Optimal Jamming using Delayed Learning

SaiDhiraj Amuru and R. Michael Buehrer
Bradley Department of Electrical and Computer Engineering
Virginia Tech, Blacksburg, VA USA
Email: {adhiraj, rbuehrer}@vt.edu

***Abstract*—Recent advances in cognitive radios for electronic warfare create the potential for dynamic environmental conditions, which makes it difficult to rely upon predict-then-adapt approaches in unfamiliar environments. It is thus imperative that radios have increasingly intelligent capabilities in order to be effective in harsh unknown surroundings. In this paper, we explore whether an intelligent jammer can learn and adapt to its surroundings in an electronic warfare-type scenario. We address this problem from a reinforcement learning perspective where the jammer has delayed information regarding the packets exchanged between a victim transmitter and the receiver. This is different from the traditional assumption that feedback is available instantaneously in reinforcement learning-based algorithms. A new framework, to enable delayed learning in scenarios where rewards are associated with state transitions rather than the states themselves is developed. The benefits of such a framework are shown by studying the optimal jamming strategies against an 802.11-type wireless network that uses the RTS-CTS protocol to communicate and deliver information.**

I. Introduction

The concept of cognitive radio [1] introduces the potential for radios to learn their surroundings. For example, spectrum sensing is used to learn the use of shared spectrum resources [2]. This knowledge enables radios to coexist along with other radios. Further, in most commercial and military applications, the underlying communication protocol of other radios is not known a priori, which makes it necessary for a radio to learn and adapt to the behavior of other radios rather than using a predict-then-adapt approach. In this paper, we explore whether a jammer can learn the behavior of other radios and develop an attack strategy in an electronic warfare-type scenario.

Reinforcement learning (RL)-based algorithms have traditionally been used to address such cognitive learning scenarios [3]. In RL, a radio (agent) learns the optimal strategy (for example, survival strategy in the presence of malicious nodes) by repeatedly interacting with the environment. During these interactions, the agent receives feedback indicating whether the actions performed were good or bad, using which it can improve its strategy to maximize rewards or minimize costs [3]. Note that rewards depend on the context and the problem being studied, for example throughput obtained or energy expended in communicating with other radios.

Most RL algorithms assume the existence of immediate observations based on which learning is performed. Here, observations correspond to either the state of the environment (most practical wireless environments can be modeled by a Markov state machine) or the rewards/costs obtained for the actions performed. In practical environments, feedback can potentially be delayed by several time instants [4]. For example, when a jammer disrupts a packet that is exchanged between the transmitter and receiver, it is not aware whether the jamming was successful or not until an acknowledgement packet is sent by the receiver. It may also not know what type of packet it jammed and hence it is not aware of the state of the system. Thus, there is delay in learning the efficacy of the actions performed and modifying the actions based on the obtained information.

Learning in scenarios where the observations are delayed was studied in [4]-[6]. Learning in cases where the impact of actions is delayed i.e., the effect of actions taken now is seen at a later time instant was considered in [5]. A framework to reformulate the learning scenario with delays to an equivalent learning problem without delays, which only differ in the size of the state space, was discussed in [4], [5]. These works also assume that rewards are associated with states of the environment which again does not always hold true in practical scenarios. This warrants the development of a new delayed learning framework that enables RL to identify the optimal strategies when rewards are associated with state transitions rather than the states themselves.

In this paper, we first develop a novel framework to address delayed learning with transition-based rewards in Section II. We then show the utility of this framework in Sections III and IV by obtaining the optimal power efficient jamming strategies that reduce the throughput of a wireless network. Protocol aware, energy efficient jamming strategies when the jammer has instantaneous knowledge of the packets exchanged, were studied via OPNET simulations in [7]; where the authors discussed the effectiveness of jamming the control packets when compared to naive periodic or continuous jamming. While the jammer can jam and receive data simultaneously, it is not always instantaneously aware of the environment state. For example, due to processing delays, the jammer cannot decode a packet until the packet transmission is finished. Hence, it is aware of the environment state only with a unit time delay. We thus explore in this paper, whether or not a jammer can learn its surroundings and disrupt a wireless network by using only the delayed knowledge, which is different from the assumptions made in [7] and references therein.

We show the benefits of the new delayed learning framework by considering the case of an 802.11-type wireless network, where the transmitter and receiver use the RTS-CTS protocol to communicate [8]. The jammer can jam either the source node or the destination node so as to disrupt the wireless network. We begin with the assumption that the jammer is aware of the protocol used by this transmit-receive pair and obtain the optimal jamming strategies. While we make several assumptions regarding the knowledge of the jammer, we show the effect of these assumptions in Section IV. Finally, conclusions are presented in Section V.

