

Deep-Reinforcement Learning Multiple Access for Heterogeneous Wireless Networks

Yiding Yu, Taotao Wang, Soungh Chang Liew

Abstract—This paper investigates the use of deep reinforcement learning (DRL) in a MAC protocol for heterogeneous wireless networking referred to as Deep-reinforcement Learning Multiple Access (DLMA). The thrust of this work is partially inspired by the vision of DARPA SC2, a 3-year competition whereby competitors are to come up with a clean-slate design that “best share spectrum with any network(s), in any environment, without prior knowledge, leveraging on machine-learning technique”. Specifically, this paper considers the problem of sharing time slots among a multiple of time-slotted networks that adopt different MAC protocols. One of the MAC protocols is DLMA. The other two are TDMA and ALOHA. The nodes operating DLMA do not know that the other two MAC protocols are TDMA and ALOHA. Yet, by a series of observations of the environment, its own actions, and the resulting rewards, a DLMA node can learn an optimal MAC strategy to coexist harmoniously with the TDMA and ALOHA nodes according to a specified objective (e.g., the objective could be the sum throughput of all networks, or a general α -fairness objective).

I. INTRODUCTION

This paper investigates a new generation of wireless multiple access control (MAC) protocol that leverages the latest advances in “deep reinforcement learning”. The work is partially inspired by our participation in the Spectrum Collaboration Challenge (SC2), a three-year competition hosted by DARPA of the United States [1].¹ Quoting DARPA, “SC2 is the first-of-its-kind collaborative machine-learning competition to overcome scarcity in the radio frequency (RF) spectrum. Today, spectrum is managed by dividing it into rigid, exclusively licensed bands. In SC2, competitors will reimagine a new, more efficient wireless paradigm in which radio networks autonomously collaborate to dynamically determine how the spectrum should be used moment to moment.” In other words, DARPA aims for a clean-slate design in which different wireless networks share spectrum in a very dynamic manner based on instantaneous supply and demand. In DARPA’s vision, “winning design is the one that best share spectrum with any network(s), in any environment, without prior knowledge, leveraging on machine-learning technique”. DARPA’s vision necessitates a total re-engineering of the PHY, MAC, and Network layers of wireless networks.

As a first step, this paper investigates a new MAC design that exploits deep Q-network (DQN) algorithm [2], a deep reinforcement learning (DRL) algorithm that combines deep neural networks [3] with Q-learning [4]. DQN was shown to

be able to achieve superhuman-level playing performance in video games. Our MAC design aims to learn an optimal way to use the time-spectrum resources by a series of observations and actions without the need to know the operating mechanisms of the MAC protocols of other coexisting networks. In particular, our MAC strives to achieve optimal performance as if it knew the MAC protocols of these networks in detail. In this paper, we refer to our MAC protocol as deep-reinforcement learning multiple access (abbreviated as DLMA), and a radio node operating DLMA as a DRL agent.

For a focus, this paper considers time-slotted systems and the problem of sharing the time slots among multiple wireless networks. In general, DLMA can adopt different objectives in time-slot sharing. We first consider the objective of maximizing the sum throughput of all the networks. We then reformulate DLMA to achieve a general α -fairness objective. In particular, we show that DLMA can achieve near-optimal sum throughput and proportional fairness when coexisting with a TDMA network, an ALOHA network, and a mix of TDMA and ALOHA networks, without knowing the coexisting networks are TDMA and ALOHA networks. Learning from the experience it gathers from a series of state-action-reward observations, a DRL agent tunes the weights of the neural network within its MAC machine to zoom to an optimal MAC strategy.

This paper also addresses the issue of why DRL is preferable to the traditional reinforcement learning (RL) [5] for wireless networking. Specifically, we demonstrate that the use of deep neural networks (DNN) in DRL affords us with two essential properties to wireless MAC: (i) fast convergence to near-optimal solutions; (ii) robustness against non-optimal parameter settings (i.e., fine parameter tuning and optimization are unnecessary with our DRL framework). Compared with MAC based on traditional RL, DRL converges faster and is more robust. Fast convergence is critical to wireless networks because the wireless environment may change quickly as new nodes arrive, and existing nodes move or depart. If the environmental “coherence time” is much shorter than the convergence time of the wireless MAC, the optimal strategy would elude the wireless MAC as it continuously tries to catch up with the environment. Robustness against non-optimal parameter settings is essential because the optimal parameter settings for DRL (and RL) in the presence of different coexisting networks may be different. Without the knowledge of the coexisting networks, DRL (and RL) cannot optimize its parameter settings a priori. If non-optimal parameter setting can also achieve roughly the same optimal throughput at roughly the same convergence rate, then optimal parameter settings are

Y. Yu, T. Wang and S. C. Liew are with the Department of Information Engineering, The Chinese University of Hong Kong. Emails:{yy016, ttwang, soungh}@ie.cuhk.edu.hk

¹Two of the authors are currently participating in this competition.

not essential for practical deployment.

In our earlier work [6], we adopted a plain DNN as the neural network in our DRL overall framework. In this work, we adopt a deep residual network (ResNet) [7]. The results of all sections in the current paper are based on ResNet, except Section III-E where we study deep ResNet versus plain DNN. A key advantage of ResNet over plain DNN is that the same static ResNet architecture can be used in DRL for different wireless network scenarios; whereas for plain DNN, the optimal neural network depth varies from case to case.

Overall, our main contributions are as follows:

- We employ DRL for the design of DLMA, a MAC protocol for heterogeneous wireless networking. Our DLMA framework is formulated to achieve general α -fairness among the heterogeneous networks. Extensive simulation results show that DLMA can achieve near-optimal sum throughput and proportional fairness objectives. In particular, DLMA achieves these objectives without knowing the operating mechanisms of the MAC protocols of the other coexisting networks.
- We demonstrate the advantages of exploiting DRL in heterogeneous wireless networking compared with the traditional RL method. In particular, we show that DRL can accelerate convergence to an optimal solution and is more robust against non-optimal parameter settings, two essential properties for practical deployment of DLMA in real wireless networks.
- In the course of our generalization to the α -fairness objective in wireless networking, we discovered an approach to generalize the Q-learning framework so that more general objectives can be achieved. In particular, we argue that for generality, we need to separate the Q function and the objective function upon which actions are chosen to optimize – in conventional Q learning, the Q function itself is the objective function. We give a framework on how to relate the objective function and the Q function in the general set-up.

A. Related Work

RL is a machine-learning paradigm, where agents learn successful strategies that yield the largest long-term reward from trial-and-error interactions with their environment [5]. The most representative RL algorithm is the Q-learning algorithm [4]. Q-learning can learn a good policy by updating an action-value function, referred to as the Q function, without an operating model of the environment. When the state-action space is large and complex, deep neural networks can be used to approximate the Q function and the corresponding algorithm is called DRL [2]. This work employs DRL to speed up convergence and increase the robustness of DLMA (see our results Section III-D).

RL was employed to develop channel access schemes for cognitive radios [8]–[11] and wireless sensor networks [12], [13]. Unlike this paper, these works do not leverage the recent advances in DRL.

There has been little prior work exploring the use of DRL to solve MAC problems, given that DRL itself is a new

research topic. The MAC scheme in [14] employs DRL in homogeneous wireless networks. Specifically, [14] considered a network in which N radio nodes dynamically access K orthogonal channels using the same DRL MAC protocol. By contrast, we are interested in heterogeneous networks in which the DRL nodes must learn to collaborate with nodes employing other MAC protocols.

The authors of [15] proposed a DRL-based channel access scheme for wireless sensor networks. Multiple frequency channels were considered. In RL terminology, the multiple frequency channels with the associated Markov interference models form the “environment” with which the DRL agent interacts. There are some notable differences between [15] and our investigation here. The Markov environmental model in [15] cannot capture the interactions among nodes due to their MAC protocols. In particular, the Markov environmental model in [15] is a “passive” model not affected by the “actions” of the DRL agent. For example, if there is one exponential backoff ALOHA node (see Section II-A for definition) transmitting on a channel, the collisions caused by transmissions by the DRL agent will cause the channel state to evolve in intricate ways not captured by the model in [15].

In [16], the authors employed DRL for channel selection and channel access in LTE-U networks. Although it also aims for heterogeneous networking in which LTE-U base stations coexist with WiFi APs, its focus is on matching downlink channels to base stations; we focus on sharing an uplink channel among users. More importantly, the scheme in [16] is model-aware in that the LTE-U base stations know that the other networks are WiFi. For example, it uses an analytical equation (equation (1) in [16]) to predict the transmission probability of WiFi stations. By contrast, our DLMA protocol is model-free in that it does not presume knowledge of coexisting networks and is outcome-based in that it derives information by observing its interactions with the other stations in the heterogeneous environment.

