Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5577756, 10 pages
https://doi.org/10.1155/2021/5577756

Research Article

Double Deep Recurrent Reinforcement Learning for Centralized Dynamic Multichannel Access

Qianhong Cong and Wenhui Lang

School of Computer and Information of Hefei University of Technology, Hefei, China

Correspondence should be addressed to Wenhui Lang; langwh@hfut.edu.cn

Received 1 March 2021; Revised 5 September 2021; Accepted 24 November 2021; Published 20 December 2021

Academic Editor: Huaming Wu

We consider the problem of dynamic multichannel access for transmission maximization in multiuser wireless communication networks. The objective is to find a multiuser strategy that maximizes global channel utilization with a low collision in a centralized manner without any prior knowledge. Obtaining an optimal solution for centralized dynamic multichannel access is an extremely difficult problem due to the large-state and large-action space. To tackle this problem, we develop a centralized dynamic multichannel access framework based on double deep recurrent Q-network. The centralized node first maps current state directly to channel assignment actions, which can overcome prohibitive computation compared with reinforcement learning. Then, the centralized node can be easy to select multiple channels by maximizing the sum of value functions based on a trained neural network. Finally, the proposed method avoids collisions between secondary users through centralized allocation policy.

1. Introduction

With the rapid development of next generation network technologies such as the mobile Internet and the Internet of Things (IoT), spectrum scarcity has been severe. Furthermore, most of spectrum resources are exclusively and unlimitedly allocated to primary users (PUs) at present, whose mechanisms enable to have no transmission opportunities for secondary users and lead to low spectrum efficiency [1]. According to Federal Communications Commission (FCC), the spectrum is severely underutilized with the utilization rate of some bands as low as 15% in general [2]. The low spectrum utilization and spectrum scarcity have triggered the development of dynamic spectrum access (DSA).

In this work, we consider an overlay DSA environment with multiple PUs, multiple secondary users (SUs), and a centralized node which can be able to detect all channel state at the current time and allocate a channel to each SU for transmitting data during the next time. This is a coordinated multichannel access problem of independent channels in a fully observable scenario. Each channel has two possible states (i.e., occupied by PUs or not) at any given time. We assumed the PUs have no ability to perceive channels state and can avoid collisions with other PUs. In this overlay DSA model, we assume that the transmission will success only if a SU accesses the idle channel. Once two or more users, i.e., PUs or SUs, transmit data using the same channel at the same time slot, the collision will occur and SU transmission will fail. PUs can occupy different channels according to their spectrum behavior. The above model can avoid collisions between SUs completely through overall channel allocation of the centralized node.

We assume the centralized node has cognitive ability that could be able to exploit time-domain holes of channels and improve spectrum utilization efficiency in an unknown environment. For this purpose, reinforcement learning (RL), especially Markov Decision Process (MDPs), is one potential solution due to good decision performance [3]. The advantage of RL can learn how to map state space to action space in situations where the characteristics about the environment are unknown for agent but can be learned through trial and error. However, traditional reinforcement learning is not suitable for scenarios with large state space where many state-action pair are rarely visited, which leads

to the curse of dimensionality and lack of generalization. Motivated by deep learning (DL) success in other domains, Google DeepMind is combined RL into a deep learning network model, i.e., Deep Q-Network (DQN) for solving the above problems, which output an approximate value function to decide its policy [4]. But DQN algorithm overestimates the value function because it employs the same value function to select action and to estimate the value function. A DQN derived algorithm, i.e., Double DQN, has the ability to alleviate overestimation compared to DQN. Besides, Deep recurrent Q-learning (DRQN, another DQN-derived algorithm) can better deal with time sequence problems. Hence, this work gives an extension to DQN approach which utilizes double DQN to stabilize approximate value function and DRQN to extract the related temporal information.

This paper greatly expands on the preliminary work of [5], which applies double deep recurrent reinforcement learning to centralized DSA scenario and gives its simulated performance in the presence of two scenarios. The main contributions of this paper are as follows:

(1) We design a centralized multiuser dynamic spectrum access model, which can effectively avoid the possibility of conflicts between secondary users. Specifically, for each environment state, the probability of each channel being idle is evaluated, and multiple channels with high idle probability are selected for communication transmission of secondary users

(2) This paper explains why reinforcement learning is widely used in the field of communication and its limitations, which leads to the rapid development and wide application of deep reinforcement learning (DRL). Furthermore, the theoretical basis of the proposed algorithm (i.e., Q learning and deep reinforcement learning) is also presented and introduced

(3) A multiuser dynamic spectrum access algorithm based on double deep recurrent reinforcement learning is proposed. The proposed algorithm uses the fully connection layer to directly estimate the channel allocation to avoid the lookup table of value function and uses the Long Short-term Memory network to improve the performance of sequential channel assignment decision

(4) The learning ability of the proposed algorithm is verified through the simulation experiment of single SU Round-Robin Switching Scenario. Simulation experiments on arbitrary switching scenario with single SU demonstrate the robustness of the proposed algorithm. The simulation results of multiple SUs scenarios show that this strategy can effectively avoid the collisions between SUs, reduce the interference of SUs to PUs

The rest of this paper is organized as follows. Section 2 presents the related algorithm application in dynamic spectrum access. Then, the system model and the problem statement are described in Section 3, and Section 4 introduces implementation of related algorithms. Section 5 describes the details of the proposed double deep recurrent reinforcement learning while Section 6 tests the performance of the proposed algorithm in the single SU scenarios and the multiple SU scenarios. Finally, Section 7 concludes the paper.

