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

Abstract

This study illustrates the effectiveness of employing deep reinforcement learning techniques for managing transmit power control in wireless networks. Unlike conventional methods that tackle a demanding optimization problem to determine nearly optimal power allocations, many of which are impractical for large networks in real-world situations due to computational complexity and the need for instantaneous cross-cell channel state information (CSI), this paper introduces a distributed dynamic power allocation scheme based on model-free deep reinforcement learning.

In this approach, each transmitter gathers CSI and quality of service (QoS) data from multiple neighbors and adjusts its transmit power accordingly. The primary goal is to maximize a weighted sum-rate utility function, which can be customized to achieve either maximum sum-rate or proportionally fair scheduling. Deep Q-learning is employed to address both random variations and delays in the CSI. The proposed algorithm is demonstrated to achieve near-optimal power allocation in real-time for a typical network architecture, relying on delayed CSI measurements available to the agents. This scheme is particularly well-suited for practical scenarios where the system model is inaccurate, and CSI delay is non-negligible.

Keywords: Deep Q-learning, radio resource management,interference mitigation,power control,Jakes fading model.

1 Introduction

In emerging and future wireless networks, effective inter-cell interference management poses a significant technological challenge, especially as access point (AP) deployment becomes denser to meet escalating demands. The dilemma arises as a transmitter may increase its transmit power to enhance its own data rate, potentially compromising the links it interferes with. Transmit power control has been a staple feature since the inception of first-generation cellular networks [3].

Various centralized and distributed optimization techniques have been employed to devise algorithms for achieving suboptimal power allocation [13]. For benchmarking purposes, we have chosen two state-of-the-art algorithms. These include the weighted minimum mean square error (WMMSE) algorithm [9] and an iterative algorithm based on fractional programming (FP) [8]. In their general formulations, both algorithms necessitate complete and up-to-date cross-cell channel state information (CSI).

This study marks the pioneering application of deep reinforcement learning to power control [7]. Previous work by Sun et al. [12] introduced a centralized supervised learning approach, employing a fast deep neural

network (DNN) that achieved a sum-rate of 90% or higher compared to the WMMSE algorithm. However, this approach still relies on complete CSI. An additional challenge is the dependence on a substantial dataset generated by the WMMSE algorithm, a process that is time-consuming due to the computational complexity of WMMSE. As the network scales up, the total number of input and output ports for the DNN also increases, raising concerns about the scalability of the centralized solution proposed in [12].

Moreover, the efficacy of supervised learning hinges on the accuracy of the system model underlying the training data. This implies that new training data is required each time the system model or key parameters undergo changes, adding a layer of complexity and potentially limiting adaptability in dynamic network environments.

2 Related works

The deep reinforcement learning framework has found application in various wireless communication problems [6]. Classical Q-learning techniques have been utilized in addressing power allocation challenges in [2]. For instance, the objectives in [10] involve reducing interference in LTE-Femtocells. In contrast to deep Q-learning, these classical algorithms construct a lookup table to represent the values of state-action pairs. As a consequence, [10] model the wireless environment using a discrete state set and impose limitations on the number of learning agents.

Amiri et al. [1] have employed cooperative Q-learning for power control to enhance the Quality of Service (QoS) for users in femtocells, although without considering channel variations. The application of deep Q-learning for power allocation to maximize the network objective is also explored in [2]. Similar to the proposed approach, the work in [2] adopts a distributed framework with a centralized training assumption. However, they benchmark their algorithm against a fixed power allocation scheme rather than state-of-the-art algorithms.

In this paper, the proposed approach introduces novelty and uniqueness in both the representation of the wireless environment and the reward function. Specifically, the approach addresses the stochastic nature of the wireless environment and handles incomplete/delayed Channel State Information (CSI). As a result, the proposed approach quickly arrives at highly competitive strategies.

2.1 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) [4]. The model has also been used to describe a simple cellular network with n APs, where each AP serves a single user device [8]. Let $N = \{1, ..., n\}$ 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

$$g_{i\to j}^{(t)} = |h_{i\to j}^{(t)}|^2 \alpha_{i\to j}, t = 1, 2, 3, \dots \tag{1}$$

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 $e_{i\to j}^{(1)}$, $e_{i\to j}^{(2)}$,... 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 J0(.) is the zeroth-order Bessel function of the first kind and fd 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 $p = [p_1, ..., p_n]^T$:

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

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:

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

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 $p^{(t)} = [p_1^{(t)}, \ldots, p_n^{(t)}]^T$.

2.2 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. 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. 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 $e^{(t+1)} = (s^{(t)}, a^{(t)}, r^{(t+1)}, s^{(t+1)})$, which describes an interaction with the environment.

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:

$$R^{(t)} = \sum_{\tau=0}^{\infty} \gamma^\tau r^{(t+\tau+1)} \tag{5}$$

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, i.e.,

3 Overview of the method

As depicted in Fig. 2, we propose a multi-agent deep reinforcement learning scheme with each transmitter as an agent. Similar to, 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 above 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. 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. 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 multiagent learning frameworks and there exists several multiagent Q-learning adaptations. However, multi-agent learning is an open research area and theoretical guarantees for these adaptations are rare and incomplete despite their good empirical performances

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 $Q^{(t)}_{target}$ 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. 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.

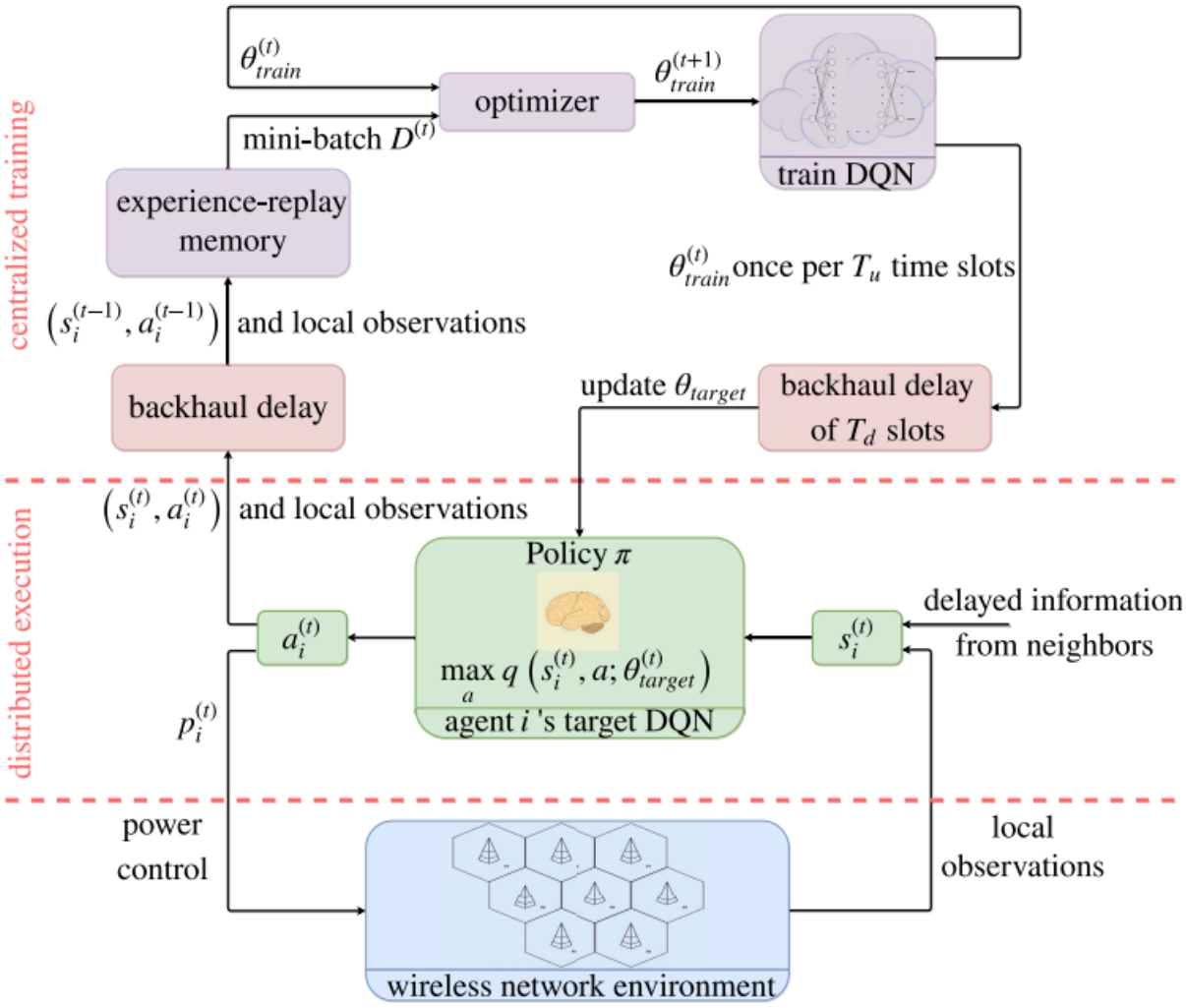

Figure 1. Illustration of the proposed multi-agent deep reinforcement learning algorithm.

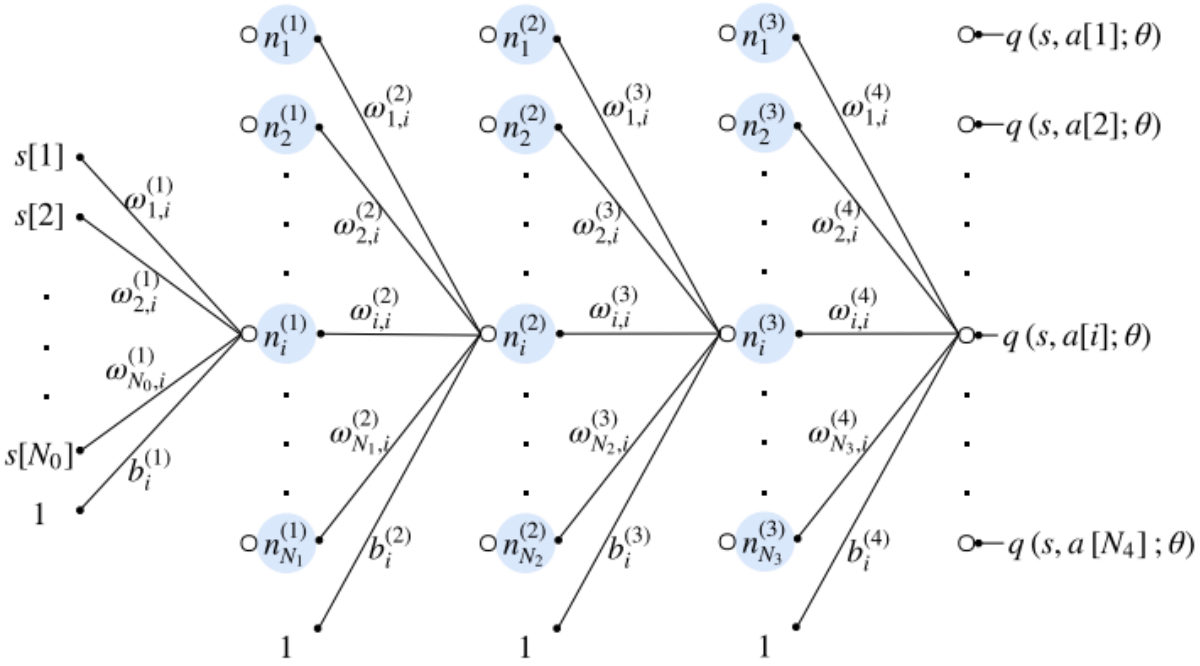

Figure 2. The illustration of all five layers of the proposed DQN: The input layeris followed by three hidden layers and an output layer. The notation n,wand b indicate DQN neurons,weights, and biases, respectively. These weightsand biases form the set of DQN parameters denoted as 0. The biases are notassociated with any neuron and we multiply these biases by the scalar value 1.

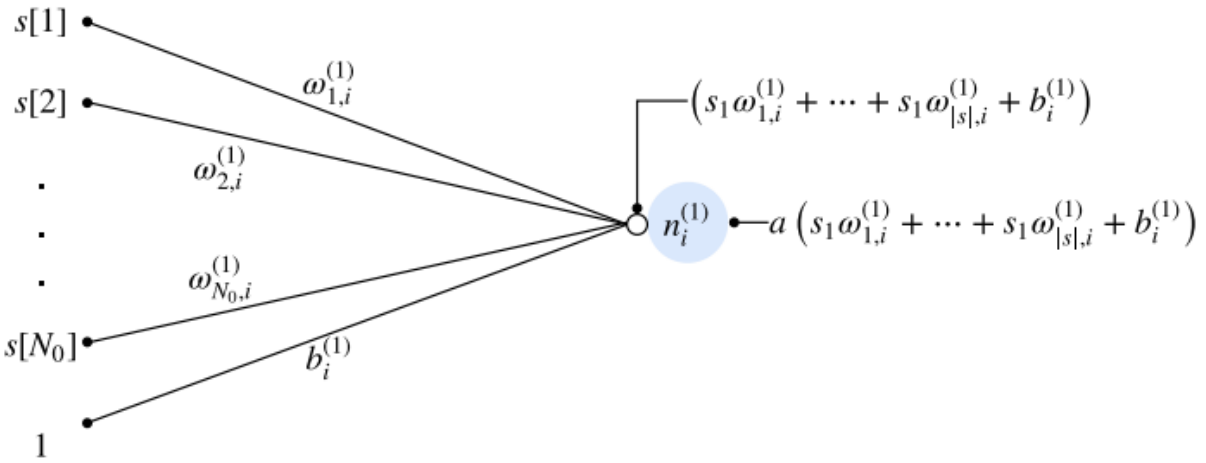

Figure 3. The functionality of a single neuron extracted from the first hidden-layer.a(.) denotes the non-linear activation function.

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 N4 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.

4 Implementation details

4.1 Comparing with the released source codes

As I am not very familiar with this field, I did not propose any improvements to the results of the paper. But while reading the paper and code, I have come to understand the general idea and structure of the paper, and I hope to have more insights in future learning.

4.2 Experimental environment setup

To begin with, we consider n links on n homogeneously deployed cells, where we choose n to be between 19 and 100. Transmitter n is located at the center of cell n and receiver n 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

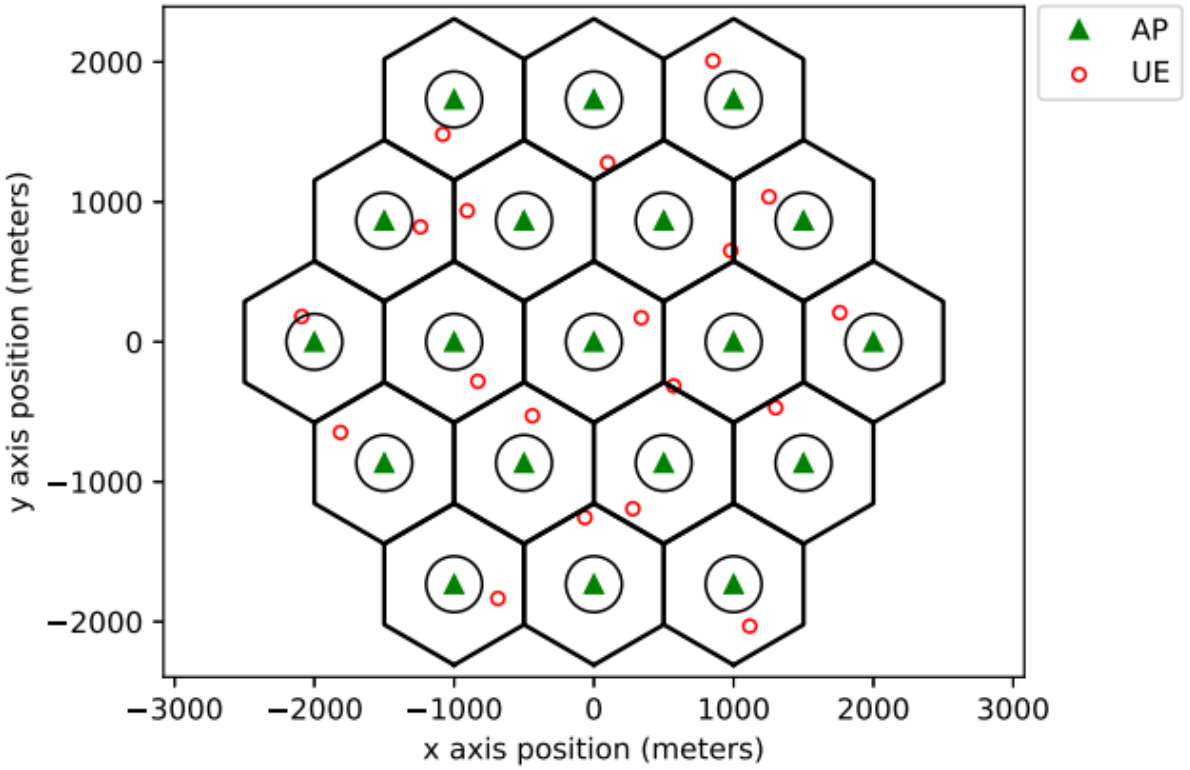

Figure 4. Single-link per cell with R = 500 m and r = 200 m.

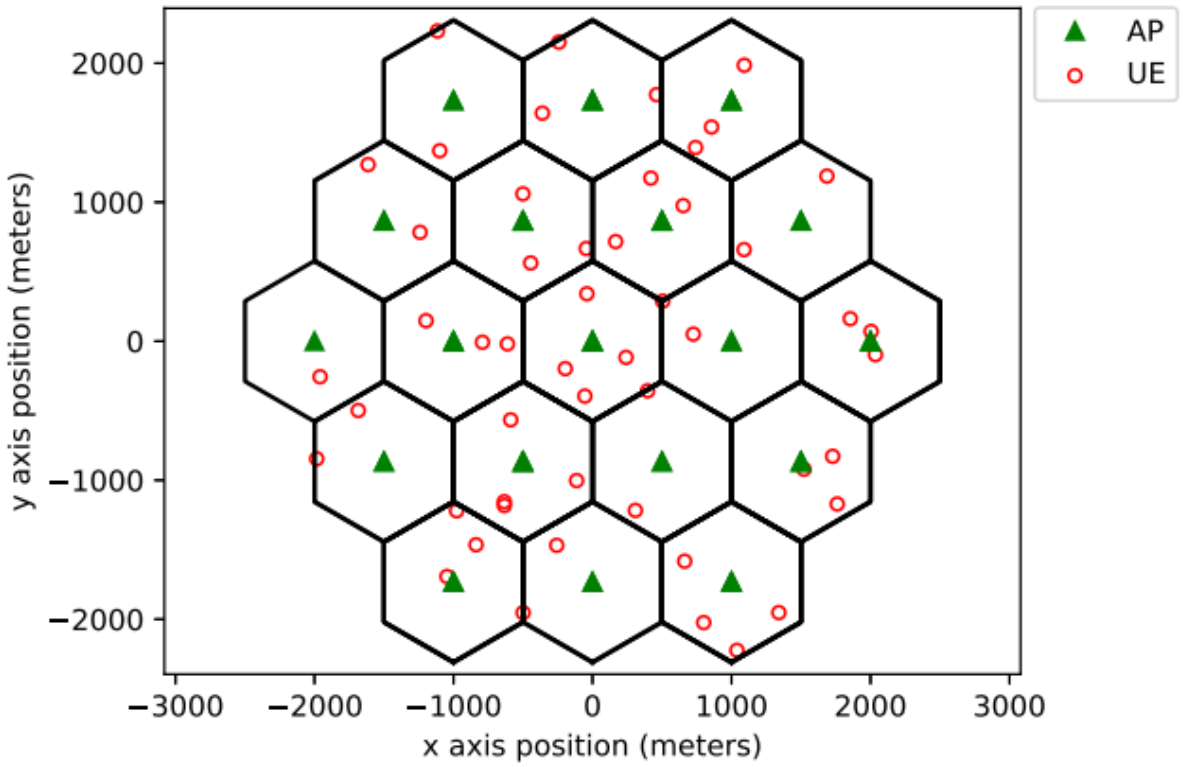

Figure 5. Multi-link per cell with R = 500 m and r = 10 m. Each cell hasa random number of links from 1 to 4 links per cell.

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 $120.9 + 37.6log_{10}(d)(indB)$, where d is transmitter-to-receiver distance in km. This path loss model is compliant with the LTE standard. The log-normal shadowing standard deviation is taken as 8 dB. The AWGN power σ^2 We set the threshold η in (9) and (10) to 5. We assume full-buffer traffic model.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

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 Pmax, 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. 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 with an adaptive learning rate $\alpha(t)$. For a more stable deep Q-learning outcome, the learning rate is reduced as $\alpha^{(t+1)} = (1 - \lambda)\alpha^{(t)}$. Here, $\alpha^{(0)}$ is 5 × 10^{-3} and $lambda$ is 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 , 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 fd case. For high γ, the DQN converges to a strategy that makesl 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 Tu 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 Tu time slots, trained parameters at the central controller will be delivered to all agents in Td 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 Td = 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. 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 $400000 * 20$ ms = 800 seconds, and each testing lasts 5,000 time slots or 100 seconds. During the testing, the trainer leaves the network and the greedy algorithm is terminated, i.e., agents stop exploring the environment.

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 other methods. 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.

5 Results and analysis

After setting up the experimental steps and configuring the environment required for code execution, the results obtained by running the code are shown in the following figure. The data graph obtained by the original

author has the same trend of change and data points.

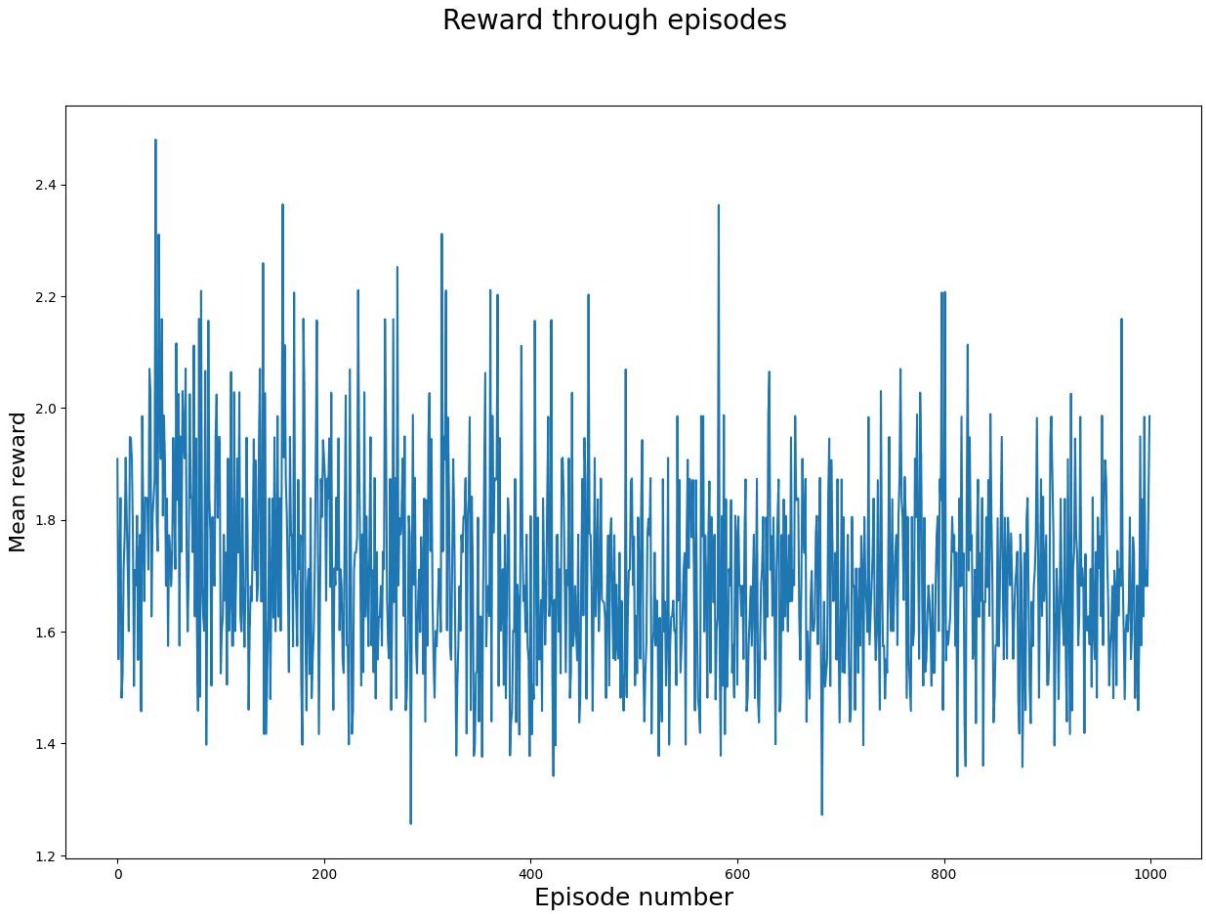

Figure 6. Experimental results

6 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.

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.

References

[1] Roohollah Amiri, Mojtaba Ahmadi Almasi, Jeffrey G Andrews, and Hani Mehrpouyan. Reinforcement learning for self organization and power control of two-tier heterogeneous networks. *IEEE Transactions on Wireless Communications*, 18(8):3933–3947, 2019.

[2] Francesco Davide Calabrese, Li Wang, Euhanna Ghadimi, Gunnar Peters, Lajos Hanzo, and Pablo Soldati. Learning radio resource management in rans: Framework, opportunities, and challenges. *IEEE Communications Magazine*, 56(9):138–145, 2018.

[3] Mung Chiang, Prashanth Hande, Tian Lan, Chee Wei Tan, et al. Power control in wireless cellular networks. *Foundations and Trends® in Networking*, 2(4):381–533, 2008.

[4] Jianwei Huang, Randall A Berry, and Michael L Honig. Distributed interference compensation for wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(5):1074–1084, 2006.

[5] Saad G Kiani, Geir E Oien, and David Gesbert. Maximizing multicell capacity using distributed power allocation and scheduling. In *2007 IEEE Wireless Communications and Networking Conference*, pages 1690–1694. IEEE, 2007.

[6] Rongpeng Li, Zhifeng Zhao, Qi Sun, I Chih-Lin, Chenyang Yang, Xianfu Chen, Minjian Zhao, and Honggang Zhang. Deep reinforcement learning for resource management in network slicing. *IEEE Access*, 6:74429–74441, 2018.

[7] Yasar Sinan Nasir and Dongning Guo. Deep reinforcement learning for distributed dynamic power allocation in wireless networks. *arXiv preprint arXiv:1808.00490*, 8:2018, 2018.

[8] Kaiming Shen and Wei Yu. Fractional programming for communication systems—part i: Power control and beamforming. *IEEE Transactions on Signal Processing*, 66(10):2616–2630, 2018.

[9] Qingjiang Shi, Meisam Razaviyayn, Zhi-Quan Luo, and Chen He. An iteratively weighted mmse approach to distributed sum-utility maximization for a mimo interfering broadcast channel. *IEEE Transactions on Signal Processing*, 59(9):4331–4340, 2011.

[10] Meryem Simsek, Andreas Czylwik, Ana Galindo-Serrano, and Lorenza Giupponi. Improved decentralized q-learning algorithm for interference reduction in lte-femtocells. In *2011 Wireless Advanced*, pages 138–143. IEEE, 2011.

[11] Illsoo Sohn. Distributed downlink power control by message-passing for very large-scale networks. *International Journal of Distributed Sensor Networks*, 11(8):902838, 2015.

[12] Haoran Sun, Xiangyi Chen, Qingjiang Shi, Mingyi Hong, Xiao Fu, and Nicholas D Sidiropoulos. Learning to optimize: Training deep neural networks for interference management. *IEEE Transactions on Signal Processing*, 66(20):5438–5453, 2018.

[13] Honghai Zhang, Luca Venturino, Narayan Prasad, Peilong Li, Sampath Rangarajan, and Xiaodong Wang. Weighted sum-rate maximization in multi-cell networks via coordinated scheduling and discrete power control. *IEEE Journal on Selected Areas in Communications*, 29(6):1214–1224, 2011.