II. DLMA PROTOCOL

This section first introduces the time-slotted heterogeneous wireless networks considered in this paper. Then a short overview of RL is given. After that, we present our DLMA protocol, focusing on the objective of maximizing the sum throughput of the overall system. A generalized DLMA protocol that can achieve α -fairness objective will be given in Section IV.

A. Time-Slotted Heterogeneous Wireless Networks

We consider time-slotted heterogeneous wireless networks in which different radio nodes transmit packets to an access point (AP) via a shared wireless channel, as illustrated in Fig. 1. We assume all the nodes can begin transmission only at the beginning of a time slot and must finish transmission within that time slot. Simultaneous transmissions of multiple nodes in the same time slot result in collisions. The nodes may not use the same MAC protocol: some may use TDMA and/or ALOHA, and at least one node uses our proposed DLMA

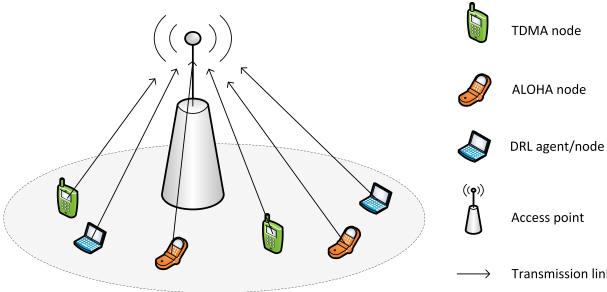


Fig. 1: A heterogeneous multiple-access system with a mix of DRL nodes and other nodes.

protocol. The detailed descriptions of different radio nodes are given below:

- **TDMA:** The TDMA node transmits in X specific slots within each frame of Y slots in a repetitive manner from frame to frame.
- **q -ALOHA:** A q -ALOHA node transmits with a fixed probability q in each time slot in an i.i.d. manner from slot to slot.
- **Fixed-window ALOHA:** A fixed-window ALOHA (FW-ALOHA) node generates a random counter value w in the range of $[0, W - 1]$ after it transmits in a time slot. It then waits for w slots before its next transmission. The parameter W is referred to as the window size.
- **Exponential backoff ALOHA:** Exponential backoff ALOHA (EB-ALOHA) is a variation of window-based ALOHA in which the window size is not fixed. Specifically, an EB-ALOHA node doubles its window size each time when its transmission encounters a collision, until a maximum window size $2^m W$ is reached, where m is the “maximum backoff stage”. Upon a successful transmission, the window size reverts back to the initial window size W .
- **DRL agent/node:** A DRL agent/node is the radio node that adopts our DLMA protocol. For a DRL node, if it transmits, it will get an immediate ACK from the AP, indicating whether the transmission is successful or not; if it does not transmit, it will listen to the channel and get an observation from the environment, indicating other nodes’ transmission results or idleness of the channel. Based on the observed results, the DRL node can set different objectives, such as maximizing the sum throughput of the overall system (as formulated in Part C of this section) and achieving a general α -fairness objective (as formulated in Section IV).

B. Overview of RL

In RL [5], an agent interacts with an environment in a sequence of discrete times, $t = 0, 1, 2, \dots$, to accomplish a task, as shown in Fig. 2. At time t , the agent observes the state of the environment $s_t \in S$, where S is the set of possible states. It then takes an action $a_t \in A_{s_t}$, where A_{s_t} is the set of possible actions at state s_t . As a result of the state-action pair (s_t, a_t) , the agent receives a reward r_{t+1} , and the environment moves to a new state s_{t+1} at time $t + 1$. The goal of the

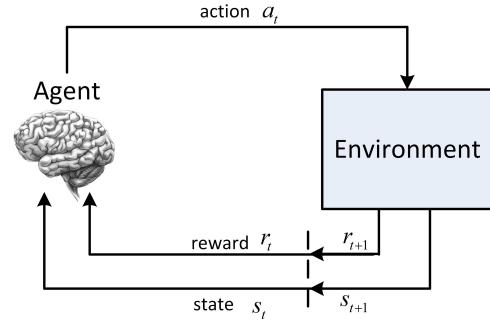


Fig. 2: The agent-environment interaction process.

agent is to effect a series of rewards $\{r_t\}_{t=1,2,\dots}$ through its actions to maximize some performance criteria. For example, the performance criterion to be maximized at time t could be $R_t \stackrel{\Delta}{=} \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_{\tau+1}$, where $\gamma \in (0, 1]$ is a discount factor for weighting future rewards. In general, the agent takes actions according to some decision policy π . RL methods specify how the agent changes its policy as a result of its experiences. With sufficient experiences, the agent can learn an optimal decision policy π^* to maximize the long-term accumulated reward [5].

Q-learning [4] is one of the most popular RL methods. A Q-learning RL agent learns an action-value function $Q^\pi(s, a)$ corresponding to the expected accumulated reward when an action a is taken in the environmental state s under the decision policy π :

$$Q^\pi(s, a) \stackrel{\Delta}{=} E[R_t | s_t = s, a_t = a, \pi]. \quad (1)$$

The optimal action-value function, $Q^*(s, a) \stackrel{\Delta}{=} \max_\pi Q^\pi(s, a)$, obeys the Bellman optimality equation [5]:

$$Q^*(s, a) = E_{s'} \left[r_{t+1} + \gamma \max_{a'} Q^*(s', a') | s_t = s, a_t = a \right], \quad (2)$$

where s' is the new state after the state-action pair (s, a) . The main idea behind Q-learning is that we can iteratively estimate $Q^*(s, a)$ at the occurrences of each state-action pair (s, a) .

Let $q(s, a)$ be the estimated action-value function during the iterative process. Upon a state-action pair (s_t, a_t) and a resulting reward r_{t+1} , Q-learning updates $q(s_t, a_t)$ as follows:

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \beta \left[r_{t+1} + \gamma \max_{a'} q(s_{t+1}, a') - q(s_t, a_t) \right], \quad (3)$$

where $\beta \in (0, 1]$ is the learning rate.

While the system is updating $q(s, a)$, it also makes decisions based on $q(s, a)$. The ε -greedy policy is often adopted, i.e.,

$$a = \begin{cases} \arg \max_{\tilde{a}} q(s, \tilde{a}), & \text{with probability } 1 - \varepsilon, \\ \text{random action,} & \text{with probability } \varepsilon. \end{cases} \quad (4)$$

A reason for randomly selecting an action is to avoid getting stuck with a $q(s, a)$ function that has not yet converged to $Q^*(s, a)$.

C. DLMA Protocol Using DRL

This subsection describes the construction of our DLMA protocol using the DRL framework.

The *action* taken by a DRL agent at time t is $a_t \in \{\text{TRANSMIT}, \text{WAIT}\}$, where *TRANSMIT* means that the agent transmits, and *WAIT* means that the agent does not transmit. We denote the *channel observation* after taking action a_t by $z_t \in \{\text{SUCCESS}, \text{COLLISION}, \text{IDLENESS}\}$, where *SUCCESS* means one and only one station transmits on the channel; *COLLISION* means multiple stations transmit, causing a collision; *IDLENESS* means no station transmits. The DRL agent determines z_t from an ACK signal from the AP (if it transmits) and listening to the channel (if it waits). We define the *channel state* at time $t+1$ as the action-observation pair $c_{t+1} \triangleq (a_t, z_t)$. There are five possibilities for c_{t+1} : $\{\text{TRANSMIT}, \text{SUCCESS}\}$, $\{\text{TRANSMIT}, \text{COLLISION}\}$, $\{\text{WAIT}, \text{SUCCESS}\}$, $\{\text{WAIT}, \text{COLLISION}\}$ and $\{\text{WAIT}, \text{IDLENESS}\}$. We define the *environmental state* at time $t+1$ to be $s_{t+1} \triangleq \{c_{t-M+2}, \dots, c_t, c_{t+1}\}$, where the parameter M is the state history length to be tracked by the agent. After taking action a_t , the transition from state s_t to s_{t+1} generates a *reward* r_{t+1} , where $r_{t+1} = 1$ if $z_t = \text{SUCCESS}$; $r_{t+1} = 0$ if $z_t = \text{COLLISION}$ or *IDLENESS*. The definition of reward here corresponds to the objective of maximizing the sum throughput. We define a reward vector in Section IV so as to generalize DLMA to achieve the α -fairness objective.

So far, the above definitions of “action”, “state” and “reward” also apply to an RL agent that adopts (3) as its learning algorithm. We next motivate the use of DRL and then provide the details of its use.