2. Related Work

In recent years, the machine learning approaches have become a potential solution in wireless communication, as it has excellent decision-making performance facing the unknown system dynamics [6–11]. The authors in [6, 7] model the radar-communications coexistence environment as Markov Decision Process and then apply policy iteration to solve the optimal channel allocation problem. Nevertheless, policy iteration is a model-based RL algorithm that requires the transition probability function and reward function which is obtained during the training phase. Q-learning is also one of the most popular algorithms since it is a model-free RL method that does not depend on environment modeling and can learn the optimal policy via trial and error in an online manner [8–11]. These works mainly use Q-learning and its derived off-policy algorithms (e.g., SARSA) to construct and tabulate value function base on each state-action pair. Compared to the traditional algorithms, these methods decrease the computational complexity and better deal with the lack of the prior knowledge. However, the complexity of tabulating the value function for each state-action pair is still high, and the performance is poor when faced with high-dimensional, large state-space problems. Thus, in order to overcome the limitation of Q-learning, the DQN has been proposed which has a good approximation of the value function based on the neural network.

Most of recent works on DSA have adopted the deep reinforcement learning approach for solving spectrum allocation. For example, Wang et al. proposed DQN approach to overcome the large space challenge and develop a channel allocation policy for single SU in the multiple correlated channel scenarios [12]. However, SUs have no complete state information and directly learn a mapping from partial observations to action space through Deep Q-Network, which limited network utility in DSA. Recurrent neural network (RNN) is proposed to solve such partial observation Markov decision problems (POMDPs) in computer games [13, 14]. Based on this advantage of RNN, Long Short-Term Memory (LSTM) layer (a RNN layer) is added to the neural network for solving such DSA problems, which can maintain an internal state and integrate sequence information [15–17]. Naparstek and Cohen consider a distributed DSA environment where each SU develops Deep Recurrent Q-Network (DRQN) to learn good policies [15]. However, this work assumes that there is no PUs in the scenario, which only represent specific situation and is difficult to apply to current radio environment where most spectrum resources have already been allocated. Different from [15–17] consider a more complex scenario where multiple PUs and multiple SUs coexist and there is no information interaction between the SUs.

Hence, the above paper assumed that wireless communication environment is considered as a partially observable MDP (POMDP), i.e., each node can only observe part of the channel state at each time. But technically, each node can be capable of sensing multiple channels in every time slot, which has been researched in [18, 19]. Besides, the above literature still has high computation complexity in the environment, where the proposed algorithm enables each SU to search spectrum hole in an online and distributed manner. In the distributed DSA model, there is no central node between SUs to coordinate spectrum allocation. This means that each SU can only update policy and select action to maximize their own transmission rate, which may lead collision with other SUs. To decrease collision in the cognitive radio networks, we consider a centralized DSA scenario where there is a centralized node to realize multiuser channel allocation and optimize its policy through deep reinforcement learning.

Other works for DSA have mainly focused on model-dependent setting (i.e., the myopic policy and the Whittle Index) and optimization of DRL. Myopic policy regards spectrum allocation as a multiarmed bandit (MAB), in which the user estimates the immediate reward for each channel and selects a channel that will maximize expected immediate reward [20]. However, the myopic policy ignores the change of communication environment and obtains near optimal only when the channel state transitions are positively correlated or are slightly negatively correlated. The Whittle Index can only acquire optimal performance in the environment that the two-state Markov chain matrix is known, and all channels are independent [21]. The authors of [22] focus on the application of the actor-critic reinforcement learning (an improved DRL algorithm) based framework to dynamic multichannel allocation which make use of the advantage of the value-based and policy-based reinforcement learning algorithms.

3. System Model and Problem Statement

We consider a cognitive radio network (CRN) where there is a centralized node, several primary users (PUs), N secondary users (SUs), and K authorized nonoverlapping channels. We assume that each SU always has transmission demand, i.e., each SU has packets to transmit in each time slot. Each channel has two possible state: good (1) or bad (0) (i.e., occupied by PUs or not). At the beginning of each time slot, the centralized node senses the state of all K channels using a specific observation pattern, chooses N of K channels, and randomly allocates one of N channels to each SU for transmission. To avoid the collisions between SUs, N is set to $1 \le N \le K$. Transmission of n^{th} SU is successful when only n^{th} SU transmits in corresponding channel and corresponding channel is in good state at a given time slot. Otherwise, transmission of n^{th} SU will fail. After each time slot (say t), the centralized node receives a binary observation $o_n(t)$ for each SU (say n), indicating whether its packet was successfully delivered or not (i.e., ACK signal). If the packet has been successfully delivered, then $o_n(t) = 1$. Otherwise, if the transmission has failed (i.e., a collision occurred), then $o_n(t) = 0$. Thus, the binary observation $o_n(t)$ can be represented as

$$o_n(t) = \begin{cases} 1 & \text{if the channel selected by } n^{\text{th}} \text{ SU is in good state,} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

In this paper, we model the cognitive radio network as the following MDP model with the goal of transmitting the maximum amount of packets in the same time. The MDP model includes state space, action space, discount factor, reward, and unknown state transition functions.

Given that there are only two possible state for each channel, the state of channels at time slot t can be defined as s_t

$$s_t = [s_{t1}, s_{t2}, \cdots, s_{tk}, \cdots, s_{tK}], \tag{2}$$

where $s_{tk} \in \{0, 1\}$ represents the state of k^{th} channel at the time slot t $(1 \le k \le K)$. For instance, $s_t = [0,0,1,0]$ represent that K is equal to 4 and only the third channel is good at the time slot t. Since the scenario we research is completely observable, the state of all channels is known to the centralized node.

