Deep Multi-User Reinforcement Learning for Dynamic Spectrum Access in Multichannel Wireless Networks

Oshri Naparstek[1] and Kobi Cohen[2]

[1]Rafael Advanced Defense Systems Ltd., Haifa 31021, Israel
oshrin@rafael.co.il
[2]Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
yakovsec@bgu.ac.il

***Abstract*— We consider the problem of dynamic spectrum access for network utility maximization in multichannel wireless networks. The shared bandwidth is divided into K orthogonal channels, and the users access the spectrum using a random access protocol. In the beginning of each time slot, each user selects a channel and transmits a packet with a certain attempt probability. After each time slot, each user that has transmitted a packet receives a local observation indicating whether its packet was successfully delivered or not (i.e., ACK signal). The objective is to find a multi-user strategy that maximizes a certain network utility in a distributed manner without online coordination or message exchanges between users. Obtaining an optimal solution for the spectrum access problem is computationally expensive in general due to the large state space and partial observability of the states. To tackle this problem, we develop a distributed dynamic spectrum access algorithm based on deep multi-user reinforcement leaning. Specifically, at each time slot, each user maps its current state to spectrum access actions based on a trained deep-Q network used to maximize the objective function. Experimental results have demonstrated that users are capable to learn good policies that achieve strong performance in this challenging partially observable setting only from their ACK signals, without online coordination, message exchanges between users, or carrier sensing.**

I. Introduction

The increasing demand for wireless communication, along with spectrum scarcity, have triggered the development of efficient dynamic spectrum access (DSA) schemes for emerging wireless network technologies. A good overview of the various DSA models for medium access control (MAC) design can be found in [1]. In this paper we focus on DSA in the open sharing model among users that acts as the basis for enabling a large number of cognitive radio users to coexist in the same limited frequency band (e.g., cognitive radio networks, cognitive radio sensor networks, WiFi and ISM band applications), so generally we will not assume that there are primary and secondary users in the network and mainly focus on the dynamic among cognitive radio users (as examined in [1]–[5]).

We consider a wireless network with N users sharing K orthogonal channels (e.g., OFDMA). In the beginning of each time slot, each user selects a channel and transmits its data with a certain attempt probability (i.e., Aloha-type narrowband transmission). After each time slot, each user that has transmitted a packet receives a local binary observation indicating whether its packet was successfully delivered or not (i.e., ACK signal). The goal of the users is to maximize a certain network utility in a distributed manner without online coordination or exchanging messages used for managing the spectrum access.

A. Learning Algorithms for Dynamic Spectrum Access

Developing distributed optimization and learning algorithms for managing efficient spectrum access among users has attracted much attention in past and recent years (see Section I-D for a detailed discussion on related work). Complete information about the network state is typically not available online for the users, which makes the computation of optimal policies intractable in general [6]. While optimal structured solutions have been developed for some special cases (e.g., [7]–[9] and references therein), most of the existing studies have been focused on designing spectrum access protocols for specific models so that efficient (though not optimal) and structured solutions can be obtained. However, model-dependent solutions cannot be effectively adapted in general for handling more complex real-world models. Model-free Q-learning has been used in [10] for Aloha-based protocol in cognitive radio networks. Handling large state space and partial observability, however, becomes inefficient under Q-learning (see Section III-A for details on Q-learning).

B. Deep Multi-User Reinforcement Learning for Dynamic Spectrum Access

Our goal is to develop a distributed learning algorithm for dynamic spectrum access that can effectively adapt for general complex real-world settings, while overcoming the expensive computational requirements due to the large state space and partial observability of the problem. We adopt a deep multi-user reinforcement learning approach to achieve this goal.

Deep reinforcement learning (DRL) (or deep Q-learning) has attracted much attention in recent years due to its capability to provide a good approximation of the objective value (referred to as Q-value) while dealing with a very large state and action spaces. In contrast to Q-learning methods that perform well for small-size models but perform poorly for large-scale models, DRL combines deep neural network with Q-learning, referred to as Deep Q-Network (DQN), for overcoming this issue. The DQN is used to map from states to actions in large-scale models so as to maximize the Q-value (for more details on DRL and related work see Sections III-A and I-D). In DeepMind's recently published Nature paper [11], a DRL algorithm has been developed to teach computers how to play Atari games directly from the on-screen pixels, and strong performance has been demonstrated in many tested games. In [12], the authors developed DRL algorithms for teaching multiple players how to communicate so as to maximize a shared utility. Strong performance has been demonstrated for several players in MNIST games and the switch riddle. In recent years, there is a growing attention on using DRL methods for other various fields. A survey on very recent studies can be found in [13].