Intuitively, subject to a non-changing or slow-changing environment, the longer the state history length M , the better the decision can be made by the agent. However, a large M induces a large state space for the RL algorithm. With a large number of state-action entries to be tracked, the step-by-step and entry-by-entry update $q(s, a)$ is very inefficient. To get a rough idea, suppose that $M = 10$ (a rather small state history to keep track of), then there are $5^{10} \approx 10$ million possible values for state s . Suppose that for convergence to the optimal solution, each state-action value must be visited at least once. If each time slot is 1 ms in duration (typical wireless packet transmission time), the convergence of RL will take at least $5^{10} \times 2$ ms, or more than 5 hours. Due to node mobility, arrivals and departures, the wireless environment will most likely to have changed well before then. Section III-D of this paper shows that applying DRL to DLMA accelerates the convergence speed significantly (convergence is obtained in seconds, not hours).

In DRL, a deep neural network [3] is used to approximate the action-value function, $q(s, a; \theta) \approx Q^*(s, a)$, where $q(s, a; \theta)$ is the approximation given by the neural network and θ is a parameter vector containing the weights of the edges in the neural network. The input to the neural network is a state s , and the outputs are approximated q values for different actions $\mathbb{Q} = \{q(s, a; \theta) | a \in A_s\}$. We refer to the neural network as the Q neural network (QNN) and the corresponding RL algorithm as DRL. Rather than following the tabular update rule of the

traditional RL in (3), DRL updates $q(s, a; \theta)$ by adjusting the θ in the QNN through a training process.

In particular, QNN is trained by minimizing prediction errors of $q(s, a; \theta)$. Suppose that at time t , the state is s_t and the weights of QNN are θ . The DRL agent takes an action $a_t = \arg \max_a q(s_t, a; \theta)$, where $q(s_t, a; \theta)$ for different actions a are given by the outputs of QNN. Suppose that the resulting reward is r_{t+1} and the state moves to s_{t+1} . Then, $(s_t, a_t, r_{t+1}, s_{t+1})$ constitutes an “experience sample” that will be used to train the QNN. For training, we define the prediction error of QNN for the particular experience sample $(s_t, a_t, r_{t+1}, s_{t+1})$ to be

$$L_{s_t, a_t, r_{t+1}, s_{t+1}}(\theta) = \left(y_{r_{t+1}, s_{t+1}}^{\text{QNN}} - q(s_t, a_t; \theta) \right)^2, \quad (5)$$

where θ are the weights in QNN, $q(s_t, a_t; \theta)$ is the approximation given by QNN, and $y_{r_{t+1}, s_{t+1}}^{\text{QNN}}$ is the target output for QNN given by

$$y_{r_{t+1}, s_{t+1}}^{\text{QNN}} = r_{t+1} + \gamma \max_{a'} q(s_{t+1}, a'; \theta). \quad (6)$$

Note that $y_{r_{t+1}, s_{t+1}}^{\text{QNN}}$ is a refined target output based on the current reward r_{t+1} plus the predicted discounted rewards going forward $\gamma \max_{a'} q(s_{t+1}, a'; \theta)$ given by QNN. We can train QNN, i.e., update θ , by applying a semi-gradient algorithm [5] in (5). The iteration process of θ is given by

$$\text{Iterate } \theta \leftarrow \theta - \rho \left[y_{r_{t+1}, s_{t+1}}^{\text{QNN}} - q(s_t, a_t; \theta) \right] \nabla q(s_t, a_t; \theta), \quad (7)$$

where ρ is the step size in each adjustment.

For algorithm stability, the “experience replay” and “quasi-static target network” techniques can be used [2]. For “experience replay”, instead of training QNN with a single experience at the end of each execution step, we could pool together many experiences for batch training. In particular, an experience memory with a fixed storage capacity is used for storing the experiences $e = (s, a, r, s')$ gathered from different time steps in an FIFO manner, i.e., once the experience memory is full, the oldest experience is removed from, and the new experience is put into, the experience memory. For a round of training, a minibatch E consisting of N_E random experiences are taken from the experience memory for the computation of the loss function. For “quasi-static target network”, a separate target QNN with parameter θ^- is used as the target network for training purpose. Specifically, the $q(\cdot)$ in (6) is computed based on this separate target QNN, while the $q(\cdot)$ in (5) is based on QNN under training. The target QNN is a copy of an earlier QNN: every F time steps, the target QNN is replaced by the latest QNN, i.e., setting θ^- to the latest θ of QNN. With these two techniques, equations (5), (6), and (7) replaced by the following:

$$L_E(\theta) = \frac{1}{N_E} \sum_{e \in E} \left(y_{r, s'}^{\text{QNN}} - q(s, a; \theta) \right)^2, \quad (8)$$

$$y_{r, s'}^{\text{QNN}} = r + \gamma \max_{a'} q(s', a'; \theta^-), \quad (9)$$

$$\text{Iterate } \theta \leftarrow \theta - \frac{\rho}{N_E} \sum_{e \in E} \left[y_{r, s'}^{\text{QNN}} - q(s, a; \theta) \right] \nabla q(s, a; \theta), \quad (10)$$

$$\text{Every } F \text{ steps, set } \theta^- \text{ to } \theta. \quad (11)$$

The pseudocode of DLMA algorithm is given in Algorithm 1.

Algorithm 1 DLMA with the sum throughput objective

```

Initialize  $s_0, \varepsilon, \gamma, \rho, N_E, F$ 
Initialize experience memory  $EM$ 
Initialize the parameter of QNN as  $\theta$ 
Initialize the parameter of target QNN  $\theta^- = \theta$ 
for  $t = 0, 1, 2, \dots$  in DLMA do
    Input  $s_t$  to QNN and output  $\mathbb{Q} = \{q(s_t, a, \theta) | a \in A_{s_t}\}$ 
    Generate action  $a_t$  from  $\mathbb{Q}$  using  $\varepsilon$ -greedy algorithm
    Observe  $z_t, r_{t+1}$ 
    Compute  $s_{t+1}$  from  $s_t, a_t$  and  $z_t$ 
    Store  $(s_t, a_t, r_{t+1}, s_{t+1})$  to  $EM$ 
    if Remainder( $t/F == 0$ ) then  $I = 1$  else  $I = 0$ 
    TRAINQNN( $\gamma, \rho, N_E, I, EM, \theta, \theta^-$ )2
end for

procedure TRAINQNN( $\gamma, \rho, N_E, I, EM, \theta, \theta^-$ )
    Randomly sample  $N_E$  experience tuples from  $EM$  as  $E$ 
    for each sample  $e = (s, a, r, s')$  in  $E$  do
        Calculate  $y_{r,s'}^{QNN} = r + \gamma \max_{a'} q(s', a'; \theta^-)$ 
    end for
    Perform Gradient Descent to update  $\theta$  in QNN:
    Iterate  $\theta \leftarrow \theta - \frac{\rho}{N_E} \sum_{e \in E} [y_{r,s'}^{QNN} - q(s, a; \theta)] \nabla q(s, a; \theta)$ 
    if  $I == 1$  then
        Update  $\theta^-$  in target QNN by setting  $\theta^- = \theta$ 
    end if
end procedure

```

III. SUM THROUGHPUT PERFORMANCE EVALUATION

This section investigates the performance of DLMA with the objective of maximizing the sum throughput of all the coexisting networks. For our investigations, we consider the interactions of DRL nodes with TDMA nodes, ALOHA nodes, and a mix of TDMA nodes and ALOHA nodes. Section IV will reformulate the DLMA framework to achieve a general α -fairness objective (which also includes maximizing sum-throughput objective as a subcase); Section V will present the corresponding results.

As illustrated in Fig. 3, the architecture of QNN used in DLMA is a six-hidden-layer ResNet with 64 neurons in each

²For convenience, in our simulation, we assume execution of decisions and training of QNN are run synchronously. In particular, the training is done at the end of each time step after an execution. In practice, for efficiency and to allow more time for training, execution and training can be done asynchronously and in parallel. In this case, the experiences resulted from executions in successive time steps are fed to the experience memory in a continuous manner. Meanwhile, training takes random minibatches of experiences from the experience memory in a parallel and asynchronous fashion continuously. There could be more than one training round (i.e., more than one minibatches used for training) per execution time step if sufficient computation resources are available. Once in a while, the QNN used in execution is replaced by the newly trained QNN by substituting the θ in QNN with the new θ in the newly trained QNN; at the same time θ^- in the target QNN is also replaced by the new θ for future training purposes.

