using the channel. After that, each SU makes its decision on channel access in the next time slot. From the perspective of each SU, the transmission patterns of coexisting users (PUs and SUs) are unknown in advance. We assume that two links (PU or SU) would collide with each other if their communication ranges overlap and they transmit on the same channel during the same time slot, i.e., we assume the so called *protocol model* for defining collisions.

III. Problem Formulation

The channel access of each SU is modeled as a hidden-mode MDP, which consists of a finite set of MDPs that share the same state space, action space, and reward functions, but have different transition probabilities caused by the non-stationary PU activities [13], [14]. Mode transitions happen at a much slower pace than state transitions. As a result, the mode remains the same during a time period consisting of multiple time slots. As shown in Fig. 1, we consider L modes, $\mathcal{L} = \{M_1, \ldots, M_L\}$. At each time slot, the system is in one of the modes, and that mode is called the active mode, which varies over time with a pattern that is unknown to all SUs.

Since the active mode at each time slot is determined by the corresponding PU activities (i.e., $\mathbf{P}_m(t)$ given in (1)), we classify the activities of each PU m according to the values of $p_m^{(01)}(t)$ and $p_m^{(10)}(t)$, $\forall m \in \{1, \ldots, M\}$. Specifically, we equally divide the range $[0, 1]$ into D non-overlapping segments given by $[0, \frac{1}{D}], [\frac{1}{D}, \frac{2}{D}], \ldots, [\frac{D-1}{D}, 1]$. Then, the activity pattern of PU m is classified according to:

$$\begin{aligned}
&\text{Pattern } 1: \quad p_m^{(01)}(t) \in [0, \frac{1}{D}], \;\; p_m^{(10)}(t) \in [0, \frac{1}{D}] \\
&\text{Pattern } 2: \quad p_m^{(01)}(t) \in [0, \frac{1}{D}], \;\; p_m^{(10)}(t) \in [\frac{1}{D}, \frac{2}{D}] \\
&\qquad \vdots \\
&\text{Pattern } D: \quad p_m^{(01)}(t) \in [0, \frac{1}{D}], \;\; p_m^{(10)}(t) \in [\frac{D-1}{D}, 1] \\
&\text{Pattern } (D+1): \quad p_m^{(01)}(t) \in [\frac{1}{D}, \frac{2}{D}], \;\; p_m^{(10)}(t) \in [0, \frac{1}{D}] \\
&\qquad \vdots \\
&\text{Pattern } D^2: \quad p_m^{(01)}(t) \in [\frac{D-1}{D}, 1], \;\; p_m^{(10)}(t) \in [\frac{D-1}{D}, 1]
\end{aligned} \tag{2}$$

A mode is defined by a unique combination of activity patterns of all PUs. Given M PUs, the total number of modes is D^{2M}. From (2), we can see that the number of modes L is determined by $L = D^2$.

The MDP under each mode $l \in \mathcal{L}$ is specified by a tuple $\mathcal{M}_l =< \mathcal{S}, \mathcal{A}, \mathcal{P}_l, \mathcal{R} >$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R}$ is the set of reward functions, and $\mathcal{P}_l$ is

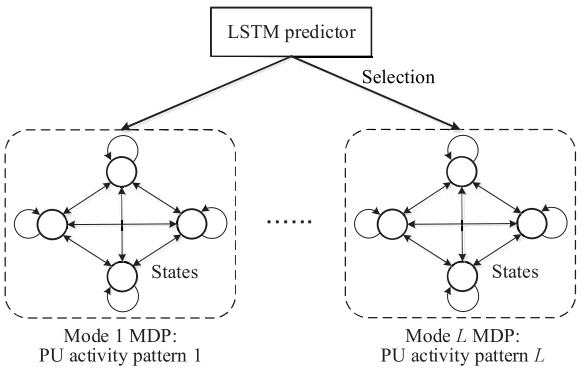


Fig. 1. Model of hidden-mode MDP for non-stationary spectrum dynamics.