Due to the large state space and the partially observed nature of spectral region management among wireless connected devices, we postulate that incorporating DRL methods in the design of DSA algorithms has a great potential for providing effective solutions to real-world complex spectrum access settings, which motivates the research in this paper.

C. Main Results

Using DRL methods in the design of spectrum access protocols is a new research direction and very little has been done in this direction so far (for details see I-D). To solve the multi-user multichannel DSA problem at hand, we develop a deep multi-user reinforcement learning-based algorithm that allows each user to adaptively adjust its transmission parameters (i.e., which channel to access, and which attempt probability to use) with the goal of maximizing a certain network utility. The algorithm is executed without online coordination or message exchanges between users used for managing the spectrum access. The proposed algorithm works as follows. While offline, we train the multi-user DQN at a central unit for maximizing the objective function. Since the network state is partially observable for each user, and the dynamic is non-Markovian and determined by the multi-user actions, we use Long Short Term Memory (LSTM) layer that maintains an internal state and aggregate observations over time. This gives the network the ability to estimate the true state using the past partial observations. Furthermore, we incorporate the dueling DQN method used to improve the estimated Q-value due to the occurrence of bad states regardless of the taken action [14]. Note that the experience replay method, suggested in [15], [11] for handling a single-agent learning from past observations, is undesirable when handling a multi-user learning for DSA due to interactions among users. Hence, instead of using experience replay, we collect M episodes at each iteration and create target values for all the episodes used for multi-user learning.

After completion of the training phase, the users only need to update their DQN weights by communicating with the central unit. In real-time, at each time slot, each user maps its local observation to spectrum access actions (i.e., which channel to select next and which attempt probability to set) based on the trained DQN. The design of the DQN enables each user to learn a good policy in online and fully distributed manners, while dealing with the large state space without online coordination or message exchanges between users. A detailed description of the algorithm is provided in Section III.

Note that the proposed algorithm is very simple for implementation using simple software defined radios (SDRs). The expensive computations at the training phase is done offline by a centralized powerful unit (e.g., service provider, cloud computing, access point). Since an extensive training over many experiences with dynamic environment and topology changes can be done offline, then updating the DQN is rarely required (e.g., once per weeks, months, only when the environment characteristics have been significantly changed and no longer reflects the training experiences).

We performed extensive numerical experiments for demonstrating the capability of the proposed algorithm to effectively adapt to different scenarios and objective functions (for details see Section IV). Under both cooperative and non-cooperative network utilities, we observed that users effectively learn in online and fully distributed manners only from their ACK signals how to access the channel so as to increase the channel throughput by reducing the number of idle time slots (i.e., when no user transmits) and collisions (i.e., when two or more users transmit). Specifically, the proposed algorithm achieves about twice the channel throughput as compared to slotted-Aloha with optimal attempt probability (which requires complete knowledge about the number of users). Second, in terms of rate allocation among users, we observed that users can learn (again, only from their ACK signals) effective policies depending on the objective network utility. When the network utility is the user sum rate we observed that in about 80% of the Mont-Carlo experiments some users are willing to get zero rate (to reduce interferences to other users) in order to increase the network utility. On the other hand, when the network utility is competitive in the sense that each user aims to maximize its own rate, we observed that in about 80% of the Mont-Carlo experiments users converge to a Pareto-optimal sharing policy. This result presents a tremendous improvement as compared to the Nash equilibrium points of competitive Aloha games which are highly inefficient [4].

D. Related work

Developing DRL-based methods for solving DSA problems is a new research direction and very little has been done in this direction so far. We discuss next the very recent studies on this topic which are relevant to the problem considered in this paper. In [16], the authors have developed a spectrum sensing policy based on DRL for a single user who interacts with an external environment. The multi-user setting (i.e., game) considered here, however, is fundamentally different in environment dynamics, network utility, and algorithm design. In [17], the authors studied a non-cooperative spectrum access problem in a different setting, in which multiple agents (i.e., base-stations in their model) compete for channels and aim at predicting the future system state using LSTM layer with REINFORCE algorithm. The neural network is trained at each agent. The problem formulation in [17] is non-cooperative in the sense that each agent aims at maximizing its own utility, while using the predicted state to reach a certain fair equilibrium point. Our algorithm and problem setting are fundamentally different. First, our algorithm uses LSTM with DQN which is different from the algorithm in [17]. Second, in our algorithm, the DQN is trained for all users at a single unit (e.g., cloud), which is more suitable to cognitive radio networks, in which cheap SDRs only need to rarely update their DQN weights by communicating with the central unit. Third, we are interested in both cooperative and non-cooperative settings, where fundamentally different operating points are reached depending on the network utility function. Furthermore, in [17] the focus is on matching channels to BSs, where in our setting we focus on sharing the limited spectrum by a large number of users (i.e., matching might be infeasible).

