NORTHWESTERN UNIVERSITY

Deep Reinforcement Learning based Wireless Resource Management

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Electrical and Computer Engineering

By

Yasar Sinan Nasir

EVANSTON, ILLINOIS

December 2021

ABSTRACT

Deep Reinforcement Learning based Wireless Resource Management

Yasar Sinan Nasir

Next generation cellular networks are expected to support a massive data traffic volume and satisfy a vast number of users that have latency-critical quality-of-service expectations. Towards serving this demand, it is envisaged that the interference management problem will be the main bottleneck due to the likeliness of a heavily interfering wireless environment caused by much denser deployment of base stations and mobiles. Due to inherently scarce shared frequency-band resources over time-varying traffic and multi-channel conditions, a scalable and practical fast-timescale resource management is an absolute necessity towards next generation cellular networks. The conventional optimization based resource management schemes are either practically infeasible, computationally challenging, or intractable due to relying on model-driven techniques. Therefore, in the past few years, there has been extensive research on model-free reinforcement learning based resource management. Reinforcement learning is purely data-driven, and its multi-agent adaptation is promising for scalability on larger networks where agents collaboratively work together towards a shared objective.

We initially show the potential of deep reinforcement learning for transmit power control in wireless networks. Existing power control 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. The proposed method is a distributively executed dynamic power allocation scheme that maximizes a weighted sum-rate objective, which can be particularized to achieve maximum sum-rate or proportionally fair scheduling. Each transmitter collects delayed channel measurements of a time-varying channel from its neighbors and adapts its own transmit power accordingly. Both random variations and delays in the CSI are inherently addressed using deep Q-learning. For a typical network architecture with single subband and full-buffer traffic, 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.

Next, we integrate the proposed power control algorithm to the case of mobile devices for which the channel conditions change not only due to fast fading but also due to the device movements. We further include the continuous power control by replacing deep Q-learning, which applies only to discrete action spaces and requires transmit power to be quantized, with deep deterministic policy gradient algorithm which is an actor-critic learning method that applies to the continuous action spaces as well. Additionally, for the case of multiple-frequency bands, we propose a novel approach for the joint subband selection and power allocation problem that consists of two layers, where the bottom layer is responsible for continuous power allocation at the physical layer by using deep deterministic policy gradient algorithm, and the top layer does discrete subband selection with deep Q-learning.

Finally, we propose a multi-agent deep reinforcement learning based resource management scheme that can instantaneously respond to the changes in both traffic and channel dynamics. With the help of a novel reward function design, each learning agent appropriately adapts its resources within each time slot to stabilize its own and its neighbors' queue lengths, and agents collaboratively maximize the long-term quality of service over their local environment by minimizing the average packet delay. The local state consists of user priorities and channel measurements. User priorities connect physical layer resource management with the network layer. User priorities can represent link weights of a proportionally fair scheme, queue lengths, or anything else specified by the network layer according to the type of user service and quality of service requirements. We also consider several practicality constraints on channel measurements, so we build the local state set with aggregated interference instead of individual channel gain measurements. Using simulations, we demonstrate the effectiveness of the proposed approach compared to an optimization based resource allocation scheme which follows proportional fairness but lacks instantaneous interaction with queue states.

Acknowledgements

Before anything else, I would like to sincerely thank my advisor Professor Dongning Guo for all of his insightful guidance, extensive advising, and continuous patience and support over my time at Northwestern. In additional to his excellent research guidance, his dedication to prioritize the practicality and applicability of our work has put great value to this thesis. Having Prof. Guo as my advisor and joining his excellent research group was one of my wisest decisions. I believe that the priceless lessons I learnt from him will continue to benefit me for the rest of my life.

I also would like to thank Professor Michael Honig, Professor Mingyi Hong and Professor Zhaoran Wang for being on my thesis committee as well as for their invaluable feedback. Initial contributions led by Professor Mingyi Hong's group were quite insightful during the early stages of this thesis. Additionally, I would like to thank Professor Randall Berry, Professor Ermin Wei and Professor Chung-Chieh Lee for their teaching over my time here at Northwestern.

I feel fortunate to all the colleagues and friends in the Communications and Networking Laboratory here at Northwestern University, including Jing, Abdu, Zhiyi, Ryan, Howard, Ding, Brendan, Xu, Hao, Yining, Tho, Fatemeh, Charikleia, and Pawan. I am also very thankful to my friends at Northwestern outside of the Commnet lab: Semih, Kerim, Can Gurkan, Can Aygen, Metehan, Erkin, Ridvan, Yigitcan, Enes, Ege, Baris, Simone, Enrico, and Ettore.

Finally, and most importantly, I want to thank my parents Nursen and Levent, and my little sister Deniz for their endless love and support which means more to me than I can express. I could not have finished this without them. To them I dedicate this dissertation.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Future wireless networks will be characterized by an immense level of complexity, so the traditional wireless resource management approaches will no longer be sufficient. Therefore, the future wireless networks will have to rely on recent advances in Artificial Intelligence.

The wireless resource management involves many optimization tasks in physical layer, link layer, and network layer to maximize an overall long-term utility. In this thesis, we start from the basic, and first consider the power control problem at the physical layer. We next extend the initial work to a joint spectrum and power allocation problem. Finally, we include varying traffic conditions in addition to varying channel conditions, and we solve the fundamental radio resource management problem using reinforcement learning. After describing how well reinforcement learning fits the fundamental radio resource management problem and showing the effectiveness of a multi-agent reinforcement learning framework for joint spectrum and power allocation under varying traffic and channel conditions, we believe that this work can be extended to additional wireless resource management tasks that involve user association and multiple-input multiple-output (MIMO) beamforming.

In this introduction, we next describe the main motivation of this thesis and describe the work done and the main contributions in Chapters 2, 3, 4, and 5.

1.1. Chapter 2: Deep Learning for Dynamic Power Allocation

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**]. Our goal in Chapter 2 is to maximize an arbitrary weighted sum-rate objective, which achieves maximum sum-rate or proportionally fair scheduling as special cases. The NP-hardness of this problem is proven in [**2**]. A number of centralized and distributed optimization techniques have been used to develop algorithms for reaching a suboptimal power allocation [**1**, **3**–**8**]. We select two state-of-the-art algorithms as benchmarks. These are the weighted minimum mean square error (WMMSE) algorithm [**3**] and an iterative algorithm based on fractional programming (FP) [**4**]. In their generic form, both algorithms require full up-to-date cross-cell channel state information (CSI).

The work in Chapter 2 is the first to apply deep reinforcement learning to power control [**9**]. Sun *et al.* [**10**] 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 [**10**]. 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.

Following [**10**]. researchers proposed various machine learning based schemes for transmit power control. For example, spatial deep learning approach in [**11**] replaces CSI measurements with geographical locations by using convolutional neural networks to extract deep representations of the channel condition from the network topology. Liang *et al.* [**12**] proposed an unsupervised learning technique to remove the supervised learning based solution's dataset requirement that runs millions of WMMSE executions on random network deployments. Further, they have shown that an unsupervised learning based strategy can outperform WMMSE sum-rate performance which would be not possible for supervised learning based approach since its training is limited to WMMSE labels. As an another example, Eisen and Ribeiro [**13**] introduced random edge graph neural networks to reduce the neural network complexity in terms of number of parameters and proposed a novel unsupervised model-free primal-dual learning algorithm to train the proposed graph neural network scheme.

Data-driven methods are promising in a realistic wireless context where varying channel conditions impose serious challenges such as imperfect or delayed CSI. Reference [**10**] uses a deep neural network to mimic an optimization algorithm that is trained by a dataset composed of many optimization runs. The main motivation in [**10**] is to reduce the computational complexity while maintaining a comparable sum-rate performance with WMMSE. However, the training dataset relies on model-based optimization algorithms. In Chapter 2, we consider a purely data-driven approach called model-free deep reinforcement learning.

In Chapter 2, we design a distributively executed algorithm to be employed by all transmitters to compute their power allocations in real time. A dynamic power allocation problem with time-varying channels for a different system model and network setup was studied in [**14**], where the delay performance of the classical dynamic backpressure algorithm was improved by exploiting the stochastic Lyapunov optimization framework. The main contributions and some advantages of the proposed scheme in Chapter 2 are summarized as follows:

(1) The proposed distributively executed algorithm is based on deep Q-learning [**15**], 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 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 [**16**]. In certain scenarios the proposed distributed algorithm even outperforms the centralized iterative algorithms introduced in [**3**, **4**]. We also address some important practical constraints that are not included in [**3**, **4**].

Deep reinforcement learning framework has been used in some other wireless communications problems [**17**–**20**]. Classical Q-learning techniques have been applied to the power allocation problem in [**21**–**25**]. The goal in [**21**, **22**] 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 [**21**] and [**22**] represent the wireless environment using a discrete state set and limit the number of learning agents. Amiri *et al.* [**23**] 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 [**24**, **25**]. Similar to the proposed approach, the work in [**24**, **25**] 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 Chapter 2, 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.

1.2. Chapter 3: Proposed Deep Reinforcement Learning Scheme Extended to Continuous Action Spaces and Mobile Devices

In Chapter 3, we replace deep Q-learning with an actor-critic method called deep deterministic policy gradient (DDPG) [**26**] algorithm that applies to continuous action spaces. Since Q-learning applies only to discrete action spaces, transmit power had to be quantized in Chapter 2. As a result, the quantizer design and the number of levels, i.e., number of possible actions, have an impact on the performance. For example, an extension of our work in Chapter 2 shows that quantizing the action space with a logarithmic step size gives better outcomes than that of a linear step size [**27**]. A distributively executed DDPG scheme has been applied to power control for fixed channel and perfect CSI [**27**]. To the best of our knowledge, we are the first to study actor-critic based dynamic power control that involves mobility of cellular devices. Our initial work in Chapter 2 assumed immobile devices where the large-scale fading component was the steady state of the channel. We adapt our previous approach to make it applicable to our new system model that involves mobility where channel conditions vary due to both small and large scale fading. In order to ensure the practicality, we assume delayed and incomplete CSI, and using simulations, we compare the sum-rate outcome with WMMSE and FP that have full perfect CSI.

1.3. Chapter 4 A Two-layer Deep Reinforcement Learning Framework for Joint Spectrum and Power Allocation

Next, in Chapter 4, we extend the reinforcement learning framework to the joint spectrum and channel allocation problem.

In today's cellular networks, the spectrum is divided into many subbands. Each cellular device suffers from the co-channel interference caused by nearby access points which use the same subbands. The interference can be particularly severe with dense and irregularly placed access points. Joint subband selection and transmit power control is a crucial tool for interference mitigation.

For the single band scenario, state-of-the-art optimization methods such as FP [**4**] have been applied to the power control problem to reach a near-optimal allocation. We assume that the number of subbands is much less than the number of cellular devices and that each link can occupy at most one subband at a time. Therefore, the joint subband selection and power allocation problem involves *mixed integer programming* [**28**].

Conventional optimization-based schemes such as fractional programming are model-driven and require a mathematically tractable and sufficiently accurate model [**29**]. Furthermore, such a scheme is in general centralized and requires instantaneous global channel state information (CSI). A centralized solution's computational complexity does not scale well for a large number of cellular devices. Therefore, its implementation is quite challenging in a practical scenario where network and channel conditions vary.

Recently, there has been extensive research on reinforcement learning based transmit power control which is purely data-driven [**29**]. For the single band scenario, deep Q-learning has been considered on a "centralized training and distributed execution" framework in [**9, 24, 30**]. Since deep Q-learning applies only to discrete power control, the continuous transmit power domain had to be quantized in [**9, 24, 30**] which may introduce a quantization error as discussed in [**31, 32**]. Reference [**31**] first showed the performance in [**9**] can be improved by quantizing the transmit power using logarithmic step size instead of linear step size, and propose replacing deep Q-learning algorithm by an actor-critic learning algorithm called deep deterministic policy gradient that applies to continuous power control.

For the multiple band scenario, Tan *et al.* [**28**] have proposed to train a single deep Q-network that jointly handles both subband selection and transmit power control. One major drawback of this approach is that the action space is the Cartesian product of available subbands and quantized transmit power levels. Therefore, the deep Q-network output layer size and the number of state action pairs to be visited for convergence during training do not scale well with increasing number of subbands. Moreover, the joint deep Q-learning approach is not directly applicable to a problem that includes both discrete and continuous variables. To overcome these challenges, we propose a novel approach that consists of two layers, where the bottom layer is responsible for continuous power allocation with deep deterministic policy gradient, and the top layer schedules discrete subbands by adapting deep Q-learning. Using simulations, we evaluate the proposed learning scheme by comparing it with the joint deep Q-learning approach and the fractional programming algorithm in terms of convergence rate and sum-rate performance.

1.4. Chapter 5: Deep Reinforcement Learning for the Fundamental Problem of Radio Resource Management

Finally, in Chapter 5, we propose a multi-agent deep reinforcement learning based resource management scheme that can respond to the changes in both traffic and channel dynamics instantaneously. In previous chapters, we assume full-buffer traffic, but we introduce a queue model in Chapter 5 and maximize users' long-term utility by minimizing the average packet delay. We formulate the fundamental problem for radio resource management. The fundamental problem resembles a Markov decision process, so the model-free reinforcement learning turns out to be the optimum tool to find optimal control policies that matches the traffic and channel conditions with resource allocations.

As a base station increases the transmit power and subband allocation of a corresponding link with heavy traffic accumulation at its queue, a quite full queue and intends to send more information at a given time to link's receiver with the goal of reducing the accumulation of packets at its queue to maintain QoS requirements, the broadcast nature of the wireless medium will cause an inadvertent interference to other nearby receivers and this may reduce the overall system performance [**12**]. Hence, there is a ubiquitous need for an intelligent resource management scheme that accordingly controls and manages the transmit power and subband allocation to enhance the overall system performance.

The conventional methods and the reinforcement learning based methods that will be introduced in previous chapters are limited to the generic weighted sum-rate maximization problem which can be particularized to achieve maximum sum-rate or proportionally fair scheduling as shown in previous chapters. As a result, the change in the objective function requires a new formulation and analysis from scratch which is not efficient and not flexible in terms of addressing the changes in network QoS and fairness requirements. In this chapter, our main goal is to describe the fundamental radio resource management problem and show a reinforcement learning based resource management scheme that can effectively solve this problem regardless of the choice of network objective.

Note that without the full-buffer assumption and for an allocation period on order of seconds, there has been a recent optimization-based study that enables centralized resource management that iteratively finds a near-optimal resource allocation scheme for a broad selection of utility functions [**33**]. The proposed centralized algorithm in [**33**] is scalable to large-scale wireless networks with an assumption on the timescale of an allocation period at the order of seconds. In this chapter, one of our goals is to design a traffic-driven radio resource management scheme on a much faster timescale, i.e. one time slot, which is typically a few-ten milliseconds.

The main contributions and some benefits of the proposed scheme in Chapter 5 are summarized as follows:

(1) We define the fundamental problem of radio resource management and introduce a novel approach that solves this problem effectively regardless of the choice of network objective.

(2) We introduce user priorities that connect physical layer resource management with network layer. Depending on the network objective, the user priorities are changed accordingly. For example, user priorities can be set to equal for all links for sum-rate maximization or they can be updated at the beginning of each time-slot to achieve proportional fairness across all users. Additionally, these user priorities can represent traffic conditions, i.e., queue lengths and traffic arrival characteristics, for the traffic-driven approach.

(3) In previous chapters, we have studied a weighted sum-rate maximization problem with full-buffer traffic. In this chapter, we introduce a traffic and queue model which makes the underlying Markov decision process in the resource management problem more intuitive. As a result, deep reinforcement learning scheme can now

make a better sense of the discount factor to maximize a long-term objective to stabilize queues and minimize average packet delay.

(4) In this chapter, we even further limit the earlier assumptions on CSI measurements and use aggregated interference power measurements instead of individual interfering channel gains. This framework can work with aggregate interference measurements in lieu of individual cross-channel gains. This improves the practicality of the solution in contrast to Chapter 2.

(5) The proposed approach has a decentralized execution framework and its complexity does not depend on the wireless network size. Therefore, it is feasible to run the proposed approach on a few-ten millisecond timescale.

After Chapter 5, we conclude in Chapter 6 and discuss several ideas on how the overall work in this thesis can be extended.

CHAPTER 2

Deep Learning for Dynamic Power Allocation

2.1. Introduction

The work in this chapter 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 chapter, 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 work in this chapter indicates that deep reinforcement learning based radio resource management can deliver highly competitive performance in a timely manner. The proposed scheme is especially suitable for practical scenarios where the system model is inaccurate and CSI delay is non-negligible.

The remainder of this chapter is organized as follows. We give the system model in Section 2.2. In Section 2.3, we formulate the dynamic power allocation problem and give our practical constraints on the local information. In Section 2.4, we give an overview of deep Q-learning. We continue with an overview of multi-agent reinforcement learning in Section 2.5. In Section 2.6, we describe the proposed algorithm. Simulation results are given in Section 2.7. We finally conclude this chapter in Section 2.8.

2.2. 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) [**6**]. The model has also been used to describe a simple cellular network with n APs, where each AP serves a single user device [**4**, **5**]. 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

$$g_{i\to j}^{(t)} = \left|h_{i\to j}^{(t)}\right|^2 \alpha_{i\to j}, \quad t = 1, 2, \ldots. \tag{2.1}$$

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 [**16**], we express the small-scale Rayleigh fading component as a first-order complex

Gauss-Markov process:

$$h_{i\to j}^{(t)} = \rho h_{i\to j}^{(t-1)} + \sqrt{1-\rho^2} e_{i\to j}^{(t)} \tag{2.2}$$

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 $\boldsymbol{p} = [p_1, \ldots, p_n]^\mathsf{T}$:

$$\gamma_i^{(t)}(\boldsymbol{p}) = \frac{g_{i\to i}^{(t)} p_i}{\sum_{j\neq i} g_{j\to i}^{(t)} p_j + \sigma^2} \tag{2.3}$$

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:

$$C_i^{(t)}(\boldsymbol{p}) = \log\left(1 + \gamma_i^{(t)}(\boldsymbol{p})\right). \tag{2.4}$$

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

2.3. 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

$$\begin{aligned} &\underset{\boldsymbol{p}}{\text{maximize}} \quad \sum_{i=1}^{n} w_i^{(t)} \cdot C_i^{(t)}(\boldsymbol{p}) \\ &\text{subject to} \quad 0 \le p_i \le P_{\max}, \quad i = 1, \ldots, n, \end{aligned} \tag{2.5}$$

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 (2.5) at the beginning of every time slot. In time slot t, the optimal power allocation solution is denoted as $\boldsymbol{p}^{(t)}$. Problem (2.5) is in general non-convex and has been shown to be NP-hard [**2**].

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 [**8**, **34**]. Specifically, at the end of time slot t, receiver i computes its weighted average spectral efficiency as

$$\bar{C}_i^{(t)} = \beta \cdot C_i^{(t)}\left(\boldsymbol{p}^{(t)}\right) + (1-\beta)\bar{C}_i^{(t-1)} \tag{2.6}$$

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

$$w_i^{(t+1)} = \left(\bar{C}_i^{(t)}\right)^{-1}. \tag{2.7}$$

This power allocation algorithm maximizes the sum of log-average spectral efficiency [**34**], i.e.,

$$\sum_{i \in N} \log \bar{C}_i^{(t)}, \tag{2.8}$$

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 [**3**] and the FP algorithm [**4**]. Both are centralized and iterative in their original form. The closed-form FP algorithm used in this chapter is formulated in [**4**, Algorithm 3]. Similarly, a detailed explanation and pseudo code of the WMMSE algorithm is given in [**10**, 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

$$I_i^{(t)} = \left\{ j \in N, j \neq i \middle| g_{j\to i}^{(t-1)} p_j^{(t-1)} > \eta\sigma^2 \right\}. \tag{2.9}$$

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

$$O_i^{(t)} = \left\{ k \in N, k \neq i \middle| g_{i\to j}^{(t-1)} p_i^{(t-1)} > \eta\sigma^2 \right\}. \tag{2.10}$$

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

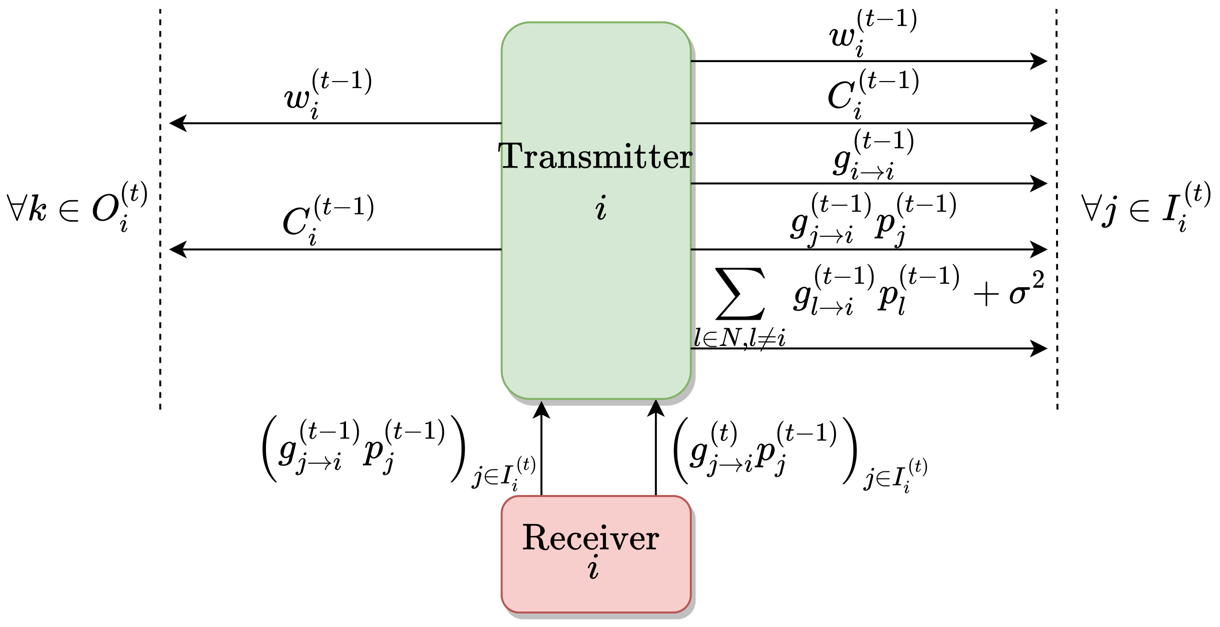

$w_i^{(t-1)}$
$C_i^{(t-1)}$
$\forall k \in O_i^{(t)}$
Transmitter i
$w_i^{(t-1)}$
$C_i^{(t-1)}$
$g_{i\to i}^{(t-1)}$
$g_{j\to i}^{(t-1)} p_j^{(t-1)}$
$\sum_{l\in N, l\neq i} g_{l\to i}^{(t-1)} p_l^{(t-1)} + \sigma^2$
$\forall j \in I_i^{(t)}$
$\left(g_{j\to i}^{(t-1)} p_j^{(t-1)}\right)_{j\in I_i^{(t)}}$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

Figure 2.1. 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.

