A modified Q Learning algorithm to solve Cognitive Radio jamming attack

Feten Slimeni*
VRIT Lab,
Military Academy of Tunisia, Nabeul, Tunisia
E-mail: feten.slimeni@gmail.com
*Corresponding author
Bart Scheers
CISS Departement,
Royal Military Academy (RMA), Brussels, Belgium
E-mail: bart.scheers@rma.ac.be
Zied Chtourou
VRIT Lab,
Military Academy of Tunisia, Nabeul, Tunisia
E-mail: ziedchtourou@gmail.com
Vincent Le Nir
CISS Departement,
Royal Military Academy (RMA), Brussels, Belgium
E-mail: vincent.lenir@rma.ac.be
Rabah Attia
SERCOM Lab,
EPT University of Carthage, Marsa, Tunisia
E-mail:rabah_attia@yahoo.fr

Abstract: Since the jamming attack is one of the most severe threats in cognitive radio networks, we study how Q-learning can be used to pro-actively avoid jammed channels. However, Q-learning needs a long training period to learn the behavior of the jammer. We take advantage of wideband spectrum sensing to speed up the learning process and we make advantage of the already learned information to minimize the number of collisions with the jammer. The learned anti-jamming strategy depends on the elected reward strategy which reflects the preferences of the cognitive radio. We start with a reward strategy based on the avoidance of the jammed channels, then we propose an amelioration to minimize the number of frequency switching. The effectiveness of our proposal is evaluated in the presence of different jamming strategies and compared to the original Q-learning algorithm. We compare also the anti-jamming strategies related to the two proposed reward strategies.

Keywords: Cognitive radio network; jamming attack; Markov decision process; Q-learning algorithm.

1 Introduction

Cognitive Radio (CR) technology is recognized as a promising solution to overcome the problems of scarcity and inefficient utilization of the radio spectrum. The CR associates learning and reconfigurability abilities in order to perform a real time adaptation to the environment modifications Mitola and Maguire (1999); Mahmoud (2007); Raval et al. (2014).

However, in addition to common wireless communication vulnerabilities, the cognitive radio networks (CRNs) are susceptible to other kinds of threats related to the intrinsic characteristics of this technology Alhakami et al. (2014); El-Saleh et al. (2011); Yang et al. (2013). Recently, research works have been done in the area of CRN security and especially the topic of opportunistic spectrum access in the presence of jammers.

The jamming attack is one of the major threats in CRNs because it can lead to network degradation and even denial of service (DoS). Furthermore, the jammer doesn't need to be a member of the network or to collect information about it to launch such attack. The jammers can be classified according to the following criteria:

1.1 Spot/Sweep/Barrage jamming

Spot jamming consists in attacking a specific frequency, while a sweep jammer will sweep across an available frequency band. A barrage jammer will jam a range of frequencies at once.

1.2 Single/Collaborative jamming

The jamming attack can be done by a single jammer or in a coordinated way between several jammers to gain more knowledge about the network and to efficiently reduce the throughput of the cognitive users.

1.3 Constant/Random jamming

The jammer can either send jamming signals continuously on a specific channel or alternate between jamming and sleeping.

1.4 Deceptive/Reactive jamming

A deceptive jammer continuously transmits signals in order to imitate a legitimate or primary user. A reactive jammer transmits only when it detects busy channel to cause collisions.

More details about the classification of CRN jamming strategies are given in Pietro and Oligeri (2013). This reference deals with the problem of spectrum coordination between CRs in the presence of jammers. CRNs are characterized by dynamic spectrum access (DSA) and by mainly distributed architectures which make it difficult to implement effective jamming countermeasures. Therefore, some coding techniques have been developed to mitigate the effects of this attack in the transmitted signal. For example, the authors in Asterjadhi and Zorzi (2010) combine random linear network coding with random channel hopping sequences to overcome the jamming effect on the transmitted control packets. Their proposed algorithm is called jamming evasive network coding neighbor discovery algorithm (JENNA). Another coding approach is presented in Balogun (2014), it consists in a hybrid forward error correction (FEC) code to mitigate the jamming impact on the transmitted data. The code is a concatenation of the raptor code to recover data loss due to jamming, and the secure hash algorithm (SHA-2) to verify the integrity of the received data. Instead of using coding technique to repair the already jammed data, an approach presented in Wang et al. (2013) consists in a multi-tier proxy based cooperative defense strategy. It exploits the time and spatial diversity of the CRs to deal with collaborative jamming attack in an infrastructure based centralized CRN. Furthermore, the concept of honeynode has been shown in Bhunia et al. (2014) to be effective in deceiving jammers about the transmitting nodes. In this reference, a single honeynode is dynamically selected for each transmitting period, to act as a normal transmitting CR in order to attract the jammer to a specific channel.

Another class of anti-jamming approaches is based on the CR ability of changing its operating frequency while maintaining continuous and proper operation. This ability can be exploited to overcome jamming attacks since the CR can hop to avoid jammed channels. In this context, markov decision process (MDP) has been widely exploited as a stochastic tool to model the CR decision making problem in jamming scenarios with fixed strategy, i.e. assuming that the jammer preserves the same tactic. The CR may use reinforcement learning (RL) algorithms to solve the MDP by learning how to take the best decisions to keep its communication unjammed. The Q-learning is the most common RL algorithm applied in CRN jamming study to deal with imperfect knowledge about the environment and the jammer's behavior. However, the application of this technique should go through two phases: the first one is a training phase during which the agent runs the Q-learning algorithm and waits until its convergence to get the optimal defense strategy. The next phase is the exploitation of the learned strategy during the real time working of the agent. An off-line application of this technique seems to be inefficient for the CR, because until the convergence of the Q-learning algorithm other jammers may emerge and legacy spectrum holders (primary users) activity may change. During the training phase of the Q-learning algorithm, the CR can already exploit the communication link, denoted as on-line learning, but it may lose many data packets because of the random learning trials.

The work developed in this paper is mainly based on Wu et al. (2010) and Chen et al. (2013). In the first paper, the authors start by deriving a frequency hopping defense strategy for the CR using an MDP model under the assumption of perfect knowledge, in terms of transition probabilities and rewards. Further, they propose two learning schemes for CRs to gain knowledge of adversaries to handle cases of imperfect knowledge: maximum likelihood estimation (MLE), and an adapted version of the Q-learning algorithm. However the modified Q-learning algorithm is given without discussion or simulation results. The second paper gives an MDP model of the CRN jamming scenario and proposes a modified Q-learning algorithm to solve it. Again, as in the previous reference no details are given on how to implement the described theoretical anti-jamming scheme.

In this paper, we aim to provide a modified version of the Q-learning algorithm to speed up the training period and to make it appropriate for on-line learning. We start in the next section by explaining how the markov decision process (MDP) can model the scenario of CRN under fixed jamming strategy. In section III, we present the standard Q-learning algorithm and we discuss its application to find an anti-jamming strategy. In the remainder of this paper, we propose an MDP model to the CRN jamming scenario and we present a modified Q-learning version. During learning, the CR tries to maximize its long term return which combines into the Q-values, the sequence of rewards related to the visited states and taken actions. So, the learned anti-jamming strategy depends on the definition of the reward strategy. In this paper, we propose two reward strategies. The first strategy gives a negative reward to the frequency on which a collision between the CR and the jammer occurs. In addition to this penalty, the second reward strategy accords a negative reward when the CR does unnecessary frequency switching; i.e. it switches to another frequency and the previous frequency still not jammed. The goal is to stay as longer as possible in the same frequency before the jammer comes to attack it, since at every handoff the CR needs to stop transmitting, do spectrum sensing, he requires also handshake and synchronization with the other side to restart transmitting, among other consequence of frequent handoff is new session every time and so new encryption keys. We evaluate the effectiveness of the modified Q-learning algorithm in the presence of different jamming strategies. The simulation results are compared to the original Q-learning algorithm applied to the same scenarios. We present and compare also the learned anti-jamming strategies related to the two proposed reward strategies.

2 The Markov decision process

The markov decision process (MDP) is a discrete time stochastic control process. It provides a mathematical framework to model the decision problem faced by an agent to optimize his outcome. The goal of solving the MDP is to find the optimal strategy for the considered agent. In CRN jamming scenario, it means finding the best actions (to hop or to stay) for the CR to avoid the jammed frequency.

An MDP is defined by four essential components:

- A finite set of states S.
- A finite set of actions A.
- $P_a(s, s') = Pr(s_{t+1} = s' | s_t = s, a_t = a)$ the transition probability from an old state s to a new state s' when taking action a.
- $R_a(s, s')$ the immediate reward after transition to state s' from state s when taking action a.

The process is played in a sequence of stages (timesteps). At every stage, the agent is in one state and at the end of that stage he selects an action, then the process moves to a new random state with the corresponding transition probability. The agent receives a payoff, also called reward, which depends on the current state and the taken action. He continues to play stages until finding the optimal policy, which is the mapping from states to actions that maximizes the state values. The standard family of algorithms used to calculate this optimal policy requires storage of two arrays indexed by state:

- State value $V(s)$, which contains a real value corresponding to the discounted sum of the rewards received when starting from each state.
- Policy $\pi(s)$ which gives the action taken in every state.

Every MDP has at least one optimal policy π^* that is stationary and deterministic. π^* is called stationary since it does not change as a function of time and it is called deterministic since the same action is always chosen whenever the agent is in one state s. At the end of the algorithm, π^* will contain the optimal solution and $V(s)$ will contain the discounted sum of the rewards to be earned by following that policy from state s.

Markov decision processes can be solved via dynamic programming (DP) when we have perfect knowledge about transition probabilities and the reward of every action. However in real situations of dynamic environment and imperfect knowledge about transition probabilities and rewards, MDP is solved using reinforcement learning (RL) algorithms Szepesvri and Littman (1996).

Dynamic programming (DP) techniques require an explicit, complete model of the process to be controlled. It is known as model based techniques, since we have to reconstruct an approximate model of the MDP and then solve it to find the optimal policy. The most popular DP techniques is the value iteration algorithm which consists in solving the following Bellman equation until convergence to the optimal values $V^*(s)$, from which we can derive the corresponding optimal policy:

$$Q(s, a) = R_a(s, s') + \gamma \sum_{s'} P_a(s, s') V^*(s') \tag{1}$$

$$V^*(s) = max_a Q(s, a) \tag{2}$$

where γ is the discount factor that controls how much effect future rewards have on the optimal decisions. Small values of γ emphasizing near-term gain and larger values giving significant weight to later rewards. Equation (1) is repeated for all possible actions in each state s. It calculates the sum of the immediate reward $R_a(s, s')$ of the taken action and the expected sum of rewards over all future steps. Then, equation (2) gives the optimal action which corresponds to the maximum $V(s)$ value. The value iteration algorithm reaches convergence when $|V_{n+1}(s) - V_n(s)| < \epsilon$ is met for all states s, where $V_n(s)$ corresponds to the calculated $V(s)$ value at timeslot n.

However, in real scenarios the CR is acting in hostile and dynamic environment without complete information. It doesn't know either the resulting new state after taking an action or the reward/cost of its action. For example, hopping to another frequency may lead to jamming situation or successful transmission. This situation can be defined as a reinforcement learning (RL) problem, in which an agent wanders in an unknown environment and tries to maximize its long term return by performing actions and receiving rewards Henrique and Ribeiro. Therefore, the CR should use learning algorithms to learn PU's and jammer's activities. After learning the jammers policy, it can predict the next action of the jammer and plan its next course of action to avoid jammed channels.

3 The Q-learning algorithm

Learning algorithms can be used as a model-free simulation tool for determining the optimal policy π^* without initially knowing the action rewards and the transition probabilities. Autonomous RL is completely based on interactive experience to update the information step by step, and based on this derive an estimate to the optimal policy. The most popular RL method is the Q-learning algorithm, which is an extension to the value iteration algorithm to be applied in non deterministic markov decision processes.

As first introduced in Watkins (1989), the Q-learning algorithm is a simple way for agents to learn how to act optimally by successively improving its evaluations of the quality of different actions at every state. It consists in approximating the unknown transition probabilities by the empirical distribution of states that have been reached as the process unfolds. The goal is finding a mapping from state/action pairs to Q-values. This result can be represented by a matrix of N_s lines, where N_s is the number of states s, and N_a columns corresponding to possible actions a. The Bellman equation (1) is replaced in this algorithm by an iterative process; at every timeslot the algorithm measures the feedback rewards of taking an action a in a state s, and updates the corresponding $Q(s,a)$:

$$Q[s,a] \leftarrow Q[s,a] + \alpha\,[R_a(s,s') + \gamma\ max_a Q(s',a) - Q[s,a]] \quad (3)$$

which gives:

$$Q[s,a] \leftarrow (1-\alpha)Q[s,a] + \alpha\,[R_a(s,s') + \gamma\ max_a Q(s',a)] \quad (4)$$

where $0 < \alpha \leq 1$ is a learning rate that controls how quickly new estimates are blended into old estimates.

The Q-value is a prediction of the sum of the discounted reinforcements (rewards) received when performing the taken action and then following the given policy thereafter. It can be considered as a measure of the goodness of that action choice.

The Q-learning algorithm updates the values of $Q(s,a)$ through many episodes (trials) until convergence to optimal Q^* values; this is known as the training/learning stage of the algorithm. Each episode starts from a random initial state s_1 and consists on a sequence of timeslots during which the agent goes from state to another and updates the corresponding Q value. Each time the agent reaches the goal state, which have to be defined depending on the scenario, the episode ends and he starts a new trial. The convergence to the optimal Q^* matrix requires visiting every state-action pair as many times as needed. In simulation, this problem is known as the exploration issue. Random exploration takes too long to focus on the best actions which leads to a long training period of many episodes. Furthermore, it does not guarantee that all states will be visited enough, as a result the learner would not expect the trained Q function to exactly match the ideal optimal Q^* matrix for the MDP Tesauro (2003). The training phase of the Q-learning process is described in algorithm 1 Sutton and Barto (1998).

Two main characteristics of the standard Q-learning algorithm are: (i) it is said to be an asynchronous process since at each timeslot the agent updates a single $Q(s,a)$ value (one matrix cell), corresponding to his current state s (line s) and his action a (column a) taken at this timeslot Abounadi et al. (2002). (ii) The Q-learning method does not specify what action a the agent should take at each timeslot during the learning period, therefore it is called OFF-policy algorithm allowing arbitrary experimentation until convergence to stationary Q values Even-Dar and Mansour (2003). The optimal Q^* matrix resulting from the learning period will be exploited by the agent as the best policy. During the exploitation phase, when he is in a state s, he has to take the action corresponding to the maximum value in the matrix line $Q^*(s,:)$.

In previous sections, we have explained the MDP and the Q-learning algorithm tools commonly used to model and solve the CRN scenario under static jamming strategy. The CR can apply the Q-learning algorithm to learn the jammer's behavior, but it have to wait for a long training period before getting the optimal anti-jamming strategy. Moreover, as the CR has to try random actions before the convergence of the Q-learning algorithm, it is not suitable to do learning in an operational communication link because the CR may loss many transmitted packets. As a solution to these challenges, we propose in the next section a modified version of the Q-learning algorithm, and we

Algorithm 1 Pseudocode of the Q-learning algorithm

```
Set the $\gamma$ parameter, and the matrix $R$ of environment rewards.
Initialize the matrix Q as a zero matrix.
for each episode do
    Select a random initial state $s = s_1$.
    while the goal state hasn't been reached do
        Select one action $a$ among all possible actions for the current state.
        Using this possible action, consider going to the next state $s'$.
        Get maximum $Q$ value for this next state based on all possible actions $max_a(Q(s', a))$.
        Compute: $Q(s, a) = R_a(s, s') + \gamma\ max_a(Q(s', a))$

        Set the next state as the current state $s = s'$.
    end while
end for
```

will denote this version as ON-policy synchronous Q-learning (OPSQ-learning) algorithm.

4 The On-policy synchronous Q-learning algorithm

We will start by defining a markov decision process to model the CR's available states and actions, with the consideration of unknown transition probabilities and unknown immediate rewards of the taken actions. Then, we will present a modified version of the Q-learning algorithm that we have implemented to solve the defined MDP model.

4.1 Markov decision process model

We consider a fixed jamming strategy to solve the decision making problem from the side of the CR trying to find an anti-jamming strategy.

Assume there are M available channels for the CR and there is a jammer trying to prevent it from an efficient exploitation of these channels. As a defense strategy, the CR have to choose at every timeslot either to keep transmitting over the same channel or to hop to another one. The challenge is to learn how to escape from jammed channels without scarifying a long training period to learn the jammer's strategy. Lets define the finite set of possible states, the finite set of possible actions at each state and the resultant rewards after taking these actions.

The state of the CR is defined by a pair of parameters: its current operating frequency and the number of successive timeslots staying in this frequency. Therefore, its state at a timeslot i is represented by the pair $s_i = (f_i, k)$, where f_i is its operating frequency at this timeslot i and k is the number of successive timeslots using this frequency. We have opt for mixing spatial and temporal properties in the state space definition to get a Markovian evolution of the environment.

At every state, the CR should choose an action to move to another state, which means that it has to choose its future frequency. Therefore, we define its possible actions as a set of M actions, which are the M available channels: $\{f_1, f_2, ..., f_M\}$. An example of the Q matrix composed by these states and actions is given in Table 1.

Assume the reward is zero $R_a(s, s') = 0$ whenever the new frequency after choosing the action a is not jammed, and $R_a(s, s') = -1$ when the CR takes an action a resulting to a jammed frequency. We consider the jammed state as a failure and a situation that should be avoided.

4.2 The learning process

We present in algorithm 2, a modified version of the Q-learning process denoted as the ON-policy synchronous Q-learning (OPSQ-learning), because of the two following modifications: (i) We have replaced the OFF-policy characterizing the standard Q-learning algorithm by an ON-policy, i.e. at each timeslot, the CR follows a greedy strategy by selecting the best action corresponding to $max_a Q(s, a)$ instead of trying random action. (ii) We have exploited the CR ability of doing wideband spectrum sensing, to do synchronous update of M Q-values instead of the asynchronous update of only one cell in the Q matrix, i.e. the CR after going to a next state can, using its wideband sensing capability, detect the frequency of the jammer at that moment and hence do an update of all state-action pairs, corresponding to the possible actions which can be taken from its previous state s (update of all columns of the Q matrix line $Q(s, :)$). Due to the second modification (the synchronous Q-values update), the modified Q-learning algorithm is no longer a model-free technique but it can be seen as a model-based technique, i.e. the CR can learn without actually apply the action.

We assume that the CR and the jammer are time synchronized. Also, we have to mention that we are assuming perfect spectrum sensing and full observations for simplicity. But we cite some interesting references dealing with the influence of the radio channel in the estimation of the detected signal. For example, Arokkiaraj and Jayasankar (2014) develops and analyzes an adaptive spectrum sensing scheme according to the variation of time-varying channels, Kyperountas et al. studies the cooperative spectrum sensing for a cognitive radio system operating in AWGN, correlated/uncorrelated shadowing, and in channels featuring composite large-scale and small scale fading. Also, Molisch et al. (2009) provides a

Algorithm 2 Pseudocode of ON-policy synchronous Q-learning

```
Set $\gamma$ and $\epsilon$ values.
Initialize matrix $Q_1$ to zero matrix.
Select a random initial state $s = s_1$
Set n=1, timeslot=1
while n<Nepisodes do
  $Q_{n-1} = Q_n$, $R_a(s, s') = 0$ $\forall$ $a$,$s$,$s'$
  Calculate the learning coefficient $\alpha = 1/timeslot$
  Select an action $a$ verifying $max_a Q_{n-1}(s, a)$
  Taking $a$, go to the new state $s'$ at frequency $f'$
  Find the new jammed frequency $f_{jam}$ %(due to wideband spectrum sensing)
  Update all $Q_n$ values of the previous state $s$ by doing:
  for $i = 1 : M$ do
    observe the fictive state $s_{tmp}$ of taking fictive action $f_i$
    if $f_i = f_{jam}$ then
      $R_{f_i}(s, s_{tmp}) = -1$
    else
      $R_{f_i}(s, s_{tmp}) = 0$
    end if
    Compute $Q_n(s, f_i) = (1 - \alpha)Q_{n-1}(s, f_i) + \alpha[R_{f_i}(s, s_{tmp}) + \gamma\ max_a Q_{n-1}(s_{tmp}, a)]$
  end for
  if $f' = f_{jam}$ %(end of episode) then
    n=n+1
    timeslot=1
    Select a random initial state $s = s_1$
  else
    $s = s'$
    timeslot=timeslot+1
  end if
  if $(abs(Q_n(s, a) - Q_{n-1}(s, a)) < \epsilon)$ $\forall$ $s$,$a$ then
    break
  end if
end while
```

comprehensive overview of the propagation channel models that will be used for the design of cognitive radio systems and deals with the time variations of the channel response which determine how often potential interference levels have to be estimated and, thus, how often transmission strategies may have to be adapted.

To evaluate the effectiveness of the proposed solution, we have applied both the standard version of the Q-learning algorithm (characterized by OFF-policy and asynchronous update) and the modified ON-policy synchronous Q-learning algorithm to the described MDP model. Note that in this algorithm, our episode starts from a random frequency, going from one state to another by taking the best action at every timeslot, and ends whenever the CR goes to a jammed frequency.

The next subsection presents the simulation results in the presence of various jamming strategies.

4.3 Simulation results

We have considered in the simulations four available frequencies ($M = 4$) for the CR. We have implemented both the standard and the modified versions of the Q-learning algorithm, under sweeping, reactive and pseudo random jamming strategies.

We started by the implementation of the standard version of Q-learning algorithm. We found, by averaging over many simulations, that it takes about one hundred episodes to converge to the matrix Q^*. Then, we have implemented the modified Q-learning version (OPSQ-learning) and we give the results in the following paragraphs. The following figures display the anti-jamming strategy in the exploitation phase, after running the learning algorithm. We have considered a discount factor $\gamma = 0.95$ and $\epsilon = 10^{-2}$ for the convergence condition. We are using the red color to indicate the jammed frequencies and the blue color to indicate the CR frequencies for an exploitation period of twenty timeslots.

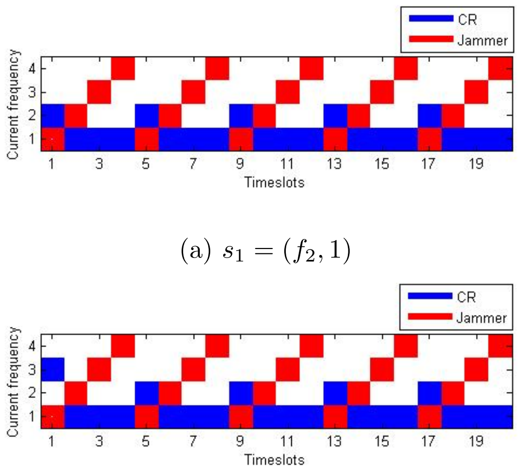
4.3.1 Scenario with a sweeping jammer

As a first scenario, we consider a jammer sweeping over the available spectrum frequencies by attacking at each timeslot one frequency. The OPSQ-learning algorithm converges after only one or two episodes depending on the initial state. The Q^* matrix is given in Table 1. The strategy given by this resulting Q^* matrix is shown in Fig. 1, when the CR starts as initial random state s_1 from the frequencies f_2 and f_3 respectively.

Table 1 The Q^* matrix in a sweeping jammer scenario

State \ Action	f_1	f_2	f_3	f_4
(f_1,1)	0	0	-0.8356	0
(f_1,2)	0	0	0	-0.6768
(f_1,3)	-0.5770	0	0	0
⋮	⋮	⋮	⋮	⋮
(f_2,1)	0	-0.3822	0	0
⋮	⋮	⋮	⋮	⋮
(f_3,1)	0	-1	0	0
⋮	⋮	⋮	⋮	⋮
(f_4,1)	0	0	0	0
⋮	⋮	⋮	⋮	⋮

4.3.2 Scenario with a reactive jammer



(a) $s_1 = (f_2, 1)$

(b) $s_1 = (f_3, 1)$

Figure 1: Exploitation of the learned policy against a sweeping jammer

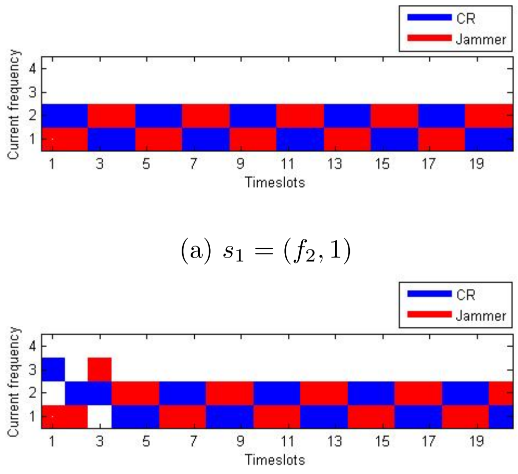
In this scenario, we consider a reactive jammer. We suppose that this jammer needs a duration of two timeslots before jamming the detected frequency, because it has to do the spectrum sensing, then make the decision and finally hop to the detected frequency. The OPSQ-learning algorithm converges in this scenario after four episodes. The Q^* matrix is given in Table 2.

According to the resulting Q^* matrix, the CR succeeds to learn that it has to change its operating frequency every two timeslots to escape from the reactive jammer. The learned strategy is given in Fig. 2 when the CR starts respectively from the frequencies f_2 and f_3 as initial state s_1.

Table 2 The $Q*$ matrix in a reactive jammer scenario

State \ Action	f_1	f_2	f_3	f_4
(f_1,1)	0	-0.8047	0	0
(f_1,2)	-0.6986	0	0	0
⋮	⋮	⋮	⋮	⋮
(f_2,1)	-1	0	0	0
(f_2,2)	0	-0.6861	0	0
⋮	⋮	⋮	⋮	⋮
(f_3,1)	-1	0	0	0
⋮	⋮	⋮	⋮	⋮
(f_4,1)	-1	0	0	0
⋮	⋮	⋮	⋮	⋮

4.3.3 Scenario with a pseudo random jammer



(a) $s_1 = (f_2, 1)$

(b) $s_1 = (f_3, 1)$

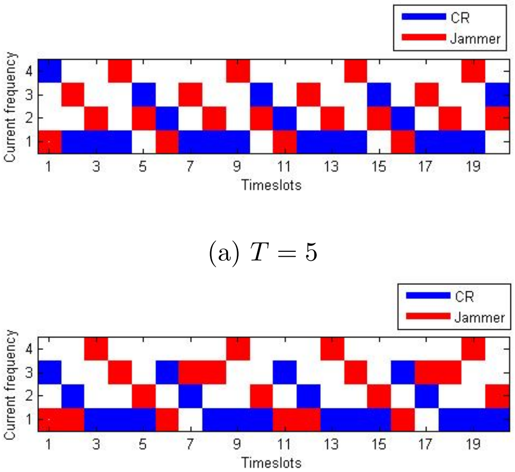
Figure 2: Exploitation of the learned policy against a reactive jammer

In this scenario, we consider a jammer with a pseudo random strategy. We suppose that at every timeslot, this jammer attacks randomly one of the four frequencies, and after a period T it repeats the same sequence of the attacked frequencies. We started with a period $T = 5$ during which the random sequence is $(1, 3, 2, 4, 2)$, we found that the OPSQ-learning algorithm converges in this scenario after four episodes. Then, we considered a period $T = 10$ during which the random sequence is $(1, 1, 4, 3, 2, 1, 3, 3, 4, 2)$, we found that the OPSQ-learning algorithm converges in this scenario after five episodes. The Q^* matrix is given in Table 3.

The CR succeeds to learn the pseudo random strategy of the jammer, and the learned anti-jamming strategies are given in Fig. 3 when the periods of the pseudo random jamming sequences are respectively $T = 5$ and $T = 10$ timeslots.

Table 3 The $Q*$ matrix in a pseudo random jammer with a period of 5 timeslots

State \ Action	f_1	f_2	f_3	f_4
(f_1,1)	0	-0.8235	0	-0.0882
(f_1,2)	0	-0.1130	0	-0.6610
(f_1,3)	-0.1100	-0.5602	0	0
⋮	⋮	⋮	⋮	⋮
(f_2,1)	0	0	-0.3236	0
⋮	⋮	⋮	⋮	⋮
(f_3,1)	-1	0	0	0
⋮	⋮	⋮	⋮	⋮
(f_4,1)	0	0	-1	0
⋮	⋮	⋮	⋮	⋮



(a) $T = 5$

(b) $T = 10$

Figure 3: Exploitation of the learned policy against a pseudo random jammer

4.4 Discussion

The standard Q-learning algorithm converges after about one hundred episodes; each episode starts from a random frequency, going randomly from one frequency to another taking random decisions until collision with the jammer. The CR applying this technique have to either wait for all this training period to get an anti-jamming strategy or to use it during real time communication and sacrifice about hundred lost packets.

The ON-policy synchronous Q-learning algorithm converges faster than the standard Q-learning algorithm; it gives a suitable defense strategy after about four training episodes against sweeping and reactive jammers. This is due to the synchronous update of all Q-values of possible actions from a current state, which helps the CR to faster improve its beliefs about all decisions without trying all of the actions. Furthermore, the choice of taking at every timeslot the best action (until the actual moment) promotes the real time exploitation of the OPSQ-learning algorithm during the CR communication. Because the OPSQ algorithm learns the safe strategy (it takes the action selection method into account when learning), it receives a higher average reward per trial than Q-learning as given by Fig. 4. But, We should mention that the proposed OPSQ-learning algorithm doesn't optimize the entire matrix Q, it just optimizes the Q-values of state/action pairs that the CR goes through until finding an anti-jamming strategy. The CR using the proposed algorithm succeeds to learn how to avoid the jammed channels, but as we can see in figures 2(b) and 3, the CR does unneeded frequency switching. It means that he learned to jump from frequency to an another even if the first one will not be jammed in the next timeslot, which costs in terms of time, frequency and power consumption. This disadvantage is due to the elected reward strategy, in which we accord -1 just to the choice of a jammed frequency, otherwise the CR receives zero as reward. In the next section, we propose an ameliorated reward strategy trying to find the optimal anti-jamming strategy.

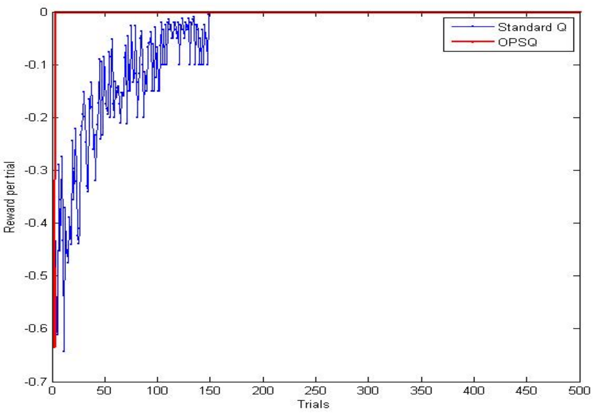


Figure 4: Comparison between Q-learning and OPSQ-learning

5 Changing the reward strategy to optimize the anti-jamming strategy

In this section, we propose another reward strategy according a penalty of $R2 = -1$ not only for the choice of a jammed frequency but also for the frequency switching without having the previous frequency attacked by the jammer. We integrate this new reward strategy in the OPSQ algorithm given by the pseudocode 2 , and the new learning algorithm is given by pseudocode 3.

We have implemented this new algorithm considering the same simulation parameters as given in subsection 4.3 with the previous simple reward strategy (penalty of -1 just for being jammed). Against a sweeping jammer we get the same simulation results as the results given in the previous section. But against reactive and pseudo random jammers, the CR succeeds to avoid the jammed channels with the minimum number of frequency switching.

5.1 Scenario with a reactive jammer

The CR succeeds to learn not only that it has to change its operating frequency every two timeslots to escape from the reactive jammer, but also that starting from frequency f_3 he doesn't need to hop to the frequency f_2 as he does in Fig. 2(b). The Q^* matrix is given in

Algorithm 3 Pseudocode of OPSQ-learning with modified reward strategy

```
Set $\gamma$ and $\epsilon$ values.
Initialize matrix $Q_1$ to zero matrix.
Select a random initial state $s = s_1$ at a frequency $f$
Set n=1, timeslot=1
while n<Nepisodes do
    $Q_{n-1} = Q_n$, $R2_a(s, s') = 0 \; \forall \; a,s,s'$
    Calculate the learning coefficient $\alpha = 1/timeslot$
    Select an action $a$ verifying $max_a Q_{n-1}(s, a)$
    Taking $a$, go to the new state $s'$ at frequency $f'$
    Find the new jammed frequency $f_{jam}$ %(due to wideband spectrum sensing)
    Update all $Q_n$ values of the previous state $s$ related to the previous frequency $f$ by doing:
    for $i = 1 : M$ do
        observe the fictive state $s_{tmp}$ of taking fictive action $f_i$
        if $f_i = f_{jam}$ then
            $R2_{f_i}(s, s_{tmp}) = -1$ % (jammed)
        else
            if $f_i = f$ then
                $R2_{f_i}(s, s_{tmp}) = 0$
            else
                if $f = f_{jam}$ then
                    $R2_{f_i}(s, s_{tmp}) = 0$
                else
                    $R2_{f_i}(s, s_{tmp}) = -1$ % (unneeded hop)
                end if
            end if
        end if
        Compute $Q_n(s, f_i) = (1 - \alpha)Q_{n-1}(s, f_i) + \alpha[R2_{f_i}(s, s_{tmp}) + \gamma \; max_a Q_{n-1}(s_{tmp}, a)]$
    end for
    if $f' = f_{jam}$ %(end of episode) then
        n=n+1
        timeslot=1
        Select a random initial state $s = s_1$
    else
        $s = s'$
        timeslot=timeslot+1
    end if
    if $(abs(Q_n(s, a) - Q_{n-1}(s, a)) < \epsilon) \; \forall \; s,a$ then
        break
    end if
end while
```

Table 4 and the ameliorated learned strategy is given in Fig. 5 when the CR starts f_3 as initial state s_1.

5.2 Scenario with a pseudo random jammer

Table 4 The $Q*$ matrix in a reactive jammer with new reward function

State \ Action	f_1	f_2	f_3	f_4
$(f_1,1)$	0	-0.727	-0.727	-0.727
$(f_1,2)$	-0.6287	0	0	0
⋮	⋮	⋮	⋮	⋮
$(f_2,1)$	-1	0	-1	-1
$(f_2,2)$	0	-0.4164	0	0
⋮	⋮	⋮	⋮	⋮
$(f_3,1)$	-1	-1	0	-1
$(f_3,1)$	-1	-1	0	-1
$(f_3,2)$	0	0	-0.75	0
⋮	⋮	⋮	⋮	⋮
$(f_4,1)$	-1	-1	-1	0
⋮	⋮	⋮	⋮	⋮

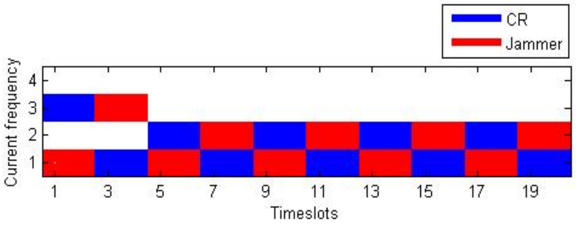


Figure 5: Exploitation of the optimal policy against a reactive jammer

In this scenario, we consider jammers with the same pseudo random strategies as the previous section: the same random sequence $(1, 3, 2, 4, 2)$ of period $T = 5$ and the same random sequence $(1, 1, 4, 3, 2, 1, 3, 3, 4, 2)$ of period $T = 10$. The CR succeeds not only to learn the pseudo random strategies of the jammers, but with the minimum number of frequency switching actions compared to Fig. 3. The Q^* matrix is given in Table 5.

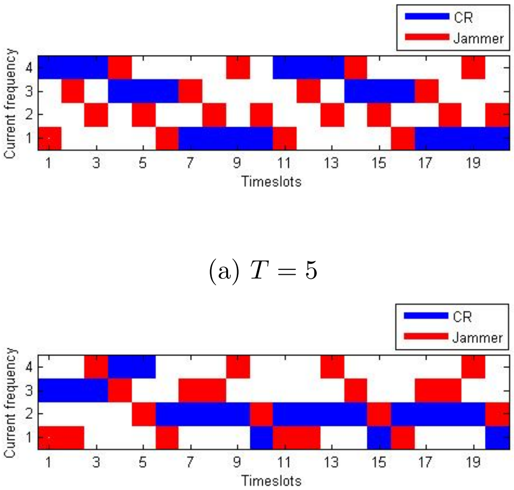
The learned anti-jamming strategies are given in Fig. 6 when the periods of the pseudo random jamming sequences are respectively $T = 5$ and $T = 10$ timeslots.

5.3 Comparison in terms of reward per trial

According to Fig. 7, we can conclude that using the new reward strategy with two penalties (one for being jammed and the other for extra frequency switching) reaches later the level $reward = 0$ than the algorithm using the simple reward strategy (according a penalty just for being jammed), but the difference is negligible compared to the gain in terms of frequency hopping.

Table 5 The $Q*$ matrix in a pseudo random jammer (5TS), with new reward function

State \ Action	f_1	f_2	f_3	f_4
(f_1,1)	-0.4334	-0.996	-1.2795	-1.2748
(f_1,2)	-0.3871	-1.052	-0.9825	-0.7728
(f_1,3)	-0.1482	-0.7613	-0.9579	-0.9555
(f_1,4)	-0.6294	-0.2356	-0.1793	-0.1727
⋮	⋮	⋮	⋮	⋮
(f_2,1)	-0.643	-0.5728	-0.4717	-0.6321
⋮	⋮	⋮	⋮	⋮
(f_3,1)	-1.4148	-1.0277	-0.3695	-1.3616
(f_3,2)	-1.028	-1.397	-0.4091	-1.3918
(f_3,3)	-0.4521	-0.4877	-0.9343	-0.4569
⋮	⋮	⋮	⋮	⋮
(f_4,1)	-1.4152	-1.4481	-1	-0.3657
(f_4,2)	-1.3753	-0.9765	-1.3048	-0.3504
(f_4,3)	-0.415	-0.4296	-0.3436	-1
⋮	⋮	⋮	⋮	⋮



(a) $T = 5$

(b) $T = 10$

Figure 6: Exploitation of the optimal policy against a pseudo random jammer

6 Conclusion

In this work, we have discussed the exploitation of the MDP model and the Q-learning algorithm to find an anti-jamming strategy in CRNs. We have modeled the scenario of fixed jamming strategy as an MDP model. Then, we have proposed a modified Q-learning algorithm to solve it, we call the proposed algorithm as the ON-policy synchronous Q-learning (OPSQ-learning) algorithm. To learn the optimal anti-jamming strategy, we have ameliorated the reward strategy in order to stay as longer as possible in the same frequency and minimize the number of frequency switching. We have presented the simulation results of the application of both the standard Q-learning and the OPSQ-learning algorithm under sweeping, reactive and pseudo random jamming strategies. We can conclude that the OPSQ-learning version speeds up the learning period and the ameliorated reward strategy optimizes the number of channel switching which enhance its application during CRN real time communication. As future work, the presented solution will be tested in real platform and real environment, considering multiple jammers and primary users.

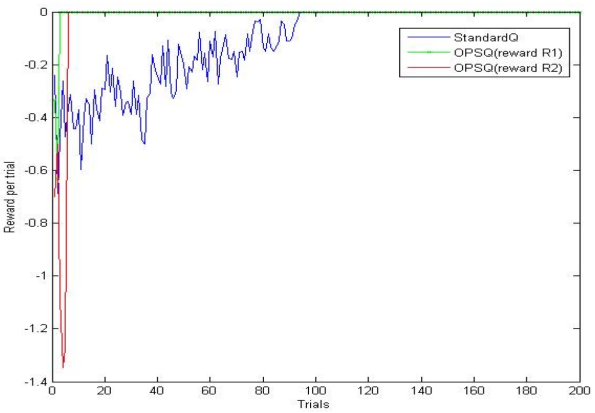


Figure 7: Comparison between OPSQ-learning with different reward strategies

Acknowledgements

This work has been supported by a grant from the Tunisian defense ministry.

References

Mitola III, J. and G.Q. Maguire Jr. (1999) 'Cognitive radio: making software radios more personal', *IEEE Personal Communications Magazine*, Vol. 6, No. 4, pp.13–18

Mahmoud, Q. (2007) 'Cognitive Networks: Towards Self-Aware Networks', *John Wiley and Sons*, pp.278–279

M. Raval, S. and B. Soni, H. and D. Trapasiya, S. (2014) 'Review on Resource Efficient Relay Selection Scheme for Cognitive Radio Networks', *International Journal Of Engineering And Science (IJES)*, Vol. 3, No. 3, pp.57–62.

Alhakami, W., Mansour, A. and A Safdar, G. 2014 'Spectrum Sharing Security and Attacks in CRNs: a Review', *International Journal of Advanced Computer Science and Applications(IJACSA)*, Vol. 5, No. 1, pp.76–87.

El-Saleh, A. and Ismail, M. and Mohd Ali, M. and Kamarudin, M.R. and Rahman, T.A. (2011) 'Analytical Simulation and Performance Optimization for Spectrum

Sensing in Cognitive Radio Networks', *International Journal on Computer Science and Engineering (IJCSE)*, Vol. 3, No. 2.

Yang, C. and Sheng, M. and Li, Ji. and Li, H. and Li, J. (2013) 'Energy-aware joint power and rate control in overlay cognitive radio networks: a Nash bargaining perspective', *International Journal on Computer Science and Engineering (IJCSE)*, Vol. 3, No. 5.

Pietro, R. Di and Oligeri, G. (2013) 'Jamming mitigation in cognitive radio networks', *IEEE Network*, Vol. 27, No. 3, pp.10–15.

Asterjadhi, A. and Zorzi, M. (2010) 'JENNA: a jamming evasive network-coding neighbor-discovery algorithm for cognitive radio networks', *IEEE Wireless Communications*, Vol. 17, No. 4,pp.24–32.

Balogun, V. (2014) 'Anti-jamming Performance of Hybrid FEC code in the Presence of CRN Random Jammers', *International Journal of Novel Research in Engineering and Applied Sciences (IJNREAS)*, Vol. 1, No. 1.

Wang, W., Bhattacharjee, S., Chatterjee, M. and Kwiat, K. (2013) 'Collaborative jamming and collaborative defense in cognitive radio networks', *Pervasive and Mobile Computing*, Vol. 9, No. 4, pp.572–587.

Bhunia, S., Su, X., Sengupta, S. and Vázquez-Abad, F.J. (2014) Stochastic Model for Cognitive Radio Networks under Jamming Attacks and Honeypot-Based Prevention, *Distributed Computing and Networking - 15th International Conference (ICDCN), Coimbatore, India*, pp.438–452.

Wu, Y., Wang, B. and Ray Liu,K. J. (2010) Optimal Defense against Jamming Attacks in Cognitive Radio Networks Using the Markov Decision Process Approach, *GLOBECOM'10*, pp.1–5.

Chen, C., Song, M., Xin, C. and Backens, J. (2013) 'A game-theoretical anti-jamming scheme for cognitive radio networks', *IEEE Network*, Vol. 27, No. 3, pp.22-27.

Szepesvri, C. and L. Littman, M. (1996) 'Generalized Markov Decision Processes: Dynamic-programming and Reinforcement-learning Algorithms'.

Henrique, C. and Ribeiro, C. 'A Tutorial on Reinforcement Learning Techniques'.

Watkins, Christopher John Cornish Hellaby (1989) 'Learning from Delayed Rewards', *King's College, Cambridge, UK*.

Tesauro, G. (2003) 'Extending Q-Learning to General Adaptive Multi-Agent Systems', *NIPS*.

Sutton, R. S. and Barto, A. G. (1998) 'Introduction to Reinforcement Learning', *MIT Press, Cambridge, MA, USA*.

Abounadi, J., Bertsekas, D.P. and Borkar, V.S. (2002) 'Stochastic Approximation for Nonexpansive Maps: Application to Q-Learning Algorithms', *SIAM J. Control and Optimization*, Vol. 41, No. 1, pp.1–22.

Even-Dar, E. and Mansour, Y. (2003) 'Learning Rates for Q-learning', *Journal of Machine Learning Research*, Vol. 5, pp.1–25.

Arokkiaraj, A. and Jayasankar, T. (2014) 'OFDM Based Spectrum Sensing In Time Varying Channel', *International Refereed Journal of Engineering and Science (IRJES)*, Vol. 3, No. 4, pp.50–55.

Kyperountas, S., Correal, N. and Shi, Q. 'A comparison of Fusion Rules for Cooperative Spectrum Sensing in Fading Channels', *EMS Research, Motorola*.

F. Molisch, A., J. Greenstein, L. and Shafi, M (2009) 'Propagation Issues for Cognitive Radio', *Proceedings of the IEEE*, Vol. 97, No. 5.