A Deep Double-Q Learning-based Scheme for Anti-Jamming Communications

Phan Khanh Ha Nguyen, Viet Hung Nguyen and Van Long Do
Viettel High Technology Industries Corporation, Viettel Group
Hoa Lac High-Tech Park, Hanoi, Vietnam
{hanpk1, hungnv152, longdv9}@viettel.com.vn

***Abstract*—Cognitive radio has become an emerging advanced wireless communication technology to achieve maximal spectrum efficiency. In cognitive radio networks, the threat of radio jamming attack arises as a big issue due to the vulnerability of radio transmission. Therefore, anti-jamming is an active research topic for a long time. Recently, with the success of deep learning, deep reinforcement learning algorithms have been applied to solve the dynamic spectrum access and anti-jamming problem. In this paper, we propose a Deep Double-Q learning-based method to learn an efficient communication policy including channel access and transmission power for tackling different jamming scenarios. The proposed scheme uses observed spectral information as input and Q-function is approximated by a neural network. Simulation results show that Double-Q learning algorithm with Convolutional Neural Network achieves effective communication strategies to avoid various jamming patterns compared with other traditional methods.**

***Index Terms*—Cognitive radio, anti-jamming, spectrum sensing, reinforcement learning, deep double-Q learning**

I. Introduction

In recent years, Cognitive Radio (CR) [1] has arisen as a potential solution to solve the spectrum shortage problem. CR allows users to adaptively access channels and thus improves the spectral utilization efficiency. However, along with reconfigurability and cognitive characteristics, CR networks also face with new security issues. Due to the shared and broadcasting nature of radio propagation, radio jamming attack is one of the most serious threats which can deteriorate significantly the performance of CR networks. Therefore, extensive research has been conducted on anti-jamming, especially with the appearance of smart jammers recently [2].

Traditional anti-jamming methods are spread spectrum based-techniques such as Frequency Hopping Spread Spectrum (FHSS), Direct-Sequence Spread Spectrum (DSSS) and Hybrid FHSS/DSSS [2], [3]. These methods require wideband spectrum and thus, are spectral inefficient. Other drawbacks include high-energy cost and high-complexity end devices. Game theory–a mathematical tool for modelling and analysing the interaction between jammer and user has recently received attention from the research community to solve the anti-jamming problem. The authors in [4] studied a defence mechanism in CR networks from a stochastic zero-sum game perspective and proposed a minimax-Q learning to find the optimal policy. Jia *et al.* [5] formulated the anti-jamming communication issue as a Stackelberg game and designed a hierarchical learning algorithm to obtain desirable solutions. Reinforcement learning such as Q-learning can also be used to find optimal anti-jamming strategies. In [6], a modified Q-learning algorithm based on Boltzmann model was proposed to find the optimal policy. Gwon *et al.* [7] considered anti-jamming problem as a stochastic game and evaluated the performance of three Q-learning techniques: Minimax-Q, Nash-Q and Friend-or-Foe Q. The authors in [8] addressed the jamming attack based on Q-learning and WoLF-Q algorithms and achieved the optimal transmission power and channel. However, traditional Q-learning algorithms is inefficient when the number of states and actions is very large.

Recently, deep reinforcement learning [11], [12] has emerged as a powerful framework to solve decision-making problems in complex environments where the number of states is enormous. Motivated by the success of Deep Double-Q Network (DDQN) [12] in learning optimal policies in video games; in this paper, first, we formulate the anti-jamming communication problem with various jamming types as a Markov Decision Process (MDP) and give a method for acquiring states based on an observed shared spectrum. Then, we design two neural network architectures and a Double-Q learning algorithm to approximate the Q-function that defines a policy for anti-jamming against various jamming scenarios.

