Accelerating Spectrum Sharing Algorithms for Cognitive Radio Transmitters in a Momentum Q-Learning Approach

Lianghui Zhu, Zhanke Zhou, Zhaochuan Peng, and Xiaojun Hei(✉)

Huazhong University of Science and Technology, Wuhan 430074, China
{unrealluver,zhankezhou,pengzc,heixj}@hust.edu.cn

Abstract. The radio frequency spectrum is a scarce resource and cognitive radio has been under heavy research to improve the utilization of spectrum in the past thirty years. It is crucial to optimize the performance of cognitive radio for high values for practical applications while it has turned out to be very technically challenging. The conventional cognitive radio methods have strong pertinence and coupling because they are generally designed for a specific application environment. To address the problem of spectrum sharing with collision avoidance mechanisms in cognitive radio, in this paper we propose a new momentum-based Q-learning algorithm to accelerate reinforcement learning based spectrum sharing algorithms for cognitive radio transmitters. We conduct a performance evaluation study based on a simulation toolkit for the reinforcement learning research and the ns-3 network simulator "ns3-gym". As a demonstrating case study, the proposed algorithm is able to capture the learnable patterns from a periodic channel occupation in a wireless environment and avoid channel collision effectively, finally improving channel efficiency and reducing the end-to-end time delay. The simulation results demonstrated that our proposed momentum Q-learning algorithm achieves a lower collision rate, faster convergence as well as stronger generalization capacity compared with two conventional algorithms including a greedy algorithm and a deep Q-learning network algorithm.

Keywords: Reinforcement learning · Cognitive radio · Spectrum sharing

1 Introduction

In the past decades, wireless communication technologies have been rapidly developed and widely applied, and various wireless communication devices compete for spectrum resources. However, spectrum resources have not been fully

L. Zhu, Z. Zhou and Z. Peng—These authors contributed equally to this work.

H. Song and D. Jiang (Eds.): SIMUtools 2020, LNICST 369, pp. 533–547, 2021.
https://doi.org/10.1007/978-3-030-72792-5_42

utilized, and only a small part of them are frequently used [6]. Cognitive radio has been proposed as one of the effective approaches to improve the utilization of spectrum resources, which utilizes the idle spectrum for dynamic network access and data transmission, to achieve transmission in a crowded spectrum space and improve the spectrum utilization [10]. Nevertheless, it is very technically challenging and has high values for practical applications to optimize the performance of cognitive radio. The traditional network access methods (e.g. carrier-sense multiple access (CSMA)) are not able to obtain the wireless environment information and effectively adapt to the changing channels, which are prone to collisions, and finally, increase the end-to-end time delay.

In the cognitive wireless network system proposed by [13], based on the cognitive characteristics, through information processing and artificial intelligence, the perception, decision-making, resource allocation and network reconstruction of the network can be enabled. A close internal logical relationship that exists between the four parts. In this architecture, cognitive wireless networks can sense, perceive and learn the states of network environments, make intelligent decisions, fine-tune the configurations and change the behaviors of nodes and network adaptively, to achieve intelligent optimizations of the network performance.

Reinforcement learning (RL) enables software agents take actions for a changing network environment in order to maximize the cumulative reward. With the rapid development of artificial intelligence algorithms, reinforcement learning has been playing an increasingly important role in computer networking and outperforming traditional control schemes in terms of performance and efficiency. In this paper, we study the optimization issues of Q-learning algorithms for spectrum sharing in cognitive radio networks. Our main contributions are summarized as follows:

(1) We propose a new momentum Q-learning based spectrum sharing algorithm for cognitive radio transmitters to accelerate the convergence and accuracy.
(2) We conduct a simulation study to evaluate the performance of the spectrum sharing algorithms based on ns3-gym [4] which utilizes the OpenAI Gym in the ns3 simulator in various wireless scenarios.

The rest of the paper is organized as follows: we the briefly review the related work in industry and academia in Sect. 2, followed by our algorithm design in Sect. 3. We present the simulation-based performance evaluation results in Sect. 4. Finally, we conclude the paper in Sect. 5.

2 Related Work

In this section, we review the representative work on spectrum sharing and reinforcement learning.

2.1 Spectrum Sharing

Wireless networks have become increasingly crowded [3]. Cognitive radio provides a promising technical approach to relieve the channel deficiency with frequency reuse [2] to improve spectrum utilization. The nonorthogonal multiple access (NOMA) has been proposed to improve transmission efficiency and minimize the overall delay with jointly optimizing computation offloading and communication [15]. An analytical study has shown that that, whether the connection-based access outperforms the packet-based access in spectrum utilization, crucially depends on the sensing capability of nodes [1]. Deep learning algorithms have been studied to improve spectrum efficiency in recent years. Deepika Rajpoot [11] proposed to optimize the frequency search algorithm for secondary users accessing to primary users in the NOMA cognitive radio networks, and finally to obtain the optimal detection time and maximum throughput to improve spectrum efficiency. In the dynamic channel access (DSA), Kassab et al. proposed a multi-agent deep deterministic policy gradient (MADDPG) learning algorithm based on the channel access event [5]. These algorithms are designed to avoid collisions and transmit redundant information, and adequately utilize the time and device-level correlation of monitored events and devices, respectively.