Other related works on learning algorithms for DSA have been mainly focused on model-dependent settings or myopic objectives so that tractable and structured solutions can be obtained. The problem has been widely studied under multi-armed bandit (and variants) formulations in [7]–[9], [18]–[20] (and references therein), game theoretic and congestion control in [3], [21]–[26] and references therein, and matching theory in [2], [5], [27]–[29] and references therein. These studies, however, focus on model and objective-dependent problem settings, and often require more involved implementations (e.g., carrier sensing, wideband monitoring), which make them fundamentally different from the problem setting considered in this paper.

II. Network Model and Problem Statement

We consider a wireless network consisting of a set $\mathcal{N} = \{1, 2, ..., N\}$ of users and a set $\mathcal{K} = \{1, 2, ..., K\}$ of shared orthogonal channels (i.e., subbands). The users transmit over the shared channels using a random access protocol. At each time slot, each user is allowed to choose a single channel for transmission with a certain attempt probability (i.e., Aloha-type narrowband transmission). We assume that users are backlogged, i.e., all users always have packets to transmit. Transmission on channel k is successful if only a single user transmits over channel k in a given time slot. After each time slot (say t), in which each user (say n) has attempted to transmit a packet, it receives a binary observation $o_n(t)$, indicating whether its packet was successfully delivered or not (e.g., ACK signal). If the packet has been successfully delivered, then $o_n(t) = 1$. Otherwise, if the transmission has failed (i.e., a collision occurred), then $o_n(t) = 0$.

Let $a_n(t) \in \{0, 1, ..., K\}$ be the action of user n at time slot t, where $a_n(t) = 0$ refers to the case in which user n chooses not to transmit a packet at time slot t (to reduce the congestion level for instance), and $a_n(t) = k$, where $1 \leq k \leq K$, refers to the case in which user n chooses to transmit a packet on channel k at time slot t. We define $a_{-n}(t) = \{a_i(t)\}_{i \neq n}$ as the action profile for all users except user n at time slot t. We consider a distributed setting without online coordination or message exchanges between users used to managing the spectrum access. As a result, the network state at time t (i.e., $a_{-n}(t)$) is only partially observed by user n through the local signal $o_n(t)$. The history $\mathcal{H}_n(t)$ of user n at time t is defined by the set of all actions and observations up to time t, $\mathcal{H}_n(t) = (\{a_n(i)\}_{i=1}^{t}, \{o_n(i)\}_{i=1}^{t})$. A strategy $\sigma_n(t)$ of user n at time t is a mapping from history $\mathcal{H}_n(t-1)$ to a probability mass function over actions $\{0, 1, ..., K\}$. The time series vector of strategies (or *policy*) for user n is denoted by $\boldsymbol{\sigma}_n = (\sigma_n(t), t = 1, 2, ...)$.

Let $r_n(t)$ be a reward that user n obtains at the beginning of time slot t. The reward depends on user n's action $a_n(t-1)$ and other users' actions $a_{-n}(t-1)$ (i.e., the unknown network stat that user n aims to learn). Let $R_n = \sum_{t=1}^{T} \gamma^{t-1} r_n(t)$ be the accumulated discounted reward, where $0 \leq \gamma \leq 1$ is a discounted factor, and T is the time-horizon of the game. We often set $\gamma = 1$, or $\gamma < 1$ when T is bounded or unbounded, respectively. The objective of each user (say n) is to find a policy $\boldsymbol{\sigma}_n$ that maximizes its expected accumulated discounted reward:

$$\max_{\boldsymbol{\sigma}_n} \ \mathbf{E}\left[R_n | \boldsymbol{\sigma}_n\right]. \tag{1}$$

Remark 1: Let $\mathbf{1}_n(t)$ be the indicator function, where $\mathbf{1}_n(t) = 1$ if user n has successfully transmitted a packet at time slot t, and $\mathbf{1}_n(t) = 0$ otherwise. Note that the reward can be social or individual. For example, we can set $r_n(t) = \mathbf{1}_n(t-1)$, so that user n aims to maximize the total number of its own successful transmissions (i.e., individual rate). On the other hand, we can set $r_n(t) = 0$, for all $1 \leq t \leq T-1$, and $r_n(T) = \sum_{n=1}^{N} f\left(\sum_{t=1}^{T} \mathbf{1}_n(t-1)\right)$, so that user n aims to maximize a social network utility (e.g., user sum rate when $f(x) = x$, proportional fairness when $f(x) = \log(x)$).

We are interested in developing a model-free distributed learning algorithm to solve (1) that can effectively adapt to topology changes, different objectives, different finite time-horizons (in which solving dynamic programming becomes very challenging, or often impossible for large T), etc.