the transition probability matrix of MDP l. The system state at time t is defined as the spectrum usage pattern, which is represented by a $N \times 1$ vector with elements 0 or 1:

$$S(t) = [S^{(1)}(t), \ldots, S^{(N)}(t)] \tag{3}$$

where $S^{(n)}(t) = 1$ indicates that channel n is busy and $S^{(n)}(t) = 0$ indicates otherwise. The action of SU k at time t is its channel access strategy, denoted by $A_k(t) \in \{0, 1, \ldots, N\}$, where $A_k(t) = n$ indicates that SU k selects to access channel n at time t and $A_k(t) = 0$ indicates that SU does not access any channel at time t. Let r_t be the reward that an SU receives at time t. This reward is determined by:

1) The SU accesses a channel that is not used by a PU or any other SUs within its communication range. In this case, no collision would happen and $r_t = 1$.
2) The SU accesses a channel that is used by a PU or another SU within its communication range. In this case, a collision would happen and $r_t = -1$.
3) The SU does not access any channel, and thus $r_t = 0$.

In our problem, each SU acts as an agent, aiming to learn from the environment and find the optimal policy that maximizes the expected long-term accumulated discounted reward. A policy determines the strategy of taking actions under certain system states. In general, a policy is in a stochastic form to enable the exploration of different actions. Let π_k be the policy of SU k, given by $\pi_k(a|s) = \Pr(A_k(t) = a|S(t) = s)$, where $a \in \mathcal{A}$ and $s \in \mathcal{S}$. The key step to obtain the optimal policy is finding the value of each state-action function, also known as Q-function, which is defined by:

$$\begin{aligned} Q_\pi(s, a) &= \mathbb{E}_\pi[G(t)|S(t) = s, A_k(t) = a] \\ &= r_s^{(a)} + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^{(a)} v_\pi(s') \end{aligned} \tag{4}$$

where $G(t)$ is the cumulative discounted reward obtained after time t, given by $G(t) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, and γ is a discount factor used to balance the long-term and short-term rewards. $G(t)$ consists of the instant reward $r_s^{(a)}$ and the expected discounted future rewards when the system transitions to other states. $P_{ss'}^{(a)}$ is the transition probability from state s to s' when action a is taken. $v_\pi(s)$ is the state-value function for state s, which defined as the expected reward when the system is in state s and follows policy $\pi(s)$. This $v_\pi(s)$ is given by $v_\pi(s) = \mathbb{E}_\pi[G(t)|S(t) = s] = \sum_{a \in \mathcal{A}} \pi(a|s) Q_\pi(s, a)$. Given the values of Q-functions, the optimal policy π^* is obtained by solving $\pi^* = \underset{\pi}{\text{argmax}} Q_\pi(s, a)$.

IV. Solution Algorithms

In this section, we solve the hidden-mode MDP problem. At each time slot, the system predicts the active mode using observed spectrum usage history. Based on this prediction, an SU can select the piecewise-stationary policy that is optimal for that mode.

A. DRL-based Spectrum Access Under a Given Mode

We first consider the spectrum access decision of an SU when the active mode is given. Because the mode is determined by PU activities, all SUs operate under the same mode.

1) MDP Solution for Each SU: A common approach to solve an MDP is Q-learning, which is based on iterative updates of the Q-values as the agent interacts with the environment. Suppose the system is in state s_t and the agent takes action a_t at time t. The agent updates the Q-value of state-action pair (s_t, a_t) based on the observed reward and state transition:[1]

$$\begin{aligned} Q(s_{t+1}, a_{t+1}) &\leftarrow Q(s_t, a_t) \\ &+ \alpha \left[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right]. \end{aligned} \tag{5}$$

