Multi-Agent Deep Reinforcement Learning for Dynamic Power Allocation in Wireless Networks

Yasar Sinan Nasir, *Student Member, IEEE*, and Dongning Guo, *Senior Member, IEEE*

***Abstract*— This work demonstrates the potential of deep reinforcement learning techniques for transmit power control in wireless networks. Existing techniques typically find near-optimal power allocations by solving a challenging optimization problem. Most of these algorithms are not scalable to large networks in real-world scenarios because of their computational complexity and instantaneous cross-cell channel state information (CSI) requirement. In this paper, a distributively executed dynamic power allocation scheme is developed based on model-free deep reinforcement learning. Each transmitter collects CSI and quality of service (QoS) information from several neighbors and adapts its own transmit power accordingly. The objective is to maximize a weighted sum-rate utility function, which can be particularized to achieve maximum sum-rate or proportionally fair scheduling. Both random variations and delays in the CSI are inherently addressed using deep *Q*-learning. For a typical network architecture, the proposed algorithm is shown to achieve near-optimal power allocation in real time based on delayed CSI measurements available to the agents. The proposed scheme is especially suitable for practical scenarios where the system model is inaccurate and CSI delay is non-negligible.**

***Index Terms*— Deep Q-learning, radio resource management, interference mitigation, power control, Jakes fading model.**

I. Introduction

IN EMERGING and future wireless networks, inter-cell interference management is one of the key technological challenges as access point (AP) deployment become denser to meet ever-increasing demand. A transmitter may increase its transmit power to improve its own data rate, but at the same time it may degrade links it interferes with. Transmit power control has been implemented since the first generation cellular networks [1]. A number of centralized and distributed optimization techniques have been used to develop algorithms for reaching a suboptimal power allocation [1]–[7]. We select two state-of-the-art algorithms as benchmarks. These are the weighted minimum mean square error (WMMSE) algorithm [2] and an iterative algorithm based on fractional programming (FP) [3]. In their generic form, both algorithms require full up-to-date cross-cell channel state information (CSI).

This work is the first to apply deep reinforcement learning to power control [8]. Sun *et al.* [9] proposed a centralized *supervised learning* approach to train a fast deep neural network (DNN) that achieves 90% or higher of the sum-rate achieved by the WMMSE algorithm. However, this approach still requires full CSI. Another issue is that training DNN depends on a massive dataset of the WMMSE algorithm's output, which takes a significant amount of time to produce due to WMMSE's computational complexity. As the network gets larger, the total number of DNN's input and output ports also increases, which raises questions on the scalability of the centralized solution of [9]. Furthermore, the success of supervised learning is highly dependent on the accuracy of the system model underlying the computed training data, which requires new training data every time the system model or key parameters change.

In this work, we design a distributively executed algorithm to be employed by all transmitters to compute their power allocations in real time.[1] The main contributions and some advantages of the proposed scheme are summarized as follows:

1) The proposed distributively executed algorithm is based on deep Q-learning [11], which is model-free and robust to unpredictable changes in the wireless environment.
2) The complexity of the distributively executed algorithm does not depend on the network size. In particular, the proposed algorithm is computationally scalable to networks that cover arbitrarily large geographical areas if the number of links per unit area remains upper bounded by the same constant everywhere.
3) The proposed algorithm learns a policy that guides all links to adjust their power levels under important practical constraints such as delayed information exchange and incomplete cross-link CSI.
4) There is no need to run an existing near-optimal algorithm to produce training data. We use a centralized network trainer approach that gathers local observations from all network agents. This approach is computationally efficient and robust. In fact, a pretrained neural network can also achieve comparable performance as that of the centralized optimization based algorithms.
5) Using simulations, we compare the reinforcement learning outcomes with state-of-the-art optimization-based

Manuscript received December 15, 2018; revised April 5, 2019; accepted May 20, 2019. Date of publication August 8, 2019; date of current version September 16, 2019. This work was supported in part by the National Science Foundation under Grant CCF-1423040. *(Corresponding author: Yasar Sinan Nasir.)*

The authors are with the Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208 USA (e-mail: snasir@u.northwestern.edu; dguo@northwestern.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2019.2933973

[1] A dynamic power allocation problem with time-varying channels for a different system model and network setup was studied in [10], where the delay performance of the classical dynamic backpressure algorithm was improved by exploiting the stochastic Lyapunov optimization framework.

algorithms, and also demonstrate the scalability and robustness of the proposed algorithm. In the simulation, we model the channel variations inconsequential to the learning algorithm using the Jakes fading model [12]. In certain scenarios the proposed distributed algorithm even outperforms the centralized iterative algorithms introduced in [2], [3]. We also address some important practical constraints that are not included in [2], [3].

Deep reinforcement learning framework has been used in some other wireless communications problems [13]–[16]. Classical Q-learning techniques have been applied to the power allocation problem in [17]–[21]. The goal in [17], [18] is to reduce the interference in LTE-Femtocells. Unlike the *deep* Q-learning algorithm, the classical algorithm builds a lookup table to represent the value of state-action pairs, so [17] and [18] represent the wireless environment using a discrete state set and limit the number of learning agents. Amiri *et al.* [19] have used cooperative Q-learning based power control to increase the QoS of users in femtocells without considering the channel variations. The deep Q-learning based power allocation to maximize the network objective has also been considered in [20], [21]. Similar to the proposed approach, the work in [20], [21] is also based on a distributed framework with a centralized training assumption, but the benchmark to evaluate the performance of their algorithm was a fixed power allocation scheme instead of state-of-the-art algorithms. In this paper, the proposed approach to the state of wireless environment and the reward function is also novel and unique. Specifically, the proposed approach addresses the stochastic nature of wireless environment as well as incomplete/delayed CSI, and arrives at highly competitive strategies quickly.

The remainder of this paper is organized as follows. We give the system model in Sec. II. In Sec. III, we formulate the dynamic power allocation problem. In Sec. IV, we first give an overview of deep Q-learning and then describe the proposed algorithm. Simulation results are given in Sec. V. We conclude with a discussion of possible future work in Sec. VI.

II. System Model

We first consider the classical power allocation problem in a network of n links. We assume that all transmitters and receivers are equipped with a single antenna. The model is often used to describe a mobile ad hoc network (MANET) [5]. The model has also been used to describe a simple cellular network with n APs, where each AP serves a single user device [3], [4]. Let $N = \{1, \ldots, n\}$ denote the set of link indexes. We consider a fully synchronized time slotted system with slot duration T. For simplicity, we consider a single frequency band with flat fading. We adopt a block fading model to denote the downlink channel gain from transmitter i to receiver j in time slot t as

$$g_{i\to j}^{(t)} = \left|h_{i\to j}^{(t)}\right|^2 \alpha_{i\to j}, \quad t = 1, 2, \ldots. \tag{1}$$

Here, $\alpha_{i\to j} \geq 0$ represents the large-scale fading component including path loss and log-normal shadowing, which remains the same over many time slots. Following Jakes fading model [12], we express the small-scale Rayleigh fading component as a first-order complex Gauss-Markov process:

$$h_{i\to j}^{(t)} = \rho h_{i\to j}^{(t-1)} + \sqrt{1-\rho^2} e_{i\to j}^{(t)} \tag{2}$$

where $h_{i\to j}^{(0)}$ and the channel innovation process $e_{i\to j}^{(1)}, e_{i\to j}^{(2)}, \ldots$ are independent and identically distributed circularly symmetric complex Gaussian (CSCG) random variables with unit variance. The correlation $\rho = J_0(2\pi f_d T)$, where $J_0(.)$ is the zeroth-order Bessel function of the first kind and f_d is the maximum Doppler frequency.

The received signal-to-interference-plus-noise ratio (SINR) of link i in time slot t is a function of the allocation $\boldsymbol{p} = [p_1, \ldots, p_n]^{\mathsf{T}}$:

$$\gamma_i^{(t)}(\boldsymbol{p}) = \frac{g_{i\to i}^{(t)} p_i}{\sum_{j\neq i} g_{j\to i}^{(t)} p_j + \sigma^2} \tag{3}$$

where σ^2 is the additive white Gaussian noise (AWGN) power spectral density (PSD). We assume the same noise PSD in all receivers without loss of generality. The downlink spectral efficiency of link i at time t can be expressed as:

$$C_i^{(t)}(\boldsymbol{p}) = \log\left(1 + \gamma_i^{(t)}(\boldsymbol{p})\right). \tag{4}$$

The transmit power of transmitter i in time slot t is denoted as $p_i^{(t)}$. We denote the power allocation of the network in time slot t as $\boldsymbol{p}^{(t)} = \left[p_1^{(t)}, \ldots, p_n^{(t)}\right]^{\mathsf{T}}$.

III. Dynamic Power Control

We are interested in maximizing a generic weighted sum-rate objective function. Specifically, the dynamic power allocation problem in slot t is formulated as

$$\begin{aligned} &\underset{\boldsymbol{p}}{\text{maximize}} \ \sum_{i=1}^{n} w_i^{(t)} \cdot C_i^{(t)}(\boldsymbol{p}) \\ &\text{subject to} \ 0 \leq p_i \leq P_{\text{max}}, \quad i = 1, \ldots, n, \end{aligned} \tag{5}$$

where $w_i^{(t)}$ is the given nonnegative weight of link i in time slot t, and P_{max} is the maximum PSD a transmitter can emit. Hence, the dynamic power allocator has to solve an independent problem in the form of (5) at the beginning of every time slot. In time slot t, the optimal power allocation solution is denoted as $\boldsymbol{p}^{(t)}$. Problem (5) is in general non-convex and has been shown to be NP-hard [22].

