Reinforcement Learning for Mixed Cooperative/Competitive Dynamic Spectrum Access

Caleb Bowyer, David Greene, Tyler Ward, Marco Menendez, John Shea, and Tan Wong
University of Florida
{c.bowyer, djgreene, tsward, marcomenendez}@ufl.edu {jshea, twong}@ece.ufl.edu,

***Abstract*—A dynamic spectrum sharing problem with a mixed collaborative/competitive objective and partial information about peers' performances that arises from the DARPA Spectrum Collaboration Challenge is considered. Because of the very high complexity of the problem and the enormous size of the state space, it is broken down into the subproblems of channel selection, flow admission control, and transmission schedule assignment. The channel selection problem is the focus of this paper. A reinforcement learning algorithm based on a reduced state is developed to select channels, and a neural network is used as a function approximator to fill in missing values in the resulting input-action matrix. The performance is compared with that obtained by a hand-tuned expert system.**

I. Introduction

Many papers have considered the application of machine learning (ML) to cognitive radio systems. Because of the page limit, we refer interested readers to consult, for instance, [1]–[3] and references therein for a complete review of the literature. In this paper, we consider the problem of determining which channels to use in a dynamic spectrum access system with a mixed collaborative/competitive objective and partial information about peers' performances. The problem scenario comes from our participation in the DARPA Spectrum Collaboration Challenge (SC2). In SC2 matches, multiple teams of radios must operate in a shared radio frequency (RF) environment that emulates mobile ad hoc networking scenarios. Each team can score points by delivering traffic flows, but teams have a cooperative objective that in each measurement period, their score will be limited to the lowest score of all of the teams unless all teams' scores are above a specified threshold. The problem is complicated by the fact that teams that are scoring above the threshold can just report the scoring threshold, and not their true score. This motivates us to apply ML to determine a good channel use strategy to maximize our team's score in this scenario.

We found that the state space for applying ML to channel use in the SC2 scenarios was huge compared to many previous works. To arrive at a practically implementable solution, we have to break down into the subproblems of channel selection, flow admission control, and transmission schedule assignment. The focus of this paper is on channel selection, and we develop a reinforcement learning algorithm to adapt our channel use. Even focusing on just channel selection, we are required to choose a vastly reduced input space for our reinforcement algorithm and to apply neural network-based matrix smoothing techniques to both be able to collect enough data for effective learning and to be able to store the resulting input-action matrices in our radio. We also describe an expert system (ES) approach to channel selection, and we compare the performance of the ML and ES approaches.

Our participation in the DARPA SC2 was supported in part by a prize from the DARPA Spectrum Collaboration Challenge, the National Science Foundation under Grant 1738065, and by AFOSR award number FA9550-19-1-0169.

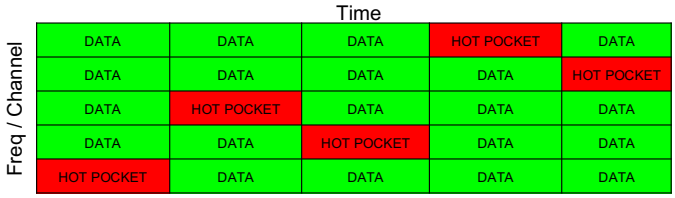
II. System Overview

The dynamic spectrum access system consists of multiple teams (networks) of radios communicating in a specified frequency band. Teams engage with each other during *matches*, which have reproducible, time-varying radio channel characteristics (produced using a channel emulator) and traffic flows. Each team receives a score based on the traffic flows that it is able to successfully deliver, as well as the traffic flows delivered by the other teams' networks. For each traffic flow, a *mandate* is provided that details the quality-of-service (QoS) that must be achieved to score points for that flow. Flows either come regularly and have specified throughput and latency requirements, or come as file bursts, for which 90% of the packets have to be delivered before a specified file transfer deadline. Each flow has an associated number of points that can be achieved in each measurement period in which the QoS is achieved, along with a hold time, which is a number of seconds for which the mandated QoS must be sustained before that flow scores any points. Each team has a *mandate threshold*, and the match score achieved in any scoring interval is limited to the lowest score among the teams if any team is below its mandate threshold.