II. Learning Framework

Reinforcement learning is a technique that allows an agent to modify its actions (without any supervision) by repeatedly interacting with the environment. A reinforcement learning task that satisfies the Markov property is called a Markov

decision process, or MDP, [3]. A MDP is defined by a tuple $\{S, A, P, R\}$ where S is the set of all possible environment states and A is the set of all possible actions that the agent can perform in any environment state. For instance, from a jammer's perspective, the environment states could be Transmission/No Transmission to reflect the cases where a packet is exchanged between the transmit-receive pair or when they are idle, and the actions of the jammer could be Jam/Don't Jam. P is the state transition probability matrix that governs the dynamics of the environment, and its entries are given by the transition probabilities $p(s'|s, a)$ which indicates the probability that the environment moves to the state s' when action a is executed in the state s. Finally R indicates the $|S| * |A|$ reward matrix whose entries are given by elements $r(s, a)$ which indicate the reward (for example, energy expended) obtained in state s when action a is executed. Here, $|S|$ and $|A|$ indicate the cardinality of the sets S and A respectively.

In the traditional undelayed RL framework, an agent observes the current state of the environment s, and chooses an action a. An optimum policy (a functional mapping between states and the actions that can be performed in these states) is one that maximizes the total expected rewards, that is more often than not discounted by a factor $\gamma \in [0, 1)$ to account for an infinite time horizon. The objective of a RL algorithm is therefore to find a policy Π, that maximizes the cumulative discounted reward

$$R(t) = \sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}), \tag{1}$$

where s_t, a_t indicate the state and action taken at time t [9]. The value of a policy Π when the environment is in state s is given by

$$V^{\Pi}(s) = E_{\Pi}\left(\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}|s_t = s)\right), \tag{2}$$

where E_{Π} indicates the averaging performed over all possible state transitions when the agent follows the policy Π. Several algorithms exist to find the optimal policy Π^*. For more details please see [3]. For ease of analysis, we assume a stationary model (state transition matrix is independent of time) and ignore the time parameter t hereafter.

When the underlying MDP model is known, policy evaluation (finding the value of a given policy) can be done via matrix inversion [3]. Specifically,

$$\begin{aligned} V^{\Pi}(s) &= r(s, a = \Pi(s)) + E_{\Pi}\left(\sum_{k=1}^{\infty} \gamma^k r(s_k, a_k|s)\right) \\ &= r(s, a = \Pi(s)) + \gamma \sum_{s'} p(s'|s, a = \Pi(s)) V^{\Pi}(s'). \end{aligned}$$

Thus, writing the above set of equations for all possible states $s \in S$ in the MDP, we have

$$\bar{\mathbf{V}}^{\Pi} = \bar{\mathbf{r}}^{\Pi} + \gamma \mathbf{P}^{\Pi}(s'|s)\bar{\mathbf{V}}^{\Pi} \Longrightarrow \bar{\mathbf{V}}^{\Pi} = \left(\mathbf{I} - \gamma \mathbf{P}^{\Pi}(s'|s)\right)^{-1} \bar{\mathbf{r}}^{\Pi} \tag{3}$$

where $\bar{\mathbf{V}}^{\Pi}$ is the $|S| * 1$ vector of values of the policy Π in states $s \in S$, $\bar{\mathbf{r}}^{\Pi}$ is the $|S| * 1$ vector of rewards obtained in state $s \in S$ using policy Π and $\mathbf{P}^{\Pi}(s'|s)$ indicates the $|S| * |S|$ state transition probability matrix when the agent uses policy Π, and $\mathbf{I}$ is an identity matrix of appropriate dimensions.

A. Delayed Learning

As discussed earlier, feedback regarding the states and the rewards is not always available instantaneously. In [4], [5], [10] it is shown that MDPs with a constant observation delay (state and rewards) can be modeled using an augmented MDP approach. This involves formulating an augmented MDP that is equivalent to the original MDP, but with a larger state space that takes the observation delay into consideration. A MDP with observation delays is represented by the tuple $\{S, A, P, R, k\}$, where k is the observation delay. The equivalent augmented MDP is represented as $\{I_k, A, P', R'\}$, where $I_k = S \times A^k$ is the augmented state space (A^k is the Cartesian product of A with itself k times and $S \times A^k$ indicates the Cartesian product of S with A^k). The state at time t i.e., $I_k^t \in I_k$ is represented as $(s_{t-k}, a_{t-k}, a_{t-k+1}, \ldots, a_{t-1})$. In the augmented state I_k^t, s_{t-k} is the most recently observed state, and $a_{t-k}, \ldots, a_{t-1}$ are the actions taken in the last k time intervals. The state transition probability is defined as

$$P'(I_k^{t+1}|I_k^t) = p(s_{t-k+1}|s_{t-k}, a_{t-k}), \tag{4}$$

where $I_k^{t+1} = (s_{t-k+1}, a_{t-k+1}, \ldots, a_t)$ and rewards are defined as $R'(I_k^t) = r(s_{t-k}, a_{t-k})$. Notice that since the state s_{t-k+1} is not known with certainty, the immediate reward obtained by the transition from the state I_k^t to I_k^{t+1} is given by $E_{\Pi}(R'(I_k^{t+1}|I_k^t)) = E_{\Pi}(R(s_{t-k}, a_{t-k}|I_k^t))$, which in other words corresponds to the expected (with respect to policy Π) immediate reward. Using these definitions, the value of the policy Π in an augmented state I_k^t is given by

$$V^{\Pi}(I_k^t) = E_{\Pi}\left(\sum_{l=1}^{\infty} \gamma^{l-1} R'(I_k^{t+l}|I_k^t)\right), \tag{5}$$

which can be rewritten as

$$\begin{aligned} V^{\Pi}(I_k^t) = \sum_{s_{t-k+1}} & p^{\Pi}(s_{t-k+1}|s_{t-k}, a_{t-k}) r^{\Pi}(s_{t-k+1}, a_{t-k+1}) \\ & + \gamma \sum_{I_k^{t+1}} P'^{\Pi}(I_k^{t+1}|I_k^t) V^{\Pi}(I_k^{t+l}), \end{aligned} \tag{6}$$

where p^{Π}, P'^{Π} and r^{Π} are the transition probabilities and rewards obtained when policy Π is used i.e., $a_{t-k} = \Pi(s_{t-k})$, $a_{t-k+1} = \Pi(s_{t-k+1})$. The matrix inversion technique in (3) can be extended to this augmented model to evaluate the value of a policy Π. Please see [4], [10] for more details.

B. A Novel Delayed Learning Framework with Transition-based Rewards

Notice that the RL algorithms discussed earlier assume state-based rewards as seen in (3) and (6). As mentioned earlier, this assumption does not always hold true in practical settings. For example, this can occur in scenarios where the MDP has a terminal state (say the QUIT state) which can be reached from several other states (say the GOOD state and the BAD state), but rewards are obtained only for some of these state transitions (for example, when the QUIT state is reached from the GOOD state only). A jamming scenario where transition-based rewards occur is discussed in Section III.

We modify the delayed learning framework in [4], [10], to address scenarios with transition-based rewards. Specifically, the value of a policy Π is given by,

$$V^{\Pi}(I_k^t) = \sum_{I_k^{t+1}} P'^{\Pi}(I_k^t, I_k^{t+1}) R'^{\Pi}(I_k^t, I_k^{t+1}) + \gamma \sum_{I_k^{t+1}} P'^{\Pi}(I_k^t, I_k^{t+1}) V^{\Pi}(I_k^{t+l}), \quad (7)$$

where $R'^{\Pi}(I_k^t, I_k^{t+1})$ indicates the reward obtained due to the state transition from the augmented state I_k^t to the augmented state I_k^{t+1}. The value function can be compactly written as

$$\bar{\mathbf{V}}^{\Pi} = \mathrm{diag}\left(\bar{\mathbf{P}}'^{\Pi}\bar{\mathbf{R}}'^{\Pi}\right) + \gamma\bar{\mathbf{P}}'^{\Pi}\bar{\mathbf{V}}^{\Pi}$$
$$\implies \bar{\mathbf{V}}^{\Pi} = \left(\mathbf{I} - \gamma\bar{\mathbf{P}}'^{\Pi}\right)^{-1} \mathrm{diag}\left(\bar{\mathbf{P}}'^{\Pi}\bar{\mathbf{R}}'^{\Pi}\right), \quad (8)$$

where $\mathbf{I}$ is an identity matrix, $\bar{\mathbf{P}}'^{\Pi}$ is the $|I_k| * |I_k|$ augmented state transition probability matrix ($|I_k|=|S| * |A|^k$ is the cardinality of the augmented set I_k), $\bar{\mathbf{R}}'^{\Pi}$ is the $|I_k| * |I_k|$ reward matrix indicating the rewards obtained for various state transitions and $\mathrm{diag}(\mathbf{X})$ indicates the leading diagonal of a matrix $\mathbf{X}$. When k is small, matrix inversion can be performed fairly easily which allows one to evaluate the value of a policy Π with transition-based rewards.

Notice that the value function V^{Π} in (8) can be shown to converge to the optimal policy Π^* and that it follows Bellman's optimality rules [3], similar to the value functions given in (3) and (6). We skip this proof due to a lack of space. Also it is important to notice that this delayed learning framework works well only with an infinite horizon since all rewards/costs will not be collected in a finite horizon [5]. This framework also allows one to capture the effect of uncertain transition probability matrices i.e., in cases where the estimates of the state transition probability P might be erroneous, which thereby leads to errors in the estimates of the value function of a policy Π. For more details on the loss bounds in the presence of uncertain transition probability matrices, please see [11]. We next demonstrate the utility of this framework with an example that considers jamming against an 802.11-type network.

III. Jamming via Delayed Learning

We consider a MAC layer jamming attack scenario where the jammer disrupts the communication between a transmitter-receiver pair. The naive way of jamming a network is to continuously transmit high power noise over the bandwidth of interest that either jams the transmitter (source node) or receiver (destination node) or both. However, this entails significant power consumption on the jammers' side which may not be available. As discussed in [7], jamming can be made energy efficient if the jammer operates based on the knowledge of the protocol used by the wireless network that it intends to jam. For example, jamming the control packets and/or the pilot symbols proves to be very effective in incapacitating the network [7].

Several metrics need to be considered when studying jamming against a wireless network. In this paper, we restrict ourselves to the energy expended in jamming and the wireless network throughput allowed as the driving metrics for obtaining the optimal jamming strategies. It is assumed that the transmitter and receiver are not aware of the presence of a jammer in their vicinity (and hence not intelligent) and assume that the packets are not received only due to bad wireless channel conditions.

A. Protocol Description

To study jamming with delayed knowledge, we consider an 802.11-type wireless network where the transmitter and receiver use the RTS-CTS handshake mechanism to communicate with each other [8]. The MDP model for the environment is based on the messages exchanged i.e., we define the environment state at time t based on the packet exchanged between the transmitter and receiver at time t. Table I shows the possible MDP state transitions which are explained in detail below.

TABLE I. MDP model state transitions

Present State	State Transition Condition	Next State
RTS	Packet Successful	CTS
	Packet Unsuccessful	WAIT
CTS	Packet Successful	DATA
	Packet Unsuccessful	WAIT
DATA	Packet Successful	ACK
	Packet Unsuccessful	DATA
	Retransmission limit reached	WAIT
ACK	Packet Successful	WAIT
	Packet Unsuccessful	DATA
	Retransmission limit reached	WAIT
WAIT	$\rightarrow$ Contention Window > 0	WAIT
	$\rightarrow$ Contention Window=0 and Data not available	WAIT
	$\rightarrow$ Contention Window=0 and Data available	RTS

When data is present for transmission, the transmitter sends a request-to-send (RTS) message to request permission from the receiver to transmit the information. The state of the environment is taken to be RTS at this stage. Upon correctly decoding the RTS, the receiver sends a clear-to-send (CTS) message to give permission to the transmitter to proceed with the data transmission; here the environment state is CTS. If the receiver does not respond to the RTS message (when the RTS packet was not received/ decoded correctly by the receiver) within a given time interval (until which the environment state remains in RTS), the environment enters the WAIT state.

Once the transmitter correctly receives the CTS message, it starts with the DATA transmission, and the environment makes a transition to the DATA state. The transmitter sends a DATA packet to the receiver, which if received will trigger an acknowledgement (ACK) or a no acknowledgement (NACK) packet. For ease of analysis, we ignore the NACK packet in this analysis and assume that a packet is not received/decoded correctly only when jamming is successful. If the transmitter does not receive the ACK packet within a specified time interval, it retransmits the data packets. During the data retransmission, the state of the environment continues to be DATA. This is done until the retransmission limit is reached, which is prespecified by the protocol. When this limit is reached, the protocol enters a WAIT state where the transmitter ignores the current data packet and starts a new data transmission by restarting the protocol with a new RTS message. A similar sequence of events holds true for the ACK state as well. The only difference here is that the DATA state is due to the transmitter and the ACK state is due to the receiver. But as mentioned before, irrespective of whether the transmitter/ receiver generates a packet, the state of the environment is only based on the packet being exchanged at a given time instant.

When the ACK is successful, the environment enters the WAIT state where a new packet can now be transmitted by

sending a RTS message. Every time the protocol enters the WAIT state, it choses a certain contention window size and then waits for an exponential number of time slots before it sends a new RTS message. For example, consider the case where the possible contention window sizes range between $[CW_{\min}, CW_{\max}]$. If the protocol enters the wait state and the contention window size chosen is CW, the protocol stays in the WAIT state for a random time interval that is uniformly chosen in the interval $[0, 2^{CW} - 1]$, where 0 indicates that the transition to the RTS state is instantaneous. Note that the contention window size doubles (until it reaches $CW_{\max}$) with every unsuccessful packet transmission.

It is assumed that the sizes of the RTS, CTS and ACK packets are the same and that the DATA packet is 10 times longer than these packets. For ease of analysis, we model this by taking into consideration that the energy expended in jamming the DATA packet is 10 times that of the other packets and that the DATA state lasts only for one time slot. However, it is easy to extend this model to the case where each DATA state actually lasts for 10 time slots.

B. Jamming Strategies

We initially assume that the jammer is aware of the protocol used by the transmitter and receiver and thus knows the MDP state transition structure, retransmission limit, and also the contention window sizes. For instance, the contention window sizes can be used to estimate the average time period over which the environment stays in the WAIT state after it enters the WAIT state. The jammer intends to learn the optimal power efficient jamming strategy against this known MDP model. We discuss the effects of not knowing the MDP model in Section IV where the jammer learns the model and the strategies by repeated interactions with the environment.

The jammer can disrupt various packets that are exchanged in the network i.e., it can jam packets that are received either at the transmitter (CTS, ACK) or at the receiver (RTS, DATA). When the jammer believes that the environment state is WAIT, it does not jam[1]. Thus, there are 4 other possible environment states that the jammer can attack; {RTS, CTS, DATA, ACK}. The MDP model with jamming is shown in Fig. 1 which reflects the belief of the jammer about the state transitions of the environment. For example, when the jammer believes that the environment state is DATA, it jams with energy $10E$ and in all other states it jams with energy E (the decision to jam in a state or not depends on the jamming policy used). With two possible actions for the jammer i.e., {Jam, Don't Jam}, a total of 16 different policies exist. Among these, the jammer has to learn the optimal policy based on 1) the environment states and 2) the costs incurred for jamming and the costs incurred for allowing a data packet to go through un-jammed, which is denoted by T (i.e., the cost for allowing throughput).

Since we only consider the MAC layer jamming scenarios, the most relevant environment condition is the jamming success probability ρ (which depends on various physical layer parameters, which are not the concern of this paper). Initially, we assume that the jammer has knowledge of this parameter ρ and later we show the effects of not knowing ρ or the MDP

[1]Jamming the WAIT state to mimic a busy wireless channel is more relevant when several transmit-receive pairs exist in the environment.

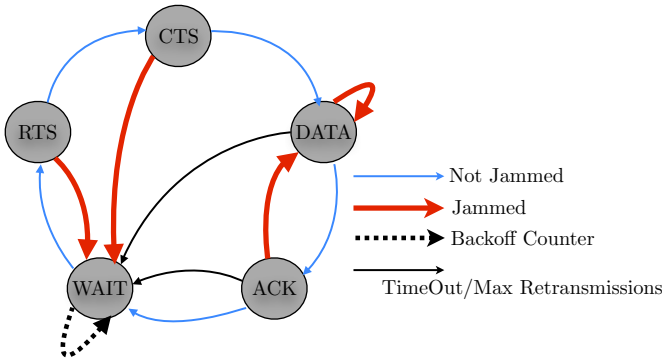

Fig. 1. MDP model of the 802.11-type wireless network with the RTS-CTS protocol. The state transitions indicate the effect of a jamming attack on the wireless network.

model that governs the environment. While the jammer can jam and receive data simultaneously, it is not aware of the environment state until the packet transmission is finished. Hence, it knows the state with a one time slot delay, which means that $k = 1$ in the delayed learning framework proposed in Section II. For instance, when the RTS packet is sent by the transmitter, the jammer understands that the RTS transmission has occurred in the previous state (once the packet transmission is finished) and it knows if it has not jammed this packet, the next transmission would be CTS i.e., $p(\text{CTS}|\text{RTS, Don't Jam}) = 1$. Based on this knowledge, the jammer decides to jam or not in the next time instant depending on the policy being used and eventually learns the optimal jamming strategy.

C. Feedback Signals

The jammer requires feedback to learn whether its strategies are optimal or not. The jammer is instantaneously aware of the energy expended when it jams (or when it makes a decision to jam). However, there is no instantaneous knowledge regarding the throughput allowed as it depends on the jammers' interaction with the environment. Since the jammer can observe the states of the environment, albeit in a delayed manner, a cost on the throughput can be taken into account when the jammer understands that the environment has made a transition from the ACK state to WAIT state and if it did not jam the ACK state. The jammer takes a probabilistic reward i.e., $(1 - \rho)T$ into consideration when the last ACK retransmission enters into WAIT state. This is because the jammer is not aware whether this state transition happened due to successful jamming (and the transmitter-receiver pair ignore the current packet) or an unsuccessful jamming attempt. In both these cases, the environment makes a transition into the WAIT state, $p(\text{ACK}|\text{WAIT}) = 1$, when the last (last in terms of the retransmission limit) ACK packet is sent.

From the above discussion it is clear that the feedback on the throughput cost is based on the state transition from ACK to the WAIT state and not on the states themselves, which is different from the RL problems usually studied. Thus the delayed learning framework with transition-based rewards is more relevant to this scenario than the traditional framework. We next present the optimal jamming strategies obtained in such delayed learning settings.

IV. Numerical Results

We study the MAC layer jamming problem by simulating the Markovian state transitions shown in Table. I. The jammer

acts on this environment to learn the optimal strategy based on its belief about the MDP model as shown in Fig. 1. The reward on the energy expended is taken to be $E = -10$ and throughput is taken to be $T = -100$, unless otherwise specified. Since both the energy and the throughput allowed are costs, they are taken to be negative rewards and thus the jammer attempts to maximize the cumulative reward. The retransmission limit is taken to be 3 and the contention window sizes are $CW_{\min} = 2$ and $CW_{\max} = 4$. In all simulation-based results, we use an explore-exploit strategy with an exponentially decreasing exploration probability [3]. In the exploration phase, the jammer chooses a random policy (from the 16 viable policies) and in the exploitation phase, it chooses the policy with the maximum cumulative reward obtained thus far. Upon choosing a policy, the jammer uses this policy to interact with the environment over 1000 time instants (one episode) and accumulates the rewards. The simulations run for 3000 episodes where the jammer has a constant policy in each episode and eventually learns the optimal policy.

A. *Learning the optimal policy: MDP model and ρ known*

Fig. 2 compares the learning performance (via the explore-exploit strategy) when the jammer has instantaneous knowledge of the state transitions and when this knowledge is delayed by one time instant. It is seen that there is a small loss in the rewards obtained due to this delayed knowledge and that the optimal policy obtained via delayed learning performs significantly better than the naive policies. Specifically, Fig. 2 also shows the rewards obtained when naive policies such as random jamming, jamming all the states and jamming none of the states are used. Table II shows the optimal jamming policies (as a function of ρ) learned by using the explore-exploit strategy when the observations are delayed.

When ρ is known, we can also find the theoretical optimal jamming policy by using the delayed learning framework in Section II. To use this framework, we need to form the matrices P'^{Π} and the reward matrix R'^{Π}, which can be done because the jammer has the knowledge of the protocol and ρ. For example, when the policy chosen is to jam CTS only, then we have p(CTS,Jam|RTS, Don't Jam)=1; p(CTS,Don't Jam|RTS, Don't Jam)=0; p(DATA,Jam|CTS,Jam)=0; p(DATA,Don't Jam|CTS,Jam)=$1-\rho$; p(WAIT,Jam|CTS,Jam)=0; p(WAIT,Don't Jam|CTS,Jam)=ρ and so on. Along these lines, the reward matrix $R(I, I')$ can also be obtained; for example, $R(\{$ACK,Don't Jam$\},\{$WAIT, Don't Jam$\})=-T$; $R(\{$RTS,Don't Jam$\},\{$CTS, Jam$\})=0$ and so on. Table II shows that the optimal policies obtained via the explore-exploit strategy match the optimal policies obtained by using the delayed learning theoretical framework.

B. *Intuition to the optimal policy*

For ease of illustration assume that $\rho = 1$, and consider the case where the transition from the WAIT state to the RTS state is instantaneous i.e., the wait time is 0. In such a model, if the jammer jams the RTS state alone, the environment enters the WAIT state and the transitions to CTS, DATA and ACK states will not occur. Thus, the throughput allowed is 0. Over a time period of 100 time slots, the RTS state is seen 50 times when this policy is used and thus the cost incurred is $50E$. However, if the jammer jams the CTS state alone, then it is easy to see that the CTS state is seen only 25 times i.e., the cost incurred is $25E$ and the throughput allowed is still 0. Jamming the DATA state or the ACK state alone is worse when compared to jamming the CTS state only due to 1) the cost incurred for jamming the DATA state is $10E$ and 2) due to the retransmission limit, whenever the DATA and ACK state is seen, the jammer also jams these states which increases the cost. Thus in this case, jamming the CTS state alone is the optimal jamming strategy. As ρ decreases, the optimal number of states to jam increases until a certain value of ρ and decreases again until it reaches a point where jamming none of the states is optimal. This behavior is also seen in Table II.