We consider two special cases. In the first case, the objective is to maximize the sum-rate by letting $w_i^{(t)} = 1$ for all i and t. In the second case, the weights vary in a controlled manner to ensure proportional fairness [7], [23]. Specifically, at the end of time slot t, receiver i computes its weighted average spectral efficiency as

$$\bar{C}_i^{(t)} = \beta \cdot C_i^{(t)}\left(\boldsymbol{p}^{(t)}\right) + (1-\beta)\bar{C}_i^{(t-1)} \tag{6}$$

where $\beta \in (0, 1]$ is used to control the impact of history. User i updates its link weight as:

$$w_i^{(t+1)} = \left(\bar{C}_i^{(t)}\right)^{-1}. \tag{7}$$

This power allocation algorithm maximizes the sum of log-average spectral efficiency [23], i.e.,

$$\sum_{i\in N}\log\bar{C}_i^{(t)}, \tag{8}$$

where a user's long-term average throughput is proportional to its long-term channel quality in some sense.

We use two popular (suboptimal) power allocation algorithms as benchmarks. These are the WMMSE algorithm [2] and the FP algorithm [3]. Both are centralized and iterative in their original form. The closed-form FP algorithm used in this paper is formulated in [3, Algorithm 3]. Similarly, a detailed explanation and pseudo code of the WMMSE algorithm is given in [9, Algorithm 1]. The WMMSE and FP algorithms are both centralized and require full cross-link CSI. The centralized mechanism is suitable for a stationary environment with slowly varying weights and no fast fading. For a network with non-stationary environment, it is infeasible to instantaneously collect all CSI over a large network.

It is fair to assume that the feedback delay T_{fb} from a receiver to its corresponding transmitter is much smaller than the slot duration T, so the prediction error due to the feedback delay is neglected. Therefore, once receiver i completes a direct channel measurement, we assume that it is also available at the transmitter i.

For the centralized approach, once a link acquires the CSI of its direct channel and all other interfering channels to its receiver, passing this information to a central controller is another burden. This is typically resolved using a backhaul network between the APs and the central controller. The CSI of cross links is usually delayed or even outdated. Furthermore, the central controller can only return the optimal power allocation as the iterative algorithm converges, which is another limitation on the scalability.

Our goal is to design a scalable algorithm, so we limit the information exchange to between nearby transmitters. We define two neighborhood sets for every $i \in N$: Let the set of transmitters whose SNR at receiver i was above a certain threshold η during the past time slot $t-1$ be denoted as

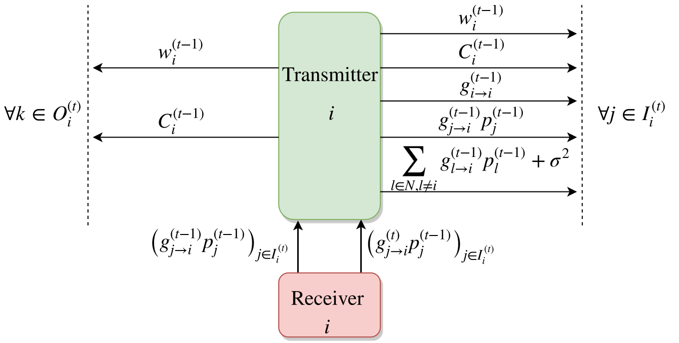
$$I_i^{(t)} = \left\{ j \in N, j \neq i \middle| g_{j\to i}^{(t-1)} p_j^{(t-1)} > \eta\sigma^2 \right\}. \tag{9}$$

Let the set of receiver indexes whose SNR from transmitter i was above a threshold in slot $t-1$ be denoted as

$$O_i^{(t)} = \left\{ k \in N, k \neq i \middle| g_{i\to j}^{(t-1)} p_i^{(t-1)} > \eta\sigma^2 \right\}. \tag{10}$$

From link i's viewpoint, $I_i^{(t)}$ represents the set of "interferers", whereas $O_i^{(t)}$ represents the set of the "interfered" neighbors.

We next discuss the local information a transmitter possesses at the beginning of time slot t. First, we assume that transmitter i learns via receiver feedback the direct downlink channel gain, $g_{i\to i}^{(t)}$. Further, transmitter i also learns the current total received interference-plus-noise power at receiver i before the global power update, i.e., $\sum_{j\in N, j\neq i} g_{j\to i}^{(t)} p_j^{(t-1)} + \sigma^2$ (as a result of the new gains and the yet-to-be-updated powers). In addition, by the beginning of slot t, receiver i has informed transmitter i of the received power from every interferer $j \in I_i^{(t)}$, i.e., $g_{j\to i}^{(t)} p_j^{(t-1)}$. These measurements can only be available at transmitter i just before the beginning of slot t. Hence, in the previous slot $t-1$, receiver i also informs transmitter i of the outdated versions of these measurements to be used in the information exchange process performed in slot $t-1$ between transmitter i and its interferers. To clarify, as shown in Fig. 1, transmitter i has sent the following outdated information to interferer $j \in I_i^{(t)}$ in return for $w_j^{(t-1)}$ and $C_j^{(t-1)}$:



$\forall k \in O_i^{(t)}$
$w_i^{(t-1)}$
$C_i^{(t-1)}$
Transmitter
i
$w_i^{(t-1)}$
$C_i^{(t-1)}$
$g_{i\to i}^{(t-1)}$
$g_{j\to i}^{(t-1)} p_j^{(t-1)}$
$\sum_{l\in N, l\neq i} g_{l\to i}^{(t-1)} p_l^{(t-1)} + \sigma^2$
$\forall j \in I_i^{(t)}$
$\left(g_{j\to i}^{(t-1)} p_j^{(t-1)}\right)_{j\in I_i^{(t)}}$
$\left(g_{j\to i}^{(t)} p_j^{(t-1)}\right)_{j\in I_i^{(t)}}$
Receiver
i

Fig. 1. The information exchange between transmitter i and its neighbors in time slot $t-1$. Note that transmitter i obtains $g_{j\to i}^{(t)} p_j^{(t-1)}$ by the end of slot $t-1$, but it is not able to deliver this information to interferer j before the beginning of slot t due to additional delays through the backhaul network.

- $w_i^{(t-1)}$: the weight of link i,
- $C_i^{(t-1)}$: link i's spectral efficiency computed from (4),
- $g_{i\to i}^{(t-1)}$: the direct gain,
- $g_{j\to i}^{(t-1)} p_j^{(t-1)}$: the interference power from transmitter j to receiver i,
- $\sum_{l\in N, l\neq i} g_{l\to i}^{(t-1)} p_l^{(t-1)} + \sigma^2$: the total interference-plus-noise power at receiver i.

As assumed earlier, these measurements are accurate, where the uncertainty about the current CSI is entirely due to the latency of information exchange (one slot). By the same token, from every interfered $k \in O_i^{(t)}$, transmitter i also obtains k's items listed above.

IV. Deep Learning for Dynamic Power Allocation

A. Overview of Deep Q-Learning

A reinforcement learning agent learns its best policy from observing the rewards of trial-and-error interactions with its environment over time [24], [25]. Let S denote a set of possible states and A denote a discrete set of actions. The state $s \in S$ is a tuple of environment's features that are relevant to the problem at hand and it describes agent's relation with its environment [20]. Assuming discrete time steps, the agent observes the state of its environment, $s^{(t)} \in S$ at time step t. It then takes an action $a^{(t)} \in A$ according to a certain policy π. The policy $\pi(s,a)$ is the probability of taking action a conditioned on the current state being s. The policy function must satisfy $\sum_{a\in A}\pi(s,a) = 1$. Once the agent takes an action $a^{(t)}$, its environment moves from the current state $s^{(t)}$ to the next state $s^{(t+1)}$. As a result of this transition, the agent gets a reward $r^{(t+1)}$ that characterizes its benefit from taking action $a^{(t)}$ at state $s^{(t)}$. This scheme forms an experience at time $t+1$, hereby defined as $e^{(t+1)} = \left(s^{(t)}, a^{(t)}, r^{(t+1)}, s^{(t+1)}\right)$, which describes an interaction with the environment [11].