We define action as $a(t)$ at time slot t

$$a(t) = [a_{t1}, a_{t2}, \cdots, a_{tk}, \cdots, a_{tK}], \tag{3}$$

where a_{tK} represents whether k^{th} channel is chosen or not for transmission. As we noted before, the centralized node only chooses N channel for multiple SUs access at each time slot, where $1 \le N \le K$. So, there are only N nonzero elements in the action vector $a(t)$ and the centralized node have the total number of valid actions equal to $C(K, N) = K!/(N!(K - N)!)$ for a specific value of N. It is easy to see that the number of valid actions tends to be much bigger than K in many cases, which can limit the network utilities when we evaluate and make decisions for each valid action. In order to decrease action space size, we consider a situation that the total number of valid actions (i.e., action space) is equal to K and action vector has only one nonzero element. Each valid action means occupying one of K channels. The centralized node will select N actions based on the algorithm proposed in this paper and randomly allocate one of the corresponding N channels to each SU at each time slot. In the multiple SU scenarios, this design can avoid collision effectively. When there are not enough channels in good state, SUs will obtain transmission opportunities with equal probability. Otherwise, if enough good channels occur simultaneously, it is possible for each SU to access a good channel for transmitting data.

Let $r_n(t)$ be a reward that n^{th} SU obtain after each time slot t. The total reward $r(t)$ can be viewed as a function of the achievable number of channels on the wireless channel, i.e., the accumulated binary observation, $\mathrm{r}(t) = \sum_{n=1}^{N} r_n(t) = \sum_{n=1}^{N} o_n(t)$. The objective of the centralized node is to find

an optimal decision policy π^* that maximizes expected discount accumulated rewards R, which can be expressed as

$$\pi^* = \arg\ \max_{\pi} R, \tag{4}$$

$$\text{Subject to } R = E\left[\sum_{t=1}^{T} \gamma^t r(t)\right], \tag{5}$$

where $0 \le \gamma \le 1$ is a discount factor, T is the time-horizon of the game. We often set $\gamma = 1$ or $0 < \gamma < 1$ when T is bounded or unbounded, respectively. However, in order to more easily measure policy over a finite time duration T, we use the following average reward instead of the above rewards

$$R = \frac{1}{T}\sum_{t=1}^{T} r(t) = \frac{1}{T}\sum_{t=1}^{T}\sum_{n=1}^{N} r_n(t). \tag{6}$$

Hence, the problem can be formulated as

$$\pi^* = \arg\max\ \frac{1}{T}\sum_{t=1}^{T}\sum_{n=1}^{N} r_n(t). \tag{7}$$

We mainly focus on overlay DSA models. Meanwhile, we assume that there are multiple PUs and multiple SUs in the wireless networks. The state of the channel is good only when the noise power of the channel is low and the channel is not occupied by primary users. We are interested in developing a model-free centralized learning algorithm to adapt to communication environment and solve dynamic programming problems. In the following section, we will introduce existing algorithms related to our proposed algorithm, i.e., Q-learning and deep reinforcement learning algorithms.

4. Implementation of Q-Learning and Deep Reinforcement Learning

Q-learning is a reinforcement learning algorithm whose goal is to search a strategy to maximize the expected discount accumulated rewards for dynamic programming problems. Meanwhile, Q-learning is also a model-free algorithm that can be able to assess the consequences of each action when the system model is unknown and adapt to environmental changes. Q-learning can evaluate the value function $Q(s, a)$ of each state and action pairs, where the state s is the environment state of the agent and the action a is performed by the agent given the state s. The policy π is derived from the value function, i.e., $\pi(s) = \arg\ \max_a Q(s, a), \forall s$. Updating the value function is a cumulative process, whose equation at the time slot t is given as follows:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha\left[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right], \tag{8}$$

where discount factor γ is set to $0 \le \gamma \le 1$, and learning rate α is also set to $0 \le \alpha \le 1$. After performing action a_t in state s_t that maximizes expected accumulated rewards, when transitioning to next state s_{t+1}, reward r_t is obtained.

While Q-learning performs well in some simple situation, it becomes impractical in the following situations: state and action space are enormous which make the storage complexity intolerable and significantly decrease performance because many states are rarely visited. Deep Q-Network (DQN) can perfectly solve the above problems with the help of the excellent fitting performance of neural network [4]. However, we use the same value function both to select and to evaluate an action base on the state of the agent in the above two algorithms. It is more likely to result in overestimation, which degrades performance. The Double DQN (DDQN) is proposed to mitigate overestimation by decoupling estimation and selection. Specifically, we use two neural networks of the same structure with parameter θ_t and θ_t^- to select an action and update value function, which is different from DQN. The update equation of target value function in DDQN at each time slot is given as follows:

$$y_t^{\text{DDQN}} = r_t + \gamma Q\left(s_{t+1}, \arg\ \max_{a_{t+1}} Q(s_t, a_t, \theta_t), \theta_t^-\right), \tag{9}$$

where θ_t represents the parameter of online network which evaluate the value of the selection, and θ_t^- represents the parameter of target network which is used to update value function. We assign the parameter θ_t^- to the parameter θ_t periodically. This approach has been shown to significantly mitigate overestimations due to estimation errors associated with the DQN and acquired high scores in many games [23].

The traditional neural network does not care about the sequential information of all inputs (and outputs). It makes decisions based on current state information and existing experience, which is trained by randomly selected samples from experience replay. To utilize sequential information, recurrent neural network (RNN) is proposed to store representations of recent input events such that it can be able to process and predict sequential data [24]. RNN has made breakthroughs in speech recognition, language modeling, machine translation, and other timing analysis domain. In this paper, we add Long Short-Term Memory (LSTM, a special kind of RNN, which can achieve long-term dependencies) between the input layer and the following fully-connected hidden layer in order to facilitate the processing of sequence decision problems. All recurrent neural networks can be considered as multiple copies of the same network, each passing output information to a successor as input information. Each copy of the same network in RNN is termed a cell. Unlike the standard RNN, the cell of LSTM is not a simple structure, such as a single tanh layer, but will be shown in the following part of this section.