2.2 Reinforcement Learning

Wu et al. proposed a cognitive radio based on a reinforcement learning algorithm to divide channel users into main users and cognitive users [14]. Lo et al. proposed a collaborative awareness method, reinforcement learning cooperative sensing (RLCS), based on reinforcement learning to reduce the collaboration cost in cognitive radio ad-hoc networks and improve the collaboration gain [7].

In recent years, reinforcement learning has attracted extensive research attention. In WiFi networks, the traditional CSMA protocol may cause serious performance degradation due to the inconsistent protocols used by different nodes. Yu et al. proposed an efficient and fair channel sharing protocol (CS-DLMA) based on reinforcement learning [17]. The experiments show that CS-DLMA can effectively improve various performance metrics such as throughput. Especially in WiFi scenarios, CS-DLMA achieves higher Pareto than CSMA. The standardization of telecommunication protocols may take a long time. With more devices entering the network, a unified and efficient protocol framework becomes a major trend in the future. Valcarce and Hoydis introduced reinforcement learning into the formulation and standardization of communication protocols [12], by using the signals transmitted between devices as an incentive channel, so that the model may effectively deploy the best channel access strategy in cellular networks, and saves costs of the manual formulation and the protocol standardization. In addition, the reinforcement learning has shown portability in more unknown scenarios [12,17].

In addition to the theoretical feasibility, simulation frameworks have been proposed to foster reinforcement learning algorithms in ns-3 [4,16]. A large number of research studies have aimed at the fairness of spectrum sharing. In this

paper, we study the optimization issues of reinforcement learning algorithms for spectrum sharing based on an ns-3 based simulation study.

3 Algorithm Design

In this section, we first illustrate the spectrum sharing problem between primary users and secondary users in cognitive radio networks. Then, we present a greedy algorithm and a deep Q-learning network algorithm as benchmark algorithms for addressing the spectrum sharing problem. To accelerate reinforcement learning based spectrum sharing algorithms for cognitive radio transmitters, we propose a new momentum-based Q-learning algorithm. We present the algorithm implementations and the ns3-gym simulation framework. Based on the ns3-gym framework, we instrument the network environment of the reinforcement learning agents and the ns-3 simulator.

3.1 Problem Statement

For the sake of simplicity, we assume that a cognitive radio network consists of 4 primary users (PU), 1 secondary user (SU), and 1 base station. PUs and SUs are able to share the same spectrum and communicate with the base station. Nevertheless, cognitive radio ensures that PUs are allocated to the available channels with preemptive priorities over SUs in that if there are idle channels in the wireless link, SUs can still utilize the spectrum for their data transmission. We consider special cases in that these 4 PUs periodically 4 channels, and the SU aims to utilize idle channel resources for transmission. As illustrated in Fig. 1, an intelligent agent is deployed to sense the entire channel utilization and then select the idle channel time slots for the SU to transmit its data.

We propose how to sense the idle channel time slots and design RL algorithms to learn the periodic patterns by PUs and evaluate the proposed algorithms based on ns-3 simulation experiments, so as to reduce the probability of collisions, improve the channel utilization, and finally achieve a better spectrum sharing strategy in cognitive radio. To compare the performance of RL algorithms, we deploy a greedy algorithm and a deep Q-learning network algorithm as benchmark.

3.2 Greedy Algorithm

In the greedy algorithm, a centralized channel intelligent assembles all information from the network. The occupied channel at the $(N-1)$th time step is defined as the state of Nth time step, and the transmitter takes the action in selecting an idle channel for its data transmission at the Nth time step. In our experiment settings, the greedy algorithm transmitter will be given a specific state-action table for that scene, in which we preset values for the transmitter to choose the best channel. Then, based on the principle of maximizing the value, the transmitter selects the best channel from the highest value to the lowest value