TABLE II. OPTIMAL JAMMING POLICIES VIA DELAYED LEARNING, $E = -10, T = -100$

ρ	Optimal Policy (Simulation)	Optimal Policy (Theory)
1	Jam CTS	Jam CTS
$[0.4, 1)$	Jam CTS & ACK	Jam CTS & ACK
$[0.2, 0.4)$	Jam CTS	Jam CTS
$[0, 0.2)$	Jam NONE	Jam NONE

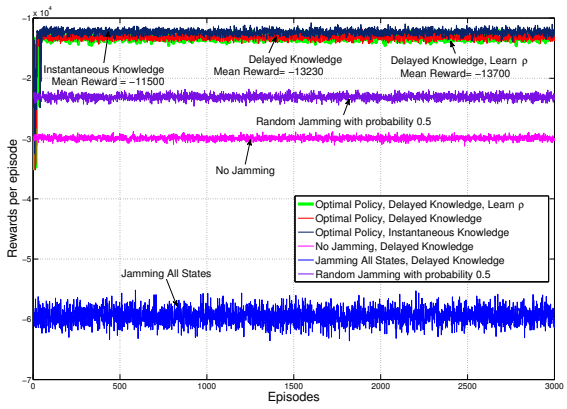

Fig. 2. Rewards obtained in various scenarios, $\rho = 0.3$. The rewards obtained with instantaneous knowledge are on average better than the rewards obtained in the delayed knowledge scenarios.

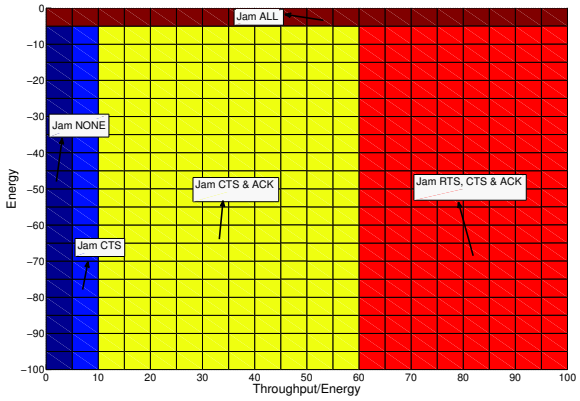

Fig. 3. Optimal jamming policies as a function of the energy and throughput costs; $\rho = 0.5$. The colors represent the various optimal jamming policies.

Fig. 3 shows the jamming policies as a function of the energy and the throughput costs, where it is seen that the optimal policies depend not only on ρ but also on the costs.

Notice that when $E = 0$ (no cost incurred due to jamming), irrespective of the value of T, it is optimal to jam all the states in order to avoid any throughput cost. Also note that as the cost of the throughput increases, it is optimal to jam more states in order to reduce the average throughput. When the observation delay is small and ρ is known, the framework in Section II can be used to find the optimal policy. Owing to matrix-inversion, this technique will be computationally complex as the value of k increases. The proposed RL framework can then be used to determine the optimal jamming policies.

C. Learning ρ and the optimal policy: MDP model known

Thus far, we have considered the case where the jammer has perfect knowledge regarding various parameters of the environment. Fig. 2 also shows the rewards obtained when the jammer must learn the jamming success probability ρ. Here, the jammer jams all the states in one episode and observes the state transitions in order to learn ρ. For example, this can be achieved by evaluating

$$\rho = \frac{N_{\{\text{RTS,Jam}\}\rightarrow\text{WAIT}}}{N_{\text{RTS}}}, \tag{9}$$

where N_{RTS} indicates the total number of times the state of the environment is RTS and $N_{\{\text{RTS,Jam}\}\rightarrow\text{WAIT}}$ is the number of times that environment made a transition to the WAIT state when the jammer jams the RTS state. Note that the RTS state should happen frequently which allows reliable estimation of ρ. Upon learning ρ, the jammer can then perform exploration/exploitation over the set of policies to learn the optimal policy. From Fig. 2, it is seen that the rewards obtained in this case are fairly close to the rewards obtained when the jammer has perfect knowledge about ρ. Depending on the accuracy of the estimate of ρ, the rewards obtained will vary as the optimal policy chosen is dependent on ρ as well.