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. 2.1, transmitter i has sent the following outdated information to interferer $j \in I_i^{(t)}$ in return for $w_j^{(t-1)}$ and $C_j^{(t-1)}$:

- the weight of link i, $w_i^{(t-1)}$,
- the spectral efficiency of link i computed from (2.4), $C_i^{(t-1)}$,
- the direct gain, $g_{i\to i}^{(t-1)}$,
- the received interference power from transmitter j, $g_{j\to i}^{(t-1)} p_j^{(t-1)}$,
- the total interference-plus-noise power at receiver i, i.e., $\sum_{l\in N, l\neq i} g_{l\to i}^{(t-1)} p_l^{(t-1)} + \sigma^2$.

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.

2.4. 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 [**35**, **36**]. 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 [**24**]. 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 [**15**].

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:

$$R^{(t)} = \sum_{\tau=0}^{\infty} \gamma^{\tau} r^{(t+\tau+1)} \tag{2.11}$$

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 [**37**], i.e.,

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[R^{(t)} \middle| s^{(t)} = s, a^{(t)} = a\right]. \tag{2.12}$$

As an action value function, the Q-function satisfies a Bellman equation [**38**]:

$$Q^{\pi}(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in S} \mathcal{P}^{a}_{ss'} \left(\sum_{a' \in A} \pi(s', a') Q^{\pi}(s', a') \right) \tag{2.13}$$

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 (2.13), 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 (2.12) [**37**]. Clearly, it suffices to let $\pi^*(s, a)$ be equal to 1 for the most favorable action. From (2.13), the optimal Q-function associated with the optimal policy is then expressed as

$$Q^*(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in S} \mathcal{P}^{a}_{ss'} \max_{a'} Q^*(s', a'). \tag{2.14}$$

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 [**15**]. 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:

$$\begin{aligned} q\left(s^{(t)}, a^{(t)}\right) \leftarrow{} & (1-\alpha) q\left(s^{(t)}, a^{(t)}\right) \\ & + \alpha \left(r^{(t+1)} + \gamma \max_{a'} q\left(s^{(t+1)}, a'\right)\right) \end{aligned} \tag{2.15}$$

where $\alpha \in (0, 1]$ is the learning rate [**37**].

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 [**39**].

Both issues can be solved with deep reinforcement learning, e.g., deep Q-learning [**15**]. 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 $q(s, a, \boldsymbol{\psi})$, where the real-valued vector $\boldsymbol{\psi}$ represents its parameters. The essence of DQN is that the function $q(\cdot, \cdot, \boldsymbol{\psi})$ is completely determined by $\boldsymbol{\psi}$. As such, the task of finding the best Q-function in a functional space of uncountably many dimensions is reduced to searching the best $\boldsymbol{\psi}$ 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 [**15**] implies, we define two DQNs: the target DQN with parameters $\boldsymbol{\psi}^{(t)}_{\text{target}}$ and the train DQN with parameters $\boldsymbol{\psi}^{(t)}_{\text{train}}$. $\boldsymbol{\psi}^{(t)}_{\text{target}}$ is updated to be equal to $\boldsymbol{\psi}^{(t)}_{\text{train}}$ once every T_u steps. From the "experience replay" [**15**], the least squares loss of train DQN for a random mini-batch $D^{(t)}$ at time t is

$$L\left(\boldsymbol{\psi}^{(t)}_{\text{train}}\right) = \sum_{(s,a,r',s') \in D^{(t)}} \left(y^{(t)}_{DQN}(r', s') - q\left(s, a; \boldsymbol{\psi}^{(t)}_{\text{train}}\right)\right)^2 \tag{2.16}$$

where the target is

$$y^{(t)}_{DQN}(r', s') = r' + \lambda \max_{a'} q\left(s', a'; \boldsymbol{\psi}^{(t)}_{\text{target}}\right). \tag{2.17}$$

Finally, we assume that each time step the stochastic gradient descent algorithm that minimizes (2.16) is used to train $\boldsymbol{\psi}^{(t)}_{\text{train}}$ 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 [**40**].

2.5. Overview of Multi-Agent Reinforcement Learning

In this section, we describe the multi-agent learning basics that enables distributively solving the resource management problem by training multiple agents collaboratively and executing control policies separately at the base stations. We design a distributively executable algorithm, so each link is associated with a learning agent that assigns a transmit power level to it. These agents work collaboratively to maximize the global network objective. In a multi-agent learning system, two or more learning agents reside in a common environment and the state transitions of the environment mainly depend on the joint actions of these learning agents. From a single agent point of view, the environment transition probability is no longer stationary as other learning agents in the system update their policies/behaviors simultaneously. This issue is called "environment non-stationarity" in the multi-agent reinforcement learning literature [**41**].

For the multi-agent case, similar to [**42**], we define the local state of agent i as $s_i \in S_i$. The local state is composed of environment features that are relevant to agent i's action $a_i \in A_i$. Intuitively, for the power allocation setting, the local state is composed of local CSI measurements and feedback from interfering/interfered neighbors. The multi-agent system composed of n agents is a stochastic game defined as a tuple $\langle S, A, P, \boldsymbol{R}, \gamma \rangle$, where $S = \Pi_i S_i$ is the joint state space, $A = \Pi_i A_i$ is the joint action space, $P : S \times A \times S \rightarrow [0, 1]$ is the state transition function, $\boldsymbol{r} = (r_1, \ldots, r_n)$ is the vector of reward functions of agents [**42**]. For this case, the reward function of agent i, $r_i : S \times A \times S$, also depends on the actions of other agents unlike the single-agent case. For fully-cooperative games, the agents share a common reward function $r_1 = \ldots r_n$ and they receive a single joint reward as a result of their joint actions.

An extreme approach that tackles the environment non-stationary issue of multi-agent reinforcement learning is modeling the system as a single meta-agent that outputs joint actions by observing the complete environment state [**43**]. A well-known example for this approach is team Q-learning [**44**,**45**]. For fully-cooperative games, a convergence guaranty is given for team Q-learning in [**45**, Theorem 6]. Although the single meta-agent approach solves the problem of non-stationarity, it has some defects too. Since team learning learns the joint-actions, the complexity of state-action space explodes with the number of agents in the system which affects the convergence rate to an optimal policy [**46**]. For a possible team deep Q-learning application, the number of DQN parameters will also be proportional to the number of agents to handle the additional complexity of the state-action set. Another problem is that the distributed execution of team learning is not feasible, since it requires the global state information and assigns the joint actions in a centralized manner.

Another straight-forward approach to learn in a multi-agent system is called concurrent learning. In concurrent learning, n independent learning agents interact with the environment by taking simultaneous actions and they learn independent policies by treating the actions of other agents as part of the environment. For fully-cooperative games, independent Q-learning is a concurrent learning scheme, in which n Q-learning agents learn independent policies by using the shared team reward that they receive after their joint actions [**47**]. Although independent Q-learning is scalable and allows distributed execution, its stability and convergence are questioned because of the environment non-stationarity issue. Still, independent-Q learning gives good empirical performance for various cooperative game applications [**48**,**49**]. Recently, Tampuu *et al.* [**50**] showed that independent deep Q-learning is promising by successfully using multiple DQN agents for the *Pong* game.

There is an extensive research to develop multi-agent learning frameworks and there exists several multi-agent Q-learning adaptations which lie between team and concurrent learning which are the two extreme approaches mentioned above [**41**, **49**–**51**]. However, multi-agent learning is an open research area and theoretical guarantees for these adaptations are rare and incomplete despite their good empirical performances [**50**, **51**]. In this thesis, we mainly focus on an adaptation to independent deep Q-learning that focuses on increasing its stability and convergence rate. Calabrese [**25**] proposed a centralized learning and distributed execution framework for power allocation in wireless networks. They employ the transfer learning and parameter sharing concepts that increase the stability and convergence rate of independent Q-learning by taking advantage of the fact that agents are learning together [**52**].

2.6. Deep Reinforcement Learning for Dynamic Power Allocation

2.6.1. Proposed Multi-Agent Deep Learning Algorithm

As depicted in Fig. 2.2, we propose a multi-agent deep reinforcement learning scheme with each transmitter as an agent. Similar to [**42**], 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 $a_i \in A_i$. The agents in Fig. 2.2 work collaboratively to maximize (2.5) by creating a balance between their assigned link's spectral efficiency and interference to nearby links. 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 (2.13) may not be stationary as other learning agents update their policies. The Markov property introduced for the single-agent case in Section 2.4 no longer holds in general [**51**]. 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 [**43**]. 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 [**50, 51**]. However, multi-agent learning is an open research area and theoretical guarantees for these adaptations are rare and incomplete despite their good empirical performances [**50, 51**].

In this chapter, 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 $Q_{\text{target}}^{(t)}$ 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 [**52**]. Since agents are working collaboratively to maximize the global objective in (2.5) with an appropriate reward function design to be discussed in Section 2.6.4, 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. 2.2, 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 [**53**, Chapter 5] that consists of five layers as shown in Fig. 2.3a. 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 Section 2.6.2. The input layer is followed by three hidden layers with N_1, N_2, and N_3 neurons, respectively. 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 Section 2.6.3. The agent finds the action that has the maximum value at the DQN output and takes this action as its transmit power.

In Fig. 2.3a, 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

$$|\psi| = \sum_{l=0}^{3} \left(N_l + 1\right) N_{l+1}. \tag{2.18}$$

In addition, Fig. 2.3b describes the functionality of a single neuron which applies a non-linear activation function to its combinatorial input.

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. During the training stage, in each time slot, the trainer

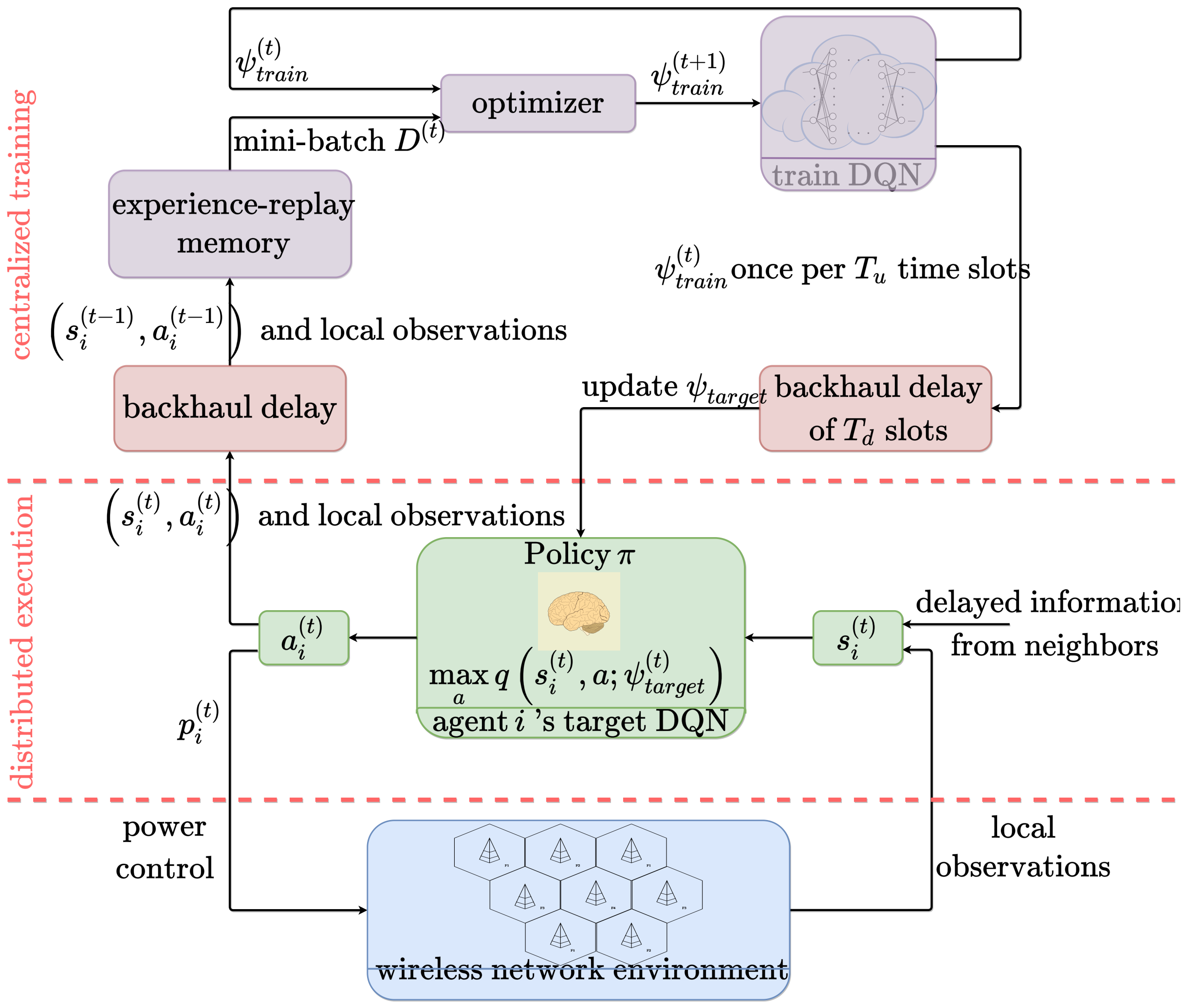


Figure 2.2. Illustration of the proposed multi-agent deep reinforcement learning algorithm.

randomly selects a mini-batch $D^{(t)}$ of M_b experiences from an experience-replay memory [**15**] that stores the experiences of all agents. The experience-replay memory is a FIFO queue [**19**] 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 (2.16) using an appropriate optimizer, e.g., the stochastic gradient descent method [**40**]. As also explained in Fig. 2.2, 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.

2.6.2. States

As described in Section 2.3, agent i builds its state $s_i^{(t)}$ using information from the interferer and interfered sets given by (2.9) and (2.10), 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:

2.6.2.1. 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 (2.5): $1/w_i^{(t)}$ 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 (2.7). 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 [**19**]. 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.

2.6.2.2. 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 (2.5). 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 (2.5). 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.

2.6.2.3. 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 (2.5). 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 (2.5) 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.

2.6.3. Actions

Unlike taking discrete steps on the previous transmit power level (see, e.g., [**24**]), 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

$$A = \left\{ 0, \frac{P_{\max}}{|A| - 1}, \frac{2P_{\max}}{|A| - 1}, \dots, P_{\max} \right\}. \tag{2.19}$$

The total number of DQN output ports denoted as N_4 in Fig. 2.3a 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 [**24**], 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 (2.2) 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.

2.6.4. Reward Function

The reward function is designed to optimize the network objective (2.5). 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

$$C_{k\backslash i}^{(t)} = \log\left(1 + \frac{g_{k\to k}^{(t)} p_k^{(t)}}{\sum_{j\neq i,k} g_{j\to k}^{(t)} p_j^{(t)} + \sigma^2}\right). \tag{2.20}$$

The network trainer computes the term $\sum_{j\neq i,k} g_{j\to k}^{(t)} p_j^{(t)} + \sigma^2$ in (2.20) 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 Section 2.3, 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 [**6**]:

$$\pi_{i\to k}^{(t)} = w_k^{(t)} \left(C_{k\backslash i}^{(t)} - C_k^{(t)}\right). \tag{2.21}$$

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

$$r_i^{(t+1)} = w_i^{(t)} C_i^{(t)} - \sum_{k\in O_k^{(t+1)}} \pi_{i\to k}^{(t)}. \tag{2.22}$$

The reward of agent i consists of two main components: its direct contribution to the network objective (2.5) 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.

2.7. Simulation Results

2.7.1. 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 Section 2.7.2. 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. 2.4 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 [**54**]. 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 (2.9) and (2.10) to 5. We assume full-buffer traffic model. Similar to [**55**], if the received SINR is greater than 30 dB, it is capped at 30 dB in the calculation of spectral efficiency by (2.4). 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. Note that we also show a situation with varying f_d. We consider five fast fading scenarios (FFS) with varying Doppler frequency, f_d. First four FFSs are can be stated that all agents have same f_d equal to 2 Hz and all agents have same f_d equal to 10 Hz called FFS 1 and FFS 2, respectively. For these first two scenarios, f_d remains unchanged. On the other hand, for the third case we consider a more flexible f_d setup compared to the first two scenarios. For the third case, at each time slot (step) t, f_d of each agent changes independently and takes a new value between 2 Hz and 10 Hz by following a Uniform distribution. Note that, the most challenging scenario is FFS 2. We picked the time slot period, T, to be 20 ms. For the FFS 1 case, the correlation term, ρ, from the Jakes fading model given in Equation (2.2) is about 0.984, whereas it is about 0.643 for the FFS 2 case. Hence, the channel conditions change more rapidly for the FFS 2 case.

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. In fact, we use a much smaller network compared to the central applications of machine learning in wireless networks. 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 Section 2.6.2. The cardinality constraint on the neighbor sets c is 5 agents. Hence, again from Section 2.6.2, 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 [**56**]. 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 [**57**] 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 $\alpha^{(t)}$ [**58**]. 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 $\epsilon^{(t+1)} = \max\left\{\epsilon_{\min}, (1-\lambda_\epsilon)\epsilon^{(t)}\right\}$, where $\epsilon_{\min} = 10^{-2}$ 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 [**58**], 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 (2.5). Further, the training cycle duration T_u is 100 time slots. After we set the parameters in (2.18), we can compute the total number of DQN parameters, i.e., $|\psi|$, 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 Section 2.6.1. 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 [**56**]. 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.

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 [**10**]. 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.

2.7.2. 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.

2.7.2.1. 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. Intuitively, since the unmatched DQN is a matched DQN of a different network initialization, it will perform worse than the matched DQN of the given initialization. The unmatched DQN approach is a handy tool to evaluate the robustness of our DQN with respect to network topology and channel variations.

In Table 2.1, 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

Table 2.1. 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

Table 2.2. 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

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. 2.5a).

Additional simulations with r and f_d taken as variables are summarized in Table 2.2 and Table 2.3, 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

Table 2.3. 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 $f_d \to \infty$ 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

remains unchanged, but the convergence time drops to 15,000 time slots. As $f_d \to \infty$, 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.

2.7.2.2. 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 2.4, 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

Table 2.4. 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

Table 2.5. 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

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.

Table 2.6. 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

2.7.2.3. 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 cell centers. We also assume that no successive interference cancellation is performed [**10**]. 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. 2.4b. The testing stage results for these multi-link per cell scenarios are given in Table 2.5. As shown in Table 2.6, we further test these scenarios using a different channel model called urban micro-cell (UMi) street canyon model of [**59**]. 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. 2.6a shows the convergence rate of training stage for 4 links per cell scenario with 76 links. In Fig. 2.6a, we also show that using a different channel model, i.e., UMi street canyon, does not

Table 2.7. 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 ($\ln(\text{bps})$)						
	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

Table 2.8. 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 ($\ln(\text{bps})$)						
	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

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.

