COGNITIVE ENGINE TESTBED FOR VEHICULAR COMMUNICATIONS

William Anderson (andersonwr@emailwofford.edu)\textsuperscript{1}, Ashley Aponik (ashley.aponik@yale.edu)\textsuperscript{2}, Youssef Daoud (ytndaoud42@students.tntech.edu)\textsuperscript{3}, Noel Teku (ntekul@email.arizona.edu)\textsuperscript{4}, Garrett Vanhoy (gvanhoy@arizona.edu)\textsuperscript{4}, and Tamal Bose (tbose@arizona.edu)\textsuperscript{4}

\textsuperscript{1}Wofford College, Spartanburg, South Carolina, US
\textsuperscript{2}Yale University, New Haven, Connecticut, US
\textsuperscript{3}Tennessee Technological University, Cookeville, Tennessee, US
\textsuperscript{4}University of Arizona, Tucson, Arizona, US

ABSTRACT

Ad-hoc networks have the potential to increase the safety and reliability of autonomous vehicles. The amount of radio spectrum available for such networks is limited, however. The use of cognitive radio, especially when integrated with reinforcement learning algorithms, may help to ease the issue of limited spectrum by finding optimal transmission policies and detecting the presence of other users, especially in a scenario where a Primary User and Secondary User are contesting for spectrum. This paper presents a testbed for simulating cognitive engines in these networks using a variety of reinforcement learning algorithms, including \( \epsilon \)-greedy, Softmax Strategy, and Q-Learning. The goal of these cognitive engines is to learn to choose the best modulation and coding rates given various channel conditions and a user-defined optimization goal (i.e. maximize throughput, minimize bit error rate). The cognitive engine then learns the optimal coding rates and modulation schemes for the given environment, and the testbed displays a visual of the performance of the cognitive engine at each step. In the case of Q-learning in a Primary User vs. Secondary User scenario, the cognitive engine also learns to choose the best channel for transmission. Because of these challenges in VANET communications, it is important to optimize the use of channel resources. The use of cognitive radio may help to ease the issue of limited spectrum. A cognitive radio is capable of sensing its environment and adjusting its decision-making process accordingly. This results in it being capable of finding optimal transmission policies and detecting the presence of other users. The cognitive engine (CE) can be described as the helper for the radio to recognize the best transmission parameters (i.e. modulation, coding rate) in a specific environment \cite{4}. Reinforcement learning, a type of machine learning that selects a model’s actions based on its environment, provides useful algorithms for decision-making processes within a cognitive engine \cite{5}.

Spectrum sensing is a fundamental part of cognitive radios, especially in a scenario where a Primary User (PU) and Secondary User (SU) are contesting for spectrum. The PU has a license to use a specific channel or band while the SU does not and should therefore yield to the PU whenever he/she is transmitting on the band. Spectrum sensing is a technique utilized by the SU to determine whether a PU is broadcasting or not. Energy detection, matched filter detection, and cyclostationary feature are the most common techniques for spectrum sensing. The benefits of energy detection are that it does not require any prior information about the PU and that it is easy to implement. The knowledge provided by the spectrum sensing techniques is important to the cognitive engine’s decision process as it seeks to avoid collisions with the PU.

Previous VANET simulations were created to reduce the interference between the nodes by adding supplementary channels \cite{6}. Other papers proposed using the device-to-device (D2D) scenario for the 5G Technology in the Long Term Evolution (LTE) band in vehicular communications \cite{7}. In \cite{3}, a Fuzzy distributive algorithm was developed for spectrum sensing and shown to have a high performance. Other simulations used a three-state occupancy model for spectrum sensing and balancing communications between vehicular nodes \cite{8}.

The rest of the paper is organized as follows: Section 2 presents background information on reinforcement learning algorithms. Section 3 discusses the implementation of the cogni-
tive engines. Section 4 explains the setup of the testbed, and Section 5 shows results from simulations using the testbed. Section 6 provides ideas for further work and some concluding remarks.

2. BACKGROUND

Reinforcement learning is one method for informing and making decisions within a cognitive engine. Reinforcement learning is a type of machine learning in which an agent is capable of taking different actions to interact with its surrounding environment. After taking an action, the agent receives a reward signal from the environment along with information about the current state of the environment. The reward is a measure of how good or successful a chosen action was. Over time, the reinforcement learning algorithm develops correlations between the actions taken in specific states of the environment and the reward received, and so an optimal action policy is learned [5]. In the context of cognitive radios, the agent is the radio, its choice of actions are different broadcast parameters, and the environment is the channel or medium through which the radio is conducting its transmissions. The reward signal at each timestep is based on the goal of the cognitive engine. In the case of maximizing throughput, the reward may be the total throughput of that time step. In the case of minimizing bit error rate (BER), the reward may be the negative of the BER of that timestep (i.e. the action cost).