III. The Proposed Algorithm

In this section we present the proposed algorithm, based on deep multi-user reinforcement learning, for solving (1). We first describe the basic idea of Q-learning and DQN.

A. Q-learning and DQN

Q-learning has been widely applied for finding good policies in various decision making problems, primarily because its ability to evaluate the expected utility among available actions without requiring prior knowledge about the system model, and its ability to adapt when stochastic transitions occur [30]. The algorithm was originally designed for a single agent who interacts with a fully observable Markovian environment and demonstrated strong performance when handling more involved settings (e.g., multi-agent, non-Markovian environment). Assume first that the network state $s_n(t) = a_{-n}(t)$ is fully observable by user n. By applying Q-learning to our setting, the algorithm updates a Q-value at each time t for each action-state pair as follows:

$$\begin{aligned} Q\left(s_n(t), a_n(t)\right) &\leftarrow Q\left(s_n(t), a_n(t)\right) \\ &+\alpha\left[r_n(t+1)+\gamma \max_{a_n(t+1)} Q\left(s_n(t+1), a_n(t+1)\right)\right. \\ &\left.\quad -Q\left(s_n(t), a_n(t)\right)\right], \end{aligned} \tag{2}$$

where $r_n(t+1)+\gamma \max_{a_n(t+1)} Q\left(s_n(t+1), a_n(t+1)\right)$ is the learned value obtained by getting reward $r_n(t+1)$ after taking action $a_n(t)$ in state $s_n(t)$, moving to next state $s_n(t+1)$, and then taking action $a_n(t+1)$ that maximizes the future Q-value seen at time $t+1$. The term $Q\left(s_n(t), a_n(t)\right)$ is the old learned value. Thus, the algorithm aims at minimizing the Time Difference (TD) error between the learned value and the current estimate value. The learning rate α is set to $0 \leq \alpha \leq 1$, where typically is set close to zero. When the problem is partially observable, the state is set to the history, i.e., $s_n(t) = \mathcal{H}_n(t)$ in our case (or a sliding window history when the problem size is too large). While Q-learning performs well when dealing with small action and state spaces, it becomes impractical when the problem size increases for mainly two reasons: (i) A stored lookup table of Q-values for all possible state-action pairs is required which makes the storage complexity intolerable for large-scale problems; (ii) As the state space increases, many states are rarely visited, which significantly decreases performance.

In recent years, a great potential was demonstrated by DRL methods that combine deep neural network with Q-learning, referred to as Deep Q-Network (DQN), for overcoming these issues. Using DQN, the deep neural network maps from the (partially) observed state to an action, instead of storing a lookup table of Q-values. Furthermore, large-scale models can be represented well by the deep neural network so that the algorithm has the ability to preserve good performance for very large-scale models.

B. Architecture of the Proposed Multi-User DQN Used for Solving (1)

In this section we describe the proposed architecture for the multi-user DQN used to solve the DSA problem. An illustration of the DQN is presented in Fig. 1.

1) Input Layer: The input $\mathbf{x}_n(t)$ to the DQN is a vector of size $2K+2$. The first $K+1$ input entries indicate the action (i.e., selected channel) taken at time $t-1$. Specifically, if the user has not transmitted at time slot $t-1$, the first entry is set to 1 and the next K entries are set to 0. If the user has chosen channel k for transmission at time $t-1$ (where $1 \leq k \leq K$), then the $(k+1)^{th}$ entry is set to 1 and the rest K entries are set to 0. The following K input entries are the capacity of each channel (i.e., the packet transmission rate over a channel conditioned on the event that the channel is

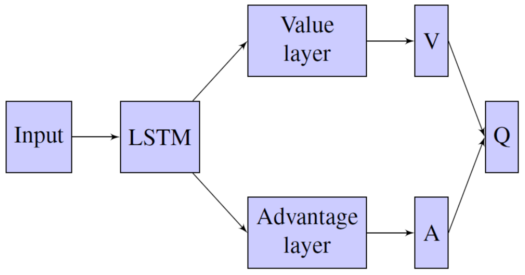

Fig. 1. Architecture of the proposed multi-user DQN used for dynamic spectrum access.

free, which is proportional to the bandwidth of the channel). In the experiments presented in Section IV, we simulated equal channels (i.e., we set 1 at all K entries). The last input is 1 if ACK signal has been received. Otherwise, if transmission has failed or no transmission has been executed, it is set to 0.

2) LSTM Layer: Since the network state is partially observable for each user, and the dynamic is non-Markovian and determined by the multi-user actions, classical DQNs do not perform well in this setting. Thus, we add an LSTM layer ([31]) to the DQN that maintains an internal state and aggregate observations over time. This gives the network the ability to estimate the true state using the history of the process. This layer is responsible of learning how to aggregate experiences over time.