At each time slot t, we acquire the hidden state h_{t-1} and the cell state c_{t-1} from the edge of a loop. Meanwhile, the input observation is s_t. The cell structure of LSTM consists of three gate layers, namely, the forget gate layer, the input gate layer, and output layer. The forget gate layer determines what information flows into the cell state, whose output is represented as

$$f_t = \text{sigmoid}\left(W_f[h_{t-1}, s_t] + b_f\right). \tag{10}$$

Here, sigmoid is activation function, which is sigmoid$(s) = 1/(1 + e^{-s})$, W_f and b_f are parameters of the forget layer.

The input gate layer decides which values we will update, which consist of a sigmoid layer and output the value i_t. Next, a tanh layer creates an input value z_t at time t:

$$i_t = \text{sigmoid}(W_i[h_{t-1}, \text{s}_t] + b_i), \tag{11}$$

$$z_t = \tanh\,(W_z[h_{t-1}, s_t] + b_z). \tag{12}$$

Here, tanh is also activation function, which is tanh $(s) = (e^s - e^{-s})/(e^s + e^{-s})$, W_i and b_i are parameters of input layer, W_z and b_z are parameter of the tanh layer.

We will update the old cell state c_{t-1} into the new cell state c_t. Specifically, we multiply the old state by f_t, forgetting old things we decide to forget earlier. Then, we add $i_t * z_t$, deciding how much we will learn new things. Thus, the new cell state can be calculated it from the following formula

$$c_t = f_t * c_{t-1} + i_t * z_t. \tag{13}$$

Finally, we introduce the output layer which decides we are going to output. This output layer is a sigmoid layer, whose output o_t is multiplied by the new cell state c_t through tanh layer to get the new hidden state h_t at time t:

$$\text{o}_t = \text{sigmoid}(W_o[h_{t-1}, s_t] + h_o], \tag{14}$$

$$h_t = o_t * \tanh\,(c_t). \tag{15}$$

Here, W_o and b_o are parameters of the output layer. It is worth noting that all parameters in LSTM are updated during learning process.

5. The Proposed Double Deep Recurrent Q-Network (DDRQN) for Dynamic Spectrum Access Algorithm

In multichannel spectrum access problem, the number of overall possible channel state and all possible channel allocation policies grows exponentially with the increase of the number of channels. It leads to the huge computational complexity of optimal channel allocation and transmission probabilities which is mathematically intractable as the network size increase. In this section, we develop a deep multiuser reinforcement learning algorithm based on double deep recurrent Q-network (DDRQN) to solve multichannel allocation problems. DDRQN applies for solving multiuser channel allocation without any prior experience in dynamic multichannel access.

5.1. Architecture of the Proposed Multiuser DDRQN Used in DDRQN Algorithm. In this section, we describe the proposed architecture for the multiuser DSA used in DDRQN algorithm to solve the DSA problem. An illustration of the DDRQN is presented in Figure 1.

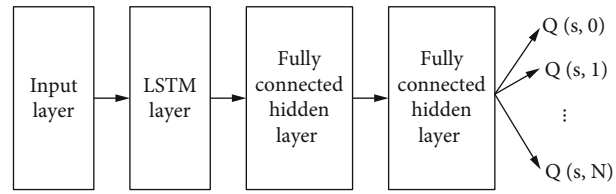

Figure 1: An illustration of the architecture of the proposed multiuser DDRQN algorithm.

(1) *Input Layer*. The input $x(t)$ to the DDRQN is a combination of historical channel conditions over previous two time slots, i.e., $x_n(t) = [s_{(t-1)1}, \cdots, s_{(t-1)K}, s_{t1}, \cdots, s_{tK}]$, where $s_{tk} \in \{0, 1\}$ represent the state of k^{th} channel at the time slot t. This design increases sequence input information (increase the input dimensionality), which make the estimated value function fit actual value through more detail information

(2) *LSTM Layer*. We add an LSTM layer to our approach that can make full use of sequence information over time. This gives the network the ability to estimate the true state transition using the history of the process. This layer is responsible of learning how to aggregate experiences over time

(3) *Fully Connected Hidden Layer*. We add two fully connected hidden layer which can estimate the true value function of state-action pair. This layer can decrease time complexity by avoiding the look-up table approach and make use of good fitting performance of the neural network. When faced with a new state, it is possible to derive a suboptimal policy through prior experience rather than random policy

(4) *Block Output Layer*. The output of the DDRQN is a vector of size K. The k^{th} entry, where $0 \le k \le K - 1$, is the estimated Q-value for transmitting on channel k at the time slot t

(5) *Double Q-Learning*. In order to decrease overestimation, we adopt double Q-leaning to decouple the selection of actions from the evaluation of Q -values. Specifically, we use two neural networks with the same structure (i.e., online network and target network) for action selection and value function estimation, respectively

5.2. Online Learning: Centralized Random Access Using DDRQN. The centralized node has no prior knowledge about the environment characteristic and makes autonomous decisions in online and centralized manners using the proposed neural network, to learn efficient spectrum access policies from its ACK signals. At the beginning of time slot t, the agent will collect the latest channels state over the past two time slot $x(t)$. Then, the centralized node will use ε-greedy method to choose N actions (i.e., N channels) according to the output of the target network, i.e., top N actions with highest scores will be selected, which can be

$$actions = \begin{cases} \text{randomly select N of the K valid actions without repeating,} & \text{rand } () < \varepsilon, \\ \text{choose N actions with highest Q-value,} & \text{rand } () \geq \varepsilon, \end{cases} \tag{16}$$

represented by the following equation:

where $0 \leq \varepsilon \leq 1$ balances between exploration and exploitation. The larger the value of ε, the more the algorithm tends to explore, and vice versa.

The centralized node will randomly allocate one of N channels to each SU. Next, reward of each channel will be sent from its ACK signals. A tuple which contains the current state $x(t)$, the next state $x(t+1)$, one action with highest scores, and the corresponding reward will be stored in replay memory. Finally, a randomly selected sample tuple from replay memory will be used to update the neural network based on mean squared error. The full framework is provided in Algorithm 1 below.