The issue of balancing exploration with exploitation is also an important consideration of reinforcement learning. When the algorithm exploits, it chooses to take the action with the current highest value in the current state. This value is the expected, or average, reward of an action in a given state. When the algorithm explores, it may choose an action other than current known best. While one action may have the current highest value, it may be revealed through more trials that a different action has the actual highest. Exploration allows these possibly higher-valued actions to be found. Reinforcement learning algorithms seek to balance exploitation and exploration in different ways.

For our testbed we implemented three reinforcement learning algorithms: $\epsilon$-greedy, Softmax Strategy, and Q-Learning. The $\epsilon$-greedy algorithm balances exploitation and exploration through the use of an $\epsilon$ value, which represents the chance that the engine will explore during each time step [5]. The Softmax algorithm handles exploration through weighted probabilities. These probabilities are based on the relative known values of each action. The probability $P$ of a given action $k$ is given by [9][10]:

$$P_k = \frac{e^{V_k/T}}{\sum_i e^{V_i/T}}$$  (1)

where $T$ is a temperature value and $V$ is the current value of an action. $T$ determines the randomness with which different actions are taken; lower temperatures decrease this randomness, while higher values increase it.

Both $\epsilon$-greedy and Softmax offer annealing versions in which $\epsilon$ and $T$, respectively, are decreased over time. This allows for increased exploitation after the algorithm has had time to explore [5]. The two annealing equations used [10] are

$$\epsilon = \frac{\epsilon_0}{1 + nd\epsilon}$$  (2)

$$T = \frac{T_0}{1 + ndT}$$  (3)

where $\epsilon_0$ and $T_0$ are the initial $\epsilon$ and temperature values, respectively, and $n$ is the number of time steps, and $d$ is the decrease rate.

While the average reward can be used as the value of a state, Q-learning offers an alternative update that takes into account possible future rewards. Q-learning does this by incorporating the maximum of the expected rewards of all actions in the next state, representing a best-case scenario for future rewards. The expected rewards, or Q-values, are stored in a lookup table known as a Q-table. These Q-values are updated over time as exploration and exploitation occur. The Q-learning function is given by [5]:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$  (4)

In the above equation, $Q$ is the value function, $S_t$ is the state in time step $t$, $A_t$ is the action taken in time step $t$, $\alpha$ is a step-size for the update, $R_{t+1}$ is the observed reward, and $\gamma$ is a discount rate. In our testbed, the action represents a modulation-coding rate pair and the state represents the channel. The discount rate adjusts the weight of the importance of future rewards. As Q-learning only pertains to updating known values, it should be combined with an action-selection algorithm such as $\epsilon$-greedy or Softmax in order to explore/exploit within different states.

Dynamic Spectrum Access (DSA) algorithms are used in cognitive radio networks to enable secondary users to efficiently access spectrum with minimal interference to primary users [2]. Several methods for spectrum access already exist, including the use of graph theory, game theory, fuzzy logic, and evolutionary algorithms, among others [11]. We propose another method for determining spectrum access using Q-learning, detailed in Section 3.

3. COGNITIVE ENGINE IMPLEMENTATION

All algorithms have been written in the Python programming language. $\epsilon$-greedy and Q-learning are based on the algorithms provided in [5], and Softmax is based on the implementation in [9]. Annealing functions are based on the equations in [10].

Along with the testbed we also provide an addition to the Q-learning algorithm intended to allow cognitive engines to take channel selection into account:
$Q_S(S_t, S_{t+1}) \leftarrow Q_S(S_t, S_{t+1}) + \\
\alpha [R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q_S(S_t, S_{t+1})] \quad (5)$

$Q_S$ represents a value function corresponding to the overall expected value of being in a particular state, rather than the expected value of individual actions. With this second set of Q-values, $\epsilon$-greedy or Softmax can be used to choose a new state to enter. In the context of our testbed, this allows the cognitive engine to not only choose broadcast parameters within a channel but also the channel to broadcast on.

4. TESTBED SETUP

The testbed was designed for two scenarios in vehicle to infrastructure communication: transmission of a single vehicle and transmission in a PU vs. SU scenario. For this iteration of the testbed, we assume stationary vehicles.

4.1. Single Vehicle

In the single vehicle scenario, the testbed allows the user to choose several parameters. One such parameter is the channel model, including AWGN, flat Rayleigh fading, and flat Rician fading. The AWGN channel model adds white Gaussian noise to the transmitted signal [12]. Flat Rayleigh fading assumes no line of sight paths from transmitter to receiver [13]. Flat Rician fading takes as an input $k$ for the number of line of sight paths [14]. For our Rayleigh and Rician fading models we assume perfect channel state information. Additionally, the user is allowed to choose the optimization goal–maximizing throughput or minimizing BER–as well as the cognitive engine used to accomplish this goal.

When the testbed is started, the cognitive engine learns the optimal modulation and coding rate pair for transmission based on the current channel conditions and the optimization goal. The engine has the choice of the following modulations: BPSK, QPSK, 16QAM, 32QAM, and 64QAM [14]. It also chooses between coding rates of 1, $\frac{3}{4}$, $\frac{1}{2}$, and $\frac{1}{3}$. The modulation-coding rate pair selected are the actions chosen by the algorithm, which returns a reward based on the various parameters and channel model selected. The symbol error rate (SER) is then calculated based on the channel model and $M$-ary modulation scheme. From the SER, we approximate the bit error rate (BER). Gray-coding is assumed so the conversion is achieved using the formula [15]:

$$P_b = \frac{P_s}{\log_2(M)} \quad (6)$$

where $P_b$ is the bit error probability, $P_s$ is the symbol error probability, and $M$ is the number of bits per symbol based on the modulation chosen. The final BER is found by multiplying $P_b$ by the chosen coding rate $R$. We calculate the throughput, defined as the number of bits per second received successfully, with the formula:

$$THR = R \cdot \log_2(M) \cdot (1 - P_b) \cdot BW \quad (7)$$

where $R$ is the coding rate chosen by the engine and $BW$ is the bandwidth. In the testbed we assumed a 6 MHz bandwidth based on the available spectrum in the Analog TV bands, which, as shown in [16], has been looked at as a viable solution for spectrum sharing in VANET technologies. At each time step, the testbed displays the vehicle, the receiver, the modulation-coding rate pair chosen by the cognitive engine, and the returned reward.

4.2. PU vs. SU

In the PU vs. SU scenario, the user is allowed to choose the optimization goal–maximizing throughput or minimizing BER–as well as the cognitive engine of the SU, similar to the single vehicle case. In contrast to the single user scenario, the user does not choose the channel model. Instead, the user chooses the characteristics of the primary user (probability of broadcasting, length in time steps of broadcast) as well as the number of available channels (1-6). Each channel is defined as either AWGN, flat Rayleigh fading, or flat Rician fading, with a fixed SNR value as shown in Table 1.

<table>
<thead>
<tr>
<th>Channel Number</th>
<th>Channel Model</th>
<th>SNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AWGN</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>Rayleigh</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>Rician (k=2)</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>AWGN</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>Rayleigh</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>Rician (k=2)</td>
<td>6</td>
</tr>
</tbody>
</table>

The user chooses whether or not the SU has perfect spectrum sensing capabilities. In the case of perfect sensing, the PU chooses randomly from the available channels. The SU is able to detect if and on what channel the PU is transmitting and thus chooses to transmit on a free channel in order to avoid a collision. If there are no free channels, the SU does not transmit.

In the case of imperfect sensing, the PU chooses randomly from a list of preferred channels (the default settings are channel 1 in the case of 1 or 2 channels, and channels 1 and 2 in the case of 3 of more channels). The SU is unable to accurately determine the presence of a PU at each time step and therefore risks a collision. When $\epsilon$-greedy or Softmax Strategy is chosen, the SU chooses its channel randomly. Due to time constraints, intelligent channel selection has not yet been implemented for these engines.

With a cognitive engine utilizing Q-learning with channel selection, the SU learns over time which channels the PU tends
to use more and learns to avoid those channels. The cognitive engine learns based on the reward received after each time step and updates its knowledge of the highest reward accordingly. In the case of a collision (i.e. the PU and SU transmit on the same channel) the SU receives a negative throughput and a BER of 1 in order to teach the SU to avoid channels in which the probability of collision is highest. Thus, the cognitive engine learns the optimal channel for transmission as well as the best modulation and coding rate pair within each channel. At each time step, the testbed displays the PU, the SU, their respective chosen channels, the receiver, the total number of channels available, the modulation-coding rate pair chosen by the cognitive engine, and the returned reward.

5. RESULTS

When the testbed is started, the CE begins with no knowledge of the channel conditions. In the single vehicle scenario, the testbed plots the BER against SNR in the case of optimizing BER and the throughput against SNR in the case of optimizing throughput. The plot shows the performance of the CE and compares it to that of a random action selector. While the CE chooses its action based on its reinforcement learning algorithm, the random action selector chooses an action at random from the list of all possible actions. Figure 1 shows such a plot in the case of the AWGN channel model, the goal of optimizing BER, and the $\epsilon$-greedy cognitive engine with $\epsilon=0.1$. Figure 2 shows the same scenario but with the goal of optimizing throughput. Figure 3 shows a plot in the case of the Rayleigh channel model, the goal of optimizing BER, and the $\epsilon$-greedy cognitive engine with $\epsilon=0.1$. Figure 4 shows the same scenario but with the goal of optimizing throughput. Figure 5 shows a plot in the case of the Rician ($k=2$) channel model, the goal of optimizing throughput, and the Softmax Strategy cognitive engine with $T=100$. Figure 6 shows a plot in the case of the AWGN channel model, the goal of optimizing throughput, and the annealing Softmax Strategy cognitive engine with $T=10000$ and $d=0.005$. Figure 7 shows a plot in the case of the AWGN channel model, the goal of optimizing BER, and the Q-learning with annealing $\epsilon$-greedy cognitive engine with $\epsilon=1$, $d=0.001$, $\alpha=0.5$ and $\gamma=0.2$. Each CE learns over 2000 timesteps at each SNR. The results indicate that the intelligent action selection outperforms the random action selection as expected.

In the PU vs. SU scenario, the testbed plots the accumulated average of either the BER or throughput depending on the optimization goal. This average is accumulated over 1000 timesteps for a single episode. Figure 8 shows the plot of the Q-learning engine with annealing $\epsilon$-greedy in the case of perfect sensing. There are 6 possible channels for transmission, and the PU has a 70% chance of transmitting and a transmission length of one time step. This plot compares the performances of the engine with channel selection (red) and without (blue). Figure 9 plots the same results but in the case of imperfect sensing.

It is clear in both Figures 8 and 9 that the engine with channel selection outperforms the engine without channel selection. This is especially dramatic in Figure 9 in which the SU has imperfect sensing capabilities. In both cases of perfect and imperfect sensing, the engine with channel selection learns to choose the channels with higher SNR values and channel models with less noise. In the case of imperfect sensing, the engine with channel selection also learns to avoid the negative reward associated with channels that have high probabilities of a collision with the PU. The engine without channel selection does not learn this and therefore collides with the PU more often, resulting in a far lower average throughput.
6. CONCLUSION AND FUTURE WORK

The testbed presents two scenarios – stationary single vehicle and stationary PU vs. SU – in V2I communication. Cognitive engines have been shown to outperform random action selection. The testbed demonstrated that the use of channel selection as a second layer of action selection in the cognitive engine improves the performance of the cognitive engine. The CE is shown to learn the best modulation-coding rate pair for each specific channel condition. Additionally, in the PU vs. SU scenario, the engine with Q-learning and channel selection learn the best channel for transmission in the case of both perfect sensing and imperfect sensing.

Future works concerning this testbed include the addition of more reinforcement learning algorithms, modulation schemes, and channel models. Specifically, the flat Rayleigh fading and flat Rician fading channel models could be adjusted to be frequency selective. Channel selection should also be added to the basic $\epsilon$-greedy and Softmax Strategy algorithms. Mobility of vehicles can also be added in the future.

In addition, we will also incorporate spectrum sensing methods into the testbed, such as in [17] and [18], in order to detect the existence of the PU instead of assuming perfect sensing. Energy detection was implemented based on the conventional methods presented in [17] and [18] using an USRP N200 and a WBX daughterboard to collect some data over different frequency bands. Due to time constraints, we weren’t able to incorporate these processes into the testbed.
Figure 7: Plot for single vehicle scenario optimizing BER in the AWGN channel model. Plots average BER against SNR performance for the Q-learning with annealing ϵ-greedy engine and a random action selector.

Figure 8: Plot of accumulated average over one episode in PU vs. SU scenario, optimizing throughput. The cognitive engine utilized is the Q-learning with ϵ-greedy annealing with ϵ=1, d=.001, α=.5 and γ=.2. The SU has perfect sensing capabilities. The engine with channel selection is shown in red, and the engine without channel selection in blue.

Figure 9: Plot of accumulated average over one episode in PU vs. SU scenario, optimizing throughput. The cognitive engine utilized is the Q-learning with ϵ-greedy annealing with ϵ=1, d=.001, α=.5 and γ=.2. The SU has imperfect sensing capabilities. The engine with channel selection is shown in red, and the engine without channel selection in blue.

7. ACKNOWLEDGEMENTS

This project was supported by the National Science Foundation (NSF Award No. 1659428) and the University of Arizona in the CAT Vehicle REU under Dr. Tamal Bose.

REFERENCES