3) Value and Advantage Layers: Another improvement that we incorporate is the use of dueling DQN, as suggested in [14]. The intuition behind this architecture lies in the fact that there is an observability problem in DQN. There are states which are good or bad regardless of the taken action. Hence, it is desirable to estimate the average Q-value of the state which is called the value of the state $V(s_n(t))$ independently from the advantage of each action. Thus, when we input $\mathbf{x}_n(t)$ to the DQN with dueling, the Q-value for selecting action $a_n(t)$ at time t is updated by:

$$Q(a_n(t)) \leftarrow V + A(a_n(t)) \tag{3}$$

where V is the value of the state and A is the advantage of each action. Note that both V and $A(a_n(t))$ depend on the state $s_n(t)$ (which is hidden and mapped by the DQN from the history).

4) Block output layer: The output of the DQN is a vector of size $K+1$. The first entry is the estimated Q-value if the user will choose not to transmit at time t. The $(k+1)^{th}$ entry, where $1 \leq k \leq K$, is the estimated Q-value for transmitting on channel k at time t.

5) Double Q-learning: The max operator in standard Q-learning and DQN (see (2)) uses the same values to both selecting and evaluating an action. Thus, it tends to select overestimated values which degrades performance. Hence, when training the DQN, we use double Q-learning [32] used to decouple the selection of actions from the evaluation of Q-values. Specifically, we use two neural networks, referred to as DQN$_1$ and DQN$_2$. DQN$_1$ is used for choosing actions and DQN$_2$ is used to estimate the Q-value associated with the selected action.

C. Training the DQN:

The DQN is trained for all users at a central unit in an offline manner. We train the DQN as follows:

1) Repeat:
2) In each iteration, repeat the following for M episodes to form a mini batch of M training examples:
3) In each episode, repeat the following for T time-slots:
4) In each time-slot, repeat the following for all $n \in \mathcal{N}$ users:
5) Observe an input $\mathbf{x}_n(t)$ and feed it into the neural network DQN$_1$. The neural network generates an estimation of the Q-values $Q(a)$ for all available actions $a \in \{0, 1, ..., K\}$.
6) Take action $a_n(t) \in \{0, 1, ..., K\}$ (according to (5) that will be discussed later) and obtain a reward $r_n(t+1)$.
7) Observe an input $\mathbf{x}_n(t+1)$ and feed it into both neural networks DQN$_1$ and DQN$_2$. The neural networks generate estimations of the Q-values $\widetilde{Q}_1(a)$ and $\widetilde{Q}_2(a)$, respectively, for all actions $a \in \{0, 1, ..., K\}$.
8) Form a target vector for the training by replacing the $a_n(t)$ entry by:

$$Q(a_n(t)) \leftarrow r_n(t+1) + \widetilde{Q}_2\left(\arg\max_a \left(\widetilde{Q}_1(a)\right)\right). \quad (4)$$

9) End loop for $n \in \mathcal{N}$ users.
10) End loop for T time-slots.
11) End loop for M episodes.
12) Train DQN$_1$ with inputs **x**s and outputs Qs.
13) Every ℓ iterations set $Q_2 \leftarrow Q_1$.
14) End outer loop.

In our experiments, we repeated the outer loop for several thousands iterations until convergence, and ℓ was set to 5. Note that unlike [15], [11], in which experience replay is used in the single-agent case to learn from past observations, in the multi-user case considered here such learning is undesirable due to interactions among users. Thus, we collect the M episodes at each iteration and create target values for all the episodes.

D. Online Learning: Distributed Random Access using DQN:

The training phase is rarely required to be updated by the central unit (e.g., once per weeks, months, only when the environment characteristics have been significantly changed and no longer reflects the training experiences). Users' SDRs only need to update their DQN weights by communicating with the central unit. In real-time, each user (say n) makes autonomous decisions in online and distributed manners using the trained DQN, so as to learn efficient channel selection policies from its ACK signals only:

1) At each time slot t, obtain observation $o_n(t)$ and feed input $\mathbf{x}_n(t)$ to the trained DQN$_1$. Output Q-values $Q(a)$ are generated by DQN$_1$ for all available actions $a \in \{0, 1, ..., K\}$.

2) Draw action $a_n(t)$ according to the following distribution:

$$P(a) = \frac{(1-\alpha)\, e^{\beta Q(a)}}{\sum\limits_{\tilde{a} \in \{0,1,...,K\}} e^{\beta Q(\tilde{a})}} + \frac{\alpha}{K+1} \quad \forall a \in \{0, 1, ..., K\}, \quad (5)$$

for small $\alpha > 0$, and β is the temperature. The game is played over a time-horizon of T time slots.