Despite the simplicity of the updating rule in (5), a traditional Q-learning approach requires the agent to visit every state-action pair and store all the Q-values, which is computational impractical in systems with large numbers of states and actions. An effective solution to this challenge is to use a neural network (NN) to approximate the Q-values, denoted by $Q(s, a, \mathbf{w}) \approx Q_\pi(s, a)$, where $\mathbf{w}$ is the weight matrix of the NN. With the data observed from the environment (i.e., training samples), the NN can be trained to map the state-action pairs to their corresponding Q-values. A key issue for the application of NN in Q-learning is the correlations between training samples [16], which potentially cause the learning process to become unstable or even diverge. To address this issue, a DRL framework based on *experience replay* was proposed in [16], in which a deep Q-network (DQN) is trained to approximate the Q-values. The input layer of the DQN is set to the system state vector $S(t)$ and the output layer is set to generate the Q-values of all actions under the input state.

The idea of experience replay is to "freeze" the agent's experience for a certain time and use it to train the DQN later, thus breaking the correlation between training samples. At each time slot, the agent takes actions according to its current policy and stores the experience, $e_t = (s_t, a_t, r_t, s_{t+1})$, in a target network. During DQN training, the experience data are randomly sampled in the form of minibatches from the target

[1]To avoid confusion, $S(t)$ is the vector for the state at time t which can be any state $s \in \mathcal{S}$, while s_t is the specific state that is observed by the agent at time t. The same applies to $A_k(t)$ and a_t.

network to break the correlation between training samples. Using these samples, the weights of the DQN are updated by minimizing a loss function defined by the mean square error between the DQN and the target network, given as:

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a', \mathbf{w}^-) - Q(s, a, \mathbf{w})\right)^2\right] \quad (6)$$

where the expectation is taken with respect to all samples of (s, a, r, s') in the minibatch, $\mathbf{w}^-$ and $\mathbf{w}$ are the weights of the target network and the DQN, respectively. The problem given in (6) can be solved via stochastic gradient descent. Note that the weights of the target network are updated less frequently to reduce the correlation with the DQN.

After the training of the DQN is completed, each SU takes action based on the estimated Q-values. Specifically, the SU first senses the system state at time t, i.e., the vector $S(t)$. Then, it inputs $S(t)$ to the DQN and obtains the Q-values of all actions (i.e., different channel access strategies). Finally, the SU adopts an ϵ-greedy approach to allow random exploration, where the SU selects the optimal action (one with the largest Q-value) with probability $1 - \epsilon$ and randomly selects another action with probability ϵ.

2) Convergence Acceleration with Importance Sampling: Because some SUs may overlap in their communication ranges, and they learn and adjust their policies simultaneously, the environment from the perspective of each SU is non-stationary and there may be a ping-pong effect. For example, two SUs may select the same channel at a given time slot and experience a collision. At the next time slot, they may both select another same channel and observe another collision. Hence, the system may take an extremely long time to converge. On the other hand, when experience replay-based training is applied where each agent's experience is "frozen" for a period of time, the SUs do not interact with each other at the same pace and are unable to learn from instantaneous feedback. As a result, the experience replay may be unstable, causing the system to fail to converge [17].

To stabilize training and accelerate convergence, a key observation is that the environment observed by an SU can be made stationary conditioned on the policies of other SUs. However, given that samples generated by experience replay are obsolete (i.e., cannot reflect current system dynamics), the SUs may not be able to disambiguate the age of the sampled data from the replay memory. Importance sampling with off-environment training is an effective approach to tackle this issue [18]. The idea of importance sampling is to assign an importance ratio to each sample in a minibatch (i.e., each tuple $e_t = (s_t, a_t, r_t, s_{t+1})$), in a way that enables each SU to learn in a reliable off-environment. Let $\{t_j\}(j = 1, \ldots, J)$ be the time when the samples with starting state s and action a are collected in a minibatch. From the perspective of SU k, denote $\mathbf{a}_{-k}$ as the joint action of other SUs at time t_j and let $\pi_{-k}^{(t_j)}(\mathbf{a}_{-k}|s)$ be the joint policy of other SUs at time t_j. Then,

$$\pi_{-k}^{(t_j)}(\mathbf{a}_{-k}|s) = \prod_{k' \notin k} \pi_{k'}^{(t_j)}(a|s). \quad (7)$$

Suppose the minibatch containing the samples generated at $\{t_1, \ldots, t_J\}$ are used for training the DQN at time t_i. Then, the importance ratio is set to $\frac{\pi_{-k}^{(t_i)}(\mathbf{a}_{-k}|s)}{\pi_{-k}^{(t_j)}(\mathbf{a}_{-k}|s)}$. Incorporating the importance ratio, the loss function at time t_i is given by:

$$\mathcal{L}(\mathbf{w}) = \sum_{j=1}^{J} \frac{\pi_{-k}^{(t_i)}(\mathbf{a}_{-k}|s)}{\pi_{-k}^{(t_j)}(\mathbf{a}_{-k}|s)} \left[(y_j - Q(s, a, \mathbf{w}))^2\right] \quad (8)$$

where y_j is the output of the target network at time t_j, given by $y_j = r_j + \gamma \max_{a'_j} Q(s'_j, a'_j, \mathbf{w}^-)$. With the *weighted* loss function (8), the error of estimating the policies of other SUs caused by time difference can be corrected. For example, if a joint policy rarely occurs when it was stored in the minibatch (i.e., $\pi_{-k}^{t_j}(\mathbf{a}_{-k}|s)$ is low) but frequently occurs when used for DQN training (i.e., $\pi_{-k}^{(t_i)}(\mathbf{a}_{-k}|s)$ is high), a higher importance is given to that joint policy. This way, each SU is learning in an off-environment, resulting in faster convergence.

B. LSTM-based Mode Prediction

To implement mode prediction, each SU sends its sensing outcome (PU activities on all channels and active PU IDs) to a network controller, which determines the state of each PU (busy or idle) by merging the outcomes from all SUs (e.g., using majority rule), and trains an LSTM-based neural network for mode prediction. Such prediction can be characterized by the transition probabilities in (1). To capture the temporal pattern of time-varying PU activity, we formulate the prediction of the transition probabilities of each PU as a regression model:

$$\mathbf{Z}_m(t) = f_m\left(\rho_m(t-1), \rho_m(t-2), \rho_m(t-3), \ldots\right), \quad m \in \{1, \ldots, M\} \quad (9)$$

where $\mathbf{Z}_m(t)$ is the prediction outcome, $\rho_m(t)$ is the PU activity indicator, where $\rho_m(t) = 1$ indicates that PU m is busy at time t and $\rho_m(t) = 0$ indicates otherwise. $f_m(\cdot)$ is a function that maps a sequence of recorded activities of PU m before time t to a prediction of its transition probabilities at time t. This function needs to represent the complex relationship between the current and previous PU activities. Intuitively, the recent "fresh" data are expected to provide more useful information about the current system dynamics, compared to the data generated a long time ago. However, if we only rely on short-term information, the prediction accuracy can be degraded by errors caused by instant variations. In addition, the long-term trend that evolves over time (e.g., a PU has a higher probability to be active during certain periods of a day) is not captured.

To capture the temporal pattern of PU activity and intelligently combine long-term and short-term data, an LSTM network is applied for mode prediction. The input layer parameters of the LSTM network are set to the on-off activities of the M PUs. The output layer parameters are the predicted values of $p_m^{(01)}(t)$ and $p_m^{(10)}(t)$. Two fully connected hidden layers are used for feature extraction and learning the nonlinear relationship between the inputs and outputs. The activation

function is set to be Rectified Linear Unit (ReLU), given by $\mathrm{ReLU}(x) = \max(x, 0)$. The ReLU function is widely used in deep NN due to its capability of sparse representation and low computational complexity. Backpropagation through time is used to train the LSTM network.

A design challenge is how to obtain the labels for training (i.e., the true values of $p_m^{(01)}(t)$ and $p_m^{(10)}(t)$). One approach is to use the statistics of the PU activity at the training sequence to generate the labels. Given a sufficient number of iterations, the values of $p_m^{(01)}(t)$ and $p_m^{(10)}(t)$ can be approximated. Another approach is to approximate the values of $p_m^{(01)}(t)$ and $p_m^{(10)}(t)$ by their time-averaged values over a short duration around t. However, such approximation may not be accurate with insufficient data. Thus, the first approach can be applied in systems that allow long training times, while the second approach is preferred in systems that require fast convergence.

After the transmission probabilities of a PU are predicted, its activity pattern can be classified according to (2). Combing the activity patterns of all PUs, the active mode can be determined.

V. Performance Evaluation

We evaluate the performance of the proposed strategies using Matlab and Python simulations. We let $N = 10$ channels, which are allocated to 4 PUs. The numbers of channels allocated to the 4 PUs are 3, 3, 2, and 2, respectively. The PU probabilities of being active vary over time during each day, as indicated in Table I. There are 10 SUs. The probability that an SU has packets to transmit is 0.5. On average, there are 3 other SUs in the communication range of each SU. We use a Keras-based framework in Python to implement the proposed DQN approach. There are 10 neurons in the input layer, which correspond to the usage pattern of the 10 channels. The output layer has 11 neurons, each corresponding to the Q-value when taking a certain action under the input system state. There are 2 hidden layers, each with 24 neurons. ReLu is utilized as the activation function.

During the DQN training, the learning rate α is set to 0.1, and the discount factor γ is 0.95. As mentioned, the SU adopts the ϵ-greedy strategy for selecting actions to balance between exploration and exploitation. Considering the fact that more information about the environment is obtained as the training proceeds, ϵ (the probability that the SU randomly takes an action) is set to decrease over time [19]. This way, the SU is more likely to take a random action at the beginning, while less likely to explore new actions as the learning continues. The initial value of ϵ is set to 0.7, and ϵ decays in every episode of training with a factor of 0.9999.

We compare five schemes for DSA. The first scheme is termed "random selection", where each SU randomly selects a channel to transmit over in each time slot. The second scheme is termed "myopic", where each SU makes channel access decisions purely based on the current sensing result, i.e., an SU will access a channel if the channel is sensed as idle at the current time slot. The myopic scheme has been used in traditional DSA systems (e.g., cognitive radio systems). The third scheme is termed "DQN only", where only the proposed DQN-based approach is applied (i.e., without mode prediction). The fourth scheme is termed "DQN+EWMA", where the exponential moving average (EWMA) method is applied for mode prediction. Specifically, the index of the PU activity pattern in the current time slot is taken as the rounded values of the weighted (exponential weights) sum of indices of PU activity patterns in the previous time slots, given by:

$$l(t) = \mathrm{Round}\left(l(t-1)w(t-1) + l(t-2)w(t-2) + l(t-3)w(t-3) + \ldots\right) \quad (10)$$

where $w(\cdot)$ are exponential variables. Finally, the proposed integrated approach is termed "DQN+LSTM".

We evaluate the collision rate under each scheme, defined as the ratio between the number of collided transmissions and the total number of transmissions. To evaluate spectrum utilization, we consider the average channel utilization rate of PUs and SUs, defined as the ratio between the number of collision-free time slots and the number of time slots that the user attempted to access over all channels. We vary the average PU transmission duration, which is an indicator of how frequent PUs switch between busy and idle states.