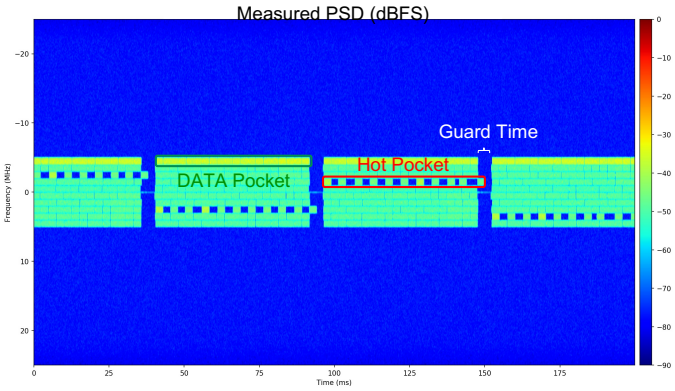
The teams do not have any information about the radio implementations and strategies of the other teams, except for information they can gather during the matches. Each team has one radio that acts as a gateway (GW), and the GW can communicate a limited set of collaboration information to peer networks over a separate collaboration channel. The information carried over the collaboration channel must adhere to a specified Collaborative Intelligent Radio Network Interaction Language (CIL) and includes:

- locations of the radios, specified as GPS coordinates,
- frequencies used and information on which radios are using which frequency bands, and
- number of achieved mandates and the mandate threshold.

978-1-7281-2376-9/19/$31.00 ©2019 IEEE



(a) Illustration of a scheduling frame



(b) Measured power spectral density

Fig. 1. Time-frequency pocket structure for channel access.

When teams are scoring below the mandate threshold, they must report their actual scores to their peers. However, when teams are scoring at or above the mandate threshold, they can report the mandate threshold instead.

We provide an overview of our team's channelization method and decision process below.

A. Channel access

Channel access in our radio network follows a time-frequency structure as shown in Fig. 1. The available frequency band is channelized into overlapping channels of 1 MHz in bandwidth. Adjacent channels are separated by 0.5 MHz. A subset of non-overlapping channels is dynamically selected to support the data flows admitted by our radio network.

The channels are subdivided in time into a repeating schedule of frames, each of which consists of a fixed number of time slots, as illustrated in Fig. 1. A given time-frequency slot is called a *pocket*. Most pockets are used for data transmission from a single source to one or more destinations. In addition, a randomized subset of pockets, referred to as *hot pockets*, is used to broadcast network management information and acknowledgments (ACKs). Each hot pocket is divided into minislots, and each radio sends its network management information and ACKs in an assigned minislot.

Transmission in each pocket is packetized into physical-layer (PHY) packets of a fixed duration. The PHY signaling is based on single-carrier frequency-domain (SC/FD) equalization with adaptive modulation and coding that is chosen based on channel conditions and flow QoS requirements. Each radio is capable of simultaneously transmitting and receiving on multiple channels.

B. Network and channel information

Each radio in our network has a spectrum sensor that can measure the power spectral density (PSD) of the whole frequency band. The PSD measurements are used to estimate the occupancy percentage of each channel. Spectrum usage and GPS information of peer networks obtained from the CIL network are fused with the PSD measurements to form an interference map at the GW. Through the use of a simple path-loss model, the GW then calculates the interference power seen at each channel of each radio and provides signal-to-interference-and-noise ratio (SINR) estimates for the current and future time.

C. Decision Engine

The decision engine is responsible for determining what channels our radio network will use and which flows can be supported using those channels. In what follows, we partition the set of flows into those that are *latency bound*, meaning that they require more than one pocket per frame and those that are *non-latency bound*. For the scenario considered in this paper, the relevant inputs to the DE consist of:

- the set of specified mandates for our team's flows,
- the estimated number of achieved mandates and the total mandates for our network,
- information on the throughput per pocket that is expected between each source-destination pair,
- the channels used by our network and by the peer networks,
- estimated channel occupancies from our spectrum sensor,
- computed SINRs from our interference map, and
- the estimated achieved and total mandates from competitor networks.