IV. Experiments

In this section we present numerical experiments to illustrate the performance of the proposed algorithm. The network model follows the description in Section II.

A. Learning to Increase the Channel Throughput

Since there is no coordination between users, inefficient channel utilization occurs when no user accesses the channel (referred to as idle time slots) or whether two or more users access the channel at the same time slot (i.e., collisions). The channel throughput is the fraction of time that packets are successfully delivered over the channel, i.e., no collisions or idle time slots occur. We first examine the throughput of a single channel that the proposed algorithm can achieve under topology uncertainty. We simulated a large-scale network with disconnected cliques with a random number of users distributed uniformly between 3 and 11. At each clique, transmission is successful if only a single user in the clique transmits over a shared channel in a given time slot. There is no interference between users located at different cliques (e.g., uplink communication with scattered hotspots). We compared the following schemes: (i) *The slotted Aloha protocol with optimal attempt probability:* In this scheme each user at clique j transmits with probability p_j at each time slot. ALOHA-based protocols are widely used in wireless communication primarily because of their ease of implementation and their random nature. Setting $p_j = 1/n_j$ is known to be optimal from both fairness (proportional fairness [33], max-min fairness) and Nash bargaining [25] perspectives. We assume that users set their attempt probability to the optimal value $p_j = 1/n_j$ when implementing the slotted-Aloha scheme. (ii) *The proposed algorithm:* We implemented the proposed algorithm, in which each user has the freedom to choose any attempt probability at each time slot.

We are interested to address the following question: Under slotted-Aloha, the expected channel throughput (conditioned on n_j) is given by $n_j p_j (1-p_j)^{n_j - 1} = (1 - 1/n_j)^{n_j - 1} \in (0.385, 0.45)$ for $3 \leq n_j \leq 11$ and decreases to $e^{-1} \approx 0.37$ as n_j increases. *We are thus interested to examine whether the users can effectively learn in a fully distributed manner only from their ACK signals how to access the channel so as to increase the channel throughput by reducing the number of idle time slots and collisions.* To make this question more challenging, the actual number of users at each clique was unknown to the users when implementing the proposed algorithm (in contrast to the implementation of slotted-Aloha).

Figure 2 provides a positive answer to this question for the experiments that we did. The blue line shows the average channel throughput achieved by the proposed algorithm when each user aims at maximizing its own rate (i.e., competitive). The red line shows the average channel throughput achieved by the proposed algorithm when each user aims at maximizing the total number of successful packet transmissions (i.e., user sum rate). The yellow line is the well known channel throughput achieved by slotted-Aloha with optimal attempt probability and it serves as a benchmark for comparison. It can be seen that the proposed algorithm significantly outperforms the slotted-Aloha protocol very quickly under both utility functions. *The algorithm was able to deliver packets successfully almost* 80% *of the time, about twice the channel throughput as compared to slotted-Aloha with optimal attempt probability. This is achieved when each user learns only from its ACK signals, without online coordination, message exchanges between users, or carrier sensing.*

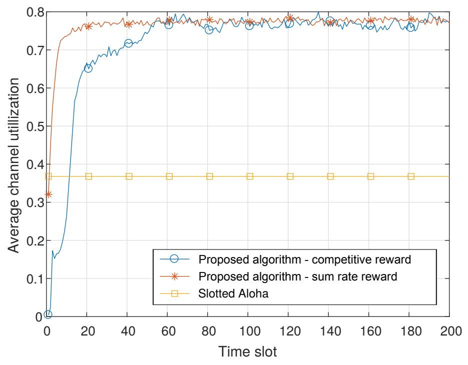

Fig. 2. Channel throughput for the experiments conducted in Section IV-A.

B. Achieving Efficient Rate Allocations Using Different Utility Functions

Channel throughput is an important measure for communication efficiency, but it does not provide an indication about the rate allocation among users. For example, if user 1 transmits 100% of the time and all other users receive rate zero, then the channel throughput is 1, but the solution might be undesirable. Hence, in this section we are interested to address the following question: *Can we train the DQN by different utilities so that the users can learn policies that result in good rate allocations depending on the desired performance?*

In what follows, we provide a positive answer to this question for the experiments that we did. We first simulated a network with 4 users and 2 channels. The DQN was trained when the users aim at maximizing the total number of successful packet transmissions (i.e., user sum rate). In Fig. 3 we show a representative example of the channel selection behavior among users. The presented policy after convergence (convergence was slightly slower occasionally than presented in the figure) is such that a single user transmits on each channel 100% of the time while the other users always do not transmit on that channel. In the figure, users $1, 4$ always transmit over channels $1, 2$, respectively, and users $2, 3$ do not transmit. This policy achieves a very good channel throughput (above 0.9). We observed such type of rate allocations in about 80% of the Monte-Carlo experiments. In terms of user sum rate, such type of rate allocations performs very well. Since each user contributes equally to the user sum rate, the users have learned a simple and efficient policy that achieves this goal.