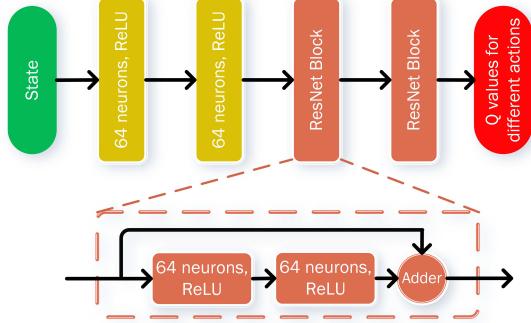


Fig. 3: ResNet Architecture

hidden layer. The activation functions used for the neurons are $ReLU$ functions [3]. The first two hidden layers of QNN are fully connected, followed by two ResNet blocks. Each ResNet block contains two fully connected hidden layers plus one “shortcut” from the input to the output of the ResNet block. The state, action and reward of DRL follow the definitions in Section II-C. The state history length M is set to 20, unless stated otherwise. When updating the weights θ of QNN, a minibatch of 32 experience samples are randomly selected from an experience-replay reservoir of 500 prior experiences for the computation of the loss function (8). The experience-replay reservoir is updated in a FIFO manner: a new experience replaces the oldest experience in it. The *RMSProp* algorithm [17] is used to conduct minibatch gradient descent for the update of θ . To avoid getting stuck with a suboptimal decision policy before sufficient learning experiences, we apply an exponential decay ε -greedy algorithm: ε is initially set to 0.1 and decreases at a rate of 0.995 every time slot until its value reaches 0.005. A reason for not decreasing ε all the way to zero is that in a general wireless setting, the wireless environment may change dynamically with time (e.g., nodes are leaving and joining the network). Having a positive ε at all time allows the decision policy to adapt to future changes. Table I summarizes the hyper-parameter settings in our investigations.

TABLE I: DLMA Hyper-parameters

Hyper-parameters	Value
State history length M	20, unless stated otherwise
Discount factor γ	0.9
ε in ε -greedy algorithm	0.1 to 0.005
Learning rate used in RMSProp	0.01
Target network update frequency F	200
Experience-replay minibatch size N_E	32
Experience-replay memory capacity	500

A salient feature of our DLMA framework is that it is model-free (it does not need to know the protocols adopted by other coexisting nodes). For benchmarking, we consider model-aware nodes. Specifically, a model-aware node knows the MAC mechanisms of coexisting nodes, and it executes an optimal MAC protocol derived from this knowledge. We will show that our model-free DRL node can achieve near-optimal throughput with respect to the optimal throughput of the model-aware node. The derivations of the optimal throughputs

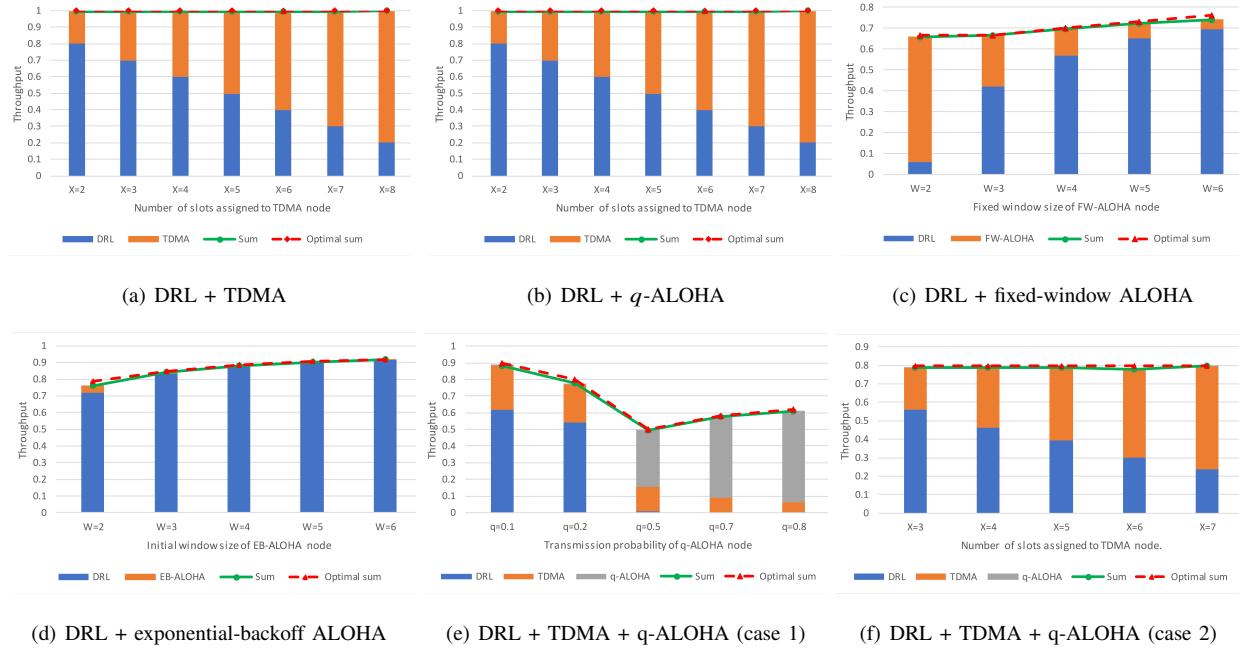


Fig. 4: Sum and individual throughputs for different cases with the objective of maximizing sum throughput.

for different cases below, which are interesting in their own right, are provided in [18]. We omit them here to save space.

A. Coexistence with TDMA networks

We first present the results of the coexistence of one DRL node with one TDMA node. The TDMA node transmits in X specific slots within each frame of Y slots in a repetitive manner from frame to frame. For benchmarking, we consider a TDMA-aware node which has full knowledge of the X slots used by the TDMA node. To maximize the overall system throughput, the TDMA-aware node will transmit in all the $Y - X$ slots not used by the TDMA node. The optimal sum throughput is one packet per time slot. The DRL agent, unlike the TDMA-aware node, does not know that the other node is a TDMA node (as a matter of fact, it does not even know how many other nodes there are) and just uses the DRL algorithm to learn the optimal strategy.

Fig. 4(a) presents the throughput³ results when $Y = 10$ and X varies from 2 to 8. The green line is the sum throughput of the DRL node and the TDMA node. We see that it is very close to 1. This demonstrates that the DRL node can capture all the unused slots without knowing the TDMA protocol adopted by the other node.

B. Coexistence with ALOHA networks

We next present the results of the coexistence of one DRL node with one q -ALOHA, one FW-ALOHA and one EB-ALOHA, respectively. We emphasize that the exact same

³Unless stated otherwise, “throughput” in this paper is the “short-term throughput”, calculated as $\sum_{\tau=t-N+1}^t r_\tau / N$, where $N = 1000$. If one time step is 1 ms in duration, then this is the throughput over the past second. In the bar charts presented in this paper, “throughput” is the average reward over the last N steps in an experiment with a length of 50000 steps and we take the average of 10 experiments for each case to get the final value.

DLMA algorithm as in Part A is used here even though the other protocols are not TDMA anymore. For benchmarking, we consider model-aware nodes that operate with optimal MACs tailored to the operating mechanisms of the three ALOHA variants [18].

Fig. 4(b) presents the experimental results for the coexistence of one DRL node and one q -ALOHA node. The results show that the DRL node can learn the strategy to achieve the optimal throughputs despite the fact that it is not aware that the other node is q -ALOHA node and what the transmission probability q is. Fig. 4(c) presents the results for the coexistence of one DRL node and one FW-ALOHA node with different fixed-window sizes. Fig. 4(d) presents the results for the coexistence of one DRL node and one EB-ALOHA node with different initial window sizes and maximum backoff stage $m = 2$. As shown, DRL node can again achieve near-optimal throughputs for these two cases.

C. Coexistence with a mix of TDMA and ALOHA networks

We now present the results of a set-up in which one DRL agent coexists with one TDMA node and one q -ALOHA node simultaneously. Again, the same DLMA algorithm is used. We consider two cases. In the first case, the TDMA node transmits in 3 slots out of 10 slots in a frame; the transmission probability q of the q -ALOHA node varies. In the second case, q of the q -ALOHA node is fixed to 0.2; X , the number of slots used by the TDMA nodes in a frame, varies. Fig. 4(e) and Fig. 4(f) present the results of the first and second cases respectively. For both cases, we see that our DRL node can approximate the optimal results without knowing the transmission schemes of the TDMA and q -ALOHA nodes.

We next consider a setup in which multiple DRL nodes coexist with a mix of TDMA and q -ALOHA nodes. Specifically, the setup consists of three DRL nodes, one TDMA

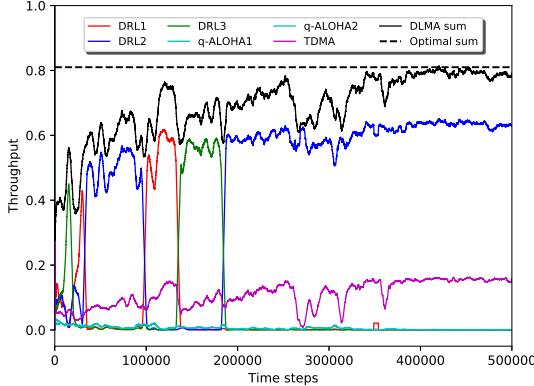


Fig. 5: Sum throughput and individual throughputs under the coexistence of three DRL nodes with one TDMA node and two q -ALOHA nodes. The throughputs at time t are computed by $\sum_{\tau=t-N+1}^t r_\tau/N$, where $N = 5000$.

node that transmits in 2 slots out of 10 slots in a frame, and two q -ALOHA nodes with transmission probability $q = 0.1$. In Fig. 5, we can see that DLMA can also achieve near-optimal sum throughput in this more complex setup. However, when we focus on the individual throughputs of each node, we find that since there is no coordination among the three DRL nodes, one DRL node may preempt all the slots not occupied by the TDMA node, causing the other two DRL nodes and two q -ALOHA nodes get zero throughputs. This observation motivates us to consider fairness among different nodes in Section IV.

D. RL versus DRL

We now present results demonstrating the advantages of the “deep” approach using the scenario where one DRL/RL agent coexists with one TDMA node. Fig. 6 compares the convergence time of the Q-learning based RL approach and the QNN-based DRL approach. The sum throughput in the figure is the “cumulative sum throughput” starting from the beginning: $\sum_{\tau=1}^t r_\tau/t$. It can be seen that DRL converges to the optimal throughput of 1 at a much faster rate than RL does. For example, DRL requires only less than 5000 steps (5 s if each step corresponds to a packet transmission time of 1 ms) to approach within 80% of the optimal throughput. Note that when state history length increases from 10 to 16, RL learns progressively slower and slower, but the convergence time of DRL varies only slightly as M increases. In general, for a model-free MAC protocol, we do not know what other MAC protocols there are besides our MAC protocol. Therefore, we will not optimize on M and will likely use a large M to cater for a large range of other possible MAC protocols. The robustness, in terms of insensitivity of convergence time to M , is a significant practical advantage of DRL.

Fig. 7 presents the throughput evolutions of TDMA+RL and TDMA+DRL versus time. Unlike in Fig. 6, in Fig. 7, the sum throughput is the “short-term sum throughput” rather than the “cumulative sum throughput” starting from the beginning.

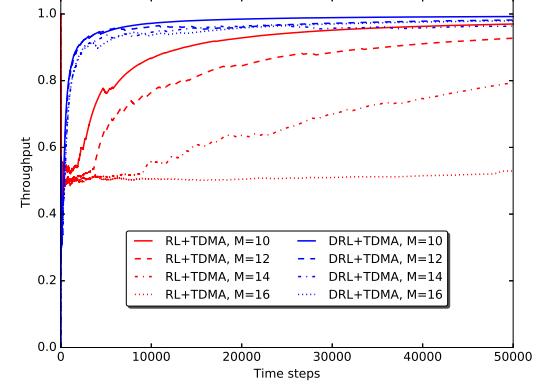


Fig. 6: Convergence speeds of RL and DRL nodes. The sum throughput at time t is computed by $\sum_{\tau=1}^t r_\tau/t$.

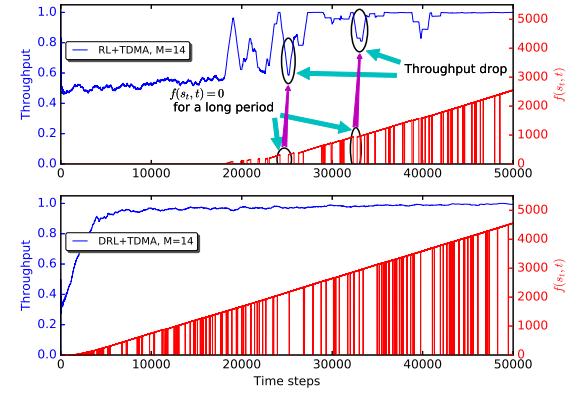


Fig. 7: Throughput evolution and cumulative number of visits to state s_t prior to time t , $f(s_t, t)$, where s_t is the particular state being visited at time t , for RL+TDMA and for DRL+TDMA. The sum throughput at time t is computed by $\sum_{\tau=t-N+1}^t r_\tau/N$, where $N = 1000$.

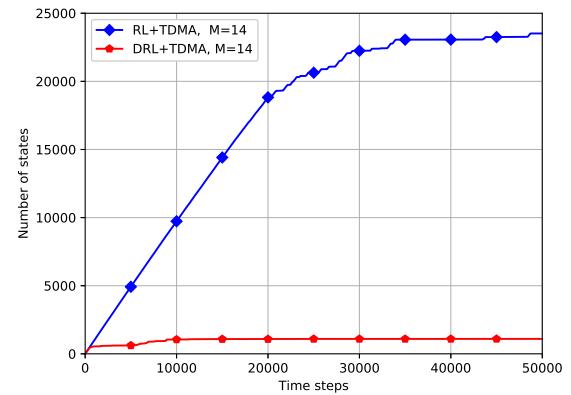
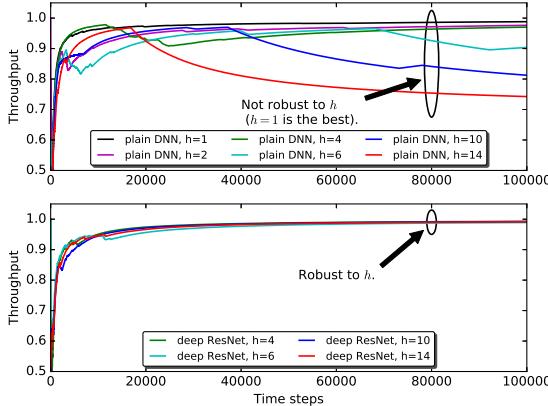
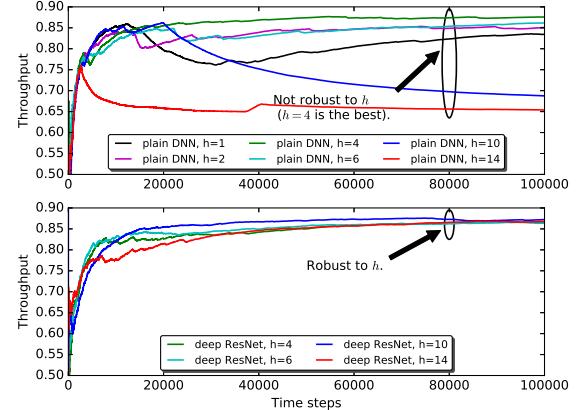


Fig. 8: The number of distinct states visited by RL/DRL agent.

Specifically, the sum throughput in Fig. 7 is $\sum_{\tau=t-N+1}^t r_\tau/N$, where $N = 1000$. If one time step is 1 ms in duration, then this is the throughput over the past second. As can be



(a) DRL + TDMA



(b) DRL + TDMA + q-ALOHA

Fig. 9: Performance comparison between plain DNN and deep ResNet based approaches. The cumulative sum throughput at time t is computed by $\sum_{\tau=1}^t r_\tau / t$.

seen, although both RL and DRL can converge to the optimal throughput in the end, DRL takes a much shorter time to do so. Furthermore, the fluctuations in throughput experienced by RL along the way are much larger. To dig deeper into this phenomenon, we examine $f(s, t)$, defined to be the number of previous visits to state s prior to time step t . Fig. 7 also plots $f(s_t, t)$: i.e., we look at the number of previous visits to state s_t before visiting s_t , the particular state being visited at time step t . As can be seen, for RL, each drop in the throughput coincides with a visit to a state s_t with $f(s_t, t) = 0$. In other words, the RL algorithm has not learned the optimal action for this state yet because of the lack of prior visits. From Fig. 7, we also see that it takes a while before RL extricates itself from persistent and consecutive visits to a number of states with $f(\cdot) = 0$. This persistency results in large throughput drops until RL extricates itself from the situation. By contrast, although DRL also occasionally visits a state s_t with $f(s_t, t) = 0$, it is able to take an appropriate action at the unfamiliar territory (due to the “extrapolation” ability of the neural network to infer a good action to take at s_t based on prior visits to states other than s_t : recall that each update of θ changes the values of $q(s, a, \theta)$ for all (s, a) , not just that of a particular (s, a)). DRL manages to extricate itself from unfamiliar territories quickly and evolve back to optimal territories where it only transmits in time slots not used by TDMA.