6. Simulation Results

In this section, we first introduce the simulation setting. Then, we test the learning capability of the proposed double deep recurrent Q-network (DDRQN) in the single SU scenarios and provide comparisons with deep Q-network (DQN), Q learning, random policy, and the optimal policy. Finally, we evaluate the performance of the proposed algorithm in the multiple SU scenarios and compare it with other algorithms.

6.1. Simulation Setting. In the proposed DDRQN algorithm, the centralized node is considered as an agent which consists of two neural networks: online network and target network. For each neural network, the first layer is LSTM layer whose size is equal to the state size, and the last two layers are fully connected hidden layers. The second layer has 100 neurons with ReLU as the activation function, and the last layer has K neurons, where K is the total number of channels. The update frequency of target network is set to 100, which means that the parameters of target network are updated by online network every 100 time slots. We set the memory size as 1,000 so that the proposed algorithm has enough samples to optimize the neural network. In each time slot, a minibatch of 32 samples is randomly selected from the memory to train the neural network. We adopt the mean squared error function as the loss function and use Adam algorithm [25] to optimize the parameters of neural network by minimizing loss function. We set learning rate of two neural networks as 0.0001 and discount factor γ as 0.9. In order to avoid falling into a local optimum, we employ ε -greedy method to encourage the agent to explore the environment and greedy value will decrease linearly from 0.1 to 0.01 in each iteration time. We set N to 1 in the single SU scenarios, which means that the centralized node selects only one channel from K channels to the SU in each time slot. We set the number of training episodes as 10 in DDRQN and other comparison algorithms. In the following context, we will introduce parameter setting of other comparison algorithms.

The Deep Q-Learning (DQN) consists of two hidden layers and maintains a memory with a size of 1,000. In each time slot, a minibatch of 32 samples is randomly extracted from the memory to update parameter of the neural network. For a comparison with the DDRQN, the DQN has the same fully connected hidden layer structure, i.e., in our implementation, the DQN has two fully connected hidden layer with the same size.

In the Q learning, the centralized node will evaluate Q -value for each state-action pair and update the value function according to the feedback reward. Relevant parameters in DQN and Q learning, such as discount factor and learning rate, are equal to our proposed algorithm.

In the random policy, there is no learning and the centralized node randomly select one channel to each SU at the beginning of each time slot, and all channels will be accessed with the same probability.

In the optimal policy, we assume that the system dynamic is known to the centralized node, and interrelationship between channels is ignored. At the beginning of each iteration, the centralized node selects one channel to the SUs. Then, according the policy, for instance, when the channel state switching probability is equal to 0.5 or greater, if the previous chosen channel is good, the centralized node will choose a channel in the next activated subset. On the other hand, if the previous chosen channel is bad, the centralized node will choose the previous chosen channel. The centralized node will adopt reverse strategy if the channel switching probability is less than 0.5.

6.2. Average Reward in the Single SU Scenarios. In this section, we consider some single SU scenarios to verify the performance of the proposed learned algorithm. Our framework is compared with Deep Q-Learning, Q-learning, random policy, and optimal decision policy.

In this experiment, we consider a system of 16 channels, and only one channel is in good state in each time slot. To evaluate the performance, we consider the average reward R for different Markov chains P scenarios. To define a Markov chain for channel distribution, we need to specify the channel states in order and the state switching probabilities. We assume that, for each state, the probability that the current state will transfer to the following state is p, and the probability that the current state will be kept is $1 - p$. Our experiment can be divided into the two cases.

(1) *Round-Robin Switching Scenario*. In this experiment, each channel is in turn a good channel from 1 to N according to a round-robin scheduling. We

1) For time-slot $t = 1, \ldots, T$ do
2) Observe an input $x(t)$ and feed it into the online network
3) Generate an estimation of Q-value $Q(a)$ for all available actions $a \in \{0, 1, \cdots, K-1\}$ by the online network
4) Take N actions $a_n(t) \in \{0, 1, \cdots, K-1\}$, $n \in \{0, 1, \cdots, N-1\}$ with ε-greedy method (according to (12)) and obtain instantaneously reward $r_n(t)$ for each SU
5) Observe an input $x(t+1)$
6) Mark $a_h(t)$, $r_h(t)$ as the action and the reward with high scores $Q(a)$
7) Store tuple $x(t)$, $a_h(t)$, $r_h(t)$, $x(t+1)$ in replay memory
8) Sample random minibatch of tuples x_j, a_j, r_j, x_{j+1} from replay memory
9) Set $y_j^{DDQN} = \begin{cases} r_j \text{ for terminal } x_{j+1} \\ r_j + \gamma Q(x_{j+1}, \arg\max_{a_{t+1}} Q(x_j, a_j, \theta_t), \theta_t^-) \text{for non-terminal } x_{j+1} \end{cases}$
10) Perform a gradient descent step on $(y_j^{DDRQN} - Q(x_{j+1}, a_j, \theta))^2$
11) End for

Algorithm 1: DDRQN algorithm.