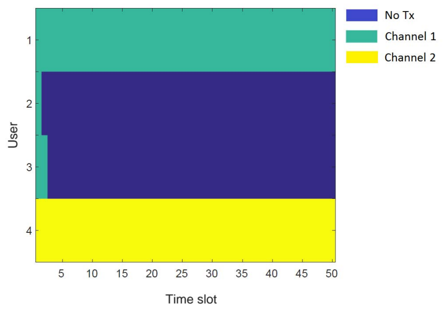

Fig. 3. A representative channel selection (observed in about 80% of the time) when maximizing the user sum rate.

It is well known that user sum rate is not a good choice when seeking for fair rate allocations. Indeed, from a fairness perspective the simple policy presented in Fig. 3 performs poorly, since 2 users receives rate zero. Hence, in the next simulation we examine the performance of the proposed algorithm when each user aims at maximizing its own individual rate (i.e., competitive reward). In Fig. 4 we show a representative example of the channel selection behavior among users observed in about 80% of the Monte-Carlo experiments. We simulated the case of 3 users and 2 channels. It can be seen that in this competitive reward case the users have learned a more complex protocol. Specifically, after convergence, a single user (user 3 in the figure) transmits on a single channel (channel 2 in the figure) 100% of the time, while the two other users (users $1, 2$ in the figure) share a channel (channel 1 in the figure) using TDMA, where each one of them transmits 50% of the time.

Interestingly, the experimental results are supported by the following insights: (i) The rate allocation presented in Fig. 4 is clearly better than the rate allocation presented in Fig. 3 from a fairness perspective. This result is reasonable since in Fig. 4 each user tries to reach a good operating point so as to maximize its own rate. Which one of the users succeeds better is affected by the initial conditions and randomness of the algorithm. (ii) From a game theoretic perspective, the Nash equilibrium of the competitive game (i.e., when each user aims at maximizing its own rate) is reached when each user transmits with probability 1 at each time slot, which is highly inefficient (for details see [34]). Thus, the fact that the DQN is trained to maximize a long-term reward for each user, pushes the users to learn a much better policy. (iii) Finally, note that the policy observed in Fig. 4 converges to a Pareto optimal operating point (for details see [34]), which further strengthens the experimental results obtained by the proposed algorithm from a game theoretic perspective.

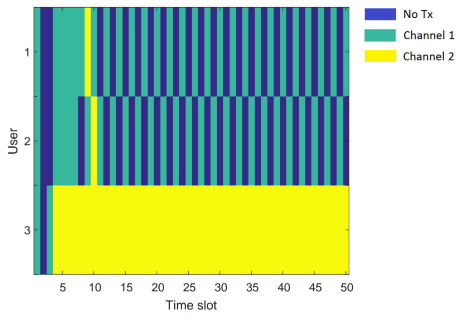

Fig. 4. A representative channel selection (observed in about 80% of the time) under individual utility maximization.

V. Conclusion

The problem of dynamic spectrum access for network utility maximization in multichannel wireless networks was considered. We developed a distributed dynamic spectrum access algorithm based on deep multi-user reinforcement leaning. The proposed algorithm enables each user to learn good policies in online and distributed manners, while dealing with the large state space without online coordination or message exchanges between users. Experimental results demonstrated strong performance of the algorithm in complex multi-user scenarios.

References

[1] Q. Zhao and B. Sadler, "A survey of dynamic spectrum access," *IEEE Signal Processing Magazine*, vol. 24, no. 3, pp. 79–89, 2007.

[2] A. Leshem, E. Zehavi, and Y. Yaffe, "Multichannel opportunistic carrier sensing for stable channel access control in cognitive radio systems," *IEEE Journal on Selected Areas in Communications*, vol. 30, pp. 82–95, Jan. 2012.

[3] L. M. Law, J. Huang, and M. Liu, "Price of anarchy for congestion games in cognitive radio networks," *IEEE Transactions on Wireless Communications*, vol. 11, no. 10, pp. 3778–3787, 2012.

[4] K. Cohen, A. Leshem, and E. Zehavi, "Game theoretic aspects of the multi-channel ALOHA protocol in cognitive radio networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, pp. 2276–2288, 2013.

[5] O. Naparstek and A. Leshem, "Fully distributed optimal channel assignment for open spectrum access," *IEEE Transactions on Signal Processing*, vol. 62, no. 2, pp. 283–294, 2014.