Fig. 8 presents the evolutions of the number of distinct states visited by RL and DRL agents in the same experiment as in Fig. 7. In this case, both RL and DRL find an optimal strategy in the end, but RL requires more time to do so. Once the optimal strategies are found, RL and DRL agents seldom explore new states, except the “exploration” step in ε -greedy algorithm. As indicated in Fig. 8, the number of distinct states visited by RL on its journal to the optimal strategy is much larger than that of DRL. From Fig. 8, we see that RL spends 35000 time steps in finding the optimal strategy, having visited 23000 distinct states before doing so. By contrast, it takes only

10000 time steps for DRL to find the optimal strategy and the number of distinct states visited is only around 1000. In other words, DRL can better narrow down its choice of states to visit in order to find the optimal strategy, hence the faster convergence speed.

E. Plain DNN versus deep ResNet

We now demonstrate the advantages of deep ResNet over plain DNN using two cases: 1) one DRL node coexisting with one TDMA node, wherein the TDMA node occupies 2 slots out of 10 slots in a frame; 2) one DRL node coexisting with one TDMA node and one q-ALOHA node, wherein the TDMA is the same as in 1) and $q = 0.1$ for the q-ALOHA node. The optimal sum throughputs for a model-aware protocol for 1) and 2) can be established analytically to be 1 and 0.9, respectively (see [18] for the derivation). For each case, we compare the cumulative sum throughputs of plain DNN based approach and deep ResNet based approach with different numbers of hidden layers h .

As can be seen from the upper parts of Fig. 9(a) and Fig. 9(b), the plain DNN is not robust against variation of h , i.e., the performance varies with h . Furthermore, $h = 1$ and $h = 4$ achieve the best performance for case 1) and 2), respectively. This implies that it is difficult to use a common plain DNN architecture for different wireless setups. In other words, the optimal h may be different under different scenarios. If the environment changes dynamically, there is no one single h that is optimal for all scenarios. In contrast to plain DNN’s non-robustness to h , deep ResNet can always achieve near-optimal performance for different h for both cases, as illustrated in the lower parts of Fig. 9(a) and Fig. 9(b).

For wireless networking, the environment may change quickly when new nodes arrive, and existing nodes move or depart. It is desirable to adopt a one-size-fits-all neural network architecture in DRL. Our results show that deep ResNet is more desirable than plain DNN in this regard.

IV. GENERAL OBJECTIVE DLMA PROTOCOL

This section first introduces the well-known α -fairness utility function [19]. Then, a multi-dimensional Q-learning algorithm is proposed to incorporate the α -fairness utility function in a general reformulation of DLMA.

A. α -fairness objective

Instead of sum throughput, we now adopt the α -fairness index as the metric of the overall system performance. The parameter $\alpha \in [0, \infty)$ is used to specify a range of the fairness criteria, e.g., when $\alpha = 0$, maximizing the α -fairness objective corresponds to maximizing the sum throughput (the corresponding results were presented in Section III); when $\alpha = 1$, maximizing the α -fairness objective corresponds to achieving proportional fairness; when $\alpha \rightarrow \infty$, the minimum throughput among nodes is being maximized. Specifically, we consider a system with N nodes and for a particular node i , its throughput is denoted by $x^{(i)}$; its α -fairness local utility function is given by

$$f_\alpha^{(i)}(x^{(i)}) = \begin{cases} \log(x^{(i)}), & \text{if } \alpha = 1, \\ (1 - \alpha)^{-1}(x^{(i)})^{1-\alpha}, & \text{if } \alpha \neq 1. \end{cases} \quad (12)$$

The objective of the overall system is to maximize the sum of all the local utility functions:

$$\begin{aligned} & \text{maximize} \quad F(x^{(1)}, x^{(2)}, \dots, x^{(N)}) = \sum_{i=1}^N f_\alpha^{(i)}(x^{(i)}) \\ & \text{subject to} \quad \sum_{i=1}^N x^{(i)} \leq 1, \\ & \quad x^{(i)} \geq 0, \quad \forall i. \end{aligned} \quad (13)$$

B. DLMA reformulation

We now reformulate our system model as a semi-distributed system that consists of several wireless networks with different MAC protocols. Nodes in different networks cannot communicate with each other. For nodes within the DLMA network, there is a DLMA central gateway that coordinates the transmissions of the nodes. Similarly, for nodes within the TDMA network, there is implicitly a TDMA central gateway to decide the time slots in which TDMA nodes transmit.

Among the N nodes in the wireless networks, let K be the number of DRL nodes in the DLMA network and $L = N - K$ be the number of non-DRL nodes. In the DLMA protocol as described in Section II-C, all DRL nodes individually adopt the single-agent DRL algorithm, and independently perform training and execution of the DRL algorithm. Unlike the DLMA protocol in Section II-C, we now consider an DRL algorithm with “centralized training at the gateway node and independent execution at DRL nodes”. The gateway in the DLMA network associates with all other DRL nodes in the DLMA network and coordinates the coexistence of the DLMA network with other networks (e.g., the TDMA and ALOHA networks). In each time slot, the gateway decides whether a node in the DLMA network should transmit or not. If YES, the gateway selects one of the DRL nodes in a round-robin manner to

transmit. After transmitting, the selected DRL node receives a feedback from the system and communicates with the gateway with this information. If NO, all DRL nodes keep silent. In this manner, the gateway can be regarded as a virtual big agent that is a combination of the K DRL nodes. The coordination information from the gateway to other DRL nodes can be sent through a control channel. For example, the control channel can be implemented as a short time slot after each time slot of information transmission. Other implementations are also possible, but we will omit the discussion here since the focus of this paper is not implementation details. The above reformulates the system to contain $L + 1$ nodes: one DRL big agent node (we index it by $i = L + 1$) and other L legacy nodes (we index them by $i = 1, 2, \dots, L$).

We now modify the original Q-learning algorithm. The original Q-learning algorithm is designed for the single-agent case under the objective of maximizing the accumulated reward of the agent. It cannot be directly applied to the multi-node/multi-agent case to meet arbitrary fairness objectives. We therefore put forth a multi-dimensional Q-learning algorithm to cater for the α -fairness objective.

In the original Q-learning algorithm, each agent receives a scalar reward from the environment. The scalar reward, representing the overall system transmission result (success, idleness or collision), is regarded as the overall reward to the system in the original Q-learning algorithm. Each agent uses the overall reward to compute the sum throughput objective. By contrast, in our new multi-dimensional Q-learning algorithm, the big agent receives an $L + 1$ dimension vector of rewards from the environment. Each element of the vector represents the transmission result of one particular node. The reward vector is used to compute the α -fairness objective. Specifically, let $r^{(i)}$ be the reward of node i and thus the received reward vector is given by $[r^{(i)}]_{i=1}^{L+1}$. For a state-action pair (s, a) , instead of maintaining an action-value scalar $Q(s, a)$, the big agent maintains an action-value vector $[Q^{(i)}(s, a)]_{i=1}^{L+1}$, where the element $Q^{(i)}(s, a)$ is the expected accumulated discounted reward of node i .

Let $q^{(i)}(s, a)$ be the estimate of the elementary action-value function $Q^{(i)}(s, a)$ in the action-value vector. Suppose at time t , the state is s_t . For decision making, we still adopt the ε -greedy algorithm. When selecting the greedy action, the objective in (13) can be applied to meet arbitrary fairness objective, i.e.,

$$a_t = \arg \max_a \left\{ \sum_{i=1}^L f_\alpha^{(i)}(q^{(i)}(s_t, a)) + K \cdot f_\alpha^{(L+1)}\left(\frac{q^{(L+1)}(s_t, a)}{K}\right) \right\}. \quad (14)$$

After taking action a_t , the big agent employs the multi-dimensional Q-learning algorithm to parallelly update the $L + 1$ elementary action-value estimates $q^{(i)}(s_t, a_t)$, $i = 1, 2, \dots, L + 1$ as

$$\begin{aligned} q^{(i)}(s_t, a_t) &\leftarrow q^{(i)}(s_t, a_t) + \\ &\beta \left[r_{t+1}^{(i)} + \gamma q^{(i)}(s_{t+1}, a_{t+1}) - q^{(i)}(s_t, a_t) \right], \end{aligned} \quad (15)$$

where

$$a_{t+1} = \arg \max_{a'} \left\{ \sum_{i=1}^L f_{\alpha}^{(i)} \left(q^{(i)}(s_{t+1}, a') \right) + K \cdot f_{\alpha}^{(L+1)} \left(\frac{q^{(L+1)}(s_{t+1}, a')}{K} \right) \right\}. \quad (16)$$