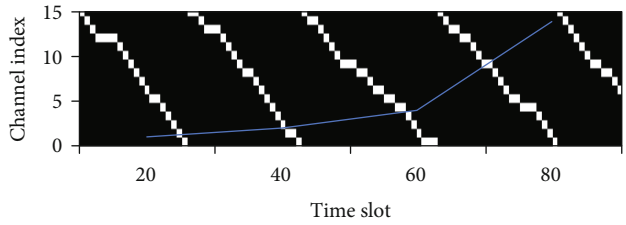

Figure 2: Round-robin switching scenario with one good channel at a given time slot and switching probability is $p = 0.75$. White square represents good channel at a given time slot.

consider five different switching probabilities $p = \{0.75, 0.80, 0.85, 0.90, 0.95\}$. A higher p is to make the state switch more frequently, so that the agent can learn more complex environmental conditions. Round-robin switching scenario with switching probability $p = 0.75$ is depicted in Figure 2, where channel with good state is indicted with a white square at the corresponding channel index value at a given time slot. The average reward and the probability of packet collision comparison between DDRQN algorithm and other algorithms in this scenario are shown in Figure 3

In Figure 3, we observe that the optimal policy achieves the highest average reward and the lowest probability of packet collision, which is close to the corresponding switching probability because the channel pattern is assumed to be known for the optimal policy. In addition, the average rewards of the random policy are kept low, meaning that the average number of good channels per time slot is low and this policy lacks adaptability. More interesting and competitive performances are displayed by the proposed DDRQN policy, DQN policy, and Q Learning. We notice that with the increase of channel switching probability, the average reward of these three algorithms increases gradually, while the probability of packet collision decreases. This is because the change of channel state is more deterministic with the increase of the switching probability. With the excellent fitting performance of neural network, DQN obtains higher average reward than Q learning. Meanwhile, the performance of DDRQN is better than DQN due to the addition of LSTM layer.

(2) *Arbitrary Switching Scenario*. In the round-robin switching scenario, the channel states switch according to a specific switching order. This switching discipline is unknown to the proposed algorithm, and of course is not being used in the process to find more good channel. But to verify the adaptability of the proposed algorithm in general switching mode, we consider many different switching sequences and fix the switching probability p at 0.9. One such switching pattern in the case is depicted in Figure 4. The performance of 10 randomly generated arbitrary switching scenarios is shown in Figure 5. As can be seen from Figure 5, there is little difference in the average reward under different arbitrary switching scenarios, demonstrating that our proposed algorithm has certain robustness

6.3. Frequency Hopping in the Multiple SU Scenario. In this section, we consider frequency hopping communication scenario with m SUs, where m is equal to 2. We assumed that there are 16 channels with 2 good channels in each time slot. The total channels are evenly divided into 8 subsets. At each time slot, only channels in one subset are good channels, and channels in the remaining subsets are all in a bad state, and good channels are switched sequentially at the next time with the probability $p = \{0.75, 0.80, 0.85, 0.90, 0.95\}$. For example, at the time slot t, the channels in one subset are good channels, and the channels in the remaining subsets are all bad. At the time slot $t + 1$, the channels in following subset are good and the remaining channels are in the bad state with the probability p; at the same time, the channels in current subset k are still good channels and the remaining channels are in the bad state with the probability $1 - p$.

The performance of the proposed algorithm and other comparison algorithm in this case is shown in Figure 6. The performance of the optimal policy based on known system dynamics is always pretty good and acquire rewards

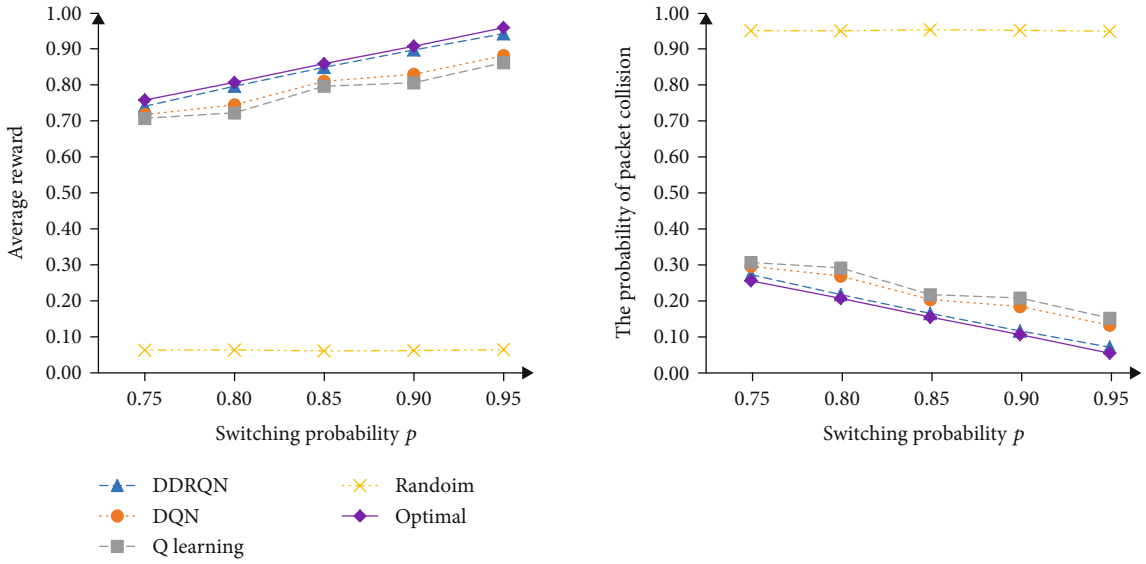

Figure 3: The simulation results in round-robin switching scenario.

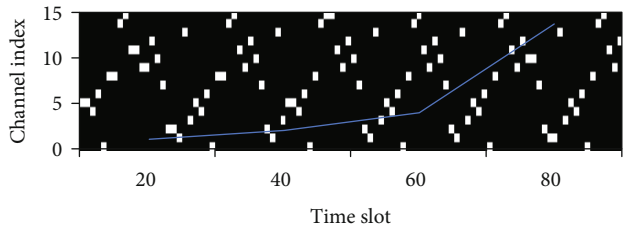

Figure 4: Arbitrary switching scenario with one good channel at a given time slot and switching probability is $p = 0.75$. White square represents good channel at a given time slot.