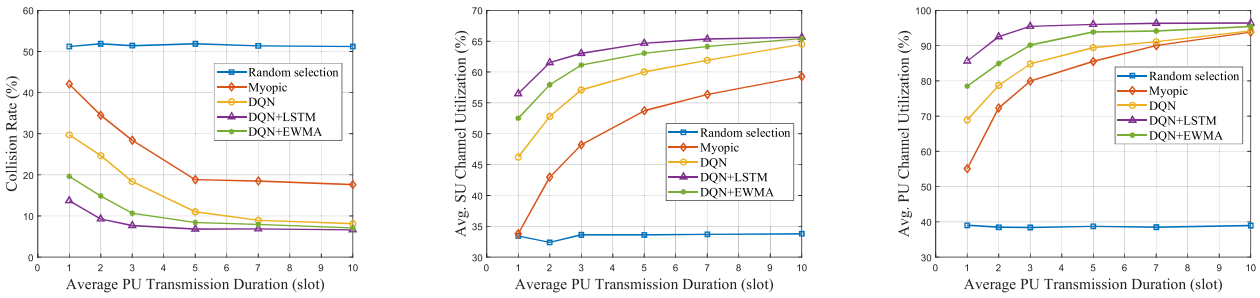
Fig. 2(a) depicts the collision rate of different schemes versus the average PU transmission duration. As expected, the collision rate of the random selection scheme is the highest among all schemes and remains high even when the average PU transmission duration is large (i.e., the PU activity varies slowly). With spectrum sensing, the myopic scheme can lower the chances of collision, especially when the PUs occupy the channels for long periods of time. The DQN-only scheme further lowers the collision rate, because the DQN of each SU has been trained to support collision avoidance. With mode prediction, the DQN+EWMA scheme achieves a lower collision rate than that of the DQN-only scheme, since the short-term spectrum usage variation can be captured. With the LSTM network, the proposed DQN+LSTM scheme achieves the best performance, since the temporal pattern spectrum usage can be accurately predicted, thereby enabling each SU to select the proper MDP.

Fig. 2(b) depicts the spectrum utilization of SUs. The DQN-only scheme achieves better performance than the random selection and myopic schemes, because each SU optimizes its channel access policy under different spectrum sensing results after its DQN has been trained. Benefiting from mode prediction, the DQN+EWMA scheme improves the SU channel utilization rate. With long-term spectrum usage prediction, the DQN+LSTM scheme achieves the highest channel utilization.

Fig. 2(c) depicts PU spectrum utilization. It can be seen that PU channel utilization under the random selection scheme is significantly lower than other schemes. This is due to a higher probability of collisions between SUs and PUs, which decreases the number of successful PU transmissions. In contrast, the number of successful PU transmissions is higher under other schemes, since collision reduction mechanisms are a part of these schemes. When the PU busy period is long, the channel utilization rates of the myopic and the proposed DQN-based schemes become comparable. This happens because

TABLE I
PU BUSY PROBABILITIES USED IN SIMULATIONS

PU ID \ Time	0:00-7:00	7:00-9:00	9:00-12:00	12:00-14:00	14:00-18:00	18:00-20:00	20:00-0:00
PU 1	0.2	0.3	0.7	0.4	0.8	0.6	0.5
PU 2	0.1	0.2	0.6	0.3	0.6	0.4	0.4
PU 3	0	0.1	0.6	0.1	0.5	0.5	0.3
PU 4	0.3	0.5	0.8	0.5	0.8	0.7	0.6



(a) (b) (c)

Fig. 2. Simulation results. (a) average collision rate of different schemes, (b) average SU channel utilization rate of different schemes, (c) Average PU channel utilization rate of different schemes.

as the PU channel usage pattern varies more slowly, the sensing result for a given time slot is likely to be valid for a longer time. However, when the PU busy period is short and the channel state changes rapidly, the proposed DQN-based schemes achieve significantly better performance, due to the fact that they can predict channel usage variations. Finally, we observe that the incorporation of LSTM-based prediction further enhances channel utilization, since mode prediction contributes to a more accurate prediction of PU activity, resulting in a higher PU channel utilization rate.