2.7.3. Proportionally Fair Scheduling

In this subsection, we change the link weights according to (2.7) to ensure fairness as described in Section 2.3. We choose the β term in (2.6) to be 0.01 and use convergence to the objective in (2.8) as performance-metric of the DQN. Since the link weights are changing a lot, we found that keeping the agents more updated during the training stage by decreasing T_u from 100 to 50 time slots is necessary to get good performance for this section. 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 (2.4). 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.

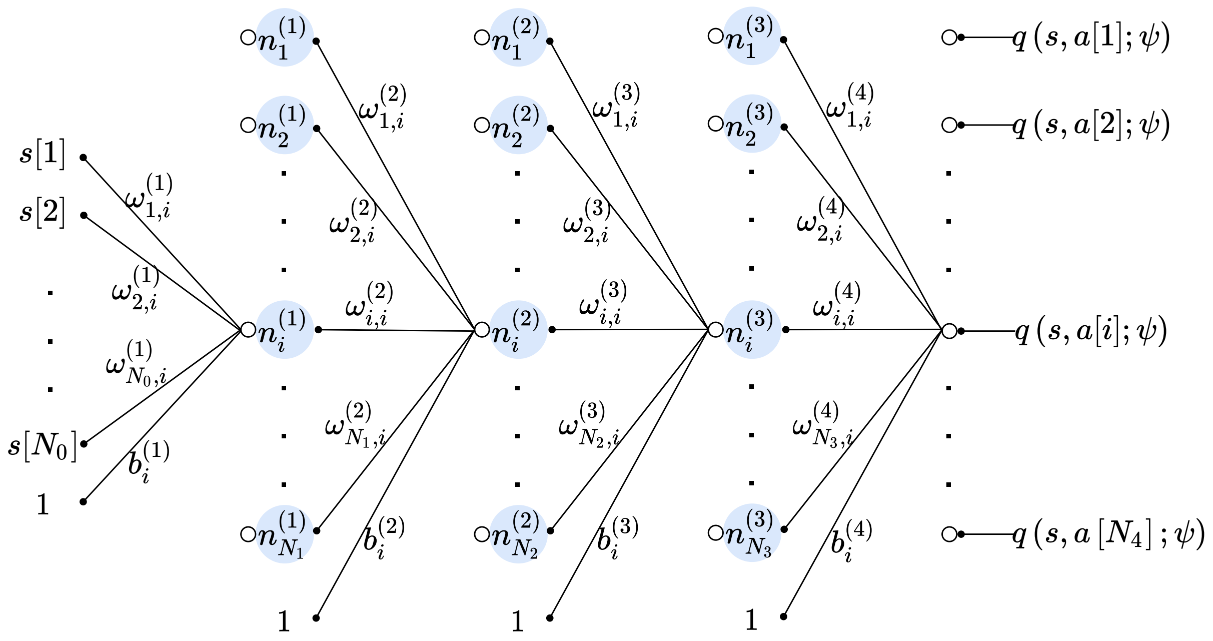
As shown in Fig. 2.7, the training stage converges to a desirable scheduling in about 30,000 time slots. Once the network is trained, as we reinitialize the link weights, our algorithm converges to an optimal scheduling in a distributed fashion as fast as the centralized algorithms. Next, we set R and r as variables to get results in Table 2.7 and Table 2.8. 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.

2.8. Conclusion

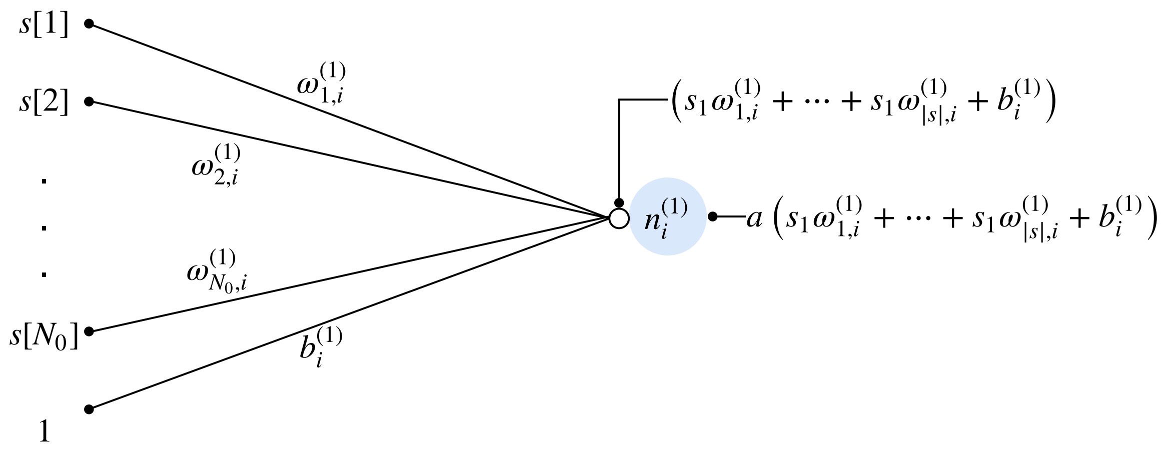
In this chapter, we have proposed a distributively executed model-free power allocation algorithm which outperforms or achieves comparable performance with existing state-of-the-art centralized algorithms. We see potentials in applying the reinforcement learning techniques on various dynamic wireless network resource management tasks in place of the optimization techniques. The proposed approach returns the new suboptimal power allocation much quicker than two of the popular centralized algorithms taken as the benchmarks in this chapter. In addition, by using the limited local CSI and some realistic practical constraints, our deep Q-learning approach usually outperforms the generic WMMSE and FP algorithms which requires the full CSI. Differently from most advanced optimization based power control algorithms, e.g., WMMSE and FP, that require both instant and accurate measurements of individual channel gains, our algorithm only requires accurate measurements of some delayed received power values that are higher than a certain threshold above noise level. An extension to an imperfect CSI case with inaccurate CSI measurements is left for future work. In addition, recently, Cui *et al.* [**11**] proposed an unsupervised learning based power control algorithm that gets geographical locations processed by a spatial convolutional filter as its input. A similar approach can also be applied to our reinforcement learning based framework to remove the dependency on CSI measurements.

Meng *et al.* [**30**] is an extension of [**9**] to multiple users in a cell, which is also addressed in the current chapter. 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, we used global training in this chapter, whereas re-initializing a local training over the regions where new links joined or performance dropped under a certain threshold is also an interesting direction to consider. Besides the regional training, completely distributed training can be considered, too. 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.



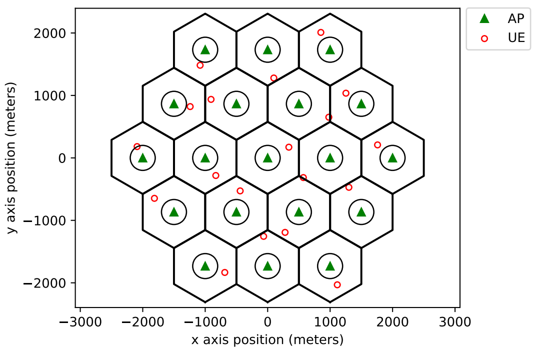
(a) The illustration of all five layers of the proposed DQN: The input layer is followed by three hidden layers and an output layer. The notation n, ω and b indicate DQN neurons, weights, and biases, respectively. These weights and biases form the set of DQN parameters denoted as ψ. The biases are not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



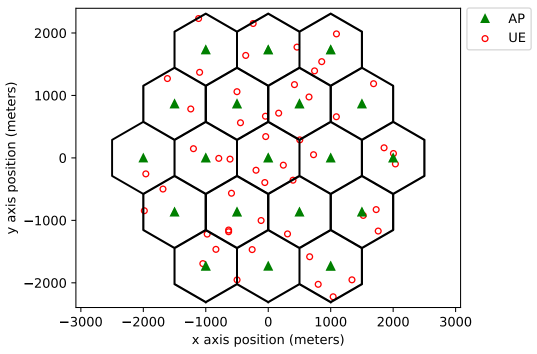
$s[1]$
$s[2]$
$s[N_0]$
1
$\omega^{(1)}_{1,i}$
$\omega^{(1)}_{2,i}$
$\omega^{(1)}_{N_0,i}$
$b^{(1)}_i$
$n^{(1)}_i$
$\left(s_1\omega^{(1)}_{1,i} + \cdots + s_1\omega^{(1)}_{|s|,i} + b^{(1)}_i\right)$
$a\left(s_1\omega^{(1)}_{1,i} + \cdots + s_1\omega^{(1)}_{|s|,i} + b^{(1)}_i\right)$

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

Figure 2.3. The overall design of the proposed DQN.

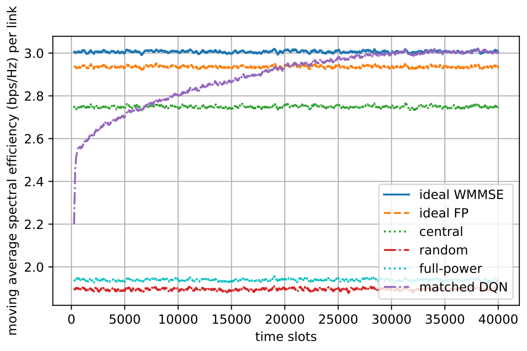


(a) Single-link per cell with $R = 500$ m and $r = 200$ m.

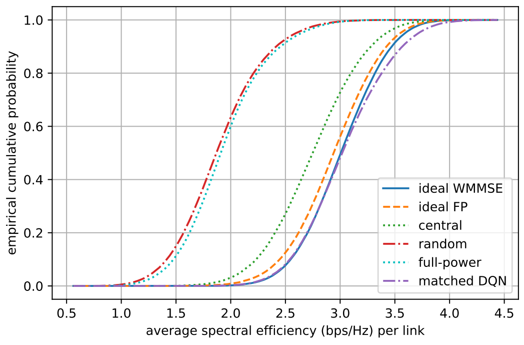


(b) Multi-link per cell with $R = 500$ m and $r = 10$ m. Each cell has a random number of links from 1 to 4 links per cell.

Figure 2.4. Network configuration examples with 19 cells

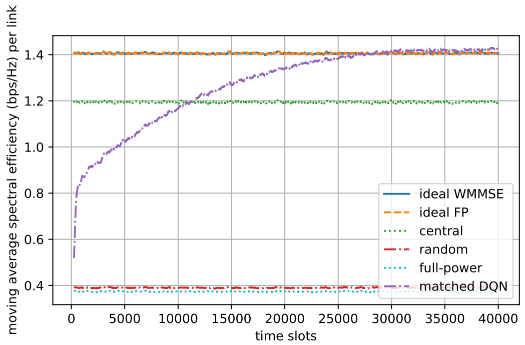


(a) Training - Moving average spectral efficiency per link of previous 250 slots.

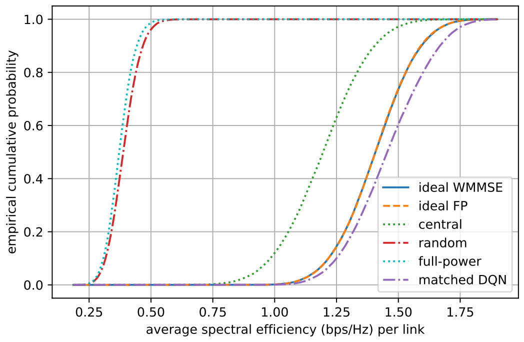


(b) Testing - Empirical CDF.

Figure 2.5. Sum-rate maximization. $n = 19$ links, $R = 100$ m, $r = 10$ m, $f_d = 10$ Hz.

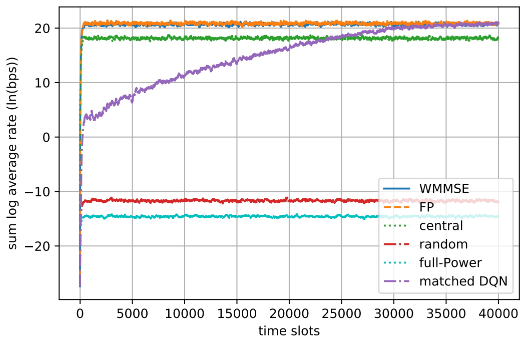


(a) Training - Moving average spectral efficiency per link of previous 250 slots.

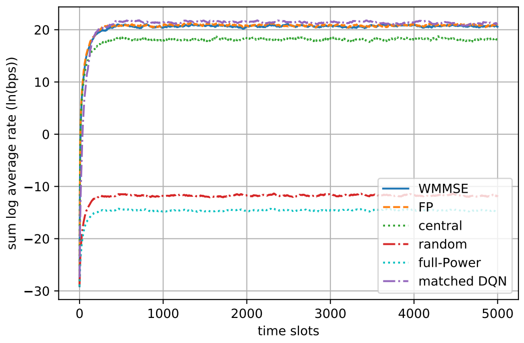


(b) Testing - Empirical CDF.

Figure 2.6. 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.



(a) Training.



(b) Testing.

Figure 2.7. Proportionally fair scheduling. $n = 19$ links, $R = 500$ m, $r = 10$ m, $f_d = 10$ Hz.

CHAPTER 3

Proposed Deep Reinforcement Learning Scheme Extended to Continuous Action Spaces and Mobile Devices

3.1. Introduction

In this chapter, we present a distributively executed continuous power control algorithm with the help of deep actor-critic learning, and more specifically, by adapting deep deterministic policy gradient. Furthermore, we integrate the proposed power control algorithm to a time-slotted system where devices are mobile and channel conditions change rapidly. We demonstrate the functionality of the proposed algorithm using simulation results and we compare the sum-rate with WMMSE and FP that have full perfect CSI.

The remainder of this chapter is organized as follows. We describe the new system model that involves user mobility and formulate the problem in Section 3.2. We next give an overview of deep actor-critic learning in Section 3.3. This is followed by proposed DDPG algorithm, simulations, and conclusion in Sections 3.4, 3.5, and 3.6, respectively.

3.2. System Model with Mobility and Problem Formulation

In this chapter, we consider a special case where N *mobile* devices are uniformly randomly placed in K homogeneous hexagonal cells. This deployment scenario is similar to the interfering multiaccess channel scenario which is also examined in [**10**, **60**]. Let $\mathcal{N} = \{1, \ldots, N\}$ and $\mathcal{K} = \{1, \ldots, K\}$ denote the sets of link and cell indexes, respectively.

Here we are not concerned with the device association problem. As device $n \in \mathcal{N}$ is inside cell $k \in \mathcal{K}$, its associated AP n is located at the center of cell k. We denote the cell association of device n as $b_n \in \mathcal{K}$ and its AP n is positioned at the center of b_n.

All transmitters and receivers use a single antenna and we consider a single frequency band with flat fading. The network is assumed to be a fully synchronized time slotted system with slot duration T. We employ a block fading model to denote the downlink channel gain from a transmitter located at the center of cell k to the receiver antenna of device n in time slot t as

$$\bar{g}_{k\to n}^{(t)} = \left|h_{k\to n}^{(t)}\right|^2 \alpha_{k\to n}^{(t)}, \quad t = 1, 2, \dots . \tag{3.1}$$

In (3.1), $\alpha_{k\to n}^{(t)} \geq 0$ represents the large-scale fading component including path loss and log-normal shadowing which varies as mobile device j changes its position. Let $\mathbf{x}_k$ denote the 2D position, i.e., (x, y)-coordinates, of cell k's center. Similarly, we represent the location of mobile device n at slot t as $\mathbf{x}_n^{(t)}$. Then, the large-scale fading can be expressed in dB as

$$\alpha_{\mathrm{dB},k\to n}^{(t)} = \mathrm{PL}\left(\mathbf{x}_k, \mathbf{x}_n^{(t)}\right) + \mathcal{X}_{k\to n}^{(t)}, \tag{3.2}$$

where PL is the distance-dependent path loss in dB and $\mathcal{X}_{k\to n}^{(t)}$ is the log-shadowing from $\mathbf{x}_k$ to $\mathbf{x}_n^{(t)}$. For each device n, we compute the shadowing from all k possible AP positions in the network. The shadowing parameter is updated by $\mathcal{X}_{k\to n}^{(t)} = \rho_{\mathrm{s},n}^{(t)} \mathcal{X}_{k\to n}^{(t)} + \sigma_{\mathrm{s}} e_{\mathrm{s},k\to n}^{(t)}$, where σ_{s} is the log-normal shadowing standard deviation and the correlation $\rho_{\mathrm{s},n}^{(t)}$ is computed by $\rho_{\mathrm{s},n}^{(t)} = e^{\frac{\Delta \mathbf{x}_n^{(t)}}{d_{\mathrm{cor}}}}$ with $\Delta \mathbf{x}_n^{(t)} = \left\|\mathbf{x}_n^{(t)} - \mathbf{x}_n^{(t-1)}\right\|_2$ being the displacement of device n during the last slot and with d_{cor} being the correlation length of the environment. Note that $\mathcal{X}_{k\to n}^{(0)} \sim \mathcal{N}\left(0, \sigma_s^2\right)$ and the shadowing innovation process $e_{\mathrm{s},k\to n}^{(1)}, e_{\mathrm{s},k\to n}^{(2)}, \ldots$ consists of independent and identically distributed (i.i.d.) Gaussian variables with distribution $\mathcal{N}\left(0, 1-\left(\rho_{\mathrm{s},n}^{(t)}\right)^2\right)$. Following [**61**], we model the change in the movement behavior of each device as incremental steps on their speed and directions.

We use Jakes fading model to model the small-scale Rayleigh fading component as a first-order complex Gauss-Markov process as explained in (2.2). Unlike Chapter 2, since users are mobile in this chapter, the correlation $\rho_n^{(t)}$ is now computed as $\rho = J_0(2\pi f_{d,n}^{(t)} T)$, where $J_0(.)$ is the zeroth-order Bessel function of the first kind and $f_{d,n}^{(t)} = v_n^{(t)} f_c/c$ is device n's maximum Doppler frequency at slot t with $v_n^{(t)} = \Delta \mathbf{x}_n^{(t)}/T$ being device n's speed, $c = 3 \times 10^8$ m/s, and f_c being carrier frequency.

Let $b_n^{(t)}$ and $p_n^{(t)}$ denote device n's associated cell and transmit power of its associated AP in time slot t, respectively. Hence the association and allocation in time slot t can be denoted as $\boldsymbol{b}^{(t)} = \left[b_1^{(t)}, \ldots, b_N^{(t)}\right]^\intercal$ and $\boldsymbol{p}^{(t)} = \left[p_1^{(t)}, \ldots, p_N^{(t)}\right]^\intercal$, respectively. The signal-to-interference-plus-noise ratio at receiver n in time slot t can be defined as a function of the association $\boldsymbol{b}^{(t)}$ and allocation $\boldsymbol{p}^{(t)}$:

$$\gamma_n^{(t)}\left(\boldsymbol{b}^{(t)}, \boldsymbol{p}^{(t)}\right) = \frac{\bar{g}_{b_n^{(t)}\to n}^{(t)} p_n^{(t)}}{\sum_{m\neq n} \bar{g}_{b_m^{(t)}\to n}^{(t)} p_m^{(t)} + \sigma^2}, \tag{3.3}$$

where σ^2 is the additive white Gaussian noise power spectral density which is assumed to be the same at all receivers without loss of generality. Then, the downlink spectral efficiency of device n at time t is

$$C_n^{(t)} = \log\left(1 + \gamma_n^{(t)}\left(\boldsymbol{b}^{(t)}, \boldsymbol{p}^{(t)}\right)\right). \tag{3.4}$$

For a given association $\boldsymbol{b}^{(t)}$, the power control problem at time slot t can be defined as a sum-rate maximization problem:

$$\begin{aligned} &\underset{\boldsymbol{p}^{(t)}}{\text{maximize}} \quad \sum_{n=1}^{N} C_n^{(t)} \\ &\text{subject to} \quad 0 \leq p_n \leq P_{\max}, \quad n = 1, \ldots, N \,, \end{aligned} \tag{3.5}$$

where $P_{\max}$ is the maximum power spectral density that an AP can emit. The real-time allocator solves the problem in (3.5) at the beginning of slot t and its solution becomes $\boldsymbol{p}^{(t)}$. For ease of notation, throughout the chapter, we use $g_{m\to n}^{(t)} = \bar{g}_{b_m^{(t)}\to n}^{(t)}$.

3.3. Deep Actor-Critic Learning Overview

A learning agent intersects with its environment, i.e., where it lives, in a sequence of discrete time steps. At each step t, agent first observes the state of environment, i.e., key relevant environment features, $s^{(t)} \in \mathcal{S}$ with $\mathcal{S}$ being the set of possible states. Then, it picks an action $a^{(t)} \in \mathcal{A}$, where $\mathcal{A}$ is a set of actions, following a policy that is either deterministic or stochastic and is denoted by μ with $a^{(t)} = \mu(s^{(t)})$ or π with $a^{(t)} \sim \pi(\cdot|s^{(t)})$, respectively. As a result of this interaction, environment moves to a new state $s^{(t+1)}$ following a transition probability matrix that maps state-action pairs onto a distribution of states at the next step. Agent perceives how good or bad taking action a^t at state $s^{(t)}$ is by a reward signal $r^{(t+1)}$. We describe the above interaction as an experience at $t+1$ denoted as $e^{(t+1)} = \big(s^{(t)}, a^{(t)}, r^{(t+1)}, s^{(t+1)}\big)$.

Model-free reinforcement learning learns directly from these interactions without any information on the transition dynamics and aims to learn a policy that maximizes agent's long-term accumulative discounted reward at time t,

$$R^{(t)} = \sum_{\tau=0}^{\infty} \gamma^{\tau} r^{(t+\tau+1)}, \tag{3.6}$$

where $\gamma \in (0, 1]$ is the discount factor.

Two main approaches to train agents with model-free reinforcement learning are value function and policy search based methods [**62**]. The well-known Q-learning algorithm is value based and learns an action-value function $Q(s, a)$. The classical Q-learning uses a lookup table to represent Q-function which does not scale well for large state spaces, i.e., a high number of environment features or some continuous environment features. Deep Q-learning overcomes this challenge by employing a deep neural network to represent Q-function in place of a lookup table. However, the action space still remains discrete which requires quantization of transmit power levels in a power control problem. Policy search methods can directly handle continuous action spaces. In addition, compared to Q-learning that indirectly optimize agent's performance by learning a value function, policy search methods directly optimize a policy which is often more stable and reliable [**63**]. By contrast, the policy search algorithms are typically on-policy which means each policy iteration only uses data that is collected by the most-recent policy. Q-learning can reuse data collected at any point during training, and consequently, more sample efficient. Another specific advantage of off-policy learning for a wireless network application is that the agents do not need to wait for the most-recent policy update and can simultaneously collect samples while the new policy is being trained. Since both value and policy based approaches have their strengths and drawbacks, there is also a hybrid approach called actor-critic learning [**62**].

Reference [**26**] proposed the DDPG algorithm which is based on the actor-critic architecture and allows continuous action spaces. DDPG algorithm iteratively trains an action-value function using a critic network and uses this function estimate to train a deterministic policy parameterized by an actor network.

For a policy π, the Q-function at state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ becomes

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[R^{(t)} \middle| s^{(t)} = s, a^{(t)} = a\right]. \tag{3.7}$$

For a certain state s, a deterministic policy $\mu : \mathcal{S} \rightarrow \mathcal{A}$ returns action $a = \mu(s)$. In a stationary Markov decision process setting, the optimal Q-function associated with the target policy μ satisfies the Bellman property and we can make use of this recursive relationship as

$$Q^{\mu}(s, a) = \mathbb{E}\left[r^{(t+1)} + \gamma Q^{\mu}(s', \mu(s')) \middle| s^{(t)} = s, a^{(t)} = a\right], \tag{3.8}$$

where the expectation is over s' which follows the distribution of the state of the environment. As the target policy is deterministic, the expectation in (3.8) depends only on the environment transition dynamics. Hence, an off-policy learning method similar to deep Q-learning can be used to learn a Q-function parameterized by a deep neural network called critic network. The critic network is denoted as $Q_{\boldsymbol{\phi}}(s, a)$ with $\boldsymbol{\phi}$ being its parameters. Similarly, we parameterize the policy using another DNN named actor network $\mu_{\boldsymbol{\theta}}(s)$ with policy parameters being $\boldsymbol{\theta}$.

Let the past interactions be stored in an experience-replay memory $\mathcal{D}$ until time t in the form of $e = (s, a, r', s')$. This memory needs to be large enough to avoid over-fitting and small enough for faster training. DDPG also applies another trick called quasi-static target network approach and define two separate networks to be used in training which are train and target critic networks with their parameters denoted as $\boldsymbol{\phi}$ and $\boldsymbol{\phi}_{\text{target}}$, respectively. To train $\boldsymbol{\phi}$, at each time slot, DDPG minimizes the following mean-squared Bellman error:

$$L\left(\boldsymbol{\phi}, \mathcal{D}\right) = \mathbb{E}_{(s,a,r',s')\sim\mathcal{D}}\left[\left(y(r',s') - Q_{\boldsymbol{\phi}}\left(s,a\right)\right)^2\right] \tag{3.9}$$

where the target $y(r',s') = r' + \gamma Q_{\boldsymbol{\phi}_{\text{target}}}\left(s', \mu_{\boldsymbol{\theta}}(s')\right)$. Hence, $\boldsymbol{\phi}$ is updated by sampling a random mini-batch $\mathcal{B}$ from $\mathcal{D}$ and running gradient descent using

$$\nabla_{\boldsymbol{\phi}} \frac{1}{|\mathcal{B}|} \sum_{(s,a,r',s')\in\mathcal{B}} \left(y(r',s') - Q_{\boldsymbol{\phi}}\left(s,a\right)\right)^2 . \tag{3.10}$$

Note that after each training iteration $\boldsymbol{\phi}_{\text{target}}$ is updated by $\boldsymbol{\phi}$.

In addition, the policy parameters are updated to learn a policy $\mu_{\boldsymbol{\theta}}(s)$ which gives the action that maximizes $Q_{\boldsymbol{\phi}}(s,a)$. Since the action space is continuous, $Q_{\boldsymbol{\phi}}(s,a)$ is differentiable with respect to action and $\boldsymbol{\theta}$ is updated by gradient ascent using

$$\nabla_{\boldsymbol{\theta}} \frac{1}{|\mathcal{B}|} \sum_{(s,\dots)\in\mathcal{B}} Q_{\boldsymbol{\phi}}\left(s, \mu_{\boldsymbol{\theta}}(s)\right) . \tag{3.11}$$

To ensure exploration during training, a noise term is added to the deterministic policy output [**26**]. In our multi-agent framework to be discussed next section, we employ ϵ-greedy algorithm of Q-learning instead for easier tuning.

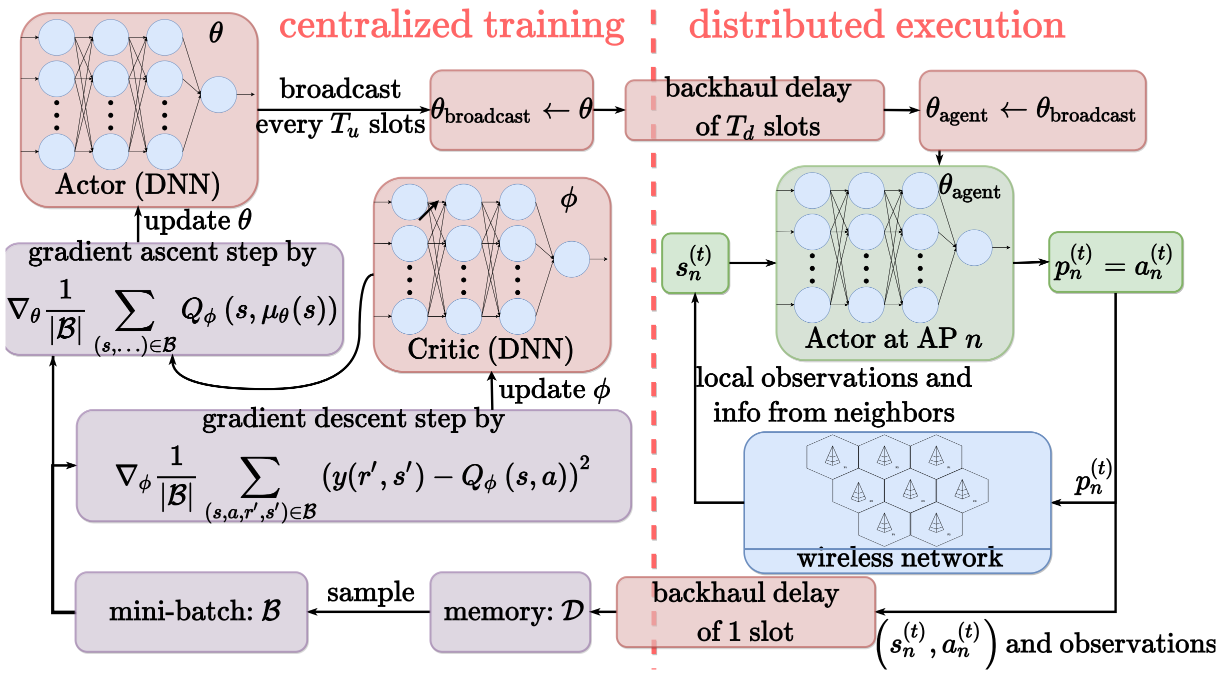

centralized training
distributed execution
θ
Actor (DNN)
broadcast every T_u slots
$\theta_{\text{broadcast}} \leftarrow \theta$
backhaul delay of T_d slots
$\theta_{\text{agent}} \leftarrow \theta_{\text{broadcast}}$
θ_{agent}
update θ
gradient ascent step by
$\nabla_\theta \frac{1}{|\mathcal{B}|} \sum_{(s,\ldots)\in\mathcal{B}} Q_\phi(s, \mu_\theta(s))$
ϕ
Critic (DNN)
update ϕ
gradient descent step by
$\nabla_\phi \frac{1}{|\mathcal{B}|} \sum_{(s,a,r',s')\in\mathcal{B}} (y(r',s') - Q_\phi(s,a))^2$
$s_n^{(t)}$
Actor at AP n
$p_n^{(t)} = a_n^{(t)}$
local observations and info from neighbors
wireless network
$p_n^{(t)}$
mini-batch: $\mathcal{B}$
sample
memory: $\mathcal{D}$
backhaul delay of 1 slot
$\left(s_n^{(t)}, a_n^{(t)}\right)$ and observations

Figure 3.1. Diagram of the proposed power control algorithm.

3.4. Proposed Multi-Agent Learning Scheme for Continuous Power Control with Mobile Users

For the proposed power control scheme in Fig. 3.1, we let each transmitter be a learning agent. Hence, the next state of each agent is determined by the joint-actions of all agents and the environment is no longer stationary. In order to avoid instability, we gather the experiences of all agents in a single replay memory and train a global actor network $\boldsymbol{\theta}_{\text{agent}}$ to be shared by all agents. At slot t, each agent $n \in \mathcal{N}$ observes its local state $s_n^{(t)}$ and sets its own action $a_n^{(t)}$ by using $\boldsymbol{\theta}_{\text{agent}}$.

For each link n, we first describe the neighboring sets that allow the distributively execution. Link n's set of interfering neighbors at time slot t consists of nearby AP indexes whose received signal-to-noise ratio (SNR) at device n was above a certain threshold

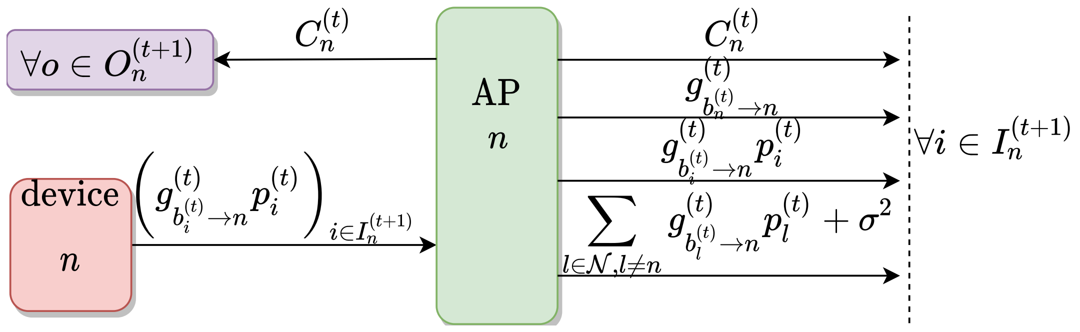

$\forall o \in O_n^{(t+1)}$
$C_n^{(t)}$
AP
n
$C_n^{(t)}$
$g_{b_n^{(t)} \to n}^{(t)}$
$g_{b_i^{(t)} \to n}^{(t)} p_i^{(t)}$
$\forall i \in I_n^{(t+1)}$
device
n
$\left(g_{b_i^{(t)} \to n}^{(t)} p_i^{(t)} \right)_{i \in I_n^{(t+1)}}$
$\sum_{l \in \mathcal{N}, l \neq n} g_{b_l^{(t)} \to n}^{(t)} p_l^{(t)} + \sigma^2$

Figure 3.2. The information exchange in time slot t.

during the past time slot and is denoted as

$$I_n^{(t)} = \left\{ i \in \mathcal{N}, i \neq n \middle| g_{i \to n}^{(t-1)} p_i^{(t-1)} > \eta \sigma^2 \right\}. \tag{3.12}$$

Conversely, we define link n's set of interfered neighbors at time slot t using the received SNR from AP n, i.e.,

$$O_n^{(t)} = \left\{ o \in \mathcal{N}, o \neq n \middle| g_{n \to o}^{(t-1)} p_n^{(t-1)} > \eta \sigma^2 \right\}. \tag{3.13}$$

To satisfy the practical constraints introduced in Chapter 2, we limit the information exchange between AP n and its neighboring APs as depicted in Fig. 3.2. Although, it is assumed in Chapter 2 that receiver n may do a more recent received power measurement from AP $i \in I_n^{(t+1)}$ just before the beginning of time slot $t+1$, i.e., $\bar{g}_{b_i^{(t)} \to n}^{(t+1)} p_i^{(t)}$, we prefer not to require it for our new model that involves mobility. Note that as device n's association changes, i.e., $b_n^{(t+1)} \neq b_n^{(t)}$, we assume that the neighboring sets are still determined with respect to the previous positioning of AP n and the feedback history from past neighbors is preserved at the new AP position to be used in agent n's state. We let the association of device change only after staying within a new cell for T_{register} consecutive slots.

For the training process, as a major modification on Chapter 2, we introduce training episodes where we execute a training process for T_{train} slots and let devices do random walk without any training for T_{travel} slots before the next training episode. The T_{travel} slot-long traveling period induces change in the channel conditions, and consequently allows policy to observe more variant states during its training which intuitively increases its robustness to the changes in channel conditions. To train a policy from scratch for a random wireless network initialization, we run E training episodes which are indexed by $\mathcal{E} = 1, \ldots, E$. The e-th training episode starts at slot $t_e = (e-1)\left(T_{\text{train}} + T_{\text{travel}}\right)$ and is composed of two parallel procedures called centralized training and distributed execution. After an interaction with the environment, each agent sends its newly acquired experience to the centralized trainer which executes the centralized training process. The trainer clears its experience-replay memory at the beginning of each training episode. Due to the backhaul delay as shown in Fig. 3.1, we assume that the most-recent experience that the trainer received from agent n at slot t is $e_n^{(t-1)}$. During slot t, after acquiring all recent experiences in the memory $\mathcal{D}$, the trainer runs one gradient step for the actor and critic networks. Since the purpose of the critic network is to guide the actor network during training, only the actor network needs to be broadcasted to the network agents and during the inference mode only the actor network is required. The trainer starts to broadcast $\boldsymbol{\theta}_{\text{broadcast}} \leftarrow \boldsymbol{\theta}_{\text{agent}}$ once every T_u slots and we assume $\boldsymbol{\theta}_{\text{broadcast}}$ is received by the agents after T_d slots again due to delay. In addition, compared to the deep Q-network in Chapter 2 that reserves an output port for each discrete action, each actor network has just one output port.

The local state of agent n at time t, i.e., $s_n^{(t)}$ is a tuple of local environment features that are significantly affected by the agent's and its neighbor's actions. As described in

Chapter 2, the state set design is a combination of three feature groups. The first feature group is called "local information" and occupies six neural network input ports. The first input port is agent n's latest transmit power $p_n^{(t-1)}$ which is followed by its contribution to the network objective (3.5), i.e., $C_n^{(t-1)}$. Next, agent n appends the last two measurements of its direct downlink channel and sum interference-plus-noise power at receiver n: $g_{n\to n}^{(t)}$, $g_{n\to n}^{(t-1)}$, $\left(\sum_{m\in\mathcal{N},m\neq n} g_{m\to n}^{(t-1)} p_m^{(t-1)} + \sigma^2\right)$, and $\left(\sum_{m\in\mathcal{N},m\neq i} g_{m\to n}^{(t-2)} p_m^{(t-2)} + \sigma^2\right)$.

These are followed by the "interfering neighbors" feature group. Since we are concerned by the scalability, we limit the number of interfering neighbors the algorithm involves to c by prioritizing elements of $I_n^{(t)}$ by their amount of interference at receiver n, i.e., $g_{i\to n}^{(t-1)} p_i^{(t-1)}$. We form $\bar{I}_n^{(t)}$ by taking first c sorted elements of $I_n^{(t)}$. As $|I_n^{(t)}| < c$, we fill this shortage by using virtual neighbors with zero downlink and interfering channel gains. We also set its spectral efficiency to an arbitrary negative number. Hence, a virtual neighbor is just a placeholder that ineffectively fills neural network inputs. Next, for each $i \in \bar{I}_n^{(t)}$, we reserve two input ports: $g_{i\to n}^{(t)} p_i^{(t-1)}$ and $C_i^{(t-1)}$. This makes a total of $2c$ input ports used for current interfering neighbors. In addition, agent n also includes the history of interfering neighbors and appends $2c$ inputs using $\bar{I}_n^{(t-1)}$.

Finally, we have the "interfered neighbors" feature group. If agent n does not transmit during slot $t-1$, $O_n^{(t)} = \emptyset$ and there will be no useful interfered neighbor information to build $s_n^{(t)}$. Hence, we define time slot t'_n as the last slot with $p_n^{(t'_n)} > 0$ and we consider $O_n^{(t'_n+1)}$ in our state set design. We also assume that as agent n becomes inactive, it will still carry on its information exchange between each $o \in O_n^{(t'_n+1)}$ without the knowledge of $g_{n\to o}^{(t-1)}$. Similar to the scheme described above, agent i regulates $O_n^{(t'_n+1)}$ to set $|\bar{O}_n^{(t)}| = c$. For $o \in O_n^{(t'_n+1)}$, the prioritization criteria is now agent i's share on the interference at receiver o, i.e., $g_{n\to o}^{(t-1)} p_n^{(t-1)} \left(\sum_{m\in\mathcal{N}, m\neq o} g_{m\to o}^{(t-1)} p_m^{(t-1)} + \sigma^2\right)^{-1}$. For each interfered neighbor $o \in O_n^{(t'_n+1)}$, $s_n^{(t)}$ accommodates three features which can be listed as: $g_{o\to o}^{(t-1)}$, $C_o^{(t-1)}$, and $g_{n\to o}^{(t'_i)} p_n^{(t'_i)} \left(\sum_{m\in\mathcal{N}, m\neq o} g_{m\to o}^{(t-1)} p_m^{(t-1)} + \sigma^2\right)^{-1}$.

The reward of agent n, $r_n^{(t+1)}$, is computed by the centralized trainer and used in the training process. Similar to Chapter 2, $r_n^{(t)}$ is defined as agent's contribution on the objective (3.5):

$$r_n^{(t+1)} = C_n^{(t)} - \sum_{o\in O_n^{(t+1)}} \pi_{n\to o}^{(t)} \tag{3.14}$$

with $\pi_{n\to o}^{(t)} = \log\left(1 + \gamma_o^{(t)}\left(\boldsymbol{b}^{(t)}, \left[\dots, p_{n-1}^{(t)}, 0, p_{n+1}^{(t)}, \dots\right]^{\mathsf{T}}\right)\right) - C_o^{(t)}$ being the externality that link n causes to interfered o.

3.5. Simulations

Following the LTE standard, the path-loss is simulated by $128.1 + 37.6\log_{10}(d)$ (in dB) with $f_c = 2$ GHz, where d is transmitter-to-receiver distance in km. We set $\sigma_{\text{s}} = 10$ dB, $d_{\text{cor}} = 10$ meters, $T = 20$ ms, $P_{\max} = 38$ dBm, and $\sigma^2 = -114$ dBm. We simulate the mobility using Haas' model [**61**] with maximum speed being 2.5 m/s. Each mobile device randomly updates its speed and direction every second uniformly within $[-0.5, 0.5]$ m/s and $[-0.175, 0.175]$ radians, respectively. Fig. 3.3 shows an example movement scenario until the end of third training episode with $T_{\text{train}} = 5,000$ and $T_{\text{travel}} = 50,000$ slots. The DDPG implementation and parameters are included in the source code [**64**]. Both WMMSE and FP start from a full power allocation, since it gives better performance than random initialization. WMMSE takes more iterations to converge than FP, resulting in higher sum-rate.

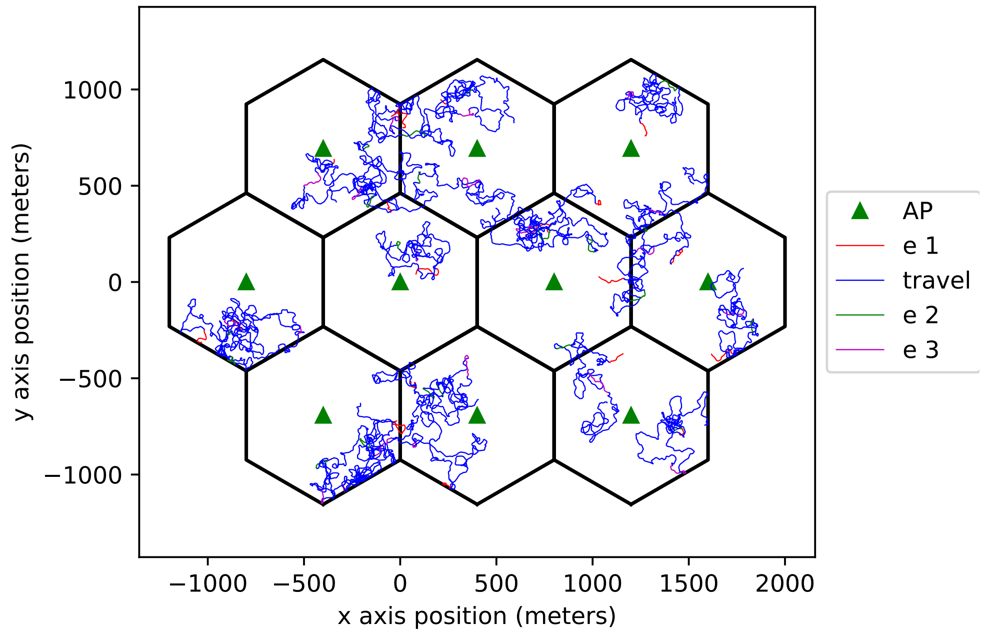


Figure 3.3. Example movement until the end of episode $e = 3$.

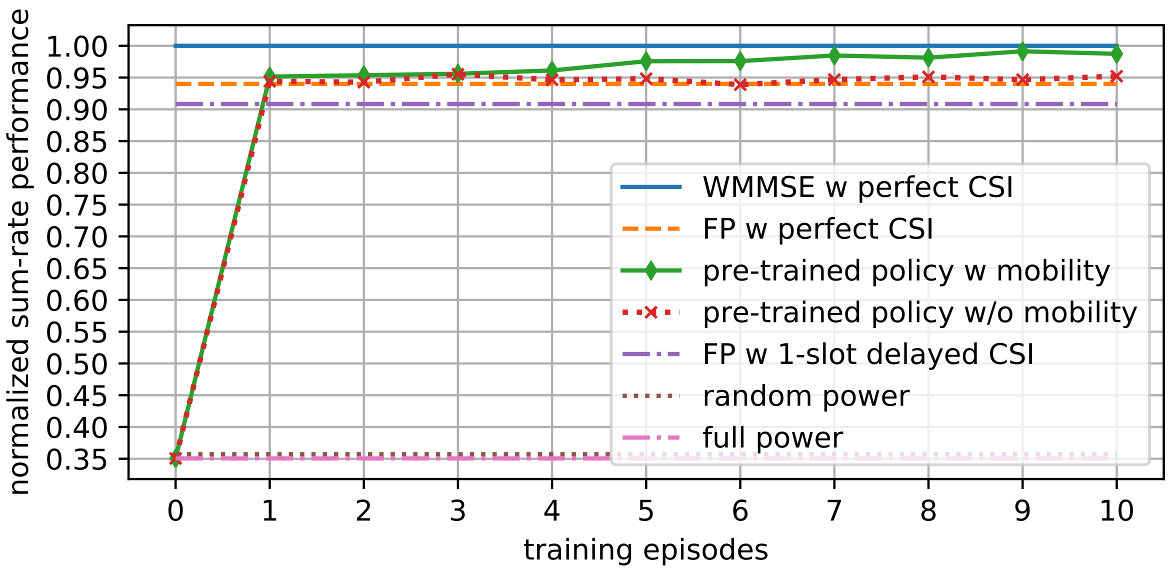


Figure 3.4. Test results for the 10 cells and 20 links scenario.

We first train two policies for $K = 10$ cells and $N = 20$ links network deployment for $E = 10$ training episodes. The first policy is trained with mobile devices, whereas the

Table 3.1. Average sum-rate performance in bps/Hz per link.

(cells,links)	policy trained for (10,20)	WMMSE	FP	FP w delay	random	full
(10,20)	2.59	2.61	2.45	2.37	0.93	0.91
(20,40)	1.97	2.09	1.98	1.87	0.68	0.68
(20,60)	1.58	1.68	1.59	1.50	0.37	0.35
(20,100)	1.14	1.23	1.15	1.09	0.18	0.17

latter is trained without mobility, i.e., with steady channel. We set f_d to 10 Hz for all time slots [**60**]. We save the policy parameters during training for testing on several random deployments with $(K, N) = (10, 20)$ and mobility. As shown in Fig. 3.4, without mobility, there is no significant sum-rate gain after the first training episode and policy converges to FP's sum-rate performance. As a remark, FP is centralized and it has full CSI, whereas actor network is distributively executed with limited information exchange. As we include device mobility and a certain travel time between training episodes, the policy is able to experience various device positions and interference conditions during training, so its sum-rate performance consistently increases. Additionally, in Table 3.1, we show that an actor network trained for $(K, N) = (10, 20)$ can keep up with the sum-rate performance of optimization algorithms as network gets larger. Hence, running centralized training from scratch is not necessary as device positions change or new devices register, since a pre-trained policy for a smaller and different deployment performs quite well. For the 20 link scenario, on average, WMMSE and FP converge in 42 and 24 iterations, respectively. For 100 links, WMMSE requires 74 iterations. Conversely, learning agent takes just one policy evaluation.

3.6. Conclusion

In this chapter, we presented a distributively executed deep actor-critic framework for power control. During training, only actor network is broadcasted to learning agents. Simulations show that a pre-trained policy gives comparable performance with WMMSE and FP, and a policy trained for a smaller deployment is applicable to a larger network without additional training thanks to the distributed execution scheme. Further, we have shown that the proposed actor-critic framework enables real-time power control under certain practical constraints and it is compatible with the case of mobile devices. DDPG in fact uses the mobility to increase its sum-rate performance by experiencing more variant channel conditions.

CHAPTER 4

Deep Reinforcement Learning for Joint Spectrum and Power Allocation in Cellular Networks

4.1. Introduction

A wireless network operator typically divides the radio spectrum it possesses into a number of subbands. In a cellular network those subbands are then reused in many cells. To mitigate co-channel interference, a joint spectrum and power allocation problem is often formulated to maximize a sum-rate objective. The best known algorithms for solving such problems generally require instantaneous global channel state information and a centralized optimizer. In fact those algorithms have not been implemented in practice in large networks with time-varying subbands. Deep reinforcement learning algorithms are promising tools for solving complex resource management problems.

A major challenge here is that spectrum allocation involves discrete subband selection, whereas power allocation involves continuous variables. In this chapter, a learning framework is proposed to optimize both discrete and continuous decision variables. Specifically, two separate deep reinforcement learning algorithms are designed to be executed and trained simultaneously to maximize a joint objective. Simulation results show that the proposed scheme outperforms both the state-of-the-art fractional programming algorithm and a previous solution based on deep reinforcement learning.

4.2. System Model

In this chapter, we consider a cellular network with N links that are placed in K cells and share M subbands. We denote the set of link and subband indexes by $\mathcal{N} = \{1, \dots, N\}$ and $\mathcal{M} = \{1, \dots, M\}$, respectively. Link n is composed of receiver n and its transmitter n. Transmitter n is placed at the corresponding cell center that includes receiver n within its cell boundaries. We consider a fully synchronized time slotted system with a fixed slot duration of T. We assume that all transmitters and receivers are equipped with a single antenna. Due to relative scarcity of available spectrum, K tends to be much larger than M, i.e., $K \gg M$. We let each link pick one subband at the beginning of each time slot.

Similar to [**16**], our channel model is composed of two parts: large and small scale fading. For simplicity, we assume that the large-scale fading is same across all subbands, whereas the small-scale fading is frequency selective, i.e., different across all subbands [**28**]. Within each subband, small-scale fading is assumed to be block-fading and flat. Let $g^{(t)}_{n\to l,m}$ denote the downlink channel gain from transmitter n to receiver l on subband m in time slot t:

$$g^{(t)}_{n\to l,m} = \beta_{n\to l} \left| h^{(t)}_{n\to l,m} \right|^2, \quad t = 1, 2, \dots\,, \tag{4.1}$$

where $\beta_{n\to l}$ is the large-scale fading that includes both path loss and log-normal shadowing, and $h^{(t)}_{n\to l,m}$ is the small-scale Rayleigh fading which is modeled by Jake's fading model (2.2). We assume that the large-scale fading remains the same through many time slots. Note that in case of mobile receivers, a time index can be associated with $\beta_{n\to l}$. Also, the cells are agnostic to the specific fading statistics a priori.

We use binary variables $\alpha_{n,m}^{(t)}$ to indicate the subband selection of link n in time slot t. If link n selects subband m, we have $\alpha_{n,m}^{(t)} = 1$ and $\alpha_{n,j}^{(t)} = 0$ for every $j \neq m$. We denote the transmit power of transmitter n in time slot t as $p_n^{(t)}$. The signal-to-interference-plus-noise at receiver n on subband m in time slot t is given by

$$\gamma_{n,m}^{(t)} = \frac{\alpha_{n,m}^{(t)} g_{n \to n,m}^{(t)} p_n^{(t)}}{\sum_{l \neq n} \alpha_{l,m}^{(t)} g_{l \to n,m}^{(t)} p_l^{(t)} + \sigma^2}, \tag{4.2}$$

where σ^2 is the additive white Gaussian noise power spectral density at receiver n. Assuming normalized bandwidth, the downlink spectral efficiency achieved by link n on subband m during time slot t is

$$C_{n,m}^{(t)} = \log\left(1 + \gamma_{n,m}^{(t)}\right). \tag{4.3}$$

4.3. Problem Formulation

Denoting subband and power vectors in time slot t as $\boldsymbol{\alpha}^{(t)} = \left[\alpha_{1,1}^{(t)}, \alpha_{1,2}^{(t)}, \ldots, \alpha_{N,M}^{(t)}\right]^{\mathsf{T}}$ and $\boldsymbol{p}^{(t)} = \left[p_1^{(t)}, \ldots, p_N^{(t)}\right]^{\mathsf{T}}$, respectively, we formulate the sum-rate maximization problem as [2, 28]:

$$\underset{\boldsymbol{p}^{(t)}, \boldsymbol{\alpha}^{(t)}}{\text{maximize}} \quad \sum_{n=1}^{N} C_n^{(t)} \tag{P1a}$$

$$\text{subject to} \quad 0 \leq p_n^{(t)} \leq P_{\max}, \forall n \in \mathcal{N}, \tag{P1b}$$

$$\alpha_{n,m}^{(t)} \in \{0, 1\}, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}, \tag{P1c}$$

$$\sum_{m \in \mathcal{M}} \alpha_{n,m}^{(t)} = 1, \forall n \in \mathcal{N}, \tag{P1d}$$

where $C_n^{(t)} = \sum_{m=1}^{M} C_{n,m}^{(t)}$ is link n's achieved spectral efficiency, and (P1b) restricts the transmit power to be nonnegative and no larger than $P_{\max}$.

Unfortunately, (P1) is in general non-convex and requires *mixed integer programming* to be carried out for each time slot as channel varies. Even for a given subband selection $\boldsymbol{\alpha}^{(t)}$, this problem has been proven to be NP-hard [**2**]. Conventional algorithms such as fractional programming are centralized solutions to (P1), but these algorithms still require many iterations to converge and their computational complexity does not scale well with increasing number of links. Besides that, obtaining instantaneous global CSI in a centralized controller and sending the allocation decisions back to all transmitters is difficult in practice.

4.4. A Two-layer Deep Reinforcement Learning Framework for Joint Spectrum and Power Allocation

4.4.1. Local Information and Neighborhood Sets

We next describe the extent of the local information at transmitter n at the beginning of time slot t. In each time slot, transmitter n has two types of neighborhood sets for each subband. The first set is called "interferers" that consists of c indexes and is denoted as $\mathcal{I}_{n,m}^{(t)}$. For subband m, transmitter n first divides nearby transmitters into two groups whether they used subband m during time slot $t-1$ or not in order to prioritize the transmitters that occupy subband m. Then, it sorts each group according to the interfering channel strength at receiver n from their transmitters during time slot $t-1$ by descending order, i.e., $g_{i\to n,m}^{(t-1)}$. Lastly, the first c sorted nearby transmitters forms $\mathcal{I}_{n,m}^{(t)}$.

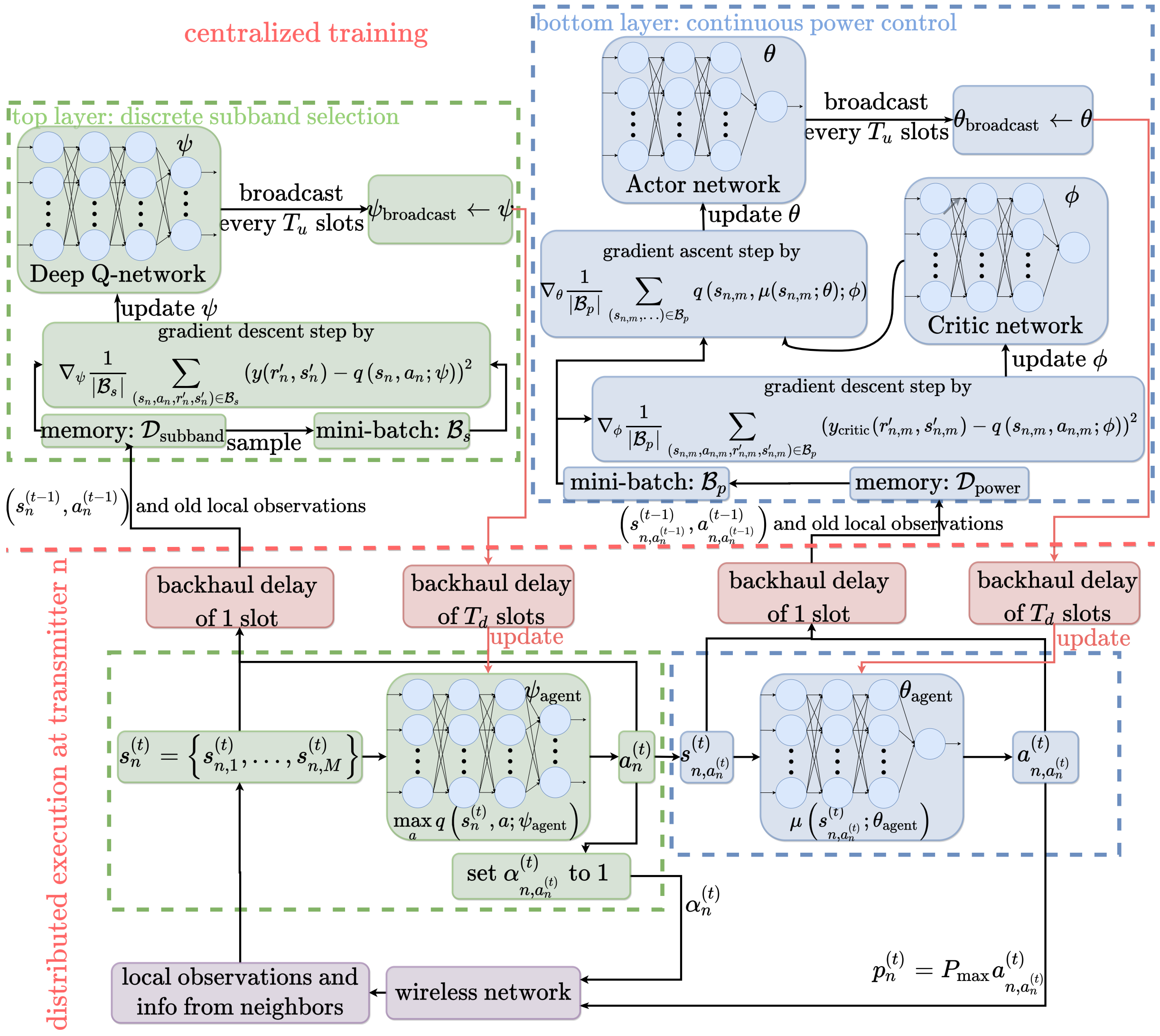

centralized training
top layer: discrete subband selection
ψ
Deep Q-network
broadcast every T_u slots
$\psi_{\text{broadcast}} \leftarrow \psi$
update ψ
gradient descent step by
$\nabla_\psi \frac{1}{|\mathcal{B}_s|} \sum_{(s_n,a_n,r'_n,s'_n)\in\mathcal{B}_s} (y(r'_n,s'_n) - q(s_n,a_n;\psi))^2$
memory: $\mathcal{D}_{\text{subband}}$
sample
mini-batch: $\mathcal{B}_s$
$\left(s_n^{(t-1)}, a_n^{(t-1)}\right)$ and old local observations
bottom layer: continuous power control
θ
Actor network
broadcast every T_u slots
$\theta_{\text{broadcast}} \leftarrow \theta$
update θ
gradient ascent step by
$\nabla_\theta \frac{1}{|\mathcal{B}_p|} \sum_{(s_{n,m},\ldots)\in\mathcal{B}_p} q(s_{n,m}, \mu(s_{n,m};\theta);\phi)$
ϕ
Critic network
update ϕ
gradient descent step by
$\nabla_\phi \frac{1}{|\mathcal{B}_p|} \sum_{(s_{n,m},a_{n,m},r'_{n,m},s'_{n,m})\in\mathcal{B}_p} (y_{\text{critic}}(r'_{n,m},s'_{n,m}) - q(s_{n,m},a_{n,m};\phi))^2$
mini-batch: $\mathcal{B}_p$
memory: $\mathcal{D}_{\text{power}}$
$\left(s_{n,a_n^{(t-1)}}^{(t-1)}, a_{n,a_n^{(t-1)}}^{(t-1)}\right)$ and old local observations
distributed execution at transmitter n
backhaul delay of 1 slot
backhaul delay of T_d slots
update
backhaul delay of 1 slot
backhaul delay of T_d slots
update
$s_n^{(t)} = \left\{s_{n,1}^{(t)}, \ldots, s_{n,M}^{(t)}\right\}$
ψ_{agent}
$\max_a q\left(s_n^{(t)}, a; \psi_{\text{agent}}\right)$
$a_n^{(t)}$
set $\alpha_{n,a_n^{(t)}}^{(t)}$ to 1
$s_{n,a_n^{(t)}}^{(t)}$
θ_{agent}
$\mu\left(s_{n,a_n^{(t)}}^{(t)};\theta_{\text{agent}}\right)$
$a_{n,a_n^{(t)}}^{(t)}$
$\alpha_n^{(t)}$
$p_n^{(t)} = P_{\max} a_{n,a_n^{(t)}}^{(t)}$
local observations and info from neighbors
wireless network

Figure 4.1. Diagram of the proposed power control algorithm.

The second set is the set of "interfered receivers" that consists of c indexes and is defined as $\mathcal{O}_{n,m}^{(t)}$. Again, each nearby receiver j is first divided into two groups based on $\alpha_{j,m}^{(t-1)}$. The sorting criteria within each group becomes the potential significance of the interference strength at receiver j from transmitter n during time slot $t-1$, i.e., $g_{n\to j,m}^{(t-1)} \left(\sum_{l\in\mathcal{N},l\neq j} \alpha_{l,m}^{(t-1)} g_{l\to j,m}^{(t-1)} p_l^{(t-1)} + \sigma^2\right)^{-1}$.

Compared to Chapter 2, we follow simpler practical constraints on the available local information to be used in the state set design, as our main goal is to show the usefulness of the proposed approach. At the beginning of time slot t, transmitter n has access to the most recent local information gathered at receiver n for each subband m such as $g^{(t)}_{n\to n,m}$, $g^{(t)}_{i\to n,m}$ $\forall i \in \mathcal{I}^{(t)}_{n,m}$, and the sum interference power at receiver n which is measured just before the new policy decisions at the beginning of time slot t and can be formulated with subband and power allocation from time slot $t-1$ and interfering channel gains from time slot t as $\sum_{l\in\mathcal{N},l\neq n} \alpha^{(t-1)}_{l,m} g^{(t)}_{l\to n,m} p^{(t-1)}_{l}$. Conversely, the measurements gathered at nearby receivers are delayed by one time slot, e.g., $g^{(t-1)}_{n\to j,m}$ $\forall j \in \mathcal{O}^{(t)}_{n,m}$. Apart from the channel related measurements, we assume that each interfered and interferer neighbor also sends crucial key performance indicators delayed by one time slot due to network latency, e.g., its achieved spectral efficiency during last slot.

4.4.2. Proposed Multi-Agent Learning Scheme

In order to allow distributed execution, each link, specifically, each transmitter, operates as an independent learning agent by treating other agents as part of its local environment. Hence, our approach is based on multiple learning agents, rather than a single learning agent that controls the entire action space whose dimensions will grow exponentially with the total number of links. The single learning agent approach has similar drawbacks as the conventional centralized optimization algorithms in terms of complexity and cost of communication. In contrast, the proposed multi-agent approach is easily scalable to larger networks and can operate with just local information after training.

At the beginning of each time slot, each agent successively executes two policies to determine its associated subband and transmit power level. The reinforcement learning component at the top layer is a deep Q-network that is responsible for the subband selection. The bottom layer uses deep deterministic policy gradient algorithm to train the actor network responsible for agent's transmit power level decisions. As described in Fig. 4.1, the actor network at the bottom layer requires the subband decision of the top layer to determine its state input before setting agent's transmit power.

We next describe key components of the proposed design:

(1) **Action set design:** All agents have the same pair of action spaces. The top layer uses a discrete action space that consists of subband indexes, i.e, $a_n^{(t)} \in \mathcal{A}_{\text{subband}} = \{1, \ldots, M\} = \mathcal{M}$. Hence, we denote the subband selection of agent n for time slot t as $a_n^{(t)}$. The bottom layer has a continuous action space defined as $\mathcal{A}_{\text{power}} = [0, 1]$. Since the bottom layer is executed after the top layer, we denote its action as $a^{(t)}_{n,a_n^{(t)}}$. We later multiply it by $P_{\max}$ to get $p_n^{(t)} = P_{\max} a^{(t)}_{n,a_n^{(t)}}$.

(2) **State set design:** To be used in the state, all agents rank the subbands at the beginning of each time slot according to their direct channel gain to the total interference power ratio. We denote the rank as $z^{(t)}_{n,m}$. Now we describe the state of agent n on subband m at time t as:

$$
\begin{aligned}
s_{n,m}^{(t)} = \Bigg\{ & \alpha_{n,m}^{(t-1)} p_n^{(t-1)}, C_n^{(t-1)}, z_{n,m}^{(t)}, g_{n\to n,m}^{(t)}, \\
& \sum_{l\neq n} \alpha_{l,m}^{(t-1)} g_{l\to n,m}^{(t)} p_l^{(t-1)}, \Big\{ g_{i\to n,m}^{(t)}, \alpha_{i,m}^{(t-1)} p_i^{(t-1)}, \\
& C_i^{(t-1)}, z_{i,m}^{(t-1)} \Big| \forall i \in \mathcal{I}_{n,m}^{(t)} \Big\}, \Big\{ g_{n\to j,m}^{(t-1)}, g_{j\to j,m}^{(t-1)}, \\
& C_j^{(t-1)}, z_{j,m}^{(t-1)}, \sum_{l\neq j} \alpha_{l,m}^{(t-1)} g_{l\to j,m}^{(t-1)} p_l^{(t-1)} \Big| \forall j \in \mathcal{O}_{n,m}^{(t)} \Big\} \Bigg\}.
\end{aligned} \tag{4.4}
$$

Since the top layer does the subband decisions that requires information from all subbands, it should have a broader environment view than the bottom layer. Thus, for the top layer, we define agent n's state as $s_n^{(t)} = \left\{ s_{n,1}^{(t)}, \ldots, s_{n,M}^{(t)} \right\}$. Then, the bottom layer uses $s_{n,a_n^{(t)}}^{(t)}$ as its input.

(3) **Reward Function Design:** Both learning layers collaboratively aim to maximize the objective in (P1a). Consequently, they share the same reward function that describes the overall contribution of agent's combined subband and power decisions on the sum-rate objective. This includes agent's own spectral efficiency and a penalty term depending on its externalities to its interfered neighbors on subband $a_n^{(t)}$ [**9**]. For the reward function, we first compute the externality of agent n to interfered $j \in \mathcal{O}_{n,a_n^{(t)}}^{(t+1)}$ during time slot t as

$$
\pi_{n\to j}^{(t)} = C_{j\backslash n,a_n^{(t)}}^{(t)} - C_{j,a_n^{(t)}}^{(t)}, \tag{4.5}
$$

where $C^{(t)}_{j\backslash n,a_n^{(t)}}$ is the spectral efficiency of j without the interference from agent n on subband $a_n^{(t)}$ during slot t:

$$C^{(t)}_{j\backslash n,a_n^{(t)}} = \log\left(1 + \frac{\alpha^{(t)}_{j,a_n^{(t)}} g^{(t)}_{j\rightarrow j,a_n^{(t)}} p_j^{(t)}}{\sum_{l\neq n,j} \alpha^{(t)}_{l,a_n^{(t)}} g^{(t)}_{l\rightarrow j,a_n^{(t)}} p_l^{(t)} + \sigma^2}\right). \tag{4.6}$$

Next, we define the reward of agent n as

$$r_n^{(t+1)} = C^{(t)}_{n,a_n^{(t)}} - \sum_{j\in\mathcal{O}^{(t+1)}_{n,a_n^{(t)}}} \pi^{(t)}_{n\rightarrow j}. \tag{4.7}$$

(4) **Centralized Training:** Since multi-agent setting violates the environment stationary assumption of the underlying Markov decision process discussed in Section 2.5, there is an extensive research to develop multi-agent learning frameworks with good empirical performance, but rarely with theoretical guarantees [**51**]. We follow some recently emerged multi-agent learning concepts like transfer learning and parameter sharing that increase the stability and convergence rate by taking advantage of the fact that agents are learning together [**52**]. Therefore, our method encourages stability by training global policy parameters shared across the network and trained by a centralized trainer that gathers experiences of all agents. As shown in Fig. 4.1, centralized training stores two experience-replay memories for each layer: $\mathcal{D}_{\text{subband}}$ and $\mathcal{D}_{\text{power}}$. At time t, the most recent experience at $\mathcal{D}_{\text{subband}}$ and $\mathcal{D}_{\text{power}}$ from agent n is $e^{(t-1)}_{n,\text{subband}} = \left(s_n^{(t-2)}, a_n^{(t-2)}, r_n^{(t-1)}, s_n^{(t-1)}\right)$ and $e^{(t-1)}_{n,\text{power}} = \left(s^{(t-2)}_{n,a_n^{(t-2)}}, a^{(t-2)}_{n,a_n^{(t-2)}}, r_n^{(t-1)}, s^{(t-1)}_{n,a_n^{(t-2)}}\right)$, respectively, due to the backhaul delay of 1 time slot. Note that the next state in $e^{(t-1)}_{n,\text{power}}$ is with respect to the old subband selection $a_n^{(t-2)}$.

Table 4.1. Testing results.

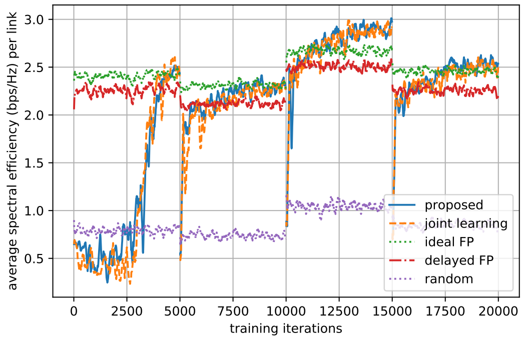
(K, N)	M	average sum-rate performance in bps/Hz per link					output layer size		average
		reinforcement learning		other schemes			reinforcement learning		iterations
(cells, links)	subbands	proposed	joint	ideal FP	delayed FP	random	proposed	joint	FP
$(5, 20)$	1	1.51	1.50	1.58	1.46	0.41	1 + 1	10	70.30
	2	2.63	2.64	2.66	2.46	0.99	2 + 1	20	102.08
	4	4.57	4.38	3.81	3.57	2.12	4 + 1	40	122.15
$(10, 50)$	1	1.26	1.26	1.31	1.21	0.25	1 + 1	10	72.83
	2	2.08	2.10	2.08	1.92	0.59	2 + 1	20	96.32
	4	3.34	3.34	2.90	2.68	1.31	4 + 1	40	185.93
	5	3.79	3.76	3.18	2.94	1.64	5 + 1	50	206.38
	10	5.71	4.41	4.44	4.08	2.99	10 + 1	100	287.70

During time slot t, the centralized training runs one gradient step for each policy. As described in Fig 4.1, it broadcasts most recent versions of ψ and θ once per T_u time slots. The broadcasting takes T_d time slots to finish, again due to the backhaul delay.

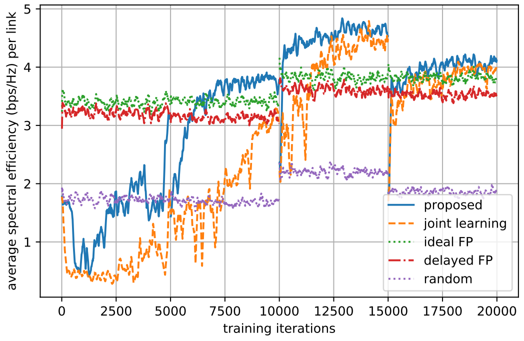
4.5. Simulation Results

In this section, we compare the performance of the proposed learning approach with some conventional optimization methods and joint learning as the number of subbands increases.

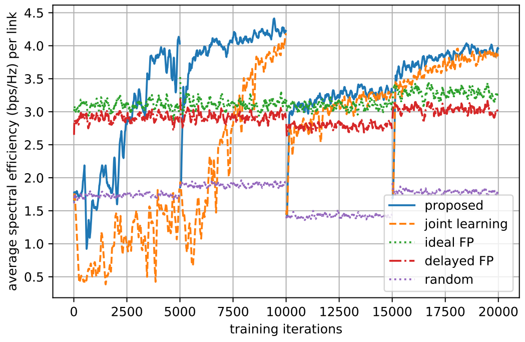
Throughout the simulations, we choose two network sizes of $(K, N) = (5 \text{ cells}, 20 \text{ links})$ and $(10 \text{ cells}, 50 \text{ links})$, respectively. Similar to previous chapters, we consider homogeneous hexagonal cells of 400 meters radius with each cell having equal number of uniformly randomly placed receivers. We vary the number of subbands M from 1 to 10. Following the LTE standard, we set the distance dependent path loss to $128.1 + 37.6 \log_{10}(d)$ (in dB), where d is transmitter-to-receiver distance in km. The log-normal shadowing standard deviation is 10 dB. We set $f_d = 10$ Hz, $T = 20$ ms, $P_{\max} = 38$ dBm, and $\sigma^2 = -114$ dBm.



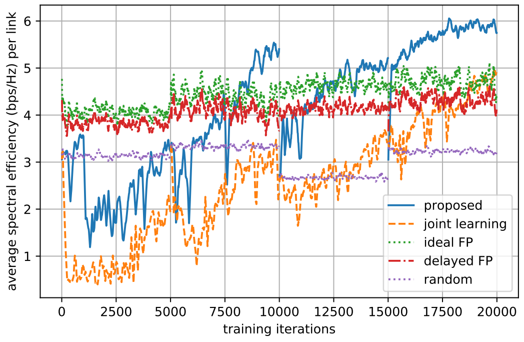
(a) $M = 2$ subbands, $(K, N) = (5, 20)$.



(b) $M = 4$ subbands, $(K, N) = (5, 20)$.



(c) $M = 5$ subbands, $(K, N) = (10, 50)$.



(d) $M = 10$ subbands, $(K, N) = (10, 50)$.

Figure 4.2. Training convergence.

Similar to Chapter 2, the signal-to-interference-plus-noise ratio is capped at 30 dB in the calculation of the spectral efficiency in (4.3) due to practical constraints on front end's dynamic range.

We compare the proposed approach with four benchmarks. The first is the joint learning approach as proposed in [**28**]. We discretize the transmit power into 10 levels. The second is called the 'ideal FP'. It runs the fractional programming algorithm with an assumption of full instant CSI. The first scenario ignores any delay during the execution of centralized optimization or passing the optimization outcomes to the transmitters. On the other hand, the third benchmark is called the ‘delayed FP’ and assumes one time slot delay to run the fractional programming algorithm. In the final benchmark, each transmitter just picks a random subband and transmit power at the beginning of every time slot.

We divide training into four episodes with each running for 5,000 time slots. At the beginning of each episode, we randomly sample a new deployment, and we reset the exploration and learning rate parameters. For faster convergence, we replace the noise term added to the deterministic policy output with Q-learning’s e-greedy algorithm. We have made the source code (including the implementation and hyper-parameters) available at [**65**]. For better stability, we ensure that the bottom layer has higher learning rate than the top layer, and it uses a higher initial value of ϵ, but with a higher decay rate. The fine-tuning of the ϵ value is important to avoid converging to undesired situations in which all agents want to transmit with $P_{\max}$ or with zero power.

In Fig. 4.2, we show the training convergence of the proposed and joint reinforcement learning scheme. For $M = 2$ subbands, as shown in Fig. 4.2a, their convergence rates are quite close. However, when we increase the number of subbands, the joint learning approach is not able to keep up with the proposed approach in terms of training convergence. This is mainly caused by the increased size of the joint learning’s action space and increased deep Q-network output layer complexity. Next, we test the performance of the trained policies on several randomly generated deployments in Table 4.1. Testing shows that a pretrained policy is still usable on new deployments and the proposed approach is better scalable than the benchmarks. In addition, compared to the joint learning and fractional programming, cross-layer learning’s complexity scales better.

4.6. Conclusion and Future Work

We have demonstrated a novel multi-agent reinforcement learning framework for the joint subband selection and power control problem. With centralized training and distributed execution, only local information is needed by the agent under practical constraints. In addition, as the number of subbands increases, the proposed learning approach has better training convergence and higher sum-rate performance than the joint learning approach. For future work, we are looking into better and easily tunable training and exploration schemes to better adapt to the environment non-stationarity of the multi-agent setting.

CHAPTER 5

Deep Reinforcement Learning for the Fundamental Problem of Radio Resource Management

5.1. Introduction

Next generation cellular networks are expected to support a massive data traffic volume and satisfy a vast number of users that have latency-critical quality-of-service expectations. Due to inherently scarce shared frequency-band resources over time-varying multi-channel and traffic conditions, a scalable fast-timescale resource management is an absolute necessity towards next generation cellular networks. This chapter proposes a multi-agent deep reinforcement learning based resource management scheme that can respond to the changes in traffic and channel dynamics instantaneously. The proposed multi-agent scheme considers each link as an individual learning agent that allocates resources according to a policy and the state of its local wireless environment. With the help of a novel reward function design, each agent appropriately adapts its resources in each time slot to queue lengths and channel qualities in its neighborhood. The agents collaboratively maximize some quality of service (QoS) over their local environment, where the QoS may be the average packet delay, the sum rate, a proportionally fair throughput, or anything else specified by the network layer.

Although the generic weighted sum-rate maximization problem can be particularized to achieve maximum sum-rate or proportionally fair scheduling as shown in previous chapters, a traffic-driven wireless resource management scheme based on conventional optimization techniques or previously considered reinforcement learning based schemes would require a different utility function or reward function design as their objectives. However, the change in the objective function would require a new formulation and analysis from scratch which is not efficient and not flexible in terms of addressing the changes in network QoS and fairness requirements. Our goal is to define the fundamental radio resource management problem and introduce a solution that works regardless of the choice of network objective and operates on a timescale of one time slot which is typically a few-ten milliseconds.

In this chapter, we also consider several additional practicality constraints on channel measurements, including the situation where only aggregate interference levels can be measured in lieu of all individual cross-channel gains. Simulations demonstrate the effectiveness of the proposed approach compared to optimization based resource allocation schemes, including a popular proportionally fair solution.

The rest of this chapter is organized as follows. In Section 5.2, we introduce the overall system model. In Section 5.3, we describe the fundamental radio resource allocation problem and the main motivation. In Section 5.4, we introduce the main components of the proposed algorithm. We give simulation results in Section 5.5. We finally conclude with a discussion of possible future directions in Section 5.6.

5.2. System Model

In this chapter, we consider N links over a K cell SISO interfering broadcast channel where the base station $k \in \mathcal{K} = 1, 2, \ldots, K$, serves N_k users in cell k on M subbands.

Let $\mathcal{N} = \{1, \ldots, N\}$ and $\mathcal{M} = \{1, \ldots, M\}$ denote the set of link and subband indexes, respectively. Since we are not concerned with the user association problem, we assume as link n's corresponding user n is inside cell k, its associated base station is located at the center of cell k. The cell association of user n is denoted as $b_n \in \mathcal{K}$.

We define the links that are associated to base station k as

$$\mathcal{N}_k = \{n \in \mathcal{N} | b_n = k\} . \tag{5.1}$$

We consider a fully synchronized time slotted system with a fixed slot duration of T.

For joint spectrum allocation and power control, existing deep learning based schemes are usually motivated by the essentially of an effective spectrum allocation scheme due to the relative scarcity of spectrum availability compared to the number of users. Therefore, it is common to assume that each link can pick at-most one subband at a time. The straight-forward approach to control spectrum and power jointly would be using a single deep Q-network as the policy with its action space being the Cartesian product of available subbands and quantized transmit power levels [**28**, **66**]. In Chapter 4, we have improved this design by a novel two layered approach where the policies at the top and bottom layers are separately responsible for spectrum allocation and power control, respectively. This approach reduces the action space complexity compared to [**28**, **66**] as the number of subbands increases. Note that we do not have a constraint on the number of subbands in this chapter for the sake of channel model simplicity, but this constraint can also be introduced as a future work using the design in [**28**] or [**67**].

Base station b_n transmits the signal $x_{n,m}^{(t)}$ to user n on subband m in time slot t using transmit power $p_{n,m}^{(t)} \geq 0$. The power constraint in each base station for subband m

becomes

$$\sum_{n\in\mathcal{N}_k} p_{n,m}^{(t)} \leq P_{\max}, \forall k \in \{1,2,\ldots,K\} \tag{5.2}$$

and restricts the total transmit power in base station k to be no larger than $P_{\max}$ on subband m. Next, we can write the received signal $y_{n,m}^{(t)}$ at receiver antenna of user n as

$$y_{n,m}^{(t)} = \underbrace{h_{b_n\to n,m}^{(t)}\sqrt{p_{n,m}}x_{n,m}^{(t)}}_{\text{desired signal}} + \underbrace{\sum_{i\in\mathcal{N}_{b_n}, i\neq n} h_{b_n\to n,m}^{(t)}\sqrt{p_{i,m}}x_{i,m}^{(t)}}_{\text{intracell interference}} + \tag{5.3}$$

$$\underbrace{\sum_{j\in\mathcal{N}, j\notin\mathcal{N}_{b_n}} h_{b_j\to n,m}^{(t)}\sqrt{p_{j,m}}x_{j,m}^{(t)} + n_{0,n}}_{\text{intercell interference plus noise}}, \forall n \in \mathcal{N}, \tag{5.4}$$

where $h_{b_n\to n,m}^{(t)}$ and $h_{b_j\to n,m}^{(t)} \in \mathbb{C}$ denote the direct and interfering downlink channel gain coefficients on subband m in time slot t from base station b_n and b_j to user n, respectively, and $n_{0,n} \sim \mathcal{CN}(0,\sigma^2)$ represents the noise in the receiver antenna of user n with σ^2 being the additive white Gaussian noise power spectral density which is assumed to be the same at all receivers without loss of generality.

Denoting the system power vector over subband m in time slot t as $\boldsymbol{p}_m^{(t)} = \left[p_{1,m}^{(t)}, p_{2,m}^{(t)}, \ldots, p_{N,m}^{(t)}\right]^\mathsf{T}$, the signal-to-interference-plus-noise ratio (SINR) at user n on subband m in time slot t can be expressed as

$$\gamma_{n,m}^{(t)}\left(\boldsymbol{p}_m^{(t)}\right) = \frac{|h_{b_n\to n,m}^{(t)}|^2 p_{n,m}^{(t)}}{\sum_{j\in\mathcal{N}, j\neq n} |h_{b_j\to n,m}^{(t)}|^2 p_{j,m}^{(t)} + \sigma^2}. \tag{5.5}$$

Following (5.5), link n's achieved spectral efficiency in time slot t on subband m becomes

$$C_{n,m}^{(t)}\left(\boldsymbol{p}_m^{(t)}\right) = \log\left(1 + \gamma_{n,m}^{(t)}\left(\boldsymbol{p}_m^{(t)}\right)\right). \tag{5.6}$$

Then, link n's total spectral efficiency in time slot t on subband m can be computed as

$$C_n^{(t)} = \sum_{m\in\mathcal{M}} C_{n,m}^{(t)}\left(\boldsymbol{p}_m^{(t)}\right). \tag{5.7}$$

5.2.1. Traffic Model

We assume that each link has an independent infinite length first-in-first-out (FIFO) queue. We denote the fixed packet length as L and link n's queue length in the unit of bits at the beginning of time slot t as $N_n^{(t)}$. For each link n, let $A_n^{(t)}$ be the newly arrived packets at link n's queue at the beginning of time slot t.

Using the spectral efficiency from the previous time slot, the queue lengths are updated as

$$N_n^{(t)} = \max\left(N_n^{(t-1)} - C_n^{(t-1)}WT, 0\right) + A_n^{(t)}L, \tag{5.8}$$

where W is the total bandwidth and $N_n^{(0)}$ is set to zero since queues start empty as also mentioned above. In addition, we also define total number of packets in link n's queue that are awaiting service at the beginning of time slot t as

$$N_{p,n}^{(t)} = \left\lceil \frac{N_n^{(t)}}{L} \right\rceil. \tag{5.9}$$

5.2.2. Channel Variations

Similar to [**9**, **16**], our channel model is composed of two parts: large and small scale fading. For simplicity, we assume that the large-scale fading is same across all subbands, whereas the small-scale fading is frequency selective, i.e., different across all subbands [**28**]. Within each subband, small-scale fading is assumed to be block-fading and flat. Let $g^{(t)}_{b_n \to n,m} = |h^{(t)}_{b_n \to n,m}|^2$ denote the downlink channel gain from transmitter b_n to receiver n on subband m in time slot t which is composed of large-scale and small-scale fading components as described in (4.1).

To simulate channel variations we adopt Jake's fading model, so the small-scale fading for each channel follows a first-order complex Gauss-Markov process as shown in (2.2).

5.3. The Fundamental Problem Formulation For Radio Resource Management

In this section, we first describe the fundamental problem of radio resource management. We analyze the problem in the downlink, but the discussion can be extended to the uplink as well using a similar reasoning.

The radio resource management can be thought as a control policy that makes decisions to allocate physical layer resources to serve the network layer traffic. In wireless cellular networks, spectrum is divided into multiple subbands, particularly M subbands, as we assume in our system model. Depending on the allocation decisions, link n can use all of these subbands. Considering rapidly varying channel conditions, the allocation decisions should ideally be instantaneous, i.e., whithin the slot duration of the wireless network. Therefore, the allocation decisions associated with link n on subband m at time slot t can fundamentally be modeled as the power spectral density base station n utilizes on subband m to transmit data to user n which is denoted as $p_{n,m}^{(t)}$.

Due to the characteristics of wireless channels, as link n is active on subband m, i.e., $p_{n,m}^{(t)} > 0$, it will suffer from interference caused by nearby links that are also active on the same subband. Therefore, the transmit powers of all links impact the service rate that determines how fast the network layer traffic can be passed to the user through the physical layer.

Link n's downlink data is generated by source n and this source is connected to a subset of base stations that are close to user n by backhaul links. For simplicity, we assume that each link has an associated base station where its downlink data is stored at a queue.

Next, we can characterize the long-term utility of user n as U_n. Intuitively, this utility function should reflect the average packet delay from the arrival at queue n to the end of transmission to user n, e.g., negative of the average packet delay for utility maximization. We can also define the long-term network utility as $U = \sum_{n \in \mathcal{N}} U_n$. Finally, the fundamental problem becomes finding an optimal control policy to maximize this long-term network utility.

To better understand this fundamental problem, we change our perspective and analyze the queue lengths from time slot to time slot. Hence, we can describe the goal of the fundamental problem as minimizing the cumulative queue lengths over time. The allocation decision at time slot t does not only influence the queue lengths of all links at time slot $t+1$ but also the queue lengths over a long time interval with a discounted effect. Hence, at time slot t optimal control policy minimizes a discounted cumulative objective and the fundamental problem at time slot t with the per-cell power constraint (5.3) can be formulated as:

$$\begin{aligned}
&\text{(P1a)} \qquad && \underset{\boldsymbol{p}^{(t)}}{\text{minimize}} \quad && \sum_{\tau=0}^{\infty} \gamma^{\tau} \sum_{n\in\mathcal{N}} N_n^{(t+\tau+1)} \\
&\text{(P1b)} && \text{subject to} && p_{n,m}^{(t)} \geq 0, \forall n \in \mathcal{I}, \\
&\text{(P1c)} && && \sum_{j\in\mathcal{N}_k} p_{j,m}^{(t)} \leq P_{\max}, \forall k \in \{1,2,\ldots,K\},
\end{aligned}$$

where $\gamma \in (0,1]$ is the discount factor and $\boldsymbol{p}^{(t)} = \left[\boldsymbol{p}_1^{(t)},\ldots,\boldsymbol{p}_N^{(t)}\right]^{\mathsf{T}}$ is the decision variables.

From (5.9), link n's queue length at time slot t, $N_n^{(t)}$, depends only on the queue length at previous time slot $N_n^{(t-1)}$ and the number of newly arrived packets $A_n^{(t)}$, and departures characterized by the service rate $\mu_n^{(t)} = C_n^{(t-1)}WT$. Consequently, from the queue length perspective, the fundamental problem for resource management resembles a Markov decision process [**68**] with the goal of optimal policy being minimizing the cumulative discounted objective P1b.

Because of the underlying Markov decision process, the model-free reinforcement learning [**68**] matches well with the fundamental problem and is a promising tool to find optimal control policies. Model-free reinforcement learning [**68**] is a trial-and-error process where an agent interacts with an unknown environment in a sequence of discrete time steps to achieve a task. At time t, agent first observes the current state of the environment which is a tuple of relevant environment features and is denoted as $s^{(t)} \in \mathcal{S}$, where $\mathcal{S}$ is the set of possible states. It then takes an action $a^{(t)} \in \mathcal{A}$ from an allowed set of actions $\mathcal{A}$ according to a policy which can be either stochastic, i.e., π with $a^{(t)} \sim \pi(\cdot|s^{(t)})$ or deterministic, i.e., μ with $a^{(t)} = \mu(s^{(t)})$ [**63**]. Since the interactions are often modeled as a Markov decision process, the environment moves to a next state $s^{(t+1)}$ following an unknown transition matrix that maps state-action pairs onto a distribution of next states, and the agent receives a reward $s^{(t+1)}$. Overall, the above process is described as an experience at $t+1$ denoted as $e^{(t+1)} = \left(s^{(t)}, a^{(t)}, r^{(t+1)}, s^{(t+1)}\right)$. The goal is to learn a policy that maximizes the cumulative discounted reward at time t, defined as

$$R^{(t)} = \sum_{\tau=0}^{\infty} \gamma^{\tau} r^{(t+\tau+1)}. \tag{5.10}$$

For the resource management problem, the policy should be both traffic and channel aware by embedding the traffic and channel conditions in the state set design. Hence, the policy essentially maps a traffic and channel state to a physical resource allocation.

Additionally, we define user priorities that connect physical layer resource management with network layer and indicate link's traffic condition. By embedding traffic conditions in the state set design through user priorities, policy can achieve any traffic related network objective with a suitably designed reward function.

We next describe deep Q-learning algorithm that is a practical reinforcement learning algorithm that trains a neural network to map state to an optimal action according to a reward function. Moreover, we explain multi-agent learning basics that lets distributively solving the fundamental problem. Finally, we give a divide-and-conquer solution for the fundamental problem.

We have already described deep Q-learning algorithm that is a practical reinforcement learning algorithm that trains a neural network to map state to an optimal action according to a reward function in Section 2.4. Moreover, in Section 2.5, we explained the multi-agent learning basics that lets distributively solving the fundamental problem.

Finally, in this section, we describe the typically used divide-and-conquer solution for the fundamental problem.

5.3.1. A Divide-And-Conquer Solution For The Fundamental Problem

For sum-rate maximization and proportionally fair scheduling, using non-negative user priorities denoted as $\alpha_n^{(t)}, \forall n \in \mathcal{N}$, we can construct a weighted sum-rate maximization problem for time slot t with per-cell power constraint (5.3) as

$$\begin{aligned}
&\text{(P2a)} && \underset{\boldsymbol{p}_m^{(t)}, \forall m \in \mathcal{M}}{\text{maximize}} && \sum_{n=1}^{N} \alpha_n^{(t)} \sum_{m \in \mathcal{M}} C_{n,m}^{(t)} \left(\boldsymbol{p}^{(t)}\right) \\
&\text{(P2b)} && \text{subject to} && p_{n,m}^{(t)} \geq 0, \forall n \in \mathcal{I}, \\
&\text{(P2c)} && && \sum_{j \in \mathcal{N}_k} p_{j,m}^{(t)} \leq P_{\max}, \forall k \in \{1, 2, \ldots, K\}.
\end{aligned}$$

The problem (P1) is in general non-convex has been proven to be NP-hard [**2**]. For a single subband, there exists centralized sub-optimal solutions that use conventional optimization algorithms such as fractional programming [**4**] and WMMSE algorithm [**3**]. For multiple subbands, Tan *et al.* [**28**] assumed a single band per link constraint and have proposed a hybrid approach using mixed integer programming for channel selections and WMMSE for power allocation. Since we do not have the single band per link constraint in our system model, for our benchmarks, we just run original WMMSE algorithm for single subband separately on all subbands with user priorities being same for all WMMSE executions on different subbands.

For proportionally fair scheduling [**34**], at the beginning of time slot t, user n's current priority $\alpha_n^{(t)}$ is updated as:

$$\alpha_n^{(t)} = \frac{1}{\bar{C}_n^{(t-1)}}, \tag{5.11}$$

where $\bar{C}_n^{(t-1)} = \beta C_n^{(t-1)} + (1-\beta)\bar{C}_n^{(t-2)}$ is user n's weighted average spectral efficiency computed at the end of time slot $t-1$ with $\beta \in (0, 1]$ being a parameter to control the impact of history. By setting user priorities according to (5.11), maximizing the network objective in (P1) will maximize the sum of log average spectral efficiency [**8**], i.e.,

$$\sum_{n \in \mathcal{N}} \log \bar{C}_n^{(t)}, \tag{5.12}$$

which will make user n's long-term average spectral efficiency proportional to its long-term channel quality.

5.4. A Deep Reinforcement Learning Framework for Traffic-Driven Resource Management

5.4.1. Channel Measurements and Neighborhood Set

In this chapter, we use a popular (suboptimal) algorithm called WMMSE [**3**] for benchmarking purposes. Similar to other conventional optimization techniques used for power control, WMMSE requires full-cross link CSI. However, this assumption is not practical for a non-stationary wireless environment that has varying channel gains due to its fast and small scaling fading components. In Chapter 2, we considered some important practicality constraints on the channel measurements. As described in [**9**, Fig. 1], we have assumed that a receiver can only measure its direct channel gain, total received interference power level, and individual interference power levels received from nearby links whose associated base station was causing an interference above a certain threshold during previous time slot. Even though these assumptions on channel measurements were quite effective to address practicality and feasibility concerns on conventional optimization algorithms that use full CSI, in this chapter we remove the latter assumption to improve the practicality even further. For convenience, we denote the aggregated interference, i.e., total received interference power level, at user n's receiver on subband m in time slot $t-1$ as

$$\zeta_{n,m}^{(t-1)} = \sum_{j\in\mathcal{N},j\neq n} |h_{b_j\to n,m}^{(t-1)}|^2 p_{j,m}^{(t-1)} + \sigma^2. \tag{5.13}$$

In addition to (5.13), we also assume that receiver n can also measure the aggregated interference at the end of time slot $t-1$ with the updated channel gains but with power allocation during $t-1$, i.e.,

$$\bar{\zeta}_{n,m}^{(t)} = \sum_{j\in\mathcal{N},j\neq n} |h_{b_j\to n,m}^{(t)}|^2 p_{j,m}^{(t-1)} + \sigma^2. \tag{5.14}$$

As explained before, we removed the assumption on measuring individual interference power levels, so we have also modified the way each link n forms a neighborhood set which was based on individual interference level passing a certain threshold in previous chapters. In this chapter, each link n has assigned to a neighborhood set based on the small scaling component which is assumed to be the steady state of the channel and measurable over multiple time slots. The neighborhood set of link n is the set of c receivers with largest $\beta_{b_n\to i}$ and it is denoted as $\mathcal{O}_n$. Note that the neighborhood set of each link is updated as

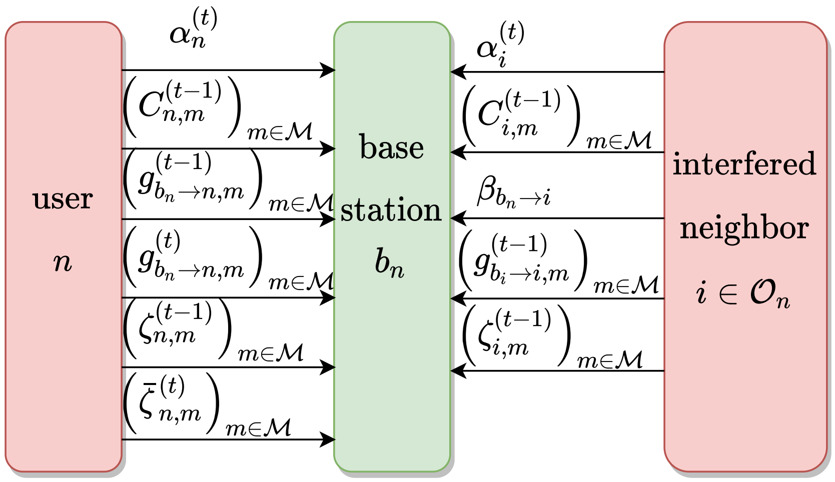

user
n
base
station
b_n
interfered
neighbor
$i \in \mathcal{O}_n$
$\alpha_n^{(t)}$
$\left(C_{n,m}^{(t-1)}\right)_{m\in\mathcal{M}}$
$\left(g_{b_n\to n,m}^{(t-1)}\right)_{m\in\mathcal{M}}$
$\left(g_{b_n\to n,m}^{(t)}\right)_{m\in\mathcal{M}}$
$\left(\zeta_{n,m}^{(t-1)}\right)_{m\in\mathcal{M}}$
$\left(\bar{\zeta}_{n,m}^{(t)}\right)_{m\in\mathcal{M}}$
$\alpha_i^{(t)}$
$\left(C_{i,m}^{(t-1)}\right)_{m\in\mathcal{M}}$
$\beta_{b_n\to i}$
$\left(g_{b_i\to i,m}^{(t-1)}\right)_{m\in\mathcal{M}}$
$\left(\zeta_{i,m}^{(t-1)}\right)_{m\in\mathcal{M}}$

Figure 5.1. The information exchange scheme on subband m which is required to build the local state of link n at the beginning of time slot t.

network topology changes. In case of mobile users, this set can also be updated regularly with a certain period.

We illustrate the information exchange to build the local state of link n at the beginning of time slot t in Fig. 5.1. The local state of link n will be formed by the base station which makes the resource allocation decisions for link n. Therefore, user n and link's interfered neighbors in $\mathcal{O}_n$ inform base station b_n with their priorities and recent channel measurements.

User n passes six important environment features to the base station. The first two features are user n's priority for time slot t and its achieved spectral efficiency on subband m during time slot $t-1$, i.e., $\alpha_n^{(t)}$ and $C_{n,m}^{(t-1)}$, respectively. Next two features are last two measurements of the direct downlink channel gain between base station b_n and receiver n on subband m, i.e., $g_{b_n\to n,m}^{(t)}$ and $g_{b_n\to n,m}^{(t-1)}$. These are followed by the aggregated interference measurements on subband m with power allocation from time slot $t-1$ and channel gains from time slots $t-1$ and t, i.e., $\zeta_{n,m}^{(t-1)}$ and $\bar{\zeta}_{n,m}^{(t)}$, respectively. Since channel variations are modeled as a first-order complex Gauss-Markov process, user n only passes last two direct channel and aggregated interference measurements.

In addition, base station b_n gathers five environment features from each interfered neighbor $i \in \mathcal{O}_n$. Similar to local information exchange, the first two features are interfered neighbor i's priority for time slot t and its achieved spectral efficiency on subband m during time slot $t-1$, i.e., $\alpha_n^{(t)}$ and $C_{n,m}^{(t-1)}$, respectively. Base station b_n also uses the small scale fading component of the interfering channel to receiver i, i.e., $\beta_{b_n \to i}$, in order to identify and prioritize neighbors according to the significance of the potential interference level they may receive from base station b_n's transmission. Finally, neighbor i passes its direct downlink channel gain from base station b_i and aggregated interference level at receiver i on subband m at the beginning of time slot $t-1$. Due to the backhaul network delay, most-recent measurements of those metrics could not be passed to base station b_n before the power allocation decision on time slot t.

5.4.2. Decentralized Multi-Agent Execution Scheme

As depicted in Fig. 5.2, we employ a decentralized execution scheme where each link acts as an individual agent, and deep reinforcement learning scheme of each link's learning agent is handled by its associated base station. Compared to the centralized execution scheme, there are three main advantages of distributively executing reinforcement learning based resource management scheme. Firstly, for the decentralized execution scheme, the state and action sets only scale with the number of subbands. Due to relative scarcity of available spectrum, we do not anticipate this to be a problem. However, for a centralized

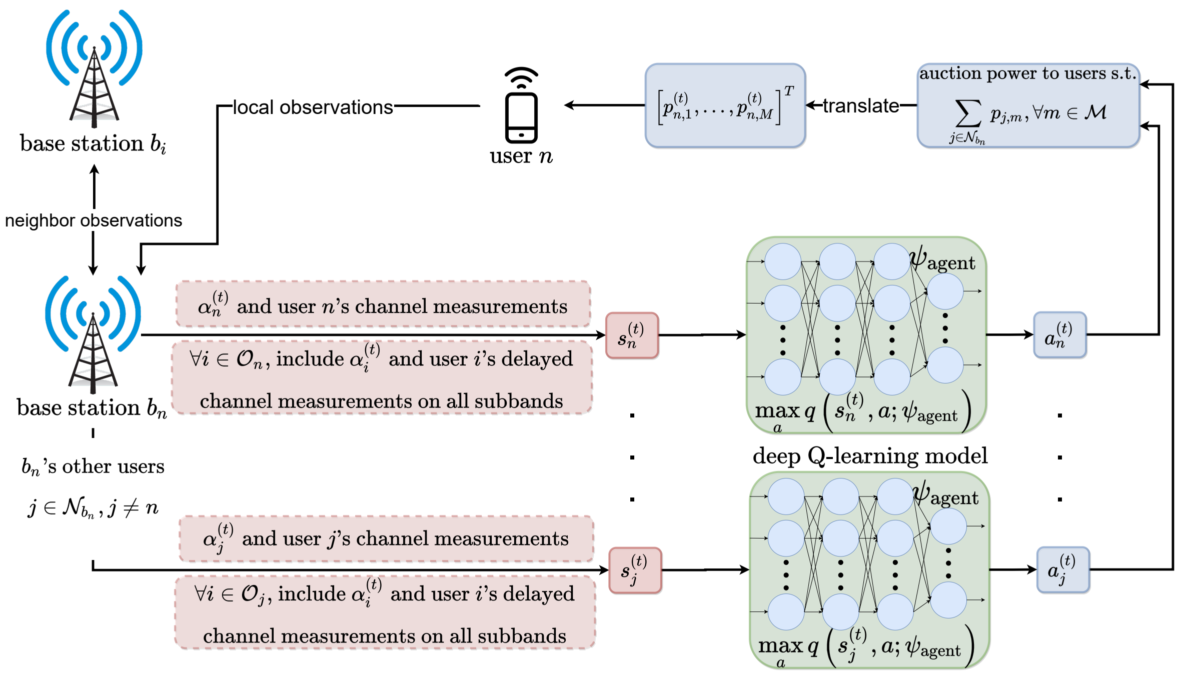
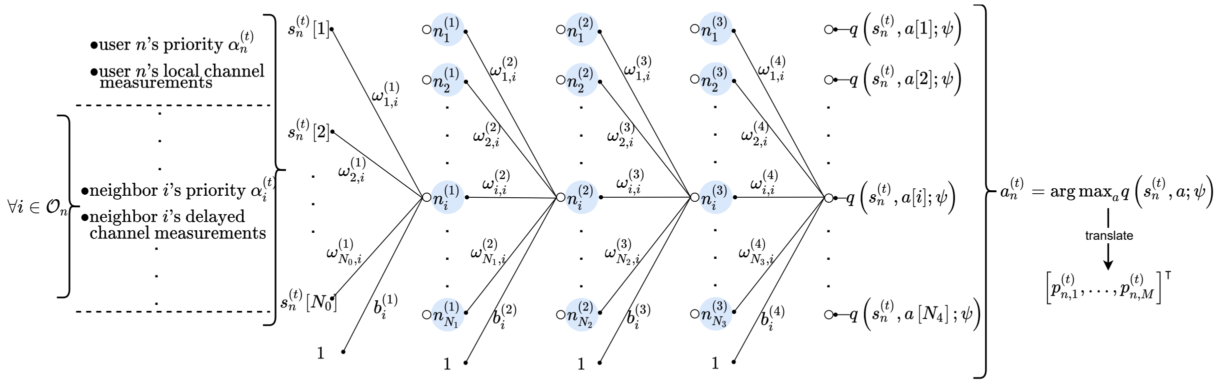

base station b_i
local observations
user n
$\left[p_{n,1}^{(t)}, \ldots, p_{n,M}^{(t)}\right]^T$
translate
auction power to users s.t. $\sum_{j \in \mathcal{N}_{b_n}} p_{j,m}, \forall m \in \mathcal{M}$
neighbor observations
$\alpha_n^{(t)}$ and user n's channel measurements
$\forall i \in \mathcal{O}_n$, include $\alpha_i^{(t)}$ and user i's delayed channel measurements on all subbands
base station b_n
b_n's other users $j \in \mathcal{N}_{b_n}, j \neq n$
$s_n^{(t)}$
ψ_{agent}
$\max_a q\left(s_n^{(t)}, a; \psi_{\text{agent}}\right)$
$a_n^{(t)}$
deep Q-learning model
$\alpha_j^{(t)}$ and user j's channel measurements
$\forall i \in \mathcal{O}_j$, include $\alpha_i^{(t)}$ and user i's delayed channel measurements on all subbands
$s_j^{(t)}$
$\max_a q\left(s_j^{(t)}, a; \psi_{\text{agent}}\right)$
$a_j^{(t)}$

Figure 5.2. Diagram of the proposed decentralized execution scheme from link n's perspective.

approach, the state and action space complexity would also scale with the number of links in addition to the number of subbands. Secondly, decentralized execution allows a policy trained for a much smaller network to be adapted on a larger scale network since state and action space remains unchanged as new links added to the network. As shown in as shown in [**32**, **67**], a pre-trained policy for a smaller network deployment can be moved to a larger network deployment without any additional training and this will cause minimal performance regression. Lastly, decentralized execution scheme is more practical, because it just requires limited information exchange between nearby base stations and it does not depend on gathering information from all base stations at a centralized agent.

As described in Fig. 5.2, we define the local state of reinforcement learning agent n as $s_n^{(t)}$. Note that reinforcement learning agent n is associated with link n and it



•user n's priority $\alpha_n^{(t)}$
•user n's local channel measurements
$\forall i \in \mathcal{O}_n$
•neighbor i's priority $\alpha_i^{(t)}$
•neighbor i's delayed channel measurements
$s_n^{(t)}[1]$
$s_n^{(t)}[2]$
$s_n^{(t)}[N_0]$
$q\left(s_n^{(t)}, a[1]; \psi\right)$
$q\left(s_n^{(t)}, a[2]; \psi\right)$
$q\left(s_n^{(t)}, a[i]; \psi\right)$
$q\left(s_n^{(t)}, a[N_4]; \psi\right)$
$a_n^{(t)} = \arg\max_a q\left(s_n^{(t)}, a; \psi\right)$
translate
$\left[p_{n,1}^{(t)}, \ldots, p_{n,M}^{(t)}\right]^\intercal$

Figure 5.3. The overall local state set design and the architecture of the 5-layered deep Q-network used in this work. The notations n, ω and b indicate deep Q-network neurons, weights, and biases, respectively. The set of deep Q-network parameters is denoted as ψ.

determines link n's resource allocation for time slot t across all available subbands, i.e., $\left[p_{n,1}^{(t)}, \ldots, p_{n,M}^{(t)}\right]^\intercal$.

We next describe the overall local state set design and the architecture of the deep Q-network in Fig. 5.3. Number of neurons at the input layer N_0 is equal to the number of environment features used in the local state design. Similarly, the size of output layer N_4 equals to the number of possible actions agent n can take. Since the policy gives the joint resource allocation across all available subbands, the number of possible actions will become M-th power of the quantization levels used in the quantization of the transmit power level. Then, at the output layer, the action which gives the maximum Q-function value is selected and translated into actual transmit power levels using the mapping that will be described in Section 5.4.4.

As described in 5.3, the local state of agent n is composed of environment features that are relevant in determining agent n's action which can be listed as follows:

(1) priority of user n, i.e., $\alpha_n^{(t)}$,

(2) most-recent channel measurements of user n that is gathered according to the information exchange between base station b_n and user n described in Fig. 5.1,

(3) priorities and delayed channel measurements of all interfered neighbors, i.e., $\alpha_i^{(t)}, \forall i \in \mathcal{O}_n$.

We further explain the details of the state set design in Section 5.4.3.

After building the local state $s_n^{(t)}$, base station b_n executes a deep Q-network with globally shared parameters ψ_{agent} to determine agent n's action $a_n^{(t)}$. Base station b_n also runs the same deep Q-network scheme for other links that share the power resources of the same base station with link n, i.e., links $j \in \mathcal{N}_{b_n}$ such that $j \neq n$. Then, base station b_n enforces the power constraint $P_{\max}$ on each subband by analyzing the actions of all of its agents. Initially, in case multiple agents asking for more power than allowed, we intuitively considered normalizing the power level to $P_{\max}$ and penalizing these agents with respect to the amount of power constraint violation. Even though, our preliminary results showed that this method works fine for sum-rate maximization problem, i.e., all priorities are set to one, the intuitive approach caused problems when we aim for minimizing average packet delay. For example, as two agents with long queues ask for full power on a given subband, operating these two on the same subband simultaneously would cause an SINR level that is less than 0 dB, and consequently, longer queues for upcoming time slots which is a vicious cycle that is not easy to recover, and less significant the power constraint penalties as the policy moves away from the objective. Hence, we instead propose to use an auction mechanism to solve situations when multiple agents ask for more power than allowed on subband m. According to this mechanism, base station b_n picks the agent that asked for the highest power and the rest gets no power on subband m. In case of a tie, the agent with longest queue gets the resources on subband m.

5.4.3. State Set Design

In this section, we will discuss the environment features used in the local state of agent n in time slot t which is denoted as $s_n^{(t)}$. Since we assume to have only aggregated interference measurements instead of individual interfering channel gains, we deeply modify the older local set design in Chapter 2. The state set design relies on some intuitive judgments and we validate its effectiveness by comparing it to the older design using the preliminary simulations executed in Section 5.5.

We now describe the environment features used in $s_n^{(t)}$. We divide these environment features into three groups as:

5.4.3.1. User n's Priority: The first feature group is intuitively the priority of user n that needs to be adjusted depending on the wireless network objective.

For traffic-aware scheduling, user n's priority consists of two entries. First entry is reserved to total number of packets in link n's queue that are awaiting service at the beginning of time slot t which can be denoted as $N_{p,n}^{(t)}$. In addition to the total accumulated load at host (base station b_n), agent n also requires an estimation of arrival rate using recent packet arrivals as the second entry of its priority. Let $\bar{\lambda}_n^{(t)}$ be the rate estimate of link n for time slot t. Agent n uses a slot-base approach which can be expressed as

$$\bar{\lambda}_n^{(t)} = \frac{\sum_{\tau=1}^{T_\mathrm{r}} \gamma_\mathrm{r}^\tau A_n^{(t-\tau)}}{\sum_{\tau=1}^{T_\mathrm{r}} \gamma_\mathrm{r}^\tau}, \tag{5.15}$$

where T_{r} is the length of history and $\gamma_{\mathrm{r}} \in (0,1)$ is the discount factor for rate estimation. Although the estimation (5.15) may not be quite accurate due to large standard deviation of Poisson packet arrival process, we intuitively expect this situation to better realize the case of bursty arrivals, consequently causing policy to act promptly.

Besides the traffic-aware scheduling, the user priorities can also be used to achieve the described objectives in Section 5.3.1. For sum-rate maximization objective, the priority is just empty and has no entry. On the other hand, for proportionally fair scheduling, we reserve a single entry for user n's priority. At the beginning of time slot t, this entry is set to user n's current priority $\alpha_n^{(t)}$

For convenience, we denote the number of deep Q-learning input ports reserved for user n's priority as $|\alpha|$ which is equal to 0 for sum-rate maximization, 1 for proportionally fair scheduling, and 2 for traffic-aware scheduling. Thus, depending on the objective, we reserve $|\alpha|$ input ports at deep Q-network's input layer for $s_n^{(t)}$'s user n's priority feature group.

5.4.3.2. User n's Local Channel Measurements: This feature group gives agent n crucial information about link n's channel quality and received interference level. This feature group can be divided into M feature subgroups corresponding to M subbands. For subband m, the first element of feature subgroup m is the spectral efficiency of link n on subband m during the previous time slot, i.e., $C_{n,m}^{(t-1)}$. For the second element, we use total power allocated to user n during the previous time slot on subband m, i.e., $p_{n,m}^{(t-1)}$. Next, we reserve next two elements to last two measurements of the direct channel gain between base station b_n and user n, i.e., $g_{b_n \to n,m}^{(t)}$ and $g_{b_n \to n,m}^{(t-1)}$. Finally, as last two elements of feature subgroup m, we put last two aggregated interference measurements $\bar{\zeta}_{n,m}^{(t)}$ and

$\zeta_{n,m}^{(t-1)}$. Therefore, we reserve a total of six input ports for each feature subgroups, making a total of $6M$ input ports reserved for user n's local channel measurements.

5.4.3.3. Neighbor Priorities and Channel Measurements: Similar to previous subsection, we can divide this feature group into few feature subgroups. This time a feature subgroup represents agent n's interfered neighbor $i \in \mathcal{O}_n$. For neighbor i's feature subgroup, the first element is the large-scale fading component $\beta_{b_n \to i}$ which is used as the criteria to determine the indices of the neighborhood set $\mathcal{O}_n$. The purpose of this first element is to identify and prioritize interfered neighbors based on the potential significance of the interference they receive caused by agent n's decisions. Next, we feed neighbor i's priority, i.e., $\alpha_i^{(t)}$, which is follows the same rationale described in Section 5.4.3. Hence, for proportional fairness we reserve one input port per neighbor for neighbor priorities, and for traffic-aware scheduling we require two input ports for the same purpose.

Next, we include channel measurements for each neighbor i on each subband m. For neighbor i on subband m, we allocate three elements that are related to the channel condition: the spectral efficiency of link i during on subband m during the previous time slot, i.e., $C_{i,m}^{(t-1)}$, the direct channel gain between base station b_i and user i on subband m in time slot $t-1$, i.e., $g_{b_i \to i,m}^{(t-1)}$, and most-recent aggregated interference measurement of user i that is available at base station b_n, i.e., $\zeta_{i,m}^{(t-1)}$.

Following the design explained above, we have a total of $(|\alpha| + 3M)c$ input ports reserved for neighbor priorities and channel measurements.

5.4.4. Action Set

Deep Q-learning based resource management scheme requires a translation mechanism that turns discrete actions into actual power levels. Since the policy parameters are the same for all agents, agents also share the same action space design.

There have been multiple approaches to design this translation mechanism. [**24**] translates deep Q-learning's action output into discrete steps on the previous transmit power level. In Chapter 2, action space is composed of discrete power levels between 0 and $P_{\max}$. Therefore, the quantizer design and the number of levels, i.e., number of possible actions, have an impact on the performance. For example, [**31**] states that quantizing actions space with a logarithmic step size instead of linear step size improves the performance. Therefore, we follow a similar approach to [**31**].

Agent n executes deep Q-network to get link n's power allocation over all available subbands, i.e., $\left[p_{n,1}^{(t)}, \ldots, p_{n,M}^{(t)}\right]^{\intercal}$. We define the allowed actions on subband $m \in \mathcal{M}$ as

$$\mathcal{A}_m = \left\{0, P_{\min}, P_{\min}\left(\frac{P_{\max}}{P_{\min}}\right)^{\frac{1}{|\mathcal{A}_m|-2}}, \ldots, P_{\max}\right\}, \tag{5.16}$$

where $P_{\min}$ is the minimum positive transmit power level and $|\mathcal{A}_m|$ is the number of quantization levels for the transmit power level. We assume that transmit power quantization $\mathcal{A}_m$ is the same for all available subbands. Next, the action space of agent n becomes the following Cartesian product:

$$\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_M. \tag{5.17}$$

5.4.5. Reward Function Design

We enable collaboration by including signal from neighbors to agent's reward, so we define the reward function as the local objective as

$$r_{\text{local objective,n}}^{(t+1)} = \pi_n^{(t)} + \sum_{i \in \mathcal{O}_n} \pi_i^{(t)}, \tag{5.18}$$

where $\pi_n^{(t)}$ is agent n's direct contribution to the network objective. The traffic-driven approach wants to minimize the average packet delay, so the objective is to minimize long-term queue lengths. Therefore, agent n's direct contribution to the network objective can be expressed as the negative of link n's queue length at the end of time slot $t-1$ before the new arrivals:

$$\pi_n^{(t)} = -\max\left(N_n^{(t)} - C_n^{(t)} WT, 0\right). \tag{5.19}$$

Therefore, the reward of agent n at time slot $t+1$ becomes the negative amount of bits in its own queue plus the negative amount of bits its neighbors' queues at the end of time slot t. If the discount factor parameter γ is set to zero, this reward function would result with a policy that pushes as many bits as possible for a given time slot and maximizes throughput. As we set discount factor parameter γ to a value that is close to 1, the reward function will intuitively lead to a policy that tries to minimize the amount of bits accumulated in the queues in long-term. Therefore, with the use of discounted reward, policy can lead to a traffic-aware resource allocation scheme that ensures queue stability and low packet delays.

One of the key advantages of the reinforcement learning compared to conventional optimization techniques is that the reinforcement learning is more adaptable to different network objectives by simply adapting the reward function accordingly. Apart from the traffic-driven approach, if the goal of the policy is to adapt the scheduling solutions in Section 5.3.1 that take weighted sum-rate maximization problem as their objective, $\pi_n^{(t)}$ is set to $\alpha_n^{(t)} C_n^{(t)}$ with user priorities $\alpha_n^{(t)}$ follow the desired fairness scheme, e.g., sum-rate maximization or proportionally fair scheduling.

Additionally, we have also tried to use global objective as an alternative, i.e.,

$$r_{\text{global objective,n}}^{(t+1)} = \sum_{j \in \mathcal{N}} \pi_j^{(t)}, \tag{5.20}$$

but this alternative resulted with significant performance regression in the preliminary simulations which is probably caused by the noise from too far links that are not in agent's neighbor set and are not represented in agent's local state.

Finally, we also would like to describe what might be a better but impractical reward function alternative This is the reward function approach based on externalities which was first introduced in Chapter 2 where the reward of agent n consists of its own contribution to the network objective and penalties that reflects its externality to its neighbors. Hence, the externality based reward of agent n at time slot $t+1$ caused by its action taken in time slot t can be expressed as

$$r_{\text{externalities,n}}^{(t+1)} = \pi_n^{(t)} - \sum_{i \in \mathcal{O}_n} \pi_{n \to i}^{(t)}, \tag{5.21}$$

where $\pi_{n \to i}^{(t)}$ is the externality caused to interfered neighbor i by the interference from base station b_n due to the power allocated to link n, i.e., $\left[p_{n,1}^{(t)}, \ldots, p_{n,M}^{(t)}\right]^{\mathsf{T}}$. Although the reward approach described in (5.21) is shown to be quite effective in Chapter 2, the externality computation would require individual interfering channel gains from base station b_n to neighbor i, i.e., $g_{b_n \to i}^{(t)}$, $\forall i \in \mathcal{O}_n$. As we described in Section 5.4.1, we have tightened the practicality constraints on channel measurements and measuring these individual interfering channel gains on a time slot scale is assumed to be impractical, so computation of the externality terms is not feasible with the new assumptions.

It is important to note that the local objective approach causes performance regression compared to the externality based reward function approach, because $\pi_i^{(t)}$ is neighbor's overall contribution that is effected by all interference, whereas externality signal $\pi_{n \to i}^{(t)}$ clearly reflects the direct impact of agent's interference on $\pi_i^{(t)}$. Fortunately, in Section 5.5, our preliminary simulations on sum-rate maximization problem show that $r_{\text{local objective,n}}^{(t+1)}$ still gives close sum-rate performance to $r_{\text{externalities,n}}^{(t+1)}$.

5.4.6. Decentralized Execution and Episodic Training Scheme with Varying Traffic Load

In this chapter, we improve the earlier proposed centralized training scheme in Chapter 2 to handle varying traffic loads. In our preliminary simulations, we have seen that it is possible to train a single policy to handle various traffic load conditions in the execution stage without further adjustment on the policy once the policy has been trained. Throughout the episodic training, the deep Q-network parameters ψ_{agent} are carried over from one episode to another episode, and we achieve comparable performance on various average traffic load conditions by systematically increasing the average traffic load during training.

Algorithm 1 Decentralized execution.

```
1: Parameters: $\epsilon$-greedy algorithm's $\epsilon$.
2: Inputs: Deep Q-network parameters at agents $\psi_{\text{agent}}$.
3: Decentralized execution ($\psi_{\textbf{agent}}$) for time slot $t$:
4: for agent $n = 1, 2, \ldots, N$ do
5:     Agent $n$ observes its local environment and uses information from its neighbors to form its current local state $s_n^{(t)}$.
6:     Agent gets Q-function estimates $q\left(s_n^{(t)}, a; \psi_{\text{agent}}\right), \forall a \in \mathcal{A}$, using deep Q-network with parameters $\psi_{\text{agent}}$.
7:     Agent sets its current action to $a_n^{(t)} = \arg\max_a q\left(s_n^{(t)}, a; \psi_{\text{agent}}\right)$.
8:     If index $n$ is divisible by $t \mod N$, apply $\epsilon$-greedy strategy for exploration during training and agent replaces $a_n^{(t)}$ with a random action with a probability of $\epsilon$.
9:     Agent computes its power allocation during time slot $t$, i.e., $\left[p_{n,1}^{(t)}, \ldots, p_{n,M}^{(t)}\right]^\intercal$, by translating action $a_n^{(t)}$ according to (5.17).
10: end for
    Output: $\boldsymbol{p}_m^{(t)}, \forall m \in \mathcal{M}$, and state-action pairs $\left(s_n^{(t)}, a_n^{(t)}\right) \forall n \in \mathcal{N}$.
```

Algorithm 2 Centralized training.

```
1: Parameters: Learning rate $\lambda_{\text{lr}}$.
2: Inputs:
3: Deep Q-network parameters $\psi$, $\psi_{\text{broadcast}}$, $\psi_{\text{agent}}$.
4: Global memory $\mathcal{D}_{\text{g}}$ and experience-replay memory of the current episode $\mathcal{D}$.
5: Centralized training ($\psi, \psi_{\textbf{broadcast}}, \psi_{\textbf{agent}}, \mathcal{D}_{\textbf{g}}, \mathcal{D}$):
6: Randomly sample a mini-batch $\mathcal{B}$ from the experiences in $\mathcal{D}_{\text{g}}$ and $\mathcal{D}$.
7: Update the parameters $\psi$ using a gradient descent step to minimize (2.16) with learning rate equal to $\lambda_{\text{lr}}$.
8: If it has been $T_u$ since last policy broadcast, update $\psi_{\text{broadcast}}$ by $\psi$ and initiate a broadcast process which will take $T_d$ time slots. At the end of the broadcast process, $\psi_{\text{agent}}$ will be set to $\psi_{\text{broadcast}}$.
    Output: Updated deep Q-network parameters $\psi$, $\psi_{\text{broadcast}}$, $\psi_{\text{agent}}$.
```

The proposed episodic training scheme is composed of multiple consecutive episodes with each episode starts with initializing a wireless network from scratch by randomly re-deploying users and resetting the queues. At the beginning of each episode, we also set average arrival rate (traffic load) to λ_{avg}. λ_{avg} tracks average rate the policy is set to be trained for the current episode in order to ensure its adaptation on varying traffic loads. The initial value of λ_{avg} is set to λ_0 at the beginning of the episodic training. As we will explain in more detail along this section, at the end of each consecutive episode, we check for the convergence of policy's average packet delay performance and increment λ_{avg} by a rate increment parameter λ_{inc}. Episodic training has additional parameters such as $T_{\max}$ and E_{fail} which control the maximum time slots per episode and the maximum consecutive episode failures, respectively.

Inside each episode, training is structured as a series of interactions between two algorithms namely "Decentralized exection" and "Centralized Training" which are described in Algorithms 1 and 2. These interactions occur on a time scale of 1 time slot, i.e., T. Decentralized execution algorithm has a parameter called ϵ which controls the frequency of exploration in ϵ-greedy strategy. As also described in Algorithm 1, each agent takes turns to replace policy's indicated action by a random action with a probability of ϵ. Note that as the policy training is over, we set ϵ to zero for testing the performance of the trained policy. On the other hand, episodic training's second algorithm, called the centralized training algorithm, has a parameter called learning rate λ_{lr} that controls how large the gradient descent steps will be. Moreover, this learning rate parameter can be decayed slowly after each training iteration for better policy stability and convergence [**58**].

At the beginning of episodic training, we initialize deep Q-network parameters ψ randomly, and the other deep Q-network parameters $\psi_{\text{broadcast}}$, ψ_{agent} with ψ. These sets of parameters are used as inputs to centralized training algorithm and trained by that algorithm as training evolves. Furthermore, we also initialize two sets of memories global memory $\mathcal{D}_{\text{g}}$ and experience-replay memory $\mathcal{D}$. The experience-replay memory $\mathcal{D}$

is cleared at the beginning of each episode and it stores most-recent experiences of agents that are gathered with current average arrival rate λ_{avg}. At the end of each episode, we append $\mathcal{D}$ to the global memory $\mathcal{D}$g, so it includes experiences from prior experiences, and consequently prior arrival rates. Since we want to