Here, it is important to point out the subtleties in (14)-(16) and how they differ from the conventional Q-learning update equation in (3). In conventional Q-learning, an action that optimizes the Q function is chosen (as explained in Section II-B). In other words, the Q function is the objective function to be optimized. However, the Q function as embodied in (3) and (15) is a projected (estimated) weighted sum of the current rewards and future rewards. To be more specific, take a look at the term $\left[r_{t+1} + \gamma \max_{a'} q(s_{t+1}, a') \right]$ in (3). It can be taken to be an estimation of $[r_{t+1} + \gamma E[r_{t+2}] + \gamma^2 E[r_{t+3}] + \dots]$, which is a weighted sum of the current reward and future rewards with discount factor γ . We can view $[r_{t+1} + \gamma E[r_{t+2}] + \gamma^2 E[r_{t+3}] + \dots]$ as a newly estimated $q(s_t, a_t)$. In (3), for the purpose of estimation smoothing, we apply a weight of β to this new estimate and a weight of $(1 - \beta)$ to the previous value of $q(s_t, a_t)$ to come up with a new value for $q(s_t, a_t)$. Nevertheless, $q(s_t, a_t)$ still embodies a weighted sum of the current and future rewards. Since in conventional Q-learning, an action that gives the maximum $q(s_t, a_t)$ is taken at each step t , the objective can be viewed as trying to maximize a weighted sum of rewards with discount factor γ . However, not all objectives can be conveniently expressed as a weighted sum of rewards. An example is the α -fairness objective of focus here. A contribution of us here is the realization that, for generality, we need to separate the objective upon which the optimizing action is chosen and the Q function itself.

Objectives can often be expressed as a function of several components, wherein each component can be expressed as a Q function (e.g., (14)). In the more general setup, the update equation of Q function still has the same form (i.e., (15) has the same form as (3)). However, the action a_{t+1} chosen a time step later in (15) is not that gives the maximum $q^{(i)}(s_{t+1}, \cdot)$, but which is based on (16). Thus, the Q function is still a projected weighted sum of rewards. But the policy that gives rise to the rewards is not based on maximizing the weighted sum of rewards, but based on maximizing a more general objective.

Returning to our wireless setting, the first term in (14) is the sum of local utility functions of all legacy nodes. Since the big agent (indexed by $L + 1$) is actually a combination of the K DRL nodes, and $q^{(L+1)}(\cdot)/K$ is the estimated accumulated reward of each DRL node, the second term in (14) is the sum of local utility functions of all the DRL nodes. We have two remarks: i) the $q^{(i)}(s_t, a_t)$ in (15) is an estimate of the expected accumulated discounted reward of node i (as expressed in (14), rather than the exact throughput $x^{(i)}$ in (13); ii) we use $q^{(i)}(s_t, a_t)$ to help the agent make decisions because the exact throughput $x^{(i)}$ is not known. Our evaluation results in section V show that this method can achieve the fairness objective. We continue the reformulation of DLMA by incorporating deep neural networks. The incorporation of

Algorithm 2 DLMA with the α -fairness objective

```

Initialize  $s_0, \varepsilon, \gamma, \rho, N_E, F$ 
Initialize experience memory  $EM$ 
Initialize the parameter of QNN as  $\theta$ 
Initialize the parameter of target QNN  $\theta^- = \theta$ 
for  $t = 0, 1, 2, \dots$  in DLMA do
    Input  $s_t$  to QNN and output
     $\mathbb{Q} = \left\{ q^{(i)}(s_t, a, \theta) | a \in A_{s_t}, i = 1, 2, \dots, L + 1 \right\}$ 
    Generate action  $a_t$  from  $\mathbb{Q}$  using  $\varepsilon$ -greedy algorithm
    Observe  $z_t, r_{t+1}^{(1)}, r_{t+1}^{(2)}, \dots, r_{t+1}^{(L)}$ 
    Compute  $s_{t+1}$  from  $s_t, a_t$  and  $z_t$ 
    Store  $(s_t, a_t, r_{t+1}^{(1)}, r_{t+1}^{(2)}, \dots, r_{t+1}^{(L+1)}, s_{t+1})$  to  $EM$ 
    if Remainder( $t/F == 0$ ) then  $I = 1$  else  $I = 0$ 
    TRAINQNN( $\gamma, \rho, N_E, I, EM, \theta, \theta^-$ )
end for

procedure TRAINQNN( $\gamma, \rho, N_E, I, EM, \theta, \theta^-$ )
    Randomly sample  $N_E$  experience tuples from  $EM$  as  $E$ 
    for each sample  $e = (s, a, r^{(1)}, r^{(2)}, \dots, r^{(L+1)}, s')$  in  $E$ 
        do
            Calculate  $y_{r,s'}^{(i)QNN} = r^{(i)} + \gamma q^{(i)}(s', a'; \theta^-)$ , where  $a'$  is selected to according (19)
        end for
        Perform Gradient Descent to update  $\theta$  in QNN:
        Iterate  $\theta \leftarrow \theta - \frac{\rho}{N_E(L+1)} \sum_{i=1}^{L+1} \sum_{e \in E} [y_{r,s'}^{(i)QNN} - q^{(i)}(s, a; \theta)] \nabla q^{(i)}(s, a; \theta)$ 
        if  $I == 1$  then
            Update  $\theta^-$  in target QNN by setting  $\theta^- = \theta$ 
        end if
end procedure

```

deep neural networks into the multi-dimensional Q-learning algorithm calls for two additional modifications. The first is to use a QNN to approximate the action-value vector $[Q^{(i)}(s, a)]_{i=1}^{L+1}$ as $[q^{(i)}(s, a; \theta)]_{i=1}^{L+1}$, where θ is the weights of QNN. The second is to augment the experience tuple to $e = (s, a, r^{(1)}, r^{(2)}, \dots, r^{(L+1)}, s')$. With these two modifications, the loss function (8), the target (9) and the update of θ (10) are now given by

$$L_E(\theta) = \frac{1}{N_E(L+1)} \sum_{i=1}^{L+1} \sum_{e \in E} \left(y_{r,s'}^{(i)QNN} - q^{(i)}(s, a; \theta) \right)^2, \quad (17)$$

$$y_{r,s'}^{(i)QNN} = r^{(i)} + \gamma q^{(i)}(s', a'; \theta^-), \quad (18)$$

where

$$a' = \arg \max_{\tilde{a}'} \left\{ \sum_{i=1}^L f_{\alpha}^{(i)} \left(q^{(i)}(s', \tilde{a}'; \theta^-) \right) + K \cdot f_{\alpha}^{(L+1)} \left(\frac{q^{(L+1)}(s', \tilde{a}'; \theta^-)}{K} \right) \right\}, \quad (19)$$

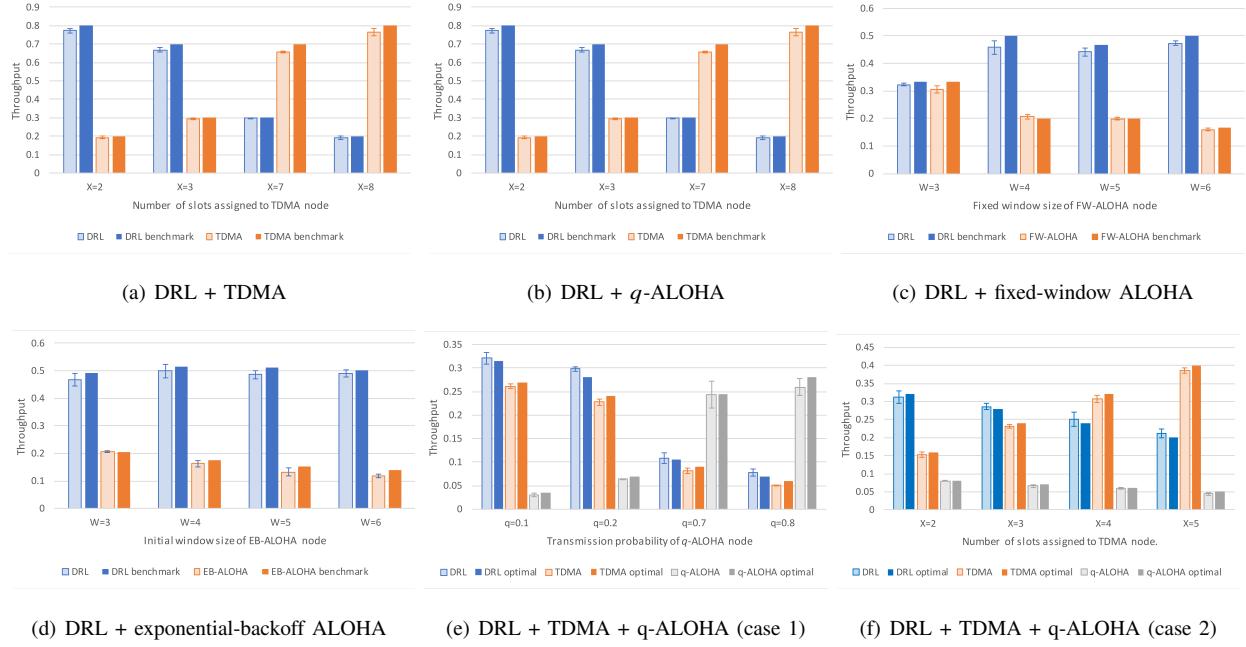


Fig. 10: Individual throughputs for different cases with the objective of achieving proportional fairness.

$$\text{Iterate } \theta \leftarrow \theta - \frac{\rho}{N_E(L+1)} \cdot \sum_{i=1}^{L+1} \sum_{e \in E} \left[y_{r,s'}^{(i)QNN} - q^{(i)}(s, a; \theta) \right] \nabla q^{(i)}(s, a; \theta). \quad (20)$$

The pseudocode of the reformulated DLMA protocol is summarized in Algorithm 2.