The decision engine, which we call the *pocket scheduler*, is responsible for determining which flows are transmitted and in which pockets they will be transmitted, with the goal of maximizing our team's match score. The output from this process is the *pocket schedule*, which is a list of pockets (time-frequency slots) and the source and destination(s) that will communicate in that slot. Because of the complexity of this optimization problem, the problem is decomposed into the following six steps:

1) The target set of channels to be used, C is determined based on the mandate performances of all of the teams, as well as channel occupancy information. The chosen set of channels has an impact on our team's performance because it limits what flows can be supported, and it impacts other teams' performances through the interference caused by our transmissions. Thus, how to chose the set of channels to use is the primary focus of this paper: in Section III, we give an overview of our expert systems approach, and in Section IV, we discuss our machine-learning approach.
2) Given the target set of channels C, admission control is performed by estimating the number of pockets or fractions of a pocket needed to support each flow, taking

into account the latency and throughput requirements of the flow, as well as the estimated bits/pocket that can be delivered for the source-destination for each flow. The cardinality of C determines the maximum number of pockets available, and an iterative process is used to choose the set of flows that maximizes the number of points that can be scored under the constraint on the number of pockets.

3) After the set of flows to be supported is determined, a linear program is used to allocate the pockets needed for sources to satisfy the latency requirements of their latency-bound flows to a set of virtual channels, which will be mapped to physical channels in a later step.
4) The linear program determines the number of pockets that each source uses on each channel but does not determine a particular set of pockets that satisfies the specified latency requirements and restrictions on the number of simultaneous transmissions. Thus, in this step, an iterative algorithm is used to search for a specific pocket assignment that can be used to satisfy the latency requirements for all sources.
5) The remaining pockets are assigned to satisfy the total throughput requirements from each source, subject to constraints on the number of possible simultaneous transmissions from a radio in a slot.
6) In the last step, the virtual channels are mapped to physical channels based on maximizing the worst-case SINR of any of the source-destination pairs assigned to the virtual channel. Let $C_A \subseteq C$ denote the set of channels that will be used for transmission; if the offered traffic requires fewer channels than are available in C, then $|C_A| < |C|$.

III. Expert System Approach to Channel Selection

The expert system (ES) approach to channel selection consists of two parts. First, a target number of channels is selected using various heuristics, depending on the relative performances of the teams. Second, the specific channels to be added to C are chosen from a sequence of channel subsets, from most desirable to least desirable. For the sake of conciseness, we give a high-level overview of each of these steps without specific details.

The number of channels starts at a high number each time a set of new mandates are received. The number of channels is then adapted based on the relative performance of our team versus our peers. Here we specify only the specific rules that apply with two peer teams. First we consider cases where at least one peer is below the mandate threshold. If both peers are doing worse than the threshold and have a score that is significantly lower than our team, we aggressively decrease the number of channels in proportion to the difference between our score and the highest score of the other teams. If one team is above the threshold but the other team is significantly below the threshold, then we maintain our current number of channels but avoid the channels of the peer whose score is below the threshold. If one or more peers are below the threshold but all peers are close to the threshold, then we will increase the maximum number of channels by one. If all peers are above the threshold, we consider two subcases. If no peer is reporting a score higher than ours, then we will increase the number of channels used by one. If a peer is reporting a higher score than ours, then we will aggressively increase the number of channels we are using in proportion to the score difference between our score and the highest score of any peer.

Once we choose a target number of channels, we add channels to C in the following order:

1) *uncontested channels*, which are channels used by our network and not by another peer,
2) *unused channels*, which are not being used by any other peer,
3) *channels from peers that are scoring above the threshold and/or have a higher score than our team*,
4) *channels used by peers that are scoring above the threshold but that may also be used by peers that are scoring below the threshold* , and
5) *channels used by other peers.*

IV. Machine Learning Approach to Channel Selection

A. Motivation

The ES approach has been shown to offer good performance across a wide variety of teams and scenarios. However, it has several limitations:

- It is not able to take into account differences in how different teams respond to interference.
- It is not able to infer peers' true scores from their reported scores and bandwidths, so it is operating in an open loop fashion when all teams are reporting the mandate threshold.
- It is not tuned to different scenarios.
- It is essentially a controller that takes time for the number of channels used to settle after a new set of mandates comes in, and the target number of channels may oscillate in some scenarios.