The well-known Q-learning algorithm aims to compute an optimal policy π that maximizes a certain expected reward without knowledge of the function form of the reward and the state transitions. Here we let the reward be the future cumulative discounted reward at time t:

$$R^{(t)} = \sum_{\tau=0}^{\infty} \gamma^{\tau} r^{(t+\tau+1)} \tag{11}$$

where $\gamma \in (0, 1]$ is the discount factor for future rewards. In the stationary setting, we define a Q-function associated with a certain policy π as the expected reward once action a is taken under state s [26], i.e.,

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[R^{(t)} \middle| s^{(t)} = s, a^{(t)} = a\right]. \tag{12}$$

As an action value function, the Q-function satisfies a Bellman equation [27]:

$$Q^{\pi}(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in S} \mathcal{P}^{a}_{ss'} \left(\sum_{a' \in A} \pi(s', a') Q^{\pi}(s', a') \right) \tag{13}$$

where $\mathcal{R}(s, a) = \mathbb{E}\left[r^{(t+1)} \middle| s^{(t)} = s, a^{(t)} = a\right]$ is the expected reward of taking action a at state s, and $\mathcal{P}^{a}_{ss'} = \Pr\left(s^{(t+1)} = s' \middle| s^{(t)} = s, a^{(t)} = a\right)$ is the transition probability from given state s to state s' with action a. From the fixed-point equation (13), the value of (s, a) can be recovered from all values of $(s', a') \in S \times A$. It has been proved that some iterative approaches such as Q-learning algorithm efficiently converges to the action value function (12) [26]. Clearly, it suffices to let $\pi^{*}(s, a)$ be equal to 1 for the most favorable action. From (13), the optimal Q-function associated with the optimal policy is then expressed as

$$Q^{*}(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in S} \mathcal{P}^{a}_{ss'} \max_{a'} Q^{*}(s', a'). \tag{14}$$

The classical Q-learning algorithm constructs a lookup table, $q(s, a)$, as a surrogate of the optimal Q-function. Once this lookup table is randomly initialized, the agent takes actions according to the ϵ-greedy policy for each time step. The ϵ-greedy policy implies that with probability $1 - \epsilon$ the agent takes the action a^{*} that gives the maximum lookup table value for a given current state, whereas it picks a random action with probability ϵ to avoid getting stuck at non-optimal policies [11]. After acquiring a new experience as a result of the taken action, the Q-learning algorithm updates a corresponding entry of the lookup table according to:

$$\begin{aligned} q\left(s^{(t)}, a^{(t)}\right) &\leftarrow (1 - \alpha) q\left(s^{(t)}, a^{(t)}\right) \\ &\quad + \alpha\left(r^{(t+1)} + \gamma \max_{a'} q\left(s^{(t+1)}, a'\right)\right) \end{aligned} \tag{15}$$

where $\alpha \in (0, 1]$ is the learning rate [26].

In case the state and action spaces are very large, as is the case for the power control problem at hand. The classical Q-learning algorithm fails mainly because of two reasons:

1) Many states are rarely visited, and
2) the storage of lookup table becomes impractical [28].

Both issues can be solved with deep reinforcement learning, e.g., deep Q-learning [11]. A deep neural network called deep Q-network (DQN) is used to estimate the Q-function in lieu of a lookup table. The DQN can be expressed as $q(s, a, \boldsymbol{\theta})$, where the real-valued vector $\boldsymbol{\theta}$ represents its parameters. The essence of DQN is that the function $q(\cdot, \cdot, \boldsymbol{\theta})$ is completely determined by $\boldsymbol{\theta}$. As such, the task of finding the best Q-function in a functional space of uncountably many dimensions is reduced to searching the best $\boldsymbol{\theta}$ of finite dimensions. Similar to the classical Q-learning, the agent collects experiences with its interaction with the environment. The agent or the network trainer forms a data set D by collecting the experiences until time t in the form of (s, a, r', s'). As the "quasi-static target network" method [11] implies, we define two DQNs: the target DQN with parameters $\boldsymbol{\theta}^{(t)}_{\text{target}}$ and the train DQN with parameters $\boldsymbol{\theta}^{(t)}_{\text{train}}$. $\boldsymbol{\theta}^{(t)}_{\text{target}}$ is updated to be equal to $\boldsymbol{\theta}^{(t)}_{\text{train}}$ once every T_u steps. From the "experience replay" [11], the least squares loss of train DQN for a random mini-batch $D^{(t)}$ at time t is

$$L\left(\boldsymbol{\theta}^{(t)}_{\text{train}}\right) = \sum_{(s,a,r',s') \in D^{(t)}} \left(y^{(t)}_{DQN}(r', s') - q\left(s, a; \boldsymbol{\theta}^{(t)}_{\text{train}}\right)\right)^2 \tag{16}$$

where the target is

$$y^{(t)}_{DQN}(r', s') = r' + \lambda \max_{a'} q\left(s', a'; \boldsymbol{\theta}^{(t)}_{\text{target}}\right). \tag{17}$$

Finally, we assume that each time step the stochastic gradient descent algorithm that minimizes (16) is used to train $\boldsymbol{\theta}^{(t)}_{\text{train}}$ over the mini-batch $D^{(t)}$. The stochastic gradient descent uses the gradient computed from just few samples of the dataset and has been shown to converge to a set of good parameters quickly [29].

B. Proposed Multi-Agent Deep Learning Algorithm

As depicted in Fig. 2, we propose a multi-agent deep reinforcement learning scheme with each transmitter as an agent. Similar to [30], we define the local state of learning agent i as $s_i \in S_i$ which is composed of environment features that are relevant to agent i's action $a_i \in A_i$. In the multi-agent learning system, the state transitions of their common environment depend on the agents' joint actions. An agent's environment transition probabilities in (13) may not be stationary as other learning agents update their policies. The Markov property introduced for the single-agent case in Sec. IV-A no longer holds in general [31]. This "environment non-stationarity" issue may cause instability during the learning process. One way to tackle the issue is to train a single meta agent with a DQN that outputs joint actions for the agents [32]. The complexity of the state-action space, and consequently the DQN complexity, will then be proportional to the total number of agents in the system. The single-meta agent approach is not suitable for our dynamic setup and the distributed execution framework, since its DQN can only forward the action assignments to the transmitters after acquiring the global state information. There is an extensive research to develop multi-agent learning frameworks and there exists several multi-agent Q-learning adaptations [31], [33]. However, multi-agent learning is an open research area and theoretical guarantees for

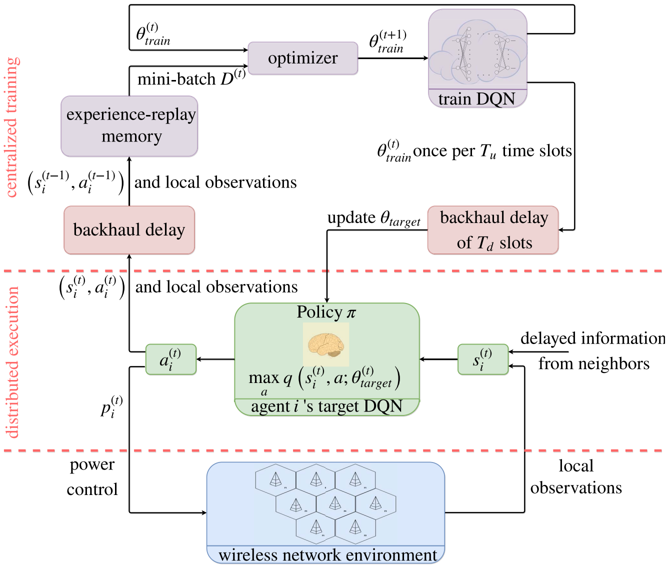

centralized training
distributed execution
$\theta^{(t)}_{train}$
optimizer
$\theta^{(t+1)}_{train}$
train DQN
mini-batch $D^{(t)}$
experience-replay memory
$\theta^{(t)}_{train}$ once per T_u time slots
$\left(s_i^{(t-1)}, a_i^{(t-1)}\right)$ and local observations
backhaul delay
update θ_{target}
backhaul delay of T_d slots
$\left(s_i^{(t)}, a_i^{(t)}\right)$ and local observations
Policy π
$\max_a q\left(s_i^{(t)}, a; \theta^{(t)}_{target}\right)$
agent i 's target DQN
$a_i^{(t)}$
$s_i^{(t)}$
delayed information from neighbors
$p_i^{(t)}$
power control
local observations
wireless network environment

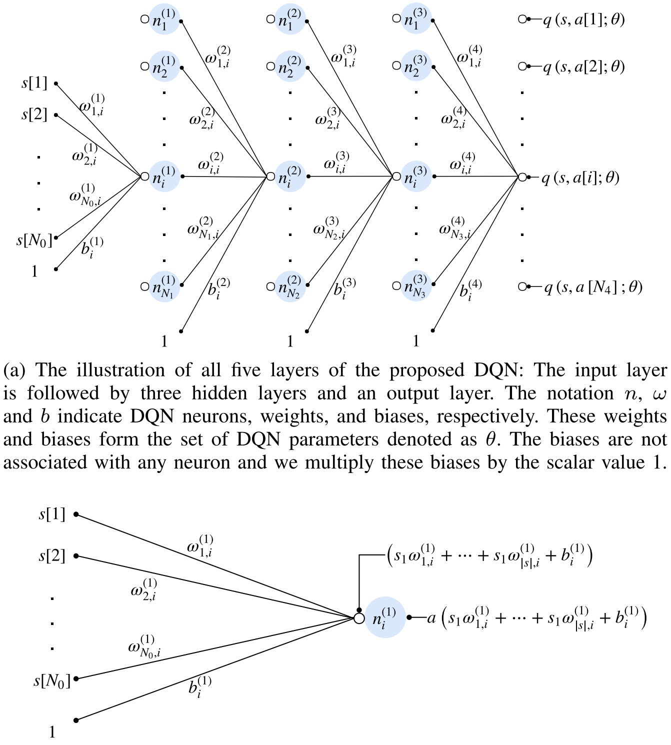
Fig. 2. Illustration of the proposed multi-agent deep reinforcement learning algorithm.

these adaptations are rare and incomplete despite their good empirical performances [31], [33].

In this work, we take an alternative approach where the DQNs are distributively executed at the transmitters, whereas training is centralized to ease implementation and to improve stability. Each agent i has the same copy of the DQN with parameters $Q^{(t)}_{\text{target}}$ at time slot t. The centralized network trainer trains a single DQN by using the experiences gathered from all agents. This significantly reduces the amount of memory and computational resources required by training. The centralized training framework is also similar to the *parameter sharing* concept which allows the learning algorithm to draw advantage from the fact that agents are learning together for faster convergence [34]. Since agents are working collaboratively to maximize the global objective in (5) with an appropriate reward function design to be discussed in Sec. IV-E, each agent can benefit from experiences of others. Note that sharing the same DQN parameters still allows different behavior among agents, because they execute the same DQN with different local states as input.