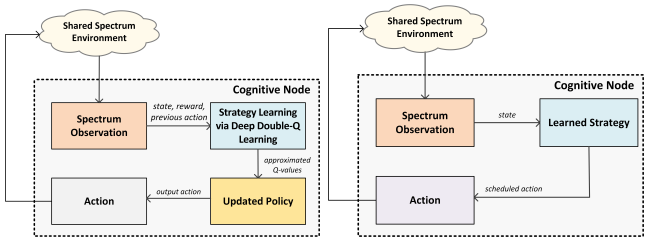
The rest of the paper is organized as follows: In Section II, we describe necessary assumptions, system model and define the anti-jamming communication problem. Our proposed scheme is shown in Section III. We provide simulation results in Section IV. Finally, Section V draws some conclusions.

II. Problem Formulation

A. System Model

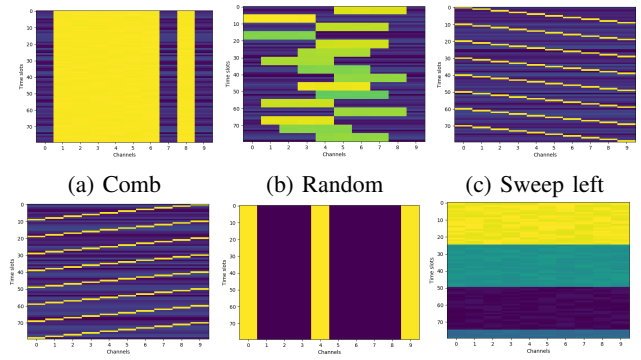
We consider the communication model of one user including a transmitter–receiver pair against the attack of one jammer operating in a shared spectrum. User has a cognitive node attached to the receiver that will learn the anti-jamming strategy in the learning phase and then in the inference phase, the cognitive node will send the scheduled action derived from learned strategy via another secured, reliable control link to the transmitter for its communication as shown in Fig. 1.

Spectrum is partitioned into spectrum blocks in both time and frequency. The whole spectrum is divided into N equal non-overlapping channels. Each channel is located at the center frequency f_i with bandwidth B_i. The occupation status of each channel i at any time, $c_{t,i}$, is either vacant ($c_{t,i} = 0$)



(a) In learning phase (b) In inference phase

Fig. 1: Processing of cognitive node.



(a) Comb (b) Random (c) Sweep left

(d) Sweep right (e) Adaptive (f) Full-band

Fig. 2: Illustration of different jamming patterns.

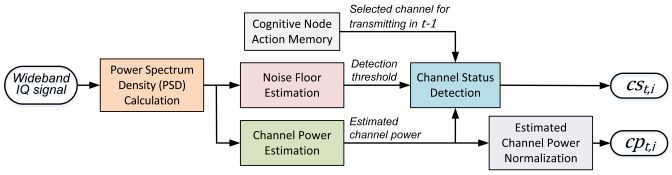


Fig. 3: Spectrum observation of cognitive node.

or occupied ($c_{t,i} = 1$). Transmissions of user and jammer are based on equal time slots. At the beginning of each time slot t, both cognitive node and jammer can sense full frequency spectrum over a duration Δt ($\Delta t \ll t$), decide which channel to be used and transmit on that channel over the remaining time of that time slot.

Denote the transmission power of transmitter and jammer as p_u and p_j, respectively; in this work, we assume p_u and p_j can be changed in each time slot but at one time slot, transmitter can communicate in only one channel while jammer can jam concurrently in several channels. For simplicity, the effects of AWGN and other background interferences are ignored and user's communication is only disrupted by the transmission of jammer with an enough power level. Therefore, a successful communication of user occurs if and only if $(p_u - p_j) \geq \beta_{th}$, where β_{th} (dB) is a specified power threshold.

To enhance the impact of attacks, jammer may utilize different jamming strategies such as comb, random, sweep (left & right), adaptive, full-band as described in Fig. 2 or randomly combine all those separate patterns. Note that, for full-band pattern, jammer can jam in all N channels with a fix power level at each time slot. The adaptive strategy is the most intelligent jamming method since the jammer will observe most recent used channels of user and only jam on those channels.

B. Markov Decision Process

We formulate the anti-jamming communication problem through a Markov Decision Process (MDP) [10]. The components of MDP is specified as follows.