All of the deficiencies identified may reduce the potential score for our team, and so we developed a channel selection algorithm that uses machine learning (ML) to overcome some of these deficiencies. The machine learning algorithm leverages our ability to identify teams that we have played before to determine appropriate channel usage versus those teams. We have developed a SARSA trained agent that is used in a generalized policy iteration framework, using discounting in a continuing task based on the data streams of states, actions, and rewards the agent experiences in the SC2 scenarios. Furthermore, a standard neural network with a single hidden layer is used as a function approximator to fill in missing gaps of knowledge in the Q-value arrays which are absent from the agents history of experience. This agent is a sample-based learner that does not require transition probabilities, which are often tedious or error prone in calculation, and are not guaranteed to be time-invariant for this problem of finding an

optimal spectrum usage policy. This is the case because of the changing environment due to competitor networks.

B. State-Space-Model, Rewards, and Actions

We now define the state-space (input space, action space, reward) used by our algorithm. Ideally, the input space would contain all of the inputs to the decision engine described in Section II-C, and the action space would consist of a list of supported flows and a complete pocket schedule indicating which channels are used and which source sends to which destination(s) in each pocket. The reward is simply the match score achieved using that action as calculated as described in Section II. However, this would result in an exceptionally large state space compared to the amount of training data and the available space to store an input-space/action matrix for use by our GW. For instance, our radio can use up to 40 channels, with 10 pockets in each channel. When our network consists of 10 radios, there are 90 source-destination pairs, thus there are on the order of 90^{400} possible pocket schedules.

In order to come up with a feasible state-space formulation, we use machine learning to perform only the channel selection step (Step 1) of the DE. The remaining steps of the DE are carried out using the same optimization algorithms as described in Section II-C. However, even after restricting to the channel selection problem, the size of the state space that fully describes the problem is still too big for practical implementation. As a result, leveraging the capability of identifying peer networks, we solve the channel selection problem by setting up a SARSA trained agent to find an approximately optimal policy to determine our appropriate action on each peer network individually. Channel selection is then accomplished by combining the actions of the agents, each of which works on an individual peer network that participates in a match.

To describe the state space employed by each agent, number our own network as network 0, and identify agent i as the agent to work on peer network i, where $i > 0$. Let t denote the time step for the agent. Let $c_{0,t}$, $c_{i,t}$, and $\bar{c}_t$ denote the estimated value of the score achieved by our own network, the score reported by peer network i, and the scoring threshold at time t. Also let $b_{i,t}$ denote the amount of bandwidth in units of MHz used by peer network i, and $b_{0,t}$ denote the amount of bandwidth out of $b_{i,t}$ that is also used by our network. Then the input state S_t for agent i at time t is the 5-tuple $S_t = (s_{1,t}, s_{2,t}, s_{3,t}, s_{4,t}, s_{5,t})$, where

$$\begin{aligned} s_{1,t} &= \log_2 \frac{c_{0,t}}{\bar{c}_t} \\ s_{2,t} &= \log_2 \frac{c_{i,t}}{\bar{c}_t} \\ s_{3,t} &= 1\left(c_{i,t} = \min_j c_{j,t}\right) \\ s_{4,t} &= b_{i,t} \\ s_{5,t} &= b_{0,t}. \end{aligned}$$

To limit the size of the state space, the state components $s_{1,t}$, $s_{2,t}$, $s_{4,t}$, and $s_{5,t}$ are quantized to 10, 10, 20, and 20 values, respectively. For $s_{3,t}$, the minimum is taken over the scores of all teams that participate in the match and the function $1(\mathcal{A})$ is the indicator function of event $\mathcal{A}$. The first two components $s_{1,t}$ and $s_{2,t}$ in the state vector are to capture the scoring performance of both our network and the peer network in relation to the scoring threshold. The use of the logarithmic transform in the two scoring ratios is to put more emphasis on scores that are near the scoring threshold. The third component $s_{3,t}$ indicates if peer network i's score is the critical factor that decides the score of the match (see Section II). The last two state components $s_{4,t}$ and $s_{5,t}$ capture the impact of the peer's spectrum usage and how much our network is overlapping with the peer in spectrum in the state-space model.