[6] Q. Zhao, L. Tong, A. Swami, and Y. Chen, "Decentralized cognitive MAC for opportunistic spectrum access in ad hoc networks: A POMDP framework," *IEEE Journal on selected areas in communications*, vol. 25, no. 3, 2007.

[7] S. H. A. Ahmad, M. Liu, T. Javidi, Q. Zhao, and B. Krishnamachari, "Optimality of myopic sensing in multichannel opportunistic access," *IEEE Transactions on Information Theory*, vol. 55, no. 9, pp. 4040–4050, 2009.

[8] K. Wang and L. Chen, "On optimality of myopic policy for restless multi-armed bandit problem: An axiomatic approach," *IEEE Transactions on Signal Processing*, vol. 60, no. 1, pp. 300–309, 2012.

[9] K. Cohen, Q. Zhao, and A. Scaglione, "Restless multi-armed bandits under time-varying activation constraints for dynamic spectrum access," in *48th Asilomar Conference on Signals, Systems and Computers*, pp. 1575–1578, 2014.

[10] H. Li, "Multiagent-learning for Aloha-like spectrum access in cognitive radio systems," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, no. 1, p. 876216, 2010.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[12] J. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, pp. 2137–2145, 2016.

[13] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[14] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[16] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access," in *International Conference on Computing, Networking and Communications (ICNC)*, 2017.

[17] U. Challita, L. Dong, and W. Saad, "Proactive resource management in LTE-U systems: A deep learning perspective," *arXiv preprint arXiv:1702.07031*, 2017.

[18] K. Liu and Q. Zhao, "Indexability of restless bandit problems and optimality of whittle index for dynamic multichannel access," *IEEE Transactions on Information Theory*, vol. 56, no. 11, pp. 5547–5567, 2010.

[19] C. Tekin and M. Liu, "Approximately optimal adaptive learning in opportunistic spectrum access," in *Proceedings IEEE INFOCOM*, pp. 1548–1556, 2012.

[20] H. Liu, K. Liu, and Q. Zhao, "Learning in a changing world: Restless multiarmed bandit with unknown dynamics," *IEEE Transactions on Information Theory*, vol. 59, no. 3, pp. 1902–1916, 2013.

[21] Z. Han, Z. Ji, and K. R. Liu, "Fair multiuser channel allocation for OFDMA networks using Nash bargaining solutions and coalitions," *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1366–1376, 2005.

[22] I. Menache and N. Shimkin, "Rate-based equilibria in collision channels with fading," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 7, pp. 1070–1077, 2008.

[23] U. O. Candogan, I. Menache, A. Ozdaglar, and P. A. Parrilo, "Competitive scheduling in wireless collision channels with correlated channel state," in *Game Theory for Networks, 2009. GameNets' 09. International Conference on*, pp. 621–630, 2009.

[24] H. Wu, C. Zhu, R. J. La, X. Liu, and Y. Zhang, "Fasa: Accelerated S-ALOHA using access history for event-driven M2M communications," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 6, pp. 1904–1917, 2013.

[25] K. Cohen and A. Leshem, "Distributed game-theoretic optimization and management of multichannel aloha networks," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1718–1731, 2016.

[26] K. Cohen, A. Nedić, and R. Srikant, "Distributed learning algorithms for spectrum sharing in spatial random access wireless networks," *IEEE Transactions on Automatic Control*, 2016.

[27] O. Naparstek, A. Leshem, and E. Jorswieck, "Distributed medium access control for energy efficient transmission in cognitive radios," *arXiv preprint arXiv:1401.1671*, 2014.

[28] Y. Gu, W. Saad, M. Bennis, M. Debbah, and Z. Han, "Matching theory for future wireless networks: fundamentals and applications," *IEEE Communications Magazine*, vol. 53, no. 5, pp. 52–59, 2015.

[29] R. Mochaourab, B. Holfeld, and T. Wirth, "Distributed channel assignment in cognitive radio networks: Stable matching and walrasian equilibrium," *IEEE Transactions on Wireless Communications*, vol. 14, no. 7, pp. 3924–3936, 2015.

[30] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[31] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," *arXiv preprint arXiv:1507.06527*, 2015.

[32] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning.," in *AAAI*, pp. 2094–2100, 2016.

[33] K. Kar, S. Sarkar, and L. Tassiulas, "Achieving proportional fairness using local information in Aloha networks," *IEEE Transactions on Automatic Control*, vol. 49, no. 10, pp. 1858–1863, 2004.

[34] O. Naparstek and K. Cohen, "Deep multi-agent reinforcement learning for dynamic spectrum access in cognitive wireless networks," *available at arXiv*.