As illustrated in Fig. 2, at the beginning of time slot t, agent i takes action $a_i^{(t)}$ as a function of $s_i^{(t)}$ based on the current policy. All agents are synchronized and take their actions at the same time. Prior to taking action, agent i has observed the effect of the past actions of its neighbors on its current state, but it has no knowledge of $a_j^{(t)}$, $\forall j \neq i$. From the past experiences, agent i is able to acquire an estimation of the impact of its own actions on future actions of its neighbors, and it can determine a policy that maximizes its discounted expected future reward with the help of deep Q-learning.

The proposed DQN is a fully-connected deep neural network [35, Chapter 5] that consists of five layers as shown in Fig. 3a. The first layer is fed by the input state vector of length N_0. We relegate the detailed design of the state vector elements to Sec. IV-C. The input layer is followed by three hidden layers with N_1, N_2, and N_3 neurons, respectively.



(a) The illustration of all five layers of the proposed DQN: The input layer is followed by three hidden layers and an output layer. The notation n, ω and b indicate DQN neurons, weights, and biases, respectively. These weights and biases form the set of DQN parameters denoted as θ. The biases are not associated with any neuron and we multiply these biases by the scalar value 1.

(b) The functionality of a single neuron extracted from the first hidden-layer. $a(.)$ denotes the non-linear activation function.

Fig. 3. The overall design of the proposed DQN.

At the output layer, each port gives an estimate of the Q-function with given state input and the corresponding action output. The total number of DQN output ports is denoted as N_4 which is equal to the cardinality of the action set to be described in Sec. IV-D. The agent finds the action that has the maximum value at the DQN output and takes this action as its transmit power.

In Fig. 3a, we also depicted the connection between these layers by using the weights and biases of the DQN which form the set of parameters. The total number of scalar parameters in the fully connected DQN is

$$|\theta| = \sum_{l=0}^{3} (N_l + 1) N_{l+1}. \tag{18}$$

In addition, Fig. 3b describes the functionality of a single neuron which applies a non-linear activation function to its combinatorial input.

During the training stage, in each time slot, the trainer randomly selects a mini-batch $D^{(t)}$ of M_b experiences from an experience-replay memory [11] that stores the experiences of all agents. The experience-replay memory is a FIFO queue [15] with a length of nM_m samples where n is the total number of agents, i.e., a new experience replaces the oldest experience in the queue and the queue length is proportional to the number of agents. At time slot t the most recent experience from agent i is $e_i^{(t-1)} = \left(s_i^{(t-2)}, a_i^{(t-2)}, r_i^{(t-1)}, s_i^{(t-1)}\right)$ due to delay. Once the trainer picks $D^{(t)}$, it updates the parameters to minimize the loss in (16) using an appropriate optimizer, e.g., the stochastic gradient descent method [29]. As also explained in Fig. 2, once per T_u time slots, the trainer broadcasts the latest trained parameters. The new parameters are available at the agents after T_d time slots due to the transmission delay through the backhaul network. Training may be terminated once the parameters converge.

C. States

As described in Sec. III, agent i builds its state $s_i^{(t)}$ using information from the interferer and interfered sets given by (9) and (10), respectively. To better control the complexity, we set $\left|\bar{I}_i^{(t)}\right| = \left|\bar{O}_i^{(t)}\right| = c$, where $c > 0$ is the restriction on the number of interferers and interfereds the AP communicating with. At the beginning of time slot t, agent i sorts its interferers by current received power from interferer $j \in I_i^{(t)}$ at receiver i, i.e., $g_{j\to i}^{(t)} p_j^{(t-1)}$. This sorting process allows agent i to prioritize its interferers. As $\left|I_i^{(t)}\right| > c$, we want to keep strong interferers which have higher impact on agent i's next action. On the other hand, if $\left|I_i^{(t)}\right| < c$, agent i adds $\left|I_i^{(t)}\right| - c$ virtual noise agents to $I_i^{(t)}$ to fit the fixed DQN. A virtual noise agent is assigned an arbitrary negative weight and spectral efficiency. Its downlink and interfering channel gains are taken as zero in order to avoid any impact on agent i's decision-making. The purpose of having these virtual agents as placeholders is to provide inconsequential inputs to fill the input elements of fixed length, like 'padding zeros'. After adding virtual noise agents (if needed), agent i takes first c interferers to form $\bar{I}_i^{(t)}$. For the interfered neighbors, agent i follows a similar procedure, but this time the sorting criterion is the share of agent i on the interference at receiver $k \in O_i^{(t)}$, i.e., $g_{i\to k}^{(t-1)} p_i^{(t-1)} \left(\sum_{j \in N, j \neq k} g_{j\to k}^{(t-1)} p_j^{(t-1)} + \sigma^2\right)^{-1}$, in order to give priority to the most significantly affected interfered neighbors by agent i's interference.

The way we organize the local information to build $s_i^{(t)}$ accommodates some intuitive and systematic basics. Based on these basics, we perfected our design by trial-and-error with some preliminary simulations. We now describe the state of agent i at time slot t, i.e., $s_i^{(t)}$, by dividing it into three main feature groups as:

1) Local Information: The first element of this feature group is agent i's previous transmit power, i.e., $p_i^{(t-1)}$. Then, this is followed by the second and third elements that specify agent i's most recent potential contribution on the network objective (5): $1/w_i^{(t)}$ and $C_i^{(t-1)}$. For the second element, we do not directly use $w_i^{(t)}$ which tends to be quite large as $\bar{C}_i^{(t)}$ is close to zero from (7). We found that using $1/w_i^{(t)}$ is more desirable. Finally, the last four elements of this feature group are the last two measurements of its direct downlink channel and the total interference-plus-noise power at receiver i: $g_{i\to i}^{(t)}$, $g_{i\to i}^{(t-1)}$, $\sum_{j \in N, j \neq i} g_{j\to i}^{(t)} p_j^{(t-1)} + \sigma^2$, and $\sum_{j \in N, j \neq i} g_{j\to i}^{(t-1)} p_j^{(t-2)} + \sigma^2$. Hence, a total of seven input ports of the input layer are reserved for this feature group. In our state set design, we take the last two measurements into account to give the agent a better chance to track its environment change. Intuitively, the lower the maximum Doppler frequency, the slower the environment changes, so that having more past measurements will help the agent to make better decisions [15]. On the other hand, this will result with having more state information which may increase the complexity and decrease the learning efficiency. Based on preliminary simulations, we include two past measurements.

2) Interfering Neighbors: This feature group lets agent i observe the interference from its neighbors to receiver i and what is the contribution of these interferers on the objective (5). For each interferer $j \in \bar{I}_i^{(t)}$, three input ports are reserved for $g_{j\to i}^{(t)} p_j^{(t-1)}$, $1/w_j^{(t-1)}$, $C_j^{(t-1)}$. The first term indicates the interference that agent i faced from its interferer j; the other two terms imply the significance of agent j in the objective (5). Similar to the local information feature explained in the previous paragraph, agent i also considers the history of its interferers in order to track changes in its own receiver's interference condition. For each interferer $j' \in \bar{I}_i^{(t-1)}$, three input ports are reserved for $g_{j'\to i}^{(t-1)} p_{j'}^{(t-2)}$, $1/w_{j'}^{(t-2)}$, $C_{j'}^{(t-2)}$. A total of $6c$ state elements are reserved for this feature group.

3) Interfered Neighbors: Finally, agent i uses the feedback from its interfered neighbors to gauge its interference to nearby receivers and their contribution to the objective (5). If agent i's link was inactive during the previous time slot, then $O_i^{(t)} = \emptyset$. For this case, if we ignore the history and directly consider the current interfered neighbor set, the corresponding state elements will be useless. Note that agent i's link became inactive when its own estimated contribution on the objective (5) was not significant enough compared to its interference to its interfered neighbors. Thus, after agent i's link became inactive, in order to decide when to reactivate its link, it should keep track of the interfered neighbors that implicitly silenced itself. We solve this issue by defining time slot t'_i which is the last time slot agent i was active. The agent i carries the feedback from interfered $k \in \bar{O}_i^{(t'_i+1)}$. We also pay attention to the fact that if $t'_i < t-1$, interfered k has no knowledge of $g_{i\to k}^{(t-1)}$, but it is still able to send its local information to agent i. Therefore, agent i reserves four elements of its state set for each interfered $k \in O_i^{(t'_i+1)}$ as $g_{k\to k}^{(t-1)}$, $1/w_k^{(t-1)}$, $C_k^{(t-1)}$, and $g_{i\to k}^{(t'_i)} p_i^{(t'_i)} \left(\sum_{j \in N, j \neq k} g_{j\to k}^{(t-1)} p_j^{(t-1)} + \sigma^2\right)^{-1}$. This makes a total of $4c$ elements of the state set reserved for the interfered neighbors.