D. Learning the MDP model, ρ and the optimal policy

We now consider a case where the jammer is not aware of the retransmission limit, the contention window sizes and the jamming success probability ρ. In such cases, it estimates the state transition probabilities by interacting with the environment. For example, the probability of transitioning to the DATA state when the environment is already in the DATA state helps learn the retransmission limit. Similar reasoning holds true for the contention window size when it sees the WAIT state i.e., it can estimate the average time period spent in the WAIT state. Fig. 4 shows the rewards obtained when the jammer is unaware of the underlying MDP model and learns the model by interacting with the environment. It is seen that the rewards obtained when the jammer learns the MDP model and ρ are lower when compared to the rewards obtained when it has perfect knowledge about the model and ρ (observations are still delayed) and higher when compared to naive jamming strategies.

V. Conclusion

In this paper, we explored whether a jammer can learn its surroundings in an electronic warfare-type scenario. We addressed the problem from a RL perspective wherein the jammer has delayed information regarding the environment which is different from the assumptions made in the previous literature. For this, we proposed a novel delayed learning framework with transition-based rewards, that allows us to handle the realistic case of delayed knowledge. This framework models the original MDP as an equivalent augmented MDP by increasing the state space to incorporate the delays. The benefits of such a framework were illustrated via a jamming scenario where we obtained the optimal jamming policy of a power efficient jammer that minimizes the combined throughput allowed and energy expended. Specifically, we modeled an 802.11-type wireless network and the associated states as a MDP and the jammer as an agent that acts over this MDP to learn the optimal actions. It was seen that the optimal policies depend on the costs incurred for jamming, throughput allowed in the network and also the jamming success probability.

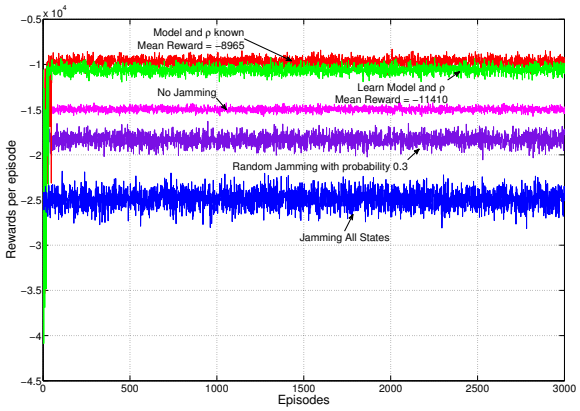

Fig. 4. Rewards obtained when jammer is uncertain about the underlying MDP model and ρ and learns it by interacting with the environment; $\rho = 0.5$.

References

[1] J. Mitola, III, "Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio," PhD thesis, KTH, Stockholm, Sweden, May 2000.

[2] S. M. Dudley *et al*., "Practical Issues for Spectrum Management With Cognitive Radios," *IEEE Proc.*, vol. 102, no. 3, pp. 242-264, Mar. 2014.

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

[4] T. J. Walsh, A. Nouri, L. Li, and M. L. Littman, "Learning and planning in environments with delayed feedback," *Auton. Agent. Multi-Agent Syst.,* vol. 18, no. 1, pp. 83-105, Aug. 2009.

[5] K. V. Katsikopoulos and S. E. Engelbrecht, "Markov Decision Processes With Delays and Asynchronous Cost Collection," *IEEE Trans. Autom. Control,* vol. 48, no. 4, pp. 568-574, Apr. 2003.

[6] P. Joulani, A. György, and C. Szepesvári, "Online Learning under Delayed Feedback," in *Proc. Int. Conf. Mach. Learning*, Atlanta, GA, Jun. 2013.

[7] D. J. Thuente and M. Acharya, "Intelligent Jamming in Wireless Networks with Applications to 802.11b and Other Networks," in *Proc. IEEE MILCOM*, Washington DC, USA, Oct. 2006, pp. 1075-1081.

[8] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 3, pp. 535-547, Mar. 2000.

[9] M. Bkassiny, Y. Li, and S. Jayaweera, "A Survey on Machine-Learning Techniques in Cognitive Radios," *IEEE Commun. Surveys and Tutorials*, vol. 15, no. 3, pp. 1136-1159, Jul. 2013.

[10] D. M. Brooks and C. T. Leondes, "Markov Decision Processes with State-Information Lag," *Opns. Res.,* vol. 20, no. 4, pp. 904-907, Aug. 1972.

[11] A. Mastin and P. Jaillet, "Loss Bounds for Uncertain Transition Probabilities in Markov Decision Processes," in *Proc. Conf. Dec. and Control.*, Maui, Hawaii, Dec. 2012, pp. 6708-6715.