VI. CONCLUSIONS

In this paper, we investigated the problem of dynamic spectrum access in non-stationary environments. We proposed an LSTM-DRL integrated approach to capture spectrum variation and make intelligent decisions on spectrum access. Simulation results show that the proposed solution significantly reduces the collision rate and improves spectrum utilization.

REFERENCES

[1] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6G networks: Use cases and technologies," *IEEE Commun. Magazine,* vol. 58, no. 3, pp. 55–61, Mar. 2020.

[2] Z. Zhang, Y. Xiao, Z. Ma, M. Xiao, Z. Ding, X. Lei, G. K. Karagiannidis, and P. Fan, "6G wireless networks: Vision, requirements, architecture, and key technologies," *IEEE Veh. Technol. Magazine,* vol. 14, no. 3, pp. 28–41, Sept. 2019.

[3] P. J. Kolodzy, "Spectrum policy task force report," FCC, Tech. Rep., Nov. 2002.

[4] Y. Xing, R. Chandramouli, S. Mangold, and S. S. N, "Dynamic spectrum access in open spectrum wireless networks," *IEEE J. Sel. Areas Commun.,* vol. 24, no. 3, pp. 626–637, Mar. 2006.

[5] Y.-C. Liang et al., "Sensing-throughput tradeoff for cognitive radio networks," *IEEE Trans. Wireless Commun.,* vol. 7 no. 4, pp. 1326–1337, Apr. 2008.

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

[7] C. Tekin and M. Liu, "Approximately optimal adaptive learning in opportunistic spectrum access," in *Proc. IEEE INFOCOM'12,* Orlando, FL, Mar. 2012, pp. 1548–1556.

[8] S. Wang, T. Lv, X. Zhang, Z. Lin, and P. Huang, "Learning-based multi-channel access in 5G and beyond networks with fast time-varying channels," *IEEE Trans. Veh. Technol.,* vol. 69, no. 5, pp. 5203–5218, May 2020.

[9] 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.

[10] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE Trans. Wireless Commun.,* vol. 18, no. 1, pp. 310–323, Jan. 2019.

[11] H.-H. Chang, H. Song, Y. Yi, J. Zhang, H. He, and L. Liu, "Distributive dynamic spectrum access through deep reinforcement learning: A reservoir computing based approach," *IEEE Internet of Things J.,* vol. 6, no. 2, pp. 1938–1948, Apr. 2018.

[12] C. Zhong, Z. Lu, M. C. Gursoy, and S. Velipasalar, "A deep actor-critic reinforcement learning framework for dynamic multichannel access," *IEEE Trans. Cogn. Commun. Netw.,* vol. 5, no. 4, pp. 1125–1139, Dec. 2019.

[13] S. P. M. Choi, D.-Y. Yeung, and N. L. Zhang, "An environment model for nonstationary reinforcement learning," in *Proc. NeurIPS'99,* pp. 987–993, Denver, Colorado, USA, Nov.–Dec. 1999.

[14] S. Padakandla, K. J. Prabuchandran, and S. Bhatnagar, "Reinforcement learning in non-stationary environments," [Online] Available: https://arxiv.org/pdf/1905.03970.pdf.

[15] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote, "A survey of learning in multiagent environments: Dealing with non-stationarity," [Online]. Available: arXiv:1707.09183.

[16] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature,* vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[17] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proc. ICML'17,* pp. 1146–1155, Sydney, Australia, Aug. 2017.

[18] K. A. Ciosek and S. Whiteson, "Offer: Off-environment reinforcement learning," in *Proc. AAAI'17,* pp. 1819–1825, San Francisco, CA, USA, Feb. 2017.

[19] J. Wu, V. Braverman, and L. Yang, "Gap-dependent unsupervised exploration for reinforcement learning," in *Proc. AISTATS'22,* online, Mar. 2022.