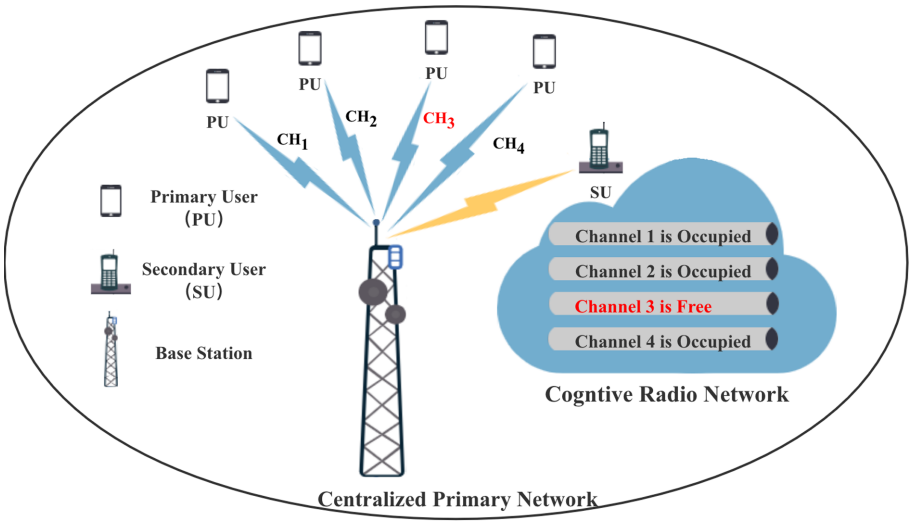

Fig. 1. Spectrum sharing between primary users and secondary users in cognitive radio networks

following the state-action corresponding table, to determine the final channel selection. The channel selection decision is determined based on the prepared state-action corresponding table. The advantages of the greedy algorithm are fast speed and low computational complexity, but the channel selection may not be optimal. The disadvantage of this greedy algorithm is that it relies heavily on the state-action table tailored for specific network environments.

3.3 Deep Q-Learning Network Algorithm

Q-learning is a representative reinforcement learning method which is to construct a control strategy to optimize the algorithm performance. For a cognitive radio transmitter, the Q-learning agent perceives and processes collision information from the network environment. The Q-learning transmitter agent can tune its parameters and change its behaviors by learning, to produce the transmit choice of cognitive radio. The transmit choice makes the agent choose a channel, and then impact the environment.

We intuitively apply a deep Q-learning network (DQN) algorithm for a cognitive radio transmitter [4] with the following parameter mapping from reinforcement learning to cognitive radio application scenarios. For a fair performance comparison, we assign the same set of parameters of our proposed momentum Q-learning algorithm as presented in Sect. 3.4.

The *Observation* represents the occupied channel corresponding to the previous time step which reflects the holistic state of the network, while the *Action* denotes selecting the channel for the next time step to transmit data. The *Reward* is defined as a positive reward that is given if there is no collision; otherwise, a negative reward.

3.4 Momentum Q-Learning Algorithm

The momentum Q-learning is a model-free reinforcement learning algorithm, which aims to learn a policy that tells a CR transmitter what action to take under certain circumstances. Note that the convergence of Q-learning is directly related to the selection of the reward. Improper selection of rewards may lead to the failure or slow convergence of the model. For the cognitive radio transmitter, we design a momentum Q-learning algorithm that meets the application requirements with enhanced performance as shown in Algorithm 1. This momentum Q-learning algorithm adopts the dynamic reward based on the adjacent running states. We assign 2 different coefficients β_s/β_f for no collision cases. For collision cases, we set the minimum value t_{smin}/t_{fmin} of successful/failing reward tokens t_s/t_f, which provide a smooth reward but also improve the convergence speed of the whole model. In our *momentum Q-learning* algorithm, a dynamic reward is employed to accelerate the convergence process of the algorithm. In addition, we introduce a *Gameover* criteria as follows: if the agent makes three mistakes in the last 10 decisions, the game ends. If the time step reaches the upper limit of simulation verification 100 times, the game also ends.

The algorithm details are stated as follows: Q-learning needs to rely on the Q-table to judge when selecting channels. If the number of channels is M, then Q-table is a $M \times M$ matrix. The corresponding values in the Q-table characterize the probability of selecting the corresponding action based on a specific observation. For example, $Q(2,3)$ is the probability that Channel 2 is occupied at the $(n-1)$th time step, and the transmitter selects Channel 3 at the nth time step. First, the Q-table is initialized to 0. In each episode, the network environment states are sensed, and the successful token and failed token are set to 0. At the beginning of each step, the action with the largest value in the Q-table is selected as the channel selected by the transmitter according to the greedy strategy, and the corresponding channel is selected and the feedback from the network environment is received. Then, the successful token and failed token are updated based on this feedback. If there is no collision, the successful token t_s is multiplied by the coefficient $\beta_s \in (t_{smin}, t_{smax})$; the failed token t_f is multiplied by the coefficient $\beta_f \in (t_{fmin}, t_{fmax})$. If the collision occurs, the successful token is reset to $t_s = t_{smin}$ and the failed token is reset to $t_f = t_{fmin}$. Then, the sum of the above two tokens is regarded as the reward r, and the state s' is predicted. Then, the Q-table is updated with $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$, and s is updated to s'. If the step reaches the default value of 100 for the maximum number of steps in the experiments or there are at least 3 collisions in the last 10 time steps, the episode is regarded as the end. We configure the experiment duration to last for 200 episodes if not explicitly specified.