1) *State space*:
At each time slot t, an observed Spectrum Status Vector (SSV) ssv_t is defined as:

$$ssv_t = [cs_{t,0},\ cp_{t,0},\ ...,\ cs_{t,N-1},\ cp_{t,N-1}] \quad (1)$$

where $cs_{t,i}$ is the status of channel i observed by cognitive node at t (if channel i is not selected by cognitive node for transmitting at $t-1$ then $cs_{t,i} = 0$ if $c_{t,i} = 0$ and $cs_{t,i} = 1$ if $c_{t,i} = 1$; otherwise, if channel i is selected by cognitive node at $t-1$ then $cs_i = 2$); $cp_{t,i}$ is the normalized estimated power of channel i at t. We propose a spectrum observation method of cognitive node for estimating SSV as illustrated in Fig. 3. Based on received wideband IQ signal of the whole spectrum at time slot t, first, Power Spectrum Density (PSD) is calculated. Then, the thresholding of PSD, or energy detection, is carried out via an image-processing based-noise floor estimation algorithm proposed in [9]. Next, power of each channel is estimated and normalized in the range $[-1; 1]$ by accumulating PSD values in each channel. Finally, status of each channel is decided using three factors: estimated channel power, detection threshold and whether this channel is selected by the cognitive node for transmitting in the previous time slot $t-1$ or not.
Derived from SSV, a state s_t is defined as a stack of SSVs in consecutive N_{ts} time slots:

$$s_t = [ssv_{t-(N_{ts}-1)},\ ...,\ ssv_{t-1},\ ssv_t] \quad (2)$$

State space is a set of all possible states: $\mathcal{S} = \{s_t\}$. It is clear to see that the size of state space is very large.

2) *Action space*:
We make an assumption that user only utilizes L different discrete transmission power levels and thus, action space is defined as:

$$\mathcal{A} = \{a_0,\ a_1,\ ...,\ a_{L\times N-1}\} \quad (3)$$

The action at time slot t is given by $a_t = a_i$. In particular, for $a_i \in \mathcal{A}$, $i = 0,\ 1,\ 2,\ ...,\ (L \times N) - 1$, user selects access channel $u = (i/L)$ and transmits with power level $l = (i\ mod\ L)$, i.e. $p_u = p_l$ with $l = 0,\ 1,\ ...,\ (L-1)$ at time slot $t+1$.

3) *State transition probabilities*:
$\mathcal{P} = (s_{t+1}|s_t)$, for $s_{t+1}, s_t \in \mathcal{S}$ represents the set of discrete state transition probabilities. It can be seen that in a dynamic complex spectrum environment, these probabilities are unknown.

4) *Reward*: The reward which user gains at time slot t, $r_t = r(a_t, s_{t+1})$ is defined as follows:

if ($\exists$ channel i such that $c_{t+1,i} = 0$)
  if ($u = i$) **then**
    if ($p_u = p_l \mid l = 0$) **then** $r_t = 1.0$
    else if ($p_u = p_l \mid l = 1$) **then** $r_t = 0.5$
    else if ($p_u = p_l \mid 1 < l < L-1$) **then** $r_t = 0.05$
  else $r_t = 0$
else
  if ($p_u = p_l \mid l = l_{min}$ such that $(p_u - p_j) \geq \beta_{th}$) **then** $r_t = 1.0$
  else $r_t = 0$

The objective of reward function is to maximize the successful transmission probability while keep the power consumption of user as low as possible.

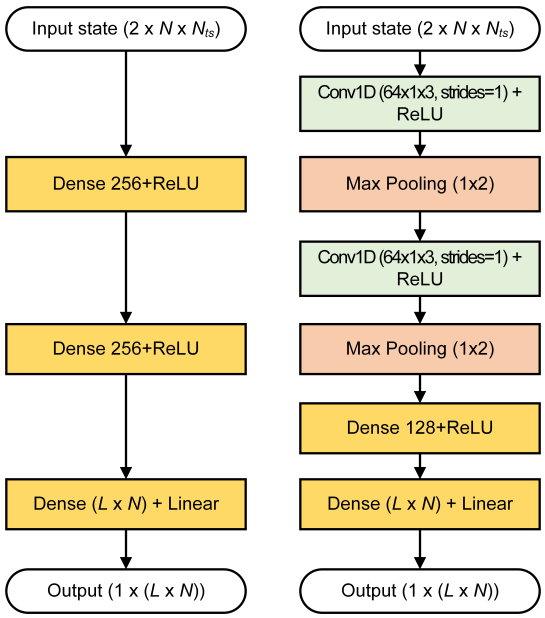
5) *Dicount factor*: γ is a real number in the interval $(0, 1)$.

The goal of cognitive node in each time slot t is to select an action that maximizes the expected accumulated future reward $\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r_t | s_0 = s\right]$ for all initial state $s \in \mathcal{S}$, where π is a policy. This goal can be achieved via finding the optimal state-action value function $Q^*(s, a)$ which is defined according to: $Q^*(s, a) = \max_\pi \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r_t | s_0 = s, a_0 = a\right]$, for $s \in \mathcal{S}, a \in \mathcal{A}$. The optimal policy can be found via an optimal Q-function or an approximation of it without the requirement of the state transition probabilities information. Therefore, it is suitable for model-free reinforcement learning algorithms.

III. Deep Double-Q Learning-based Scheme

The traditional Q-learning [10], which creates a lookup-table to select the best action, i.e. the values of state-value Q-function are stored in a table, is unable to cope with the anti-jamming problem formulated in this work since the size of the state space is very large. In recent years, Deep Q-Network (DQN) algorithm, which is the combination of Q-learning and neural network, has emerged as a powerful method to approximate Q-function. However, one of the big issue of DQN is unstable performance because of non-linear function, i.e. neural network. Several methods are proposed in [11] to improve the stability as well as performance of DQN such as experience replay, ϵ-greedy policy selection and using separate weights for networks (prediction and target) to avoid non-stationary target. In [12], the authors argue the maximum over-estimation problem in the original DQN and propose a Double-DQN algorithm. The results show that Double-DQN overcomes the maximum bias issue and achieves a more stable and better learning performance compared to DQN.

In this paper, we adapt the idea of Double-DQN and propose a deep Double-Q learning method for anti-jamming problem where a prediction Q-network Q_θ is used for selecting action and another target Q-network Q_{θ^-} is used for evaluating action. Their weights θ and θ^- are trained using experience replay. We also design two different neural network architectures for those Q-network and evaluate their performance in the next session. The architecture of two proposed networks is shown in Fig. 4. The first model is a Dense Neural Network (DNN) consisting of 2 fully-connected layers, each of which has 256 neurons. The second model is a Convolutional Neural Network (CNN) that includes 2 Conv1D layers following by a fully-connected layer. The output of both models is a vector of size $1 \times (L \times N)$ that contains the value of Q-function corresponding to state s and each of the $(L \times N)$ actions.



(a) DNN (b) CNN

Fig. 4: Architecture of proposed Deep-Q Neural Networks for anti-jamming communications.

Algorithm 1 describes the proposed Deep Double-Q learning algorithm for anti-jamming communications. At the beginning, the replay memory is set empty. Weights of prediction Q-network and target Q-network are initialized randomly. Cognitive node observes wideband spectrum to obtain the first SSV ssv_0 and puts this SSV in the state stack to get state s_0. To train the prediction Q-network, experiences are stored in the replay memory $\mathcal{M}$ using ϵ-greedy policy. At each time slot, state s_t is obtained from current SSV ssv_t and previous state s_{t-1}. Then, when $\mathcal{M}$ is big enough, a minibatch $\mathcal{B}$ of k experiences $\{(s_{t-1}, a_t, r_t, s_t)\}$, $t \in \mathcal{B}$ is randomly sampled from $\mathcal{M}$. The weights θ_t of prediction Q-network are updated with learning rate α according to gradient descent algorithm to minimize the loss function $L(\theta_t)$ as follows:

$$L(\theta_t) = \frac{1}{k}\sum_{t\in B}\Big(r_t + \gamma Q\big(s_t, \operatorname*{argmax}_a Q(s_t, a, \theta_t), \theta_t^-\big) - Q\big(s_{t-1}, a_t, \theta_t\big)\Big)^2 \quad (4)$$

In training phase, the exploration rate ϵ is gradually decreased after each time slot to balance the exploration-

Algorithm 1 Deep Double-Q Learning for Anti-jamming

Input: number of time slots T, update target network period U, number of stacked spectrum status vectors N_{ts}, mini-batch size k, discount factor γ, learning rate α.

Output: weights θ of the prediction network.

Initialize replay memory $\mathcal{M} = \emptyset$; exploration rate ϵ = 0.85
Initialize prediction Q-network and target Q-network with random weights θ and θ^-, respectively
Initialize zeros for state $s_0 = queue\{ssv_i\}$, with $i = 0, 1, 2, ..., N_{ts} - 1$
Observe spectrum to get first spectrum status vector ssv_0 and state $s_0 = s_0$.Enqueue(ssv_0)
for $t = 1$ to T **do**
    With probability ϵ, select randomly an action $a_t \in \mathcal{A}$; otherwise select $a_t = argmax_{a'} Q_\theta(s_{t-1}, a')$
    Execute action a_t and collect reward r_t
    Observe spectrum to get ssv_t and next state $s_t = s_{t-1}$.Enqueue(ssv_t)
    if $(t \geq N_{ts})$ **then**
        Store experience (s_{t-1}, a_t, r_t, s_t) in memory $\mathcal{M}$
    end if
    if $(t \geq k + N_{ts})$ **then**
        $\epsilon = \epsilon_0(T - t)/T$
        Sample randomly from $\mathcal{M}$ a minibatch $\mathcal{B}$ containing k samples: $\{(s_{t-1}, a_t, r_t, s_t)\}$, $t \in \mathcal{B}$
        Calculate loss function $L(\theta_t)$ as in Equation. (4)
        Update θ_t with α to minimize $L(\theta_t)$
        After each U time slots, update $\theta_t^- = \theta_t$
    end if
end for

exploitation trade-off. Besides, to synchronize two networks, after each period U, weights θ_t of prediction Q-network are copied to weights θ_t^- of target Q-network.

IV. Performance Evaluation

A. Simulation Setting

We implement Deep Double-Q learning algorithm and other competitors in Python with Keras library and TensorFlow backend. Table. I describes simulation parameters and values used in training Q-networks.

TABLE I: Parameters for training Q-networks.

Parameters	Value
# of channels N	10
# of training time slots T	300,000
# of SSVs in a state N_{ts}	8
# of transmitter power levels L	3 (10, 20 and 40 dBm)
Power threshold β_{th}	10 dB
Replay memory size $\|\mathcal{M}\|$	1,500
Batch size k	32
Discount factor γ	0.8
Learning rate α	0.001
Update target period U	2,000

For jamming scenarios, we deploy five different jamming patterns: 1) Comb jamming, 2) Random jamming, 3) Sweep jamming includes Sweep left jamming and Sweep right jamming, 4) Adaptive jamming and 5) Full-band jamming. Parameters of these jamming types are given in Table II. For each separate jamming pattern, jamming parameters (e.g., number of jammed channels, jamming power, hopping duration...) are varied randomly between $[30 - 200]$ time slots. We also consider a combined mode, i.e. a jammer can use a jamming pattern continuously in a duration between $[30 - 200]$ time slots and then randomly switch to a new jamming pattern. The testing process is carried out in 5,000 time slots and 20,000 time slots for each separate jamming pattern and combined mode, respectively.

TABLE II: Simulation parameters of various jamming patterns.

Patterns	Parameters	Value
Comb	# of jammed channels	6→8
	Jamming power p_j (same for all jammed channels)	5→20 dBm
Random	# of consecutive jammed channels in 1 hop	4
	Hop duration	5→10 time slots
	Jamming power p_u (same for all channels in 1 hop)	5→20 dBm
Sweep	Jamming power p_u	5→20 dBm
Adaptive	# of jammed channels	3
	Buffer size for tracking user's action	30 time slots

	Jamming power:
	if (detected $p_u \leq$ 10dBm):
	if (10 dBm $<$ detected $p_u \leq$ 20dBm):
	otherwise:
		
		8 dBm
		15 dBm
		20 dBm

Full-band	# of jammed channels	10 (all channels)
	Jamming power p_u (same for all jammed channels)	5→20 dBm

To evaluate the effectiveness of anti-jamming algorithms in terms of selecting optimal action that maximizes the successful transmission probability and minimizes user's power consumption, three performance metrics are used:

- *Average reward* $\bar{r}$: average achieved reward of user per time slot.
- *Percentage of successful transmission* $\%TX_{success}$:

$$\%TX_{success} = \frac{\text{\# of sucessful transmissions}}{\text{\# of testing time slots}} * 100 \quad (5)$$

- *Average user's transmission power* $\bar{p}_u$: average transmission power of user per time slot.

In order to compare their performances with our two proposed schemes *Double-Q DNN* and *Double-Q CNN*, three traditional decision-making models are considered as follows:

- *Random Channel, Random Power (RC-RP)*: user randomly selects an access channel and a transmission power level at each time slot.
- *Fix Channel, Random Power (FC-RP)*: user randomly selects a transmission power level but uses a fixed channel at each time slot.
- *Random Channel, Fix Power (RC-FP)*: user randomly selects an access channel but transmits with a fixed power of 20 dBm at each time slot.

TABLE III: Performance results of different schemes against various jamming patterns.

Jamming Patterns	Metrics	Schemes				
		Double-Q DNN	Double-Q CNN	RC-RP	FC-RP	RC-FP
Combined mode	$\bar{r}$	0.84	**0.89**	0.36	0.35	0.34
	$\%TX_{success}$	86.00	**90.26**	75.83	71.14	69.83
	$\bar{p}_u$ (dBm)	15.47	**14.27**	23.39	23.52	20.00
Comb	$\bar{r}$	0.77	**0.80**	0.18	0.14	0.17
	$\%TX_{success}$	78.16	**80.28**	63.49	60.55	53.09
	$\bar{p}_u$ (dBm)	10.33	**10.19**	23.53	23.62	20.00
Random	$\bar{r}$	0.88	**0.93**	0.32	0.22	0.29
	$\%TX_{success}$	90.14	**93.18**	76.88	66.85	69.43
	$\bar{p}_u$ (dBm)	10.44	**10.12**	23.10	23.36	20.00
Sweep Right	$\bar{r}$	0.90	**0.99**	0.45	0.46	0.44
	$\%TX_{success}$	93.38	**99.72**	94.44	94.28	92.48
	$\bar{p}_u$ (dBm)	10.67	**10.07**	23.91	23.31	20.00
Sweep Left	$\bar{r}$	0.91	**0.99**	0.46	0.47	0.45
	$\%TX_{success}$	94.66	**99.76**	94.16	94.24	92.80
	$\bar{p}_u$ (dBm)	10.81	**10.12**	23.29	23.27	20.00
Adaptive	$\bar{r}$	0.58	**0.63**	0.30	0.09	0.29
	$\%TX_{success}$	61.35	**67.95**	68.81	48.77	59.59
	$\bar{p}_u$ (dBm)	10.77	**10.24**	23.34	23.48	20.00
Full-band	$\bar{r}$	0.94	**0.93**	0.38	0.38	0.27
	$\%TX_{success}$	96.02	**95.28**	42.35	42.35	27.23
	$\bar{p}_u$ (dBm)	34.59	**32.86**	23.31	23.25	20.00

B. Performance Results

Table. III reports the performance of 5 algorithms in various jamming patterns in terms of average reward, percentage of successful transmission and average user's transmission power. In general, our two proposed deep Double-Q learning approaches outperform other schemes in all compared metrics. In comparison with DNN, CNN achieves a slight improvement.

For instance, in the combined jamming mode, Double-Q CNN can achieve an average reward of 0.89 and a successful transmission rate of about 90%, which are 0.53 and 14.43% higher than those of RC-RP scheme, respectively; while keep the average power transmission at only 14.27 dBm (note that the lowest possible power level which a user can use is 10 dBm). When an intelligent jammer uses adaptive jamming pattern, the performance of all schemes degrades noticeably. For this jamming scenario, it should be noted that one of the best policy is randomly select an access channel but use an adaptive power level in each timeslot. This is verified by the simulation results. We can see that the successful transmission rate of Double-Q CNN and RC-RP is quite similar. However, Double-Q CNN obtains a higher average reward and reduces significantly power consumption compared to RC-RP because our proposed scheme can learn and use a lower power level for successful transmission.

V. Conclusion

In this paper, we have studied the anti-jamming communication problem. In the dynamic shared spectrum environment, state transition probabilities are unknown and the state space size is very large. To overcome these issues, we design two anti-jamming neural networks: DNN and CNN. Additionally, a Double-Q learning algorithm is proposed to learn the best anti-jamming strategy in various jamming patterns. Simulation results show that Double-Q learning algorithm with CNN outperforms other methods to solve the decision-making for anti-jamming problem. For future research, we plan to investigate more complex scenarios such as reinforcement learning for the jammer and multi-agent reinforcement learning which form a complete Competing Cognitive Radio Network.

References

[1] J. Mitola, and G. Q. Maguire, "Cognitive radio: Making software radios more personal," *IEEE Pers. Commun.*, vol. 6, no. 4, pp. 13–18, 1999.

[2] K. Grover, A. Lim, and Q. Yang, "Jamming and anti-jamming techniques in wireless networks: A survey," *Int. J. of Adhoc and Ubiquitous Comput.*, vol. 17, no. 4, pp. 197–215, 2014.

[3] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "A survey on jamming attacks and countermeasures in WSNs," *IEEE Commun. Survey & Tutorials*, vol. 11, no. 4, pp. 42–56, 2009.

[4] B. Wang, Y. Wu, K. J. R. Liu, and T. C. Clancy, "An anti-jamming stochastic game for cognitive radio networks," *IEEE J. on Selected Areas in Commun.*, vol. 30, no. 1, pp. 4–15, 2011.

[5] L. Jia, Y. Xu, Y. Sun, S. Feng, and A. Anpalagan, "Stackelberg game approaches for anti-jamming defence in wireless networks," *IEEE Wireless Commun.*, vol. 25, no. 6, pp. 120–128, 2018.

[6] C. Han, Y. Niu, T. Pang, and Z. Xia, "Intelligent anti-jamming communication based on the modified Q-learning," in *Proc. of the Eighth Int. Congress. of Information. and Commun. Tech. (ICICT)*, pp. 1023–1031, 2018.

[7] Y. Gwon, S. Dastangoo, C. Fossa, and H. T. Kung, "Competing mobile network game: embracing antijamming and jamming strategies with reinforcement learning," in *Proc. IEEE Conf. on Commun. and Network Security (CNS)*, pp. 28–36, 2013.

[8] T. Chen, J. Liu, L. Xiao, and L. Huang, "Anti-jamming transmissions with learning in heterogeneous cognitive radio networks," in *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC)*, pp. 293–298, 2015.

[9] M. J. Ready, M. L. Downey, and L. J. Corbalis, "Automatic noise floor spectrum estimation in the presence of signals," in *Proc. IEEE ASILOMAR*, pp. 877–881, 1997.

[10] R. S. Sutton, and A. G. Barto, *Reinforcement learning: An introduction, 2nd edition*. MIT Press, 2018.

[11] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[12] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. of the Thirtieth AAAI Conf. on Artificial Intelligence*, pp. 2094–2100, 2016.