V. PROPORTIONAL FAIRNESS PERFORMANCE EVALUATION

This section investigates the performance when DRL nodes aim to achieve proportional fairness among nodes, as a representative example of the general α -fairness DLMA formulation. We investigate the interaction of DRL nodes with TDMA nodes, ALOHA nodes, and a mix of TDMA nodes and ALOHA nodes, respectively. The optimal results for benchmarking purposes can also be derived by imagining a model-aware node for different cases (the derivations are provided in [18] and omitted here.)

A. Coexistence with TDMA networks

We first present the results of the coexistence of one DRL node with one TDMA node. In this trivial case, achieving proportional fairness is the same as maximizing sum throughput. That is, to achieve proportional fairness, the optimal strategy of the DRL node is to transmit in the slots not occupied by the TDMA node and keep silent in the slots occupied by the TDMA node. Fig. 10(a) presents the results when the number of slots assigned to TDMA node is 2, 3, 7 and 8 out of 10 slots within a frame. We can see that the reformulated DLMA protocol can achieve proportional fairness in this case.

B. Coexistence with ALOHA networks

We next present the results of the coexistence of one DRL node with one q -ALOHA node, one FW-ALOHA node, and one EB-ALOHA, respectively. Fig. 10(b) presents the results with different transmission probabilities for the coexistence of one DRL node with one q -ALOHA node. Fig. 10(c) presents the results with different fixed-window sizes for the coexistence of one DRL node with one FW-ALOHA node. Fig. 10(d) presents the results with different initial window sizes and $m = 2$ for the coexistence of one DRL node with one EB-ALOHA node. As shown in these results, the reformulated DLMA protocol can again achieve proportional fairness without knowing the transmission schemes of different ALOHA variants.

C. Coexistence with a mix of TDMA and ALOHA networks

We now present the results of a setup where one DRL node coexists with one TDMA node and one q -ALOHA node simultaneously. We also consider the two cases investigated in Section III-C, but the objective now is to achieve proportional fairness among all the nodes. Fig. 10(e) and Fig. 10(f) present the results of the two cases. We can see that with the reformulated DLMA protocol, the individual throughputs achieved approximate the optimal individual throughputs achieved by imaging a model aware node.

We now present the results when three DRL nodes coexist with one TDMA node and two q -ALOHA nodes. The case investigated here is the same as the case presented in Fig. 5, but the three DRL nodes are now formulated to be one big agent and the objective is modified to achieve proportional fairness among all the nodes. The optimal results for the big agent, the TDMA node and each q -ALOHA node are derived

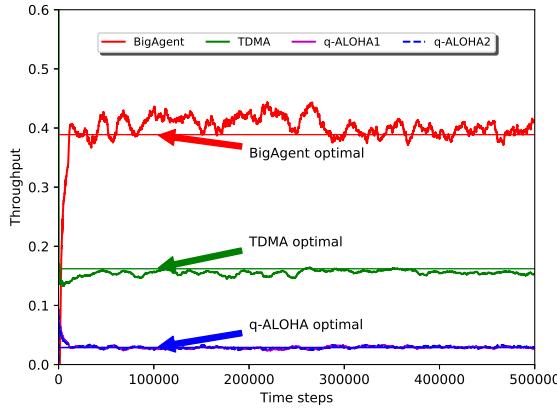


Fig. 11: Individual throughputs under the coexistence of three DRL nodes (BigAgent) with one TDMA node and two q -ALOHA nodes. The throughput at time t is computed $\sum_{\tau=t-N+1}^t r_\tau / N$, where $N = 5000$.

in [18]. As shown in Fig. 11, the optimal results can also be approximated using the reformulated DLMA protocol.

VI. CONCLUSION

This paper proposed and investigated a MAC protocol based on DRL for heterogeneous wireless networking, referred to as DLMA. A salient feature of DLMA is that it can learn to achieve an overall objective (e.g., α -fairness objective) by a series of state-action-reward observations while operating in the heterogeneous environment. In particular, it can achieve near-optimal performance with respect to the objective without knowing the detailed operating mechanisms of the other coexisting MACs.

This paper also demonstrated the advantages of using neural networks in reinforcement learning for wireless networking. Specifically, compared with the traditional RL, DRL can acquire the near-optimal strategy and performance with faster convergence time and higher robustness, two essential properties for practical deployment of the MAC protocol in dynamically changing wireless environments.

Last but not least, in the course of doing this work, we discovered an approach to generalize the Q-learning framework so that more general objectives can be achieved. In particular, for generality, we argued that we need to separate the Q function and the objective function upon which actions are chosen to optimize. A framework on how to relate the objective function and the Q function in the general set-up was presented in this paper.

REFERENCES

- [1] DARPA SC2 Website: <https://spectrumcollaborationchallenge.com/>.
- [2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [4] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
- [5] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.
- [6] Y. Yu, T. Wang, and S. C. Liew, “Deep-reinforcement learning multiple access for heterogeneous wireless networks,” in *Communications (ICC), 2018 IEEE International Conference on*. IEEE, 2018, pp. 1–7.
- [7] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [8] 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.
- [9] K.-L. A. Yau, P. Komisarczuk, and D. T. Paul, “Enhancing network performance in distributed cognitive radio networks using single-agent and multi-agent reinforcement learning,” in *2010 IEEE 35th Conference on Local Computer Networks (LCN)*. IEEE, 2010, pp. 152–159.
- [10] C. Wu, K. Chowdhury, M. Di Felice, and W. Meleis, “Spectrum management of cognitive radio using multi-agent reinforcement learning,” in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Industry track*. International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 1705–1712.
- [11] M. Bkassiny, S. K. Jayaweera, and K. A. Avery, “Distributed reinforcement learning based mac protocols for autonomous cognitive secondary users,” in *2011 20th Annual Wireless and Optical Communications Conference (WOCC)*. IEEE, 2011, pp. 1–6.
- [12] Z. Liu and I. Elhanany, “Rl-mac: A qos-aware reinforcement learning based mac protocol for wireless sensor networks,” in *Networking, Sensing and Control, 2006. ICNSC’06. Proceedings of the 2006 IEEE International Conference on*. IEEE, 2006, pp. 768–773.
- [13] Y. Chu, P. D. Mitchell, and D. Grace, “Aloha and q-learning based medium access control for wireless sensor networks,” in *2012 International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2012, pp. 511–515.
- [14] O. Naraparastek and K. Cohen, “Deep multi-user reinforcement learning for dynamic spectrum access in multichannel wireless networks,” *arXiv preprint arXiv:1704.02613*, 2017.
- [15] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, “Deep reinforcement learning for dynamic multichannel access in wireless networks,” *IEEE Transactions on Cognitive Communications and Networking*, 2018.
- [16] U. Challita, L. Dong, and W. Saad, “Deep learning for proactive resource allocation in lte-u networks,” in *Proceedings of 23th European Wireless Conference*. VDE, 2017, pp. 1–6.
- [17] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.
- [18] Y. Yu, T. Wang, and S. C. Liew, “Model-aware nodes in heterogeneous networks: A supplementary document to paper ‘deep-reinforcement learning multiple access for heterogeneous wireless networks’,” *Technical report*, available at: <https://github.com/YidingYu/DLMA/blob/master/DLMA-benchmark.pdf>.
- [19] J. Mo and J. Walrand, “Fair end-to-end window-based congestion control,” *IEEE/ACM Transactions on Networking (ToN)*, vol. 8, no. 5, pp. 556–567, 2000.