3.5 Simulation Framework

We deploy an ns-3 layer to simulate the interaction between the RL agents and the network environment. The *timestep* represents the incremental change in time. We study two cases of the periodic channel usage patterns by the primary

Algorithm 1. The momentum Q-learning algorithm for the cognitive radio transmitter.

Input: State(s): The occupied channel in the $(N-1)$th time step.
Input: Channel collision status: if the channel which the transmitter selects at the $(N-1)$th time step occurs collision.
Output: Action(a): The channel the transmitter selects in the Nth time step.

Initialize $Q(s,a)$ by zero.
repeat(for each episode):
  Get initial state(s) from the network environment.
  Initialize successful token $t_s \leftarrow 0$ and failed token $t_f \leftarrow 0$.
  repeat(for each step of episode):
    Choose action(a) from state using ϵ-greedy policy from Q.
    Take actions, and pass it to the network environment:
    if no collision **then**
      $t_s \leftarrow min(t_s * \beta_s, t_{smax})$.
      $t_f \leftarrow min(t_f * \beta_f, t_{fmax})$.
    else
      $t_s \leftarrow t_{smin}$.
      $t_f \leftarrow t_{fmin}$.
    end if
    Get reward $r \leftarrow (t_s + t_f)$, and predict state s'.
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$.
    $s \leftarrow s'$.
  until The time steps equal to 100 or 3 collisions occur in the latest 10 time steps.
until through each episode

users in the cognitive radio network as shown in Fig. 2. With the progress of time steps, 4 PUs occupied the channels periodically in order from Channel 1 to 4. The brown blocks represent the occupied channel time slots by the primary users at this time point and the white blocks indicate the idle time slots.

Figure 3 is depicted to illustrate a sample path of the channel usage of the CR transmitter. After the reinforcement learning agent makes a decision in selecting a channel, the SU uses this channel for data transmission in the current time step. We use the blue block to mark the selected idle channel time slot by the RL agent. If the RL agent selects occupied channel time slots by the primary users, those time slots are marked as red in Fig. 3(a), which is plotted as a demonstrating case for the periodic channel usage pattern by the primary users in case 1. In this example, among the total 60 time slots the SU utilizes, 14 time slots experience collisions for case 1. Given the same sample path of the 60 time slots selected by the SU, 17 time slots experience collisions for case 2 as shown in Fig. 3(b). The RL agent aims to minimize the probability of red blocks by learning the periodic pattern by PUs and selecting those idle time slots in this example.

As shown in Fig. 4, the ns3-gym simulation framework consists of three parts: the ns-3 network model, OpenAI Gym, and the reinforcement learning agent. The ns-3 network model provides us a basic network simulation environment that builds the cognitive radio network and statistic functions. The middle layer,

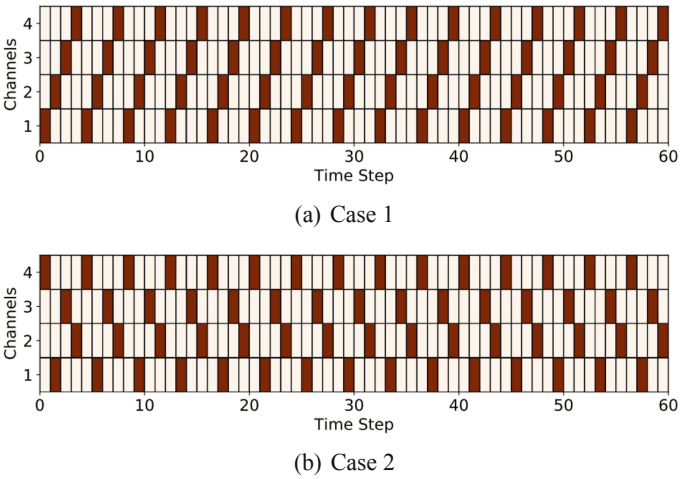

Fig. 2. Periodic channel usage patterns by the primary users in the cognitive radio network

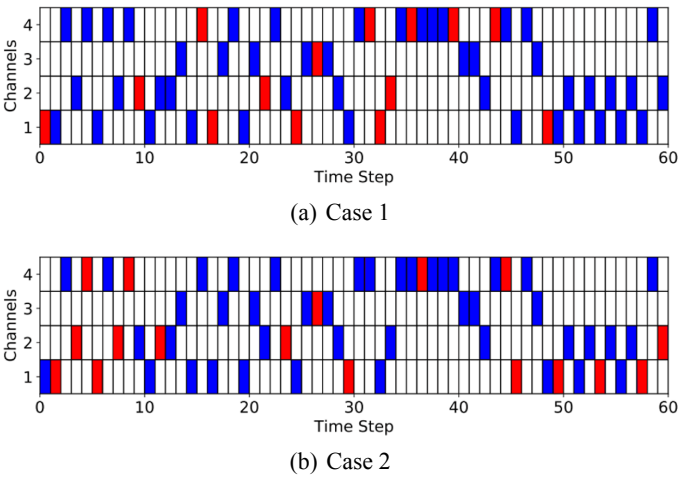

Fig. 3. A sample path of the channel usage of the CR transmitter with collisions

OpenAI Gym, plays an important role in the middle-ware to integrate the reinforcement learning agent and the simulated network environment. This layer passes the actions that the RL agent launches to the network environment and return the reactions of the network to the reinforcement learning layer. In this way, we are capable to deploy the reinforcement learning agent at the top layer to implement decision-making on the low layer. Exactly in this layer, we deploy the deep Q-learning network algorithm and the proposed momentum-based Q-learning algorithm.