D. Actions

Unlike taking discrete steps on the previous transmit power level (see, e.g., [20]), we use discrete power levels taken between 0 and P_{max}. All agents have the same action space, i.e., $A_i = A_j = A$, $\forall i, j \in N$. Suppose we have $|A| > 1$ discrete power levels. Then, the action set is given by

$$A = \left\{0, \frac{P_{\text{max}}}{|A|-1}, \frac{2P_{\text{max}}}{|A|-1}, \ldots, P_{\text{max}}\right\}. \tag{19}$$

The total number of DQN output ports denoted as N_4 in Fig. 3a is equal to $|A|$. Agent i is only allowed to pick an action $a_i(t) \in A$ to update its power strategy at time slot t. This way of approaching the problem could increase the number of DQN output ports compared to [20], but it will increase the robustness of the learning algorithm. For example, as the maximum Doppler frequency f_d or time slot duration T increases, the correlation term ρ in (2) is going to decrease and the channel state will vary more. This situation may require the agents to react faster, i.e., possible transition from zero-power to full-power, which can be addressed efficiently with an action set composed of discrete power levels.

E. Reward Function

The reward function is designed to optimize the network objective (5). We interpret the reward as how the action of agent i through time slot t, i.e., $p_i^{(t)}$, affects the weighted sum-rate of its own and its future interfered neighbors $O_i^{(t+1)}$. During the time slot $t+1$, for all agent $i \in N$, the network trainer calculates the spectral efficiency of each link $k \in O_i^{(t+1)}$ without the interference from transmitter i as

$$C_{k\setminus i}^{(t)} = \log\left(1+\frac{g_{k\to k}^{(t)}p_k^{(t)}}{\sum_{j\neq i,k} g_{j\to k}^{(t)}p_j^{(t)}+\sigma^2}\right). \tag{20}$$

The network trainer computes the term $\sum_{j\neq i,k} g_{j\to k}^{(t)}p_j^{(t)} + \sigma^2$ in (20) by simply subtracting $g_{i\to k}^{(t)}p_i^{(t)}$ from the total interference-plus-noise power at receiver k in time slot t. As assumed in Sec. III, since transmitter $i \in I_k^{(t+1)}$, its interference to link k in slot t, i.e., $g_{i\to k}^{(t)}p_i^{(t)} > \eta\sigma^2$, is accurately measurable by receiver k and has been delivered to the network trainer.

In time slot t, we account for the externality that link i causes to link k using a price charged to link i for generating interference to link k [5]:

$$\pi_{i\to k}^{(t)} = w_k^{(t)}\left(C_{k\setminus i}^{(t)} - C_k^{(t)}\right). \tag{21}$$

Then, the reward function of agent $i \in N$ at time slot $t+1$ is defined as

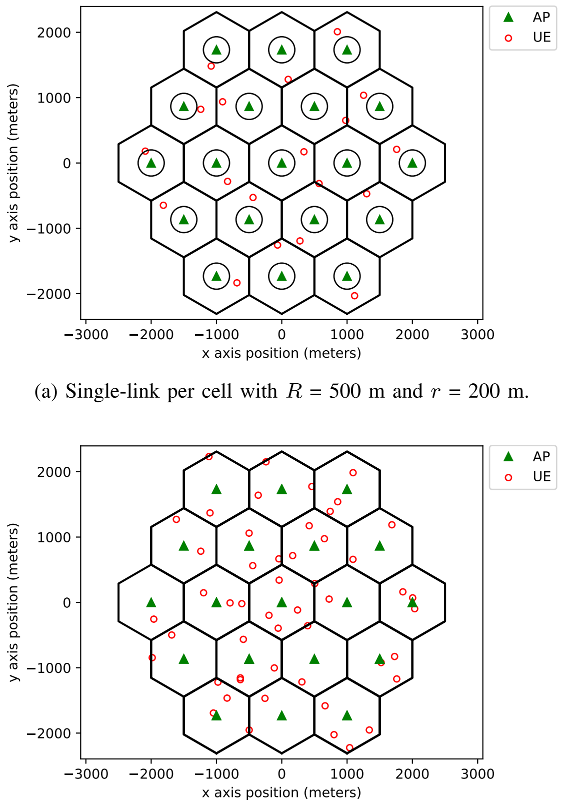
$$r_i^{(t+1)} = w_i^{(t)}C_i^{(t)} - \sum_{k\in O_i^{(t+1)}} \pi_{i\to k}^{(t)}. \tag{22}$$

The reward of agent i consists of two main components: its direct contribution to the network objective (5) and the penalty due to its interference to all interfered neighbors. Evidently, transmitting at peak power $p_i^{(t)} = P_{\max}$ maximizes the direct contribution as well as the penalty, whereas being silent earns zero reward.

V. Simulation Results

A. Simulation Setup

To begin with, we consider n links on n homogeneously deployed cells, where we choose n to be between 19 and 100. Transmitter i is located at the center of cell i and receiver i is located randomly within the cell. We also discuss the extendability of our algorithm to multi-link per cell scenarios in Sec. V-B. The half transmitter-to-transmitter distance is denoted as R and it is between 100 and 1000 meters. We also define an inner region of radius r where no receiver is allowed to be placed. We set the r to be between 10 and $R-1$ meters. Receiver i is placed randomly according to a uniform distribution on the area between out of the inner region of radius r and the cell boundary. Fig. 4 shows two network configuration examples. We set $P_{\max}$, i.e., the maximum transmit power level of transmitter i, to 38 dBm over 10 MHz frequency band which is fully reusable across all links. The distance dependent path loss between all transmitters and receivers is simulated by $120.9 + 37.6\log_{10}(d)$ (in dB), where d is transmitter-to-receiver distance in km. This path loss model is compliant with the LTE standard [36]. The log-normal shadowing standard deviation is taken as 8 dB. The AWGN power σ^2 is -114 dBm. We set the threshold η in (9) and (10) to 5. We assume full-buffer traffic model. Similar to [37], if the received SINR is greater than 30 dB, it is capped at 30 dB in the calculation of spectral efficiency by (4). This is to account for typical limitations of finite-precision digital processing. In addition to these parameters, we take the period of the time-slotted system T to be 20 ms. Unless otherwise stated, the maximum Doppler frequency f_d is 10 Hz and identical for all receivers.



(a) Single-link per cell with R = 500 m and r = 200 m.

(b) Multi-link per cell with R = 500 m and r = 10 m. Each cell has a random number of links from 1 to 4 links per cell.

Fig. 4. Network configuration examples with 19 cells.

We next describe the hyper-parameters used for the architecture of our algorithm. Since our goal is to ensure that the agents make their decisions as quickly as possible, we do not over-parameterize the network architecture and we use a relatively small network for training purposes. Our algorithm trains a DQN with one input layer, three hidden layers, and one output layer. The hidden layers have $N_1 = 200$, $N_2 = 100$, and $N_3 = 40$ neurons, respectively. We have 7 DQN input ports reserved for the local information feature group explained in Sec. IV-C. The cardinality constraint on the neighbor sets c is 5 agents. Hence, again from Sec. IV-C, the input ports reserved for the interferer and the interfered neighbors are $6c = 30$ and $4c = 20$, respectively. This makes a total of $N_0 = 57$ input ports reserved for the state set. (We also normalize the inputs with some constants depending on P_{max}, maximum intra-cell path loss, etc., to optimize the performance.) We use ten discrete power levels, $N_4 = |A| = 10$. Thus, the DQN has ten outputs. Initial parameters of the DQN are generated with the truncated normal distribution function of the TensorFlow [38]. For our application, we observed that the rectifier linear unit (ReLU) function converges to a desirable power allocation slightly slower than the hyperbolic tangent (tanh) function, so we used tanh as DQN's activation function. Memory parameters at the network trainer, M_b and M_m, are 256 and 1000 samples, respectively. We use the RMSProp algorithm [39] with an adaptive learning rate $\alpha^{(t)}$. For a more stable deep Q-learning outcome, the learning rate is reduced as $\alpha^{(t+1)} = (1 - \lambda)\alpha^{(t)}$, where $\lambda \in (0, 1)$ is the decay rate of $\alpha^{(t)}$ [40]. Here, $\alpha^{(0)}$ is 5×10^{-3} and λ is 10^{-4}. We also apply adaptive ϵ-greedy algorithm: $\epsilon^{(0)}$ is initialized to 0.2 and it follows $\epsilon^{(t+1)} = \max\left\{\epsilon_{\text{min}}, (1 - \lambda_\epsilon)\epsilon^{(t)}\right\}$, where $\epsilon_{\text{min}} = 10^{-2}$ and $\lambda_\epsilon = 10^{-4}$.

Although the discount factor γ is nearly arbitrarily chosen to be close to 1 and increasing γ potentially improves the outcomes of deep Q-learning for most of its applications [40], we set γ to 0.5. The reason we use a moderate level of γ is that the correlation between agent's actions and its future rewards tends to be smaller for our application due to fading. An agent's action has impact on its own future reward through its impact on the interference condition of its neighbors and consequences of their unpredictable actions. Thus, we set $\gamma \geq 0.5$. We observed that higher γ is not desirable either. It slows the DQN's reaction to channel changes, i.e., high f_d case. For high γ, the DQN converges to a strategy that makes the links with better steady-state channel condition greedy. As f_d becomes large, due to fading, the links with poor steady-state channel condition may become more advantageous for some time-slots. Having a moderate level of γ helps detect these cases and allows poor links to be activated during these time slots when they can contribute the network objective (5). Further, the training cycle duration T_u is 100 time slots. After we set the parameters in (18), we can compute the total number of DQN parameters, i.e., $|\theta|$, as 36,150 parameters. After each T_u time slots, trained parameters at the central controller will be delivered to all agents in T_d time slots via backhaul network as explained in Sec. IV-B. We assume that the parameters are transferred without any compression and the backhaul network uses pure peer-to-peer architecture. As $T_d = 50$ time slots, i.e., 1 second, the minimum required downlink/uplink capacity for all backhaul links is about 1 Mbps. Once the training stage is completed, the backhaul links will be used only for limited information exchange between neighbors which requires negligible backhaul link capacity.