The action $A_{i,t}$ taken by agent i is to determine how much bandwidth from the part of spectrum used by peer network i that our network should use. In other words, agent i's action is to determine a new value for $b_{0,t}$. The action space is again quantized to contain 20 possible values as above. The match score is taken as the reward.

C. Agent policy training

We employ the SARSA algorithm [4, Ch. 4] to estimate the Q-value array for agent i:

$$\begin{aligned} &Q(S_t, A_t) \\ &\quad \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)) \end{aligned}$$

where γ is the discount rate and α is the update step size. Missing gaps of knowledge in the Q-value array are further filled in by the use of a standard neural network with a single hidden layer, which serves as a function approximator. Training is performed offline using data obtained from matches. The policy

$$\pi(s) = \arg\max_a Q(s, a)$$

is applied to obtain the number of channels from peer network i's set of channels that we should use. Aggregating the actions from all agents, we obtain the total number of channels from all peer networks to be used by us. Then the set of channels C determined in Step 1) of the DE is obtained following the same procedure described in Section III.

V. Results

To assess the relative performances of the ES and ML channel selection algorithms proposed in Section III and Section IV, we conducted a series of "freeplay" matches using DARPA's Colosseum, a system consisting of 128 standard radio nodes (SRNs), each with 2 transmit and 2 receive chains, and a 256×256 channel emulator. The freeplay designation indicates that each match was run at our request using randomly selected radio images that other teams in the DARPA Spectrum Collaboration Challenge (SC2) had submitted for the purpose of testing their radio algorithms. Because each algorithm's channel choices depend on past choices by the algorithm, each match that we present results for only used either the Expert System or the ML algorithm. To provide reasonable comparisons of algorithm performance, we found

matches for each algorithm that had the same set of peer teams. We had data on a total of 14 matches with our ML algorithm, but because of the stochastic nature of the freeplay process, we only found 10 matches with the same set of teams in the same scenario for the ES algorithm. Consequently, this paper presents the data for only these 10 matches with the same set of teams. After each set of 10 matches were selected, we paired matches (one ES, one ML) with the same sets of teams. To make some of the figures easier to interpret, we assigned these matches normalized match numbers from 1 to 10, such that the overall match scores for the ML algorithm are in decreasing order. The particular identities of the peer teams in each match are not known, but we are able to discriminate among the teams using their CIL characteristics. In Table I, we show which teams were involved in each match, where the different teams are coded using a letter A–H.

Normalized Match Number(s)	Peer Team 1	Peer Team 2
1,2	A	B
3	A	C
4	A	D
5	D	E
6	E	F
7	D	G
8,9	G	H
10	F	G

TABLE I
TEAMS PARTICIPATING IN EACH MATCH. ALTHOUGH EACH TEAM IS UNIQUELY IDENTIFIED, WE DO NOT HAVE THE MAPPING TO THE TEAM'S TRUE IDENTITY.

The scenario for each run is the 7013 "Alleys of Austin" scenario, which DARPA describes as follows: "A platoon from the Texas Army National Guard at Camp Mabry is practicing urban maneuvers and communications in Austin. The platoon is split into five squads consisting of 9 squad members and one UAV. The squads move through the Heritage neighborhood in the following three stages..." The allowed channel bandwidth is 20 MHz, and the scenario contains three stages, each of which is 300 s long. Traffic loads increase at each stage change, and the amount of spatial reuse possible varies across the stages. In the first stage, the traffic is primarily Voice over IP (VOIP) and Blue Force Tracking (BFT) data, which require low throughput ($<$ 40 kbps). Stage 2 and 3 add file and video flows, which require much higher throughput. Note that although the teams are the same for matches with the same "Normalized Match Number", the teams may be in different positions, and this might cause the inter-team interference to be very different.

The results in Fig. 2 show the cumulative end of game/end of stage 3 match scores achieved for the ES and ML algorithms for each of the 10 matches. Although not shown, the cumulative scores at the ends of stages 1 and 2 are very similar in terms of the relative performances of the algorithms in different matches. The results show that for matches 1, 2, 3, 4, and 10, the performance of the ES and ML algorithms are approximately equal. These correspond to matches in which

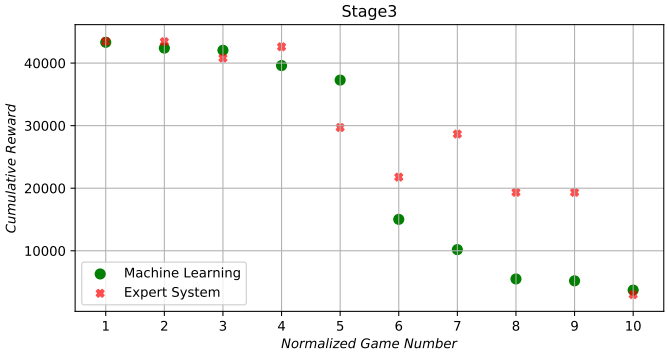


Fig. 2. Cumulative match scores for 10 matches with both expert system (ES) and machine learning (ML) algorithms.

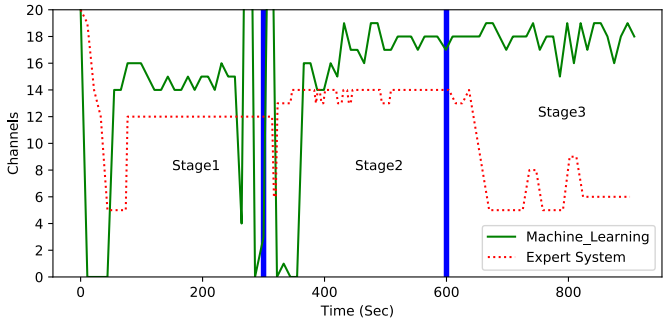


Fig. 3. Target total number of channels for expert systems and ML algorithms as a function of time for match 5.

the scores are very high or very low. For the other matches, the ES algorithm achieves the highest overall score in 4 matches, and the ML algorithm achieves the highest overall score in 1 match.

To provide a bit more insight into the choices made by these algorithms, we investigate the target and actual total number of channels used in several matches. For match 5, the target and actual total number of channels are shown in Fig. 3 and Fig. 4. The channel use is mostly constrained by the offered loads in stages 1 and 2. The ML algorithm outscores

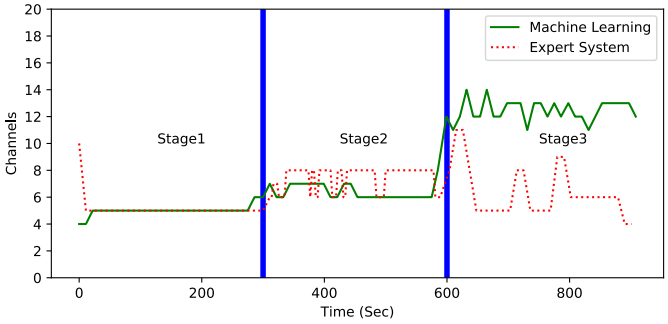


Fig. 4. Actual total number of channels for expert systems and ML algorithms as a function of time for match 5.

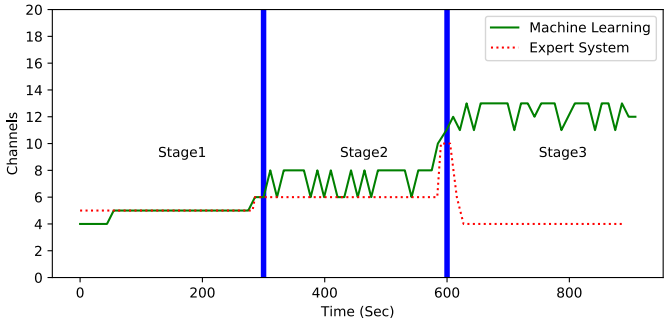


Fig. 5. Actual total number of channels for expert systems and ML algorithms as a function of time for match 7.