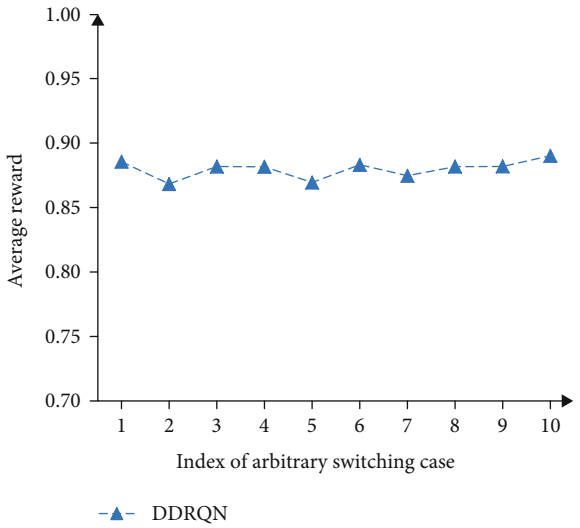

Figure 5: The simulation results in arbitrary switching scenario.

with approximately twice the probability of state transition. The random policy has low average rewards due to lack of learning adaptability about channel dynamics. The DQN and Q Learning acquire similar reward value. However, the performance of DDRQN is better than DQN and Q learning.

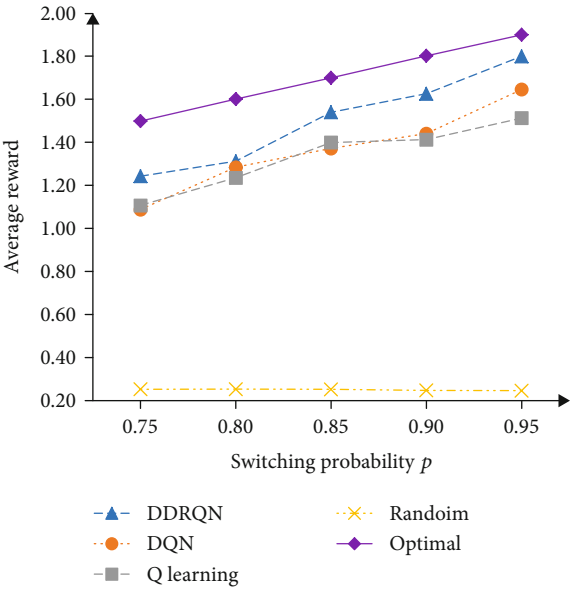

Figure 6: The simulation results in frequency hopping communication scenario.

6.4. Real Data Trace. This section uses the data provided in [12] to carry out simulation experiments under real data trace. The data is collected from the Tutornet platform of the University of Southern California, which consists of TelosB, MicaZ, and OpenMote nodes that meet the IEEE 802.15.4 MAC protocol. There are 8 Wi-Fi access nodes in the tutornet platform, which well simulates the dynamic scenario of multichannel access. This section uses the data to test the proposed DDRQN algorithm and other algorithms with 2 SUs. The simulation results of real data trace will be shown in Figure 7.

As can be seen from Figure 7, although the average reward of random policy remained constant with the number of episodes, it was significantly larger than the two

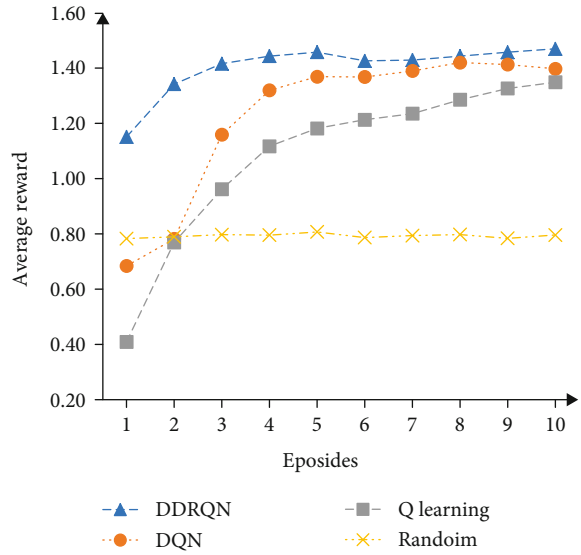

Figure 7: The simulation results in real data trace.

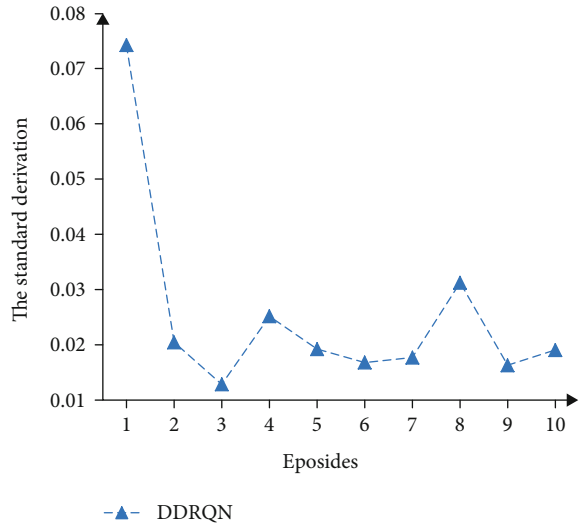

Figure 8: The standard derivation of the average reward in real data trace.

scenarios above. This phenomenon indicates that there are more good channels per time slot in this scenario. As the number of training episodes increases, the performance of all algorithms improves except random policy, which means that these algorithms except random policy can adapt to real data trace. The performance of DQN and Q learning in the first episode is lower than random policy, indicating that the real data trace is more complex than the above scenarios. Meanwhile, for the same number of training episodes, the proposed DDRQN algorithm had better performance than DQN, and the Q learning was the worst.

In this communication scenario, the variation of the standard derivation of the average reward in DDRQN with the number of training episodes is shown in Figure 8. As can be seen from Figure 8, the standard derivation of the proposed algorithm is relatively low. As the number of training episodes increases, the standard deviation of the average reward of the proposed algorithm will gradually decrease, indicating that stability and convergence will become better.