We empirically validate the functionality of our algorithm. We implemented the proposed algorithm with TensorFlow [38]. Each result is an average of at least 10 randomly initialized simulations. We have two main phases for the simulations: training and testing. Each training lasts 40,000 time slots or $40,000 \times 20$ ms $= 800$ seconds, and each testing lasts 5,000 time slots or 100 seconds. During the testing, the trainer leaves the network and the ϵ-greedy algorithm is terminated, i.e., agents stop exploring the environment.

TABLE I

TESTING RESULTS FOR VARIANT HALF TRANSMITTER-TO-TRANSMITTER DISTANCE. $n = 19$ LINKS, $r = 10$ m, $f_d = 10$ Hz

	average sum-rate performance in bps/Hz per link						
	DQN		benchmark power allocations				
R (m)	matched	unmatched	WMMSE	FP	central	random	full-power
100	3.04	2.83	3.01	2.94	2.75	1.89	1.94
300	2.76	2.49	2.69	2.61	2.46	1.45	1.47
400	2.80	2.49	2.70	2.63	2.48	1.40	1.42
500	2.78	2.50	2.66	2.58	2.44	1.36	1.37
1000	2.71	2.43	2.61	2.54	2.40	1.31	1.33

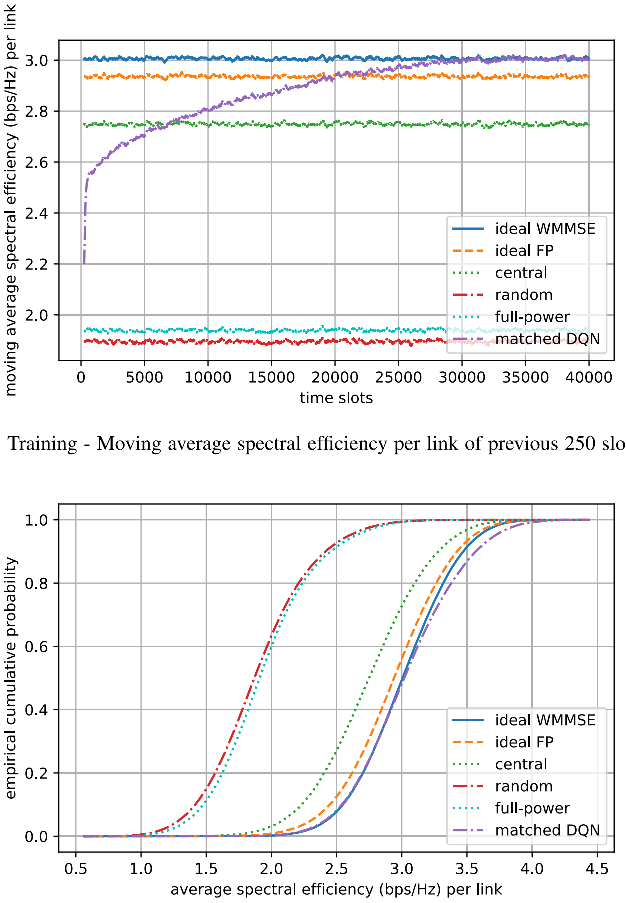
We have five benchmarks to evaluate the performance of our algorithm. The first two benchmarks are centralized 'ideal WMMSE' and 'ideal FP' with instantaneous full CSI. The third benchmark is the 'central power allocation' (central), where we introduce one time slot delay on the full CSI and feed it to the FP algorithm. Even though the single time slot delay to acquire the full CSI is not scalable in practical settings, it is a useful approach to reflect potential performance of negligible computation time achieved with the centralized supervised learning approach in [9]. The next benchmark is the 'random' allocation, where each agent chooses its transmit power for each slot at random uniformly between 0 and P_{max}. The last benchmark is the 'full-power' allocation, i.e., each agent's transmit power is P_{max} for all slots.

B. Sum-Rate Maximization

In this subsection, we focus on the sum-rate by setting the weights of all network agents to 1 through all time slots.

1) Robustness: We fix $n = 19$ links and use two approaches to evaluate performance. The first approach is the 'matched' DQN where we train a DQN from scratch during the first 40,000 time slots, whereas for the 'unmatched' DQN we skip the training stage and directly run the testing (the last 5,000 time slots) using a randomly picked DQN from the memory that was trained for another initialization with the same R and r parameters. Here an unmatched DQN is always trained for a random initialization with $n = 19$ links and $f_d = 10$ Hz.

In Table I, we vary R and see that training a DQN from scratch for the specific initialization is able to outperform both state-of-the-art centralized algorithms that are under ideal conditions such as full CSI and no delay. Interestingly, the unmatched DQN approach converges to the central power allocation where we feed the FP algorithm with delayed full CSI. The DQN approach achieves this performance with distributed execution and incomplete CSI. In addition, training a DQN from scratch enables our algorithm to learn to compensate for CSI delays and specialize for its network initialization scenario. Training a DQN from scratch swiftly converges in about 25,000 time slots (shown in Fig. 5a).



(a) Training - Moving average spectral efficiency per link of previous 250 slots.

(b) Testing - Empirical CDF.

Fig. 5. Sum-rate maximization. $n = 19$ links, $R = 100$ m, $r = 10$ m, $f_d = 10$ Hz.

TABLE II

Testing Results for Variant Inner Region Radius. $n = 19$ Links, $R = 500$ m, $f_d = 10$ Hz

	average sum-rate performance in bps/Hz per link						
	DQN		benchmark power allocations				
r (m)	matched	unmatched	WMMSE	FP	central	random	full-power
10	2.78	2.50	2.66	2.58	2.44	1.36	1.37
200	2.33	2.04	2.28	2.20	2.06	0.92	0.93
400	2.06	1.84	2.00	1.93	1.80	0.70	0.70
499	2.09	1.87	2.05	1.98	1.84	0.65	0.64

TABLE III

Testing Results for Variant Maximum Doppler Frequency. $n = 19$ Links, $R = 500$ m, $r = 10$ m. (‘Random’ Means f_d of Each Link Is Randomly Picked Between 2 Hz and 15 Hz for Each Time Slot t. ‘Uncorrelated’ Means That We Set $f_d \rightarrow \infty$ and ρ Becomes Zero)

	average sum-rate performance in bps/Hz per link						
	DQN		benchmark power allocations				
f_d (Hz)	matched	unmatched	WMMSE	FP	central	random	full-power
2	2.80	2.48	2.64	2.55	2.54	1.36	1.37
5	2.83	2.47	2.68	2.58	2.52	1.21	1.21
10	2.78	2.50	2.66	2.58	2.44	1.36	1.37
15	2.85	2.45	2.72	2.64	2.47	1.35	1.36
random	2.88	2.55	2.80	2.71	2.59	1.47	1.49
uncorrelated	2.82	2.41	2.68	2.61	2.39	1.55	1.57

TABLE IV

Testing Results for Variant Total Number of Links. $R = 500$ m, $r = 10$ m, $f_d = 10$ Hz

	average sum-rate performance in bps/Hz per link						
	DQN		benchmark power allocations				
n (links)	matched	unmatched	WMMSE	FP	central	random	full-power
19	2.78	2.50	2.66	2.58	2.44	1.36	1.37
50	2.28	1.99	2.17	2.13	2.00	1.01	1.02
100	1.92	1.68	1.90	1.88	1.74	0.87	0.89

Additional simulations with r and f_d taken as variables are summarized in Table II and Table III, respectively. As the area of receiver-free inner region increases, the receivers get closer to the interfering transmitters and the interference mitigation becomes more necessary. Hence, the random and full-power allocations tend to show much lower sum-rate performance compared to the central algorithms. For that case, our algorithm still shows decent performance and the convergence rate is still about 25,000 time slots. We also stress the DQN under various f_d scenarios. As we reduce f_d, its sum-rate performance remains unchanged, but the convergence time drops to 15,000 time slots. As $f_d \rightarrow \infty$, i.e., we set $\rho = 0$ to remove the temporal correlation between current channel condition and past channel conditions, the convergence takes more than 35,000 time slots. Intuitively, the reason of this effect on the convergence rate is that the variation of states visited during the training phase is proportional to f_d. Further, the comparable performance of the unmatched DQN with the central power allocation shows the robustness of our algorithm to the changes in interference conditions and fading characteristics of the environment.

2) Scalability: We increase the total number of links to investigate the scalability of our algorithm. As we increase n to 50 links, the DQN still converges in 25,000 time slots with high sum-rate performance. As we keep on increasing n to 100 links, from Table IV, the matched DQN's sum-rate outperformance drops because of the fixed input architecture of the DQN, i.e., each agent only considers $c = 5$ interferer and interfered neighbors. The performance of DQN can be improved for that case by increasing c at a higher computational complexity. Additionally, the unmatched DQN trained for just 19 links still shows good performance as we increase the number of links.

It is worth pointing out that each agent is able to determine its own action in less than 0.5 ms on a personal computer. Therefore, our algorithm is suitable for dynamic power allocation. In addition, running a single batch takes less than $T = 20$ ms. Most importantly, because of the fixed architecture of the DQN, increasing the total number of links from 19 to 100 has no impact on these values. It will just increase the queue memory in the network trainer. For the FP algorithm it takes about 15 ms to converge for $n = 19$ links, but with $n = 100$ links it becomes 35 ms. The WMMSE algorithm converges slightly slower, and the convergence time is still proportional to n which limits its scalability.