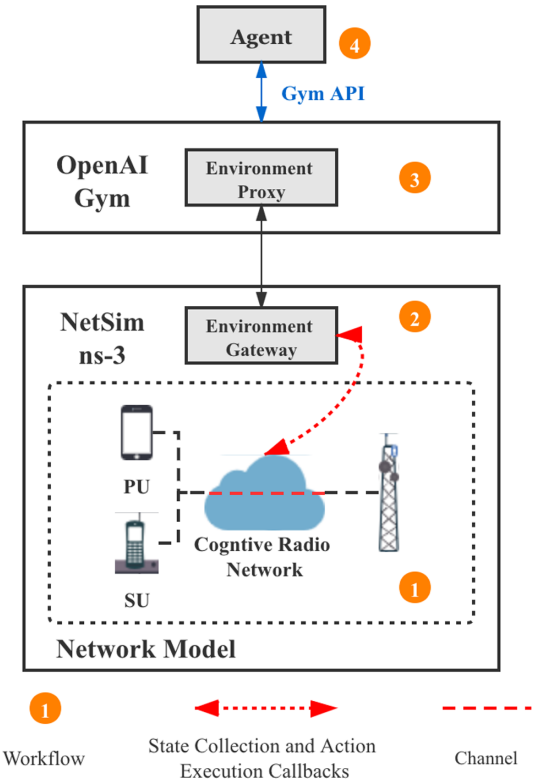

Fig. 4. The ns3-gym simulation framework for spectrum sharing by the cognitive radio transmitter

4 Performance Evaluation

We conduct a simulation study of the proposed momentum Q-learning based channel sharing algorithm for the cognitive radio transmitters. In this section, we present the simulation results to evaluate the performance of the proposed momentum Q-learning algorithm against a greedy algorithm and a deep Q-learning network algorithm.

4.1 Simulation Setup

With the manual tuning of the hyper parameters of the proposed MQL algorithm, we recommend the following parameters utilizing ns3-gym in our simulation. We control the wireless network environment following the suggested settings in ns3-gym. The stochastic gradient descent (SGD) is used as the optimizer of the deep reinforcement learning algorithm. We find that the deep Q-learning rate may lead to reliable performance while it is set as 0.001. For the proposed momentum Q-learning algorithm, we set the exploration rate to 0.3 to keep the

ability to seek appropriate parameters in the Q-learning table. To improve the performance, we set the successful coefficient $\beta_s = 1.1$, the maximum of the successful token is $t_{smax} = 2$ and the minimum value is $t_{smin} = 0$. The failing coefficient is $\beta_f = 0.9$ and the maximum value of the failing token is $t_f = 0$ and the minimum value is $t_f = -2.5$. Each hyper parameter of the momentum Q-learning algorithm requires skills for fine-tuning. In order to maximize the performance of the algorithms, we fine-tuned only one hyper parameter via debugging each time. At last, the *Disturbance Bias* is introduced in our experiments to simulate the potential errors of sensing the collisions in that the network channel will be assigned with a certain deviation probability at other channels. This probability is set as 5% if not explicitly specified.

4.2 Performance Metrics

In our simulation study, we focus on three aspects of the algorithms: generalization capacity, convergence speed and time complexity. The convergence of the algorithms are the central performance metric. The default x-axis in the result figures is the episode by default, and the y-axis quantifies the time step, in which the transmitter agents make a channel selection decision. For each algorithm, we continuously simulate 200 episodes, and the maximum time step of each episode is 100 steps. The advantages and disadvantages of the algorithm are characterized by the time step of the episode. The closer the time step approaches 100, the better the algorithm performs in this episode. For the reinforcement learning algorithm, there will be an extra backward curve. The reward curve represents the cumulative reward of the reinforcement learning algorithm in each episode. It captures the learning performance that the reinforcement learning algorithm achieves for multiple episodes.

4.3 Simulation Results

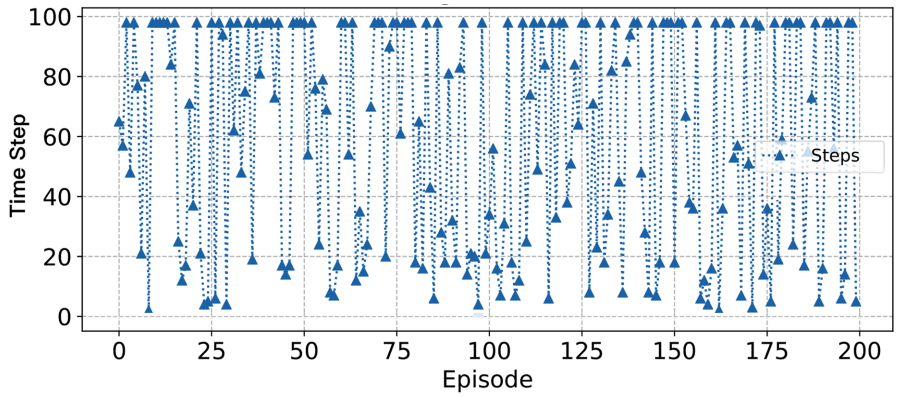

Fig. 5. The performance of the greedy algorithm with disturbance bias in case 1

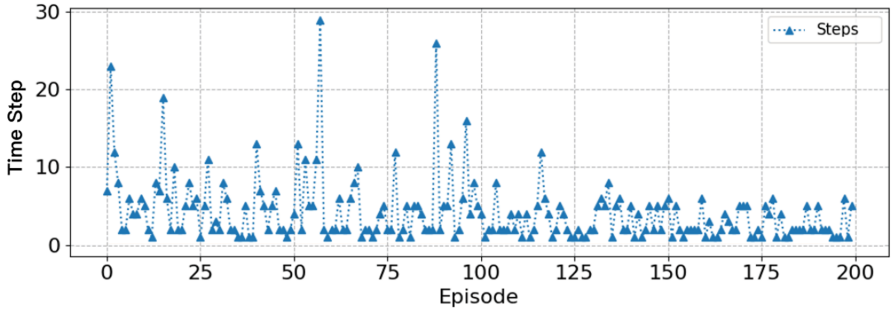

Fig. 6. The performance of the greedy algorithm with disturbance bias in case 2

Algorithm Generalization. As shown in Fig. 5, the greedy method has lots of episodes with over 100 steps, indicating that the greedy algorithm made the transmitter could achieve the whole transmit process well. We use different cases to perform different application environments. We assume the situation showed in Fig. 2 is case 1, and case 2 adopt a totally different periodic channel usage pattern by primary users. Given a state-action corresponding table designed for the case 1 in advance, the transmitter will make a corresponding greedy selection through the value given in the table. It has strong pertinence because it is generally designed for a specific application environment which brings strong coupling to its system. Although it is easy for the greedy algorithm to achieve considerable results for a specific networking environment, such as case 1, it will end up with a poor effect (see Fig. 6) when the environment fluctuates or changes to case 2 as shown in Fig. 2(b). However, for both case 1 and case 2, reinforcement learning is able to construct the most appropriate state-action table from the current environment because it does not need to set the state-action table in advance. We select the most representative case 1 environment to conduct more experiments to evaluate DQN and MQL. The representative experimental results are depicted in Fig. 7 and Fig. 8. From the time step curves, we observe that the adaptive trend of the reinforcement learning algorithm for the changing network environment gradually improves.

Convergence Speed. Following the experiments in ns3-gym [4], we adopt a deep Q-learning network algorithm with a one-hidden-layer neural network for training. Obtaining the training effect as shown in Fig. 7. In the meantime, we design a momentum Q-learning algorithm as presented in Sect. 3.4. The adaptive reward helps us learn the features more quickly in Fig. 8. From both time and reward curves, we observe that the DQN algorithm converges at about 70-th episodes but the momentum Q-learning algorithm convergence much faster in the 20-th episode and achieves better performance. After convergence, the steep curve of sustains at 100 steps, and the reward curve also maintains at a high level. The performance of the momentum Q-learning algorithm is stable, and finally, we can harvest the highest reward.

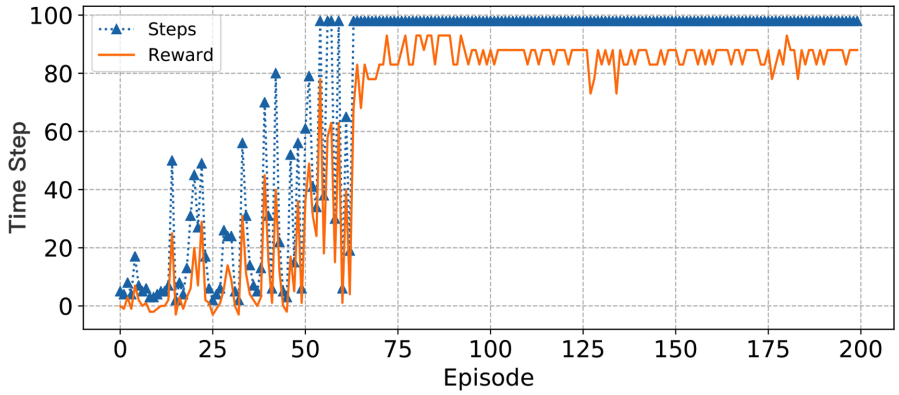

Fig. 7. The performance of the deep Q-learning network algorithm with disturbance bias

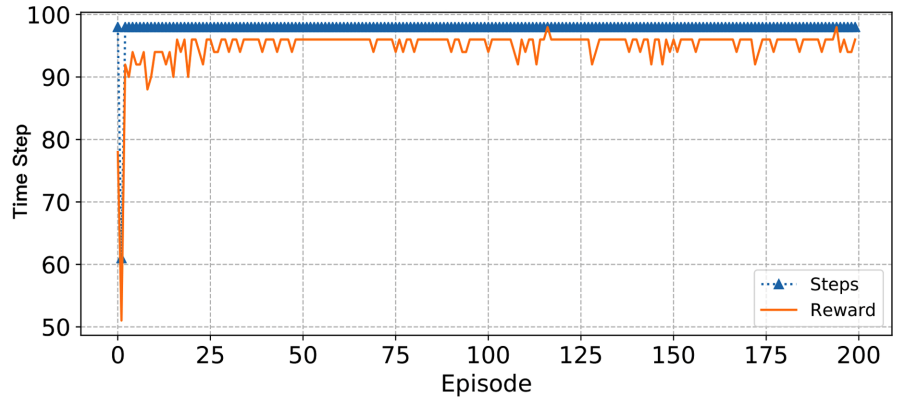

Fig. 8. The performance of the momentum Q-learning algorithm with disturbance bias

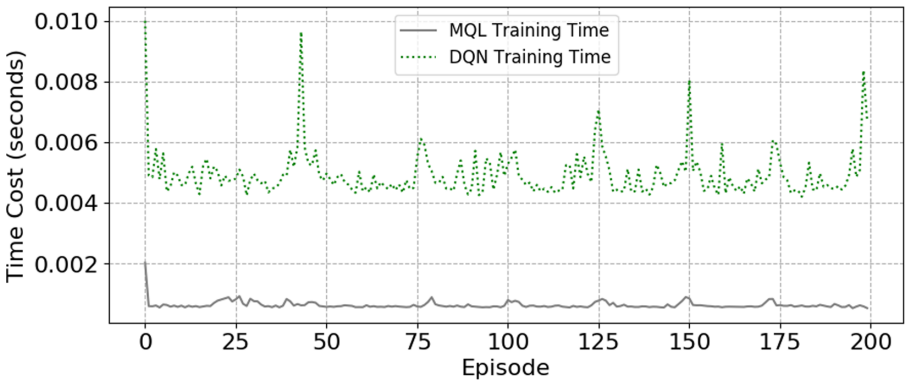

Fig. 9. Training time: momentum Q-learning v.s. DQN

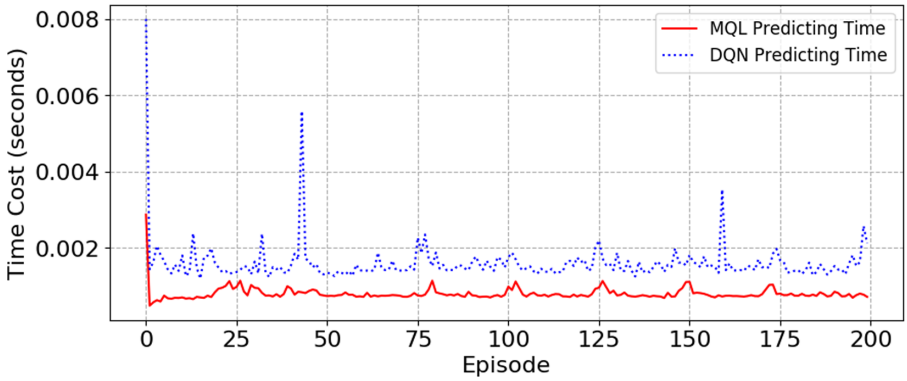

Fig. 10. Predicting time: momentum Q-learning v.s. DQN

Time Complexity. As shown in Fig. 9 and Fig. 10, we apply "MQL Training/Predicting Time" to represent the training average time-cost of training step of each episode with the momentum Q-learning algorithm, and "DQN Training/Predicting Time" for the deep Q-learning network algorithm. The x-axis depicts each episode of the algorithm episode, and the y-axis indicates the time consumed in seconds. The momentum Q-learning average predicting time is only half of deep Q-learning, the average training time of each step is only 1/5 of that of DQN and the curves' fluctuation range of MQL will be smaller, more stable, and there will be no significant fluctuation changes. The training time and the predicting time of the momentum Q-learning is far less than that of deep Q-learning, and more stable too. In summary, in the actual application scenarios, we recommend the momentum Q-learning algorithm for the cognitive radio transmitter to achieve lower judgment delay and training time cost in this way.

Discussions. The reason for the difference in convergence speed and time is mainly due to the backward update mode and the parameter number of DQN and MQL. For DQN, the number of parameters includes the parameters in the neural network, which makes the total number of parameters required is the number of network parameters. If we want to improve the effectiveness of the model and increase the depth of the network, these parameters will increase. MQL only needs the number of M^2 Q-table parameters. Assume we have M channels in the environment, plus a constant number of momentum reward control parameters. The number of parameters is fewer, and the reward with momentum ensures that MQL requires less training time and achieves faster convergence speed.

5 Conclusion

In this paper, we presented a momentum-based Q-learning based spectrum sharing algorithm for cognitive radio transmitters to ensure effectiveness and applicability. We conducted a simulation based performance evaluation study based on

a popular open-source simulator, ns3-gym. The simulation results show that the proposed momentum-based Q-learning algorithm achieves lower collision rate, faster convergence as well as stronger generalization capacity compared with the deep Q-learning network algorithm. The proposed algorithm can also be applied in numerous similar network scenarios. We plan to study an automatic parameter adjustment scheme and deploy the proposed algorithm for the practical application of a real network environment on a software defined wireless network testbed [8,9].

Acknowledgment. The authors would like to express their gratitude to the anonymous reviewers for their constructive comments which help us to improve the quality of this paper very much. This work was supported in part by the National Natural Science Foundation of China (no. 61972172) and the teaching research fund by the Huazhong University of Science and Technology (no. 2018077).

References

1. Gao, Y., Dai, L.: Random access: packet-based or connection-based? IEEE Trans. Wirel. Commun. **18**(5), 2664–2678 (2019)
2. Gao, Y., Roy, S.: Achieving proportional fairness for LTE-LAA and Wi-Fi coexistence in unlicensed spectrum. IEEE Trans. Wirel. Commun **19**(5), 3390–3404 (2020)
3. Gao, Y., Dai, L., Hei, X.: Throughput optimization of multi-BSS IEEE 802.11 networks with universal frequency reuse. IEEE Trans. Commun. **65**(8), 3399–3414 (2017)
4. Gawłowicz, P., Zubow, A.: Ns-3 meets OpenAI Gym: the playground for machine learning in networking research. In: Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 113–120 (2019)
5. Kassab, R., Destounis, A., Tsilimantos, D., Debbah, M.: Multi-agent deep stochastic policy gradient for event based dynamic spectrum access (2020). arXiv
6. Leibovitz, J.S.: The great spectrum debate: a commentary on the FCC spectrum policy task force's report on spectrum rights and responsibilities. Yale J. Law Technol. **6**, 390–414 (2003)
7. Lo, B., Akyildiz, I.: Reinforcement learning for cooperative sensing gain in cognitive radio ad hoc networks. Wirel. Netw. **19**, 1237–1250 (2012)
8. Manzoor, S., Chen, Z., Gao, Y., Hei, X., Cheng, W.: Towards QoS-aware load balancing for high density software defined Wi-Fi networks. IEEE Access **8**, 117623–117638 (2020)
9. Manzoor, S., Yin, Y., Gao, Y., Hei, X., Cheng, W.: A systematic study of IEEE 802.11 DCF network optimization from theory to testbed. IEEE Access **8**, 154114–154132 (2020)
10. Mitola, J., Maguire, G.Q.: Cognitive radio: making software radios more personal. IEEE Pers. Commun. **6**(4), 13–18 (1999)
11. Rajpoot, D.: Sensing-throughput analysis in NOMA-based CR network (2020). arXiv
12. Valcarce, A., Hoydis, J.: Towards joint learning of optimal signaling and wireless channel access (2020) arXiv

13. Wei, J.B., Wang, S., Zhao, H.T.: Cognitive wireless networks: key techniques and sate of the art. Tongxin Xuebao/J. Commun **32**, 147–158 (2011)
14. Wu, C., Chowdhury, K., Di Felice, M., Meleis, W.: Spectrum management of cognitive radio using multi-agent reinforcement learning. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Industry Track, pp. 1705–1712. International Foundation for Autonomous Agents and Multiagent Systems, Richland (2010)
15. Wu, Y., Qian, L.P., Ni, K., Zhang, C., Shen, X.: Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading. IEEE J. Sel. Topics Signal Process. **13**(3), 392–407 (2019)
16. Yin, H., et al.: NS3-AI: fostering artificial intelligence algorithms for networking research. In: Proceedings of the 2020 Workshop on Ns-3, pp. 57–64 (2020)
17. Yu, Y., Liew, S.C., Wang, T.: Non-uniform time-step deep Q-network for carrier-sense multiple access in heterogeneous wireless networks. IEEE Trans. Mob. Comput. (2020)