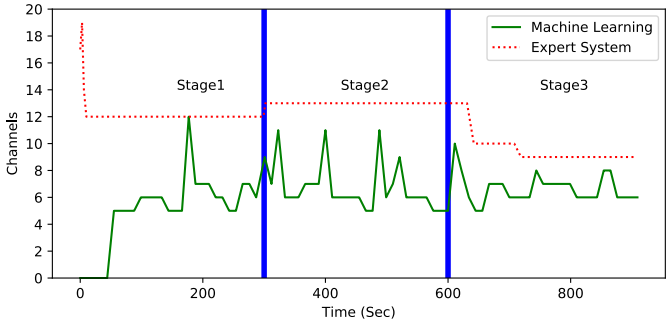


Fig. 6. Target total number of channels for expert systems and ML algorithms as a function of time for match 10.

the ES algorithm significantly in stages 1 and 3. However, we observe that the difference in stage 1 is achieved with a similar number of channels. Thus, either the ML algorithm does a better job of choosing the particular channels to use, or the interference topology is more favorable in the ML match. In both matches, all 3 teams are in adjacent positions, which have the worse SINRs, but in the ES match, our team is in between the other two teams; in the ML match, our team is one of the teams on the ends. In stage 3, the ML algorithm is much more aggressive in using channels and achieves a higher overall score.

For match 7, the actual total number of channels are shown in Fig. 5. The target number of channels is not shown but is similar in trend to the result for match 5. Again, the channel use is constrained by the offered load in stage 1. The ES algorithm significantly outperforms the ML algorithm in all stages, even while using much fewer channels in Stage 3. The fact that the ES algorithm does better than the ML algorithm in Stage 1, when both teams use the same number of channels for most of the stage suggests that the difference could be due to other factors, such as the interference topology. However, the topology is similar in both matches, with all three teams in adjacent positions and our team on one end. The effect may be because the particular order of the other teams and differences in their sensitivity to interference. If this is the case, then the ES strategy of reducing channels and avoiding the channels of low scoring teams may explain the better performance.

Finally, Fig. 6 shows the target maximum number of channels for match 10. This result is interesting because, unlike the matches investigated above, the ML algorithm consistently targets a smaller number of channels than the ES algorithm. However, both algorithms achieve a terrible performance. These observations may be attributed to two factors. First, both peer teams reported high scores while actually achieving low scores, causing the ES algorithm to drive the target number of channels up. However, the ML algorithm apparently "understands" that the reported scores for these teams cannot be trusted and chooses a small number of channels. Ultimately, neither strategy can improve the score because at this stage of the SC2 competition, we did not observe these two teams achieving scores above the mandate threshold in any of the games we played with them.

VI. Conclusion

We presented two approaches for determining how many and which channels to use in a competitive/cooperative spectrum sharing system. The expert system (ES) approach is based on using a set of rules, thresholds, and functions for adapting channel use in response to the team scores at a given time in a match. It depends on hand-tuning by an experienced communications engineer. The machine learning (ML) algorithm uses reinforcement learning and a neural network for matrix completion. It depends on careful selection of a reduced state space that is small enough that we can apply SARSA to learn the input-action matrices and be able to store those matrices on our radio. The results show that in many cases, the ML algorithm achieves similar performance to the ES algorithm; however, we found that the ML algorithm may do worse in certain situations because it uses too many channels. One explanation for this is that the algorithm actually is applied across a variety of interference configurations, and many points can be scored by aggressive channel use when the topology provides spatial reuse. The algorithm may find that aggressive channel use is the overall best strategy to maximize the average score, even if the score for many individual matches may be lower if there is less spatial reuse. Thus, metrics that better capture the spatial reuse may be important to include in the state space in future implementations.

References

[1] M. Bkassiny, Y. Li, and S. K. Jayaweera, "A survey on machine-learning techniques in cognitive radios," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1136–1159, 2012.

[2] J. Lunden, V. Koivunen, and H. V. Poor, "Spectrum exploration and exploitation for cognitive radio: Recent advances," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 123–140, 2015.

[3] Y. Wang, Z. Ye, P. Wan, and J. Zhao, "A survey of dynamic spectrum allocation based on reinforcement learning algorithms in cognitive radio networks," *Artificial Intelligence Review*, vol. 51, no. 3, pp. 493–506, 2019.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.