3) Extendability to Multi-Link per Cell Scenarios and Different Channel Models: We first consider a special homogeneous cell deployment case with co-located transmitters at the

TABLE V

TESTING RESULTS FOR VARIANT NUMBER OF LINKS PER CELL. 19 CELLS, $R = 500$ m, $r = 10$ m

	average sum-rate performance in bps/Hz per link						
	DQN		benchmark power allocations				
links per cell	matched	unmatched	WMMSE	FP	central	random	full-power
2	1.84	1.58	1.78	1.74	1.59	0.58	0.57
4	1.25	1.06	1.24	1.22	1.10	0.25	0.25
random	1.61	1.37	1.57	1.53	1.40	0.44	0.44

TABLE VI

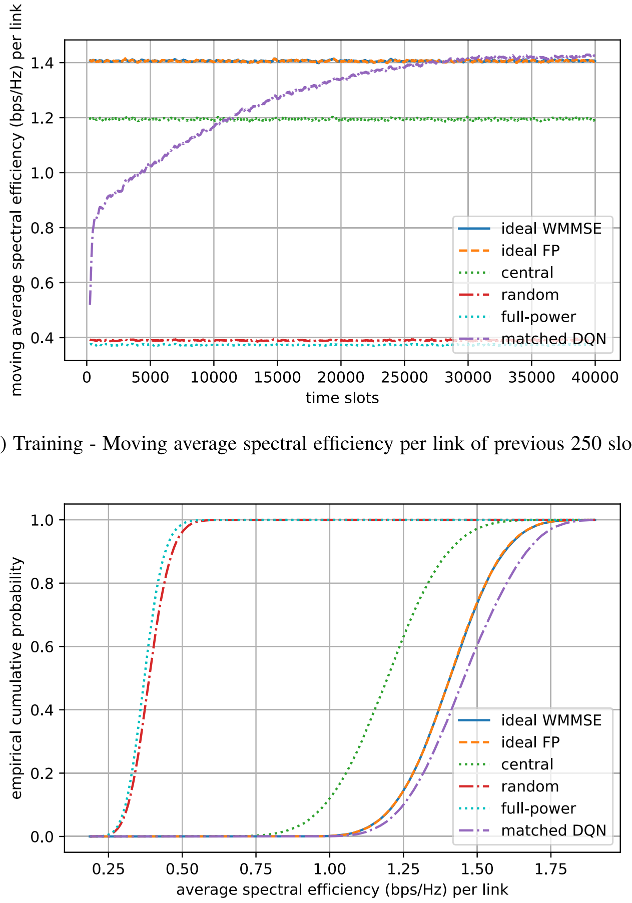
TESTING RESULTS FOR VARIANT NUMBER OF LINKS PER CELL AND UMI STREET CANYON MODEL. 19 CELLS, $R = 500$ m, $r = 10$ m

	average sum-rate performance in bps/Hz per link						
	DQN		benchmark power allocations				
links per cell	matched	unmatched	WMMSE	FP	central	random	full-power
2	2.60	2.29	2.53	2.52	2.27	1.04	0.99
4	1.46	1.23	1.41	1.41	1.19	0.39	0.37
random	2.09	1.78	2.01	2.01	1.77	0.78	0.76

cell centers. We also assume that no successive interference cancellation is performed [9]. The WMMSE and FP algorithms apply to this multi-link per cell scenario without any modifications.

We fix R and r to 500 and 10 meters, respectively. We set f_d to 10 Hz and the total number of cells to 19. We first consider two scenarios where each cell has 2 and 4 links, respectively. The third scenario assigns each cell a random number of links from 1 to 4 links per cell as shown in Fig. 4b. The testing stage results for these multi-link per cell scenarios are given in Table V. As shown in Table VI, we further test these scenarios using a different channel model called urban micro-cell (UMi) street canyon model of [41]. For this model, we take the carrier frequency as 1 GHz. The transmitter and receiver antenna heights are assumed to be 10 and 1.5 meters, respectively.

Our simulations for these scenarios show that as we increase number of links per cell, the training stage still converges in about 25,000 time slots. Fig. 6a shows the convergence rate of training stage for 4 links per cell scenario with 76 links. In Fig. 6a, we also show that using a different channel model, i.e., UMi street canyon, does not affect the convergence rate. Although the convergence rate is unaffected, the proposed algorithm's average sum-rate performance decreases as we increase number of links per cell. Our algorithm still outperforms the centralized algorithms even for 4 links per cell scenario for both channel models. Another interesting fact is that although the unmatched DQN was trained for a single-link deployment scenario and can not handle the delayed CSI constraint as good as the matched DQN, it gives comparable performance with the 'central' case. Thus, the unmatched DQN is capable of finding good estimates of optimal actions for unseen local state inputs.



(a) Training - Moving average spectral efficiency per link of previous 250 slots.

(b) Testing - Empirical CDF.

Fig. 6. Sum-rate maximization. 4 links per cell scenario. UMi street canyon. $n = 76$ links deployed on 19 cells, $R = 500$ m, $r = 10$ m, $f_d = 10$ Hz.

TABLE VII

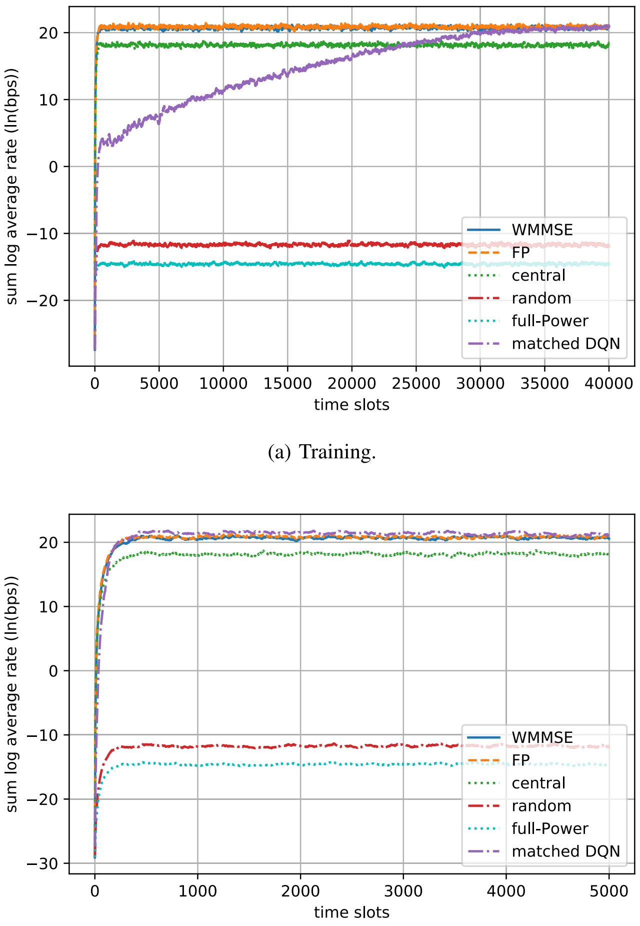
PROPORTIONAL FAIR SCHEDULING WITH VARIANT HALF TRANSMITTER-TO-TRANSMITTER DISTANCE. $n = 19$ LINKS, $r = 10$ m, $f_d = 10$ Hz

	convergence of the network sum log-average rate (ln (bps))						
	DQN		benchmark power allocations				
R (m)	matched	unmatched	WMMSE	FP	central	random	full-power
100	26.25	24.75	29.12	28.27	25.21	15.03	14.36
300	22.95	21.53	23.80	23.31	20.57	-2.64	-4.88
400	22.72	20.91	22.64	22.48	19.85	-7.52	-10.05
500	21.25	18.45	20.69	20.88	18.19	-11.76	-14.59
1000	18.37	14.67	17.27	17.34	14.53	-16.66	-19.64

C. Proportionally Fair Scheduling

In this subsection, we change the link weights according to (7) to ensure fairness as described in Sec. III. We choose the β term in (6) to be 0.01 and use convergence to the objective in (8) as performance-metric of the DQN. We also make some additions to the training and testing stage of DQN. We need an initialization for the link weights. This is done by letting all transmitters to serve their receivers with full-power at $t = 0$, and initialize weights according to the initial spectral efficiencies computed from (4). For the testing stage, we reinitialize the weights after the first 40,000 slots to see whether the trained DQN can achieve fairness as fast as the centralized algorithms.

As shown in Fig. 7, the training stage converges to a desirable scheduling in about 30,000 time slots. Once the network is trained, as we reinitialize the link weights, our algorithm converges to an optimal scheduling in a distributed



(a) Training.

(b) Testing.