7. Conclusion

In this work, a double deep recurrent reinforcement learning is proposed and implemented to improve the spectrum utilization in the dynamic spectrum access. The proposed algorithm has a LSTM layer to utilize sequential information and two fully connected hidden layer to evaluate value function. We tested the single SU scenarios in the round-robin and arbitrary switching cases and compared the average reward with that of the DQN, the Q learning, random policy, and optimal policy. Then, we test the multiple SU scenarios in the frequency hopping and real data trace. Experimental results show that the proposed algorithm can enable secondary users to obtain more spectrum access opportunities and quickly adapt to the dynamic spectrum access environment. In addition, the proposed algorithm also avoids collision between SUs in the multiple SU scenarios.

Data Availability

The data in real communication scenario of Section 6 is collected from the Tutornet platform of the University of Southern California. More information about the testbed on http://anrg.usc.edu/www/tutornet/. The data can be found in [6] of this paper, i.e., E. Selvi, R. M. Buehrer, A. Martone, and K. Sherbondy, "Reinforcement Learning for Adaptable Bandwidth Tracking Radars," in IEEE Transactions on Aerospace and Electronic Systems, vol. 56, no. 5, pp. 3904-3921, Oct. 2020, doi:10.1109/TAES.2020.2987443.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] B. H. Kirk, R. M. Narayanan, K. A. Gallagher, A. F. Martone, and K. D. Sherbondy, "Avoidance of time-varying radio frequency interference with software-defined cognitive radar," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 3, pp. 1090–1107, 2019.

[2] X. Li, J. Fang, W. Cheng, H. Duan, Z. Chen, and H. Li, "Intelligent power control for spectrum sharing in cognitive radios: a deep reinforcement learning approach," *IEEE Access*, vol. 6, pp. 25463–25473, 2018.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction Vol. 1*, MIT Press, Cambridge, MA, USA, 1998.

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

[5] Q. Cong and W. Lang, "Deep multi-user reinforcement learning for centralized dynamic multichannel access," in *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*, pp. 824–827, Xi'an, China, 2021.

[6] E. Selvi, R. M. Buehrer, A. Martone, and K. Sherbondy, "On the use of Markov decision processes in cognitive radar: an

application to target tracking," in *2018 IEEE Radar Conference (RadarConf18)*, pp. 537–542, Oklahoma City, OK, USA, 2018.

[7] E. Selvi, R. M. Buehrer, A. Martone, and K. Sherbondy, "Reinforcement learning for adaptable bandwidth tracking radars," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 5, pp. 3904–3921, 2020.

[8] Y. Yao and Z. Feng, "Centralized channel and power allocation for cognitive radio networks: a Q-learning solution," in *2010 Future Network & Mobile Summit*, pp. 1–8, 2010, https://ieeexplore.ieee.org/document/5722451.

[9] J. Lunden, S. R. Kulkarni, V. Koivunen, and H. V. Poor, "Multiagent reinforcement learning based spectrum sensing policies for cognitive radio networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 5, pp. 858–868, 2013.

[10] Y. Li, H. Ji, X. Li, and V. C. M. Leung, "Dynamic channel selection with reinforcement learning for cognitive WLAN over fiber," *International Journal of Communication Systems*, vol. 25, no. 8, pp. 1077–1090, 2012.

[11] J. Oksanen, J. Lundén, and V. Koivunen, "Reinforcement learning based sensing policy optimization for energy efficient cognitive radio networks," *Neurocomputing*, vol. 80, pp. 102–110, 2012.

[12] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 2, pp. 257–265, 2018.

[13] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable MDPS," in *2015 AAAI Fall Symposium Series*, pp. 29–37, 2015, https://arxiv.org/abs/1507.06527.

[14] C. Schulze and M. Schulze, "ViZDoom: DRQN with prioritized experience replay, double-Q learning and snapshot ensembling," in *Proceedings of SAI Intelligent Systems Conference*, pp. 1–17, Springer, Cham, Switzerland, Sep. 2018.

[15] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic Spectrum access," *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, pp. 310–323, 2019.

[16] Y. Xu, J. Yu, and R. M. Buehrer, "The application of deep reinforcement learning to distributed spectrum access in dynamic heterogeneous environments with partial observations," *IEEE Transactions on Wireless Communications*, vol. 19, no. 7, pp. 4494–4506, 2020.

[17] A. Wang, L. Zhang, D. Chen, and J. Chen, "Deep reinforcement learning for dynamic multichannel access in multi-cognitive radio networks," *Journal of Physics Conference Series*, vol. 1550, no. 3, article 032135, 2020.

[18] Y. Xu, J. Yu, W. C. Headley, and R. M. Buehrer, "Deep reinforcement learning for dynamic spectrum access in wireless networks," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, pp. 207–212, Los Angeles, CA, USA, 2018.

[19] H. Q. Nguyen, B. T. Nguyen, T. Q. Dong, D. T. Ngo, and T. A. Nguyen, "Deep Q-learning with multiband sensing for dynamic spectrum access," in *2018 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pp. 1–5, Seoul, Korea, 2018.

[20] Q. Zhao, B. Krishnamachari, and K. Liu, "On myopic sensing for multi-channel opportunistic access: structure, optimality, and perfor- mance," *IEEE Transactions on Wireless Communications*, vol. 7, no. 12, pp. 5431–5440, 2008.

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

[22] C. Zhong, Z. Lu, M. C. Gursoy, and S. Velipasalar, "A deep actor-critic reinforcement learning framework for dynamic multichannel access," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1125–1139, 2019.

[23] H. van Hassel, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, pp. 2094–2099, 2016.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 2094–2099, 1997.

[25] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, http://arxiv.org/abs/1412.6980.