Fig. 7. Proportionally fair scheduling. $n = 19$ links, $R = 500$ m, $r = 10$ m, $f_d = 10$ Hz

TABLE VIII

PROPORTIONAL FAIR SCHEDULING WITH VARIANT INNER REGION RADIUS. $n = 19$ LINKS, $R = 500$ m, $f_d = 10$ Hz

	convergence of the network sum log-average rate (ln (bps))						
	DQN		benchmark power allocations				
r (m)	matched	unmatched	WMMSE	FP	central	random	full-power
10	21.25	18.45	20.69	20.88	18.19	-11.76	-14.59
200	20.24	17.78	19.01	19.25	16.58	-16.31	-19.43
400	16.65	14.82	16.70	16.84	13.92	-26.82	-30.35
499	13.99	12.43	14.12	14.60	11.56	-35.46	-39.29

fashion as fast as the centralized algorithms. Next, we set R and r as variables to get results in Table VII and Table VIII. We see that the trained DQN from scratch still outperforms the centralized algorithms in most of the initializations, using the unmatched DQN also achieves a high performance similar to the previous sections.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a distributively executed model-free power allocation algorithm which outperforms or achieves comparable performance with existing state-of-the-art centralized algorithms. We see potentials in applying the reinforcement learning techniques on various dynamic wireless network resource management tasks in place of the optimization techniques. The proposed approach returns a suboptimal power allocation much quicker than two popular centralized algorithms. In contrast to most advanced optimization based power control algorithms (e.g., WMMSE and FP) which require both instant and accurate measurements of all channel gains, the proposed algorithm only requires delayed accurate measurements of some received power values that are sufficiently strong. An extension to the case with inaccurate CSI measurements is left for future work.

Meng *et al.* [42] is an extension of [8] to multiple users in a cell, which is also addressed in the current paper. Although the centralized training phase seems to limit scalability, we have shown that a DQN trained for a smaller wireless network can be applied to a larger wireless network. Also, a jump-start on the training of DQN can also be implemented by using initial parameters taken from another DQN previously trained for a different setup.

Finally, while a centralized training approach saves computational resources and converges faster, distributed training may beat a path for an extension of the proposed algorithm to some other channel deployment scenarios. The main hurdle on the way to apply distributed training is to avoid the instability caused by the environment non-stationarity.

ACKNOWLEDGEMENT

The authors would like to thank Dr. Gang Qian, Dr. Mingyi Hong, Dr. Wei Yu, and Dr. Georgios Giannakis for stimulating discussions.

REFERENCES

[1] M. Chiang, P. Hande, T. Lan, and C. W. Tan, "Power control in wireless cellular networks," *Found. Trends Netw.*, vol. 2, no. 4, pp. 381–533, Apr. 2008.

[2] Q. Shi, M. Razaviyayn, Z.-Q. Luo, and C. He, "An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel," *IEEE Trans. Signal Process.*, vol. 59, no. 9, pp. 4331–4340, Sep. 2011.

[3] K. Shen and W. Yu, "Fractional programming for communication systems—Part I: Power control and beamforming," *IEEE Trans. Signal Process.*, vol. 66, no. 10, pp. 2616–2630, May 2018.

[4] I. Sohn, "Distributed downlink power control by message-passing for very large-scale networks," *Int.J. Distrib. Sensor Netw.*, vol. 11, no. 8, p. e902838, 2015.

[5] J. Huang, R. A. Berry, and M. L. Honig, "Distributed interference compensation for wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 5, pp. 1074–1084, May 2006.

[6] S. G. Kiani, G. E. Oien, and D. Gesbert, "Maximizing multicell capacity using distributed power allocation and scheduling," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Mar. 2007, pp. 1690–1694.

[7] H. Zhang, L. Venturino, N. Prasad, P. Li, S. Rangarajan, and X. Wang, "Weighted sum-rate maximization in multi-cell networks via coordinated scheduling and discrete power control," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 6, pp. 1214–1224, Jun. 2011.

[8] Y. S. Nasir and D. Guo, "Deep reinforcement learning for distributed dynamic power allocation in wireless networks," *arXiv:1808.00490*, Aug. 2018.

[9] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for interference management," *IEEE Trans. Signal Process.*, vol. 66, no. 20, pp. 5438–5453, Oct. 2018.

[10] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Jan. 2005.

[11] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[12] L. Liang, J. Kim, S. C. Jha, K. Sivanesan, and G. Y. Li, "Spectrum and power allocation for vehicular communications with delayed CSI feedback," *IEEE Wireless Commun. Lett.*, vol. 6, no. 4, pp. 458–461, Aug. 2017.

[13] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, to be published.

[14] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.

[15] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1277–1290, Jun. 2019.

[16] R. Li *et al.*, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74429–74441, 2018.

[17] M. Bennis and D. Niyato, "A Q-learning based approach to interference avoidance in self-organized femtocell networks," in *Proc. IEEE Global Commun. Conf. Workshops (GLOBECOM)*, Dec. 2010, pp. 706–710.

[18] M. Simsek, A. Czylwik, A. Galindo-Serrano, and L. Giupponi, "Improved decentralized Q-learning algorithm for interference reduction in LTE-femtocells," in *Proc. Wireless Adv.*, Jun. 2011, pp. 138–143.

[19] R. Amiri, M. A. Almasi, J. G. Andrews, and H. Mehrpouyan, "Reinforcement learning for self organization and power control of two-tier heterogeneous networks," *IEEE Trans. Wireless Commun.*, to be published.

[20] E. Ghadimi, F. D. Calabrese, G. Peters, and P. Soldati, "A reinforcement learning approach to power control and rate adaptation in cellular networks," in *Proc. IEEE Int. Conf. Commun.*, May 2017, pp. 1–7.

[21] F. D. Calabrese, L. Wang, E. Ghadimi, G. Peters, L. Hanzo, and P. Soldati, "Learning radio resource management in RANs: Framework, opportunities, and challenges," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 138–145, Sep. 2018.

[22] Z.-Q. Luo and S. Zhang, "Dynamic spectrum management: Complexity and duality," *IEEE J. Sel. Topics Signal Process.*, vol. 2, no. 1, pp. 57–73, Feb. 2008.

[23] D. N. C. Tse and P. Viswanath, *Fundamentals Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[24] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, Jan. 1996.

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

[26] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári, "Convergence results for single-step on-policy reinforcement-learning algorithms," *Mach. Learn.*, vol. 38, no. 3, pp. 287–308, 2000.

[27] A. Galindo-Serrano and L. Giupponi, "Distributed Q-learning for interference control in OFDMA-based femtocell networks," in *Proc. IEEE Veh. Technol. Conf. (VTC Spring)*, May 2010, pp. 1–5.

[28] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 310–323, Jan. 2019.

[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[30] J. Hu and M. P. Wellman, "Online learning about other agents in a dynamic multiagent system," in *Proc. Int. Conf. Auto. Agents*, vol. 10, no. 13, pp. 239–246, 1998.

[31] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications," 2018, *arXiv:1812.11794*.

[32] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, Aug. 2017, pp. 1146–1155.

[33] A. Tampuu *et al.*, "Multiagent cooperation and competition with deep reinforcement learning," *PLoS ONE*, vol. 12, no. 4, p. e0172396, 2017.

[34] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.* Cham, Switzerland: Springer, 2017, pp. 66–83.

[35] J. Watt, R. Borhani, and A. K. Katsaggelos, *Machine Learning Refined: Foundations, Algorithms, and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2016.

[36] *Radio Frequency (RF) System Scenarios*, document 3GPP TR 25.942, v.14.0.0, 2017.

[37] B. Zhuang, D. Guo, and M. L. Honig, "Energy-efficient cell activation, user association, and spectrum allocation in heterogeneous networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 4, pp. 823–831, Apr. 2016.

[38] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. USENIX Symp. Operating Syst. Design Implement.*, 2016, pp. 265–283.

[39] S. Ruder, "An overview of gradient descent optimization algorithms," Sep. 2016, *arXiv:1609.04747*.

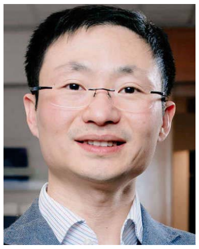
[40] V. François-Lavet, R. Fonteneau, and D. Ernst, "How to discount deep reinforcement learning: Towards new dynamic strategies," in *Proc. NIPS Workshop Deep Reinforcement Learning*, Dec. 2015, pp. 1–9.

[41] *Study on Channel Model for Frequencies From 0.5 to 100 GHz*, document 3GPP TR 38.901, v.14.0.0, 2017.

[42] F. Meng, P. Chen, and L. Wu, "Power allocation in multi-user cellular networks with deep Q learning approach," in *Proc. IEEE Int. Conf. Commun.*, May 2019, pp. 1–6.

Yasar Sinan Nasir (S'17) received the B.S. degree (as valedictorian) in electrical engineering from Bilkent University, Ankara, Turkey, in 2016, and the M.S. degree in electrical engineering from Northwestern University, Evanston, IL, USA, in 2018, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering.

He has held summer research internships at Fraunhofer IIS in 2015, 5G Radio Research, Nokia Bell Labs, in 2018, and SmartRF Team, Qualcomm, in 2019. His research interests include machine learning with a focus on multi-agent deep reinforcement learning applications, software-defined radios, wireless communications, and networking with emphasis on large-scale dynamic resource management problems in future generation cellular networks.



Dongning Guo (S'97–M'05–SM'11) received the Ph.D. degree from Princeton University, Princeton, NJ, USA. He then joined the faculty of Northwestern University, Evanston, IL, USA, where he is currently a Professor with the Department of Electrical and Computer Engineering.

Dr. Guo received the IEEE Marconi Prize Paper Award in Wireless Communications in 2010 and the Best Paper Award at the 2017 IEEE Wireless Communications and Networking Conference. He was a recipient of the National Science Foundation Faculty Early Career Development (CAREER) Award in 2007. He has been an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION THEORY and a Guest Editor of a Special Issue of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. He is an Editor of *Foundations and Trends in Communications and Information Theory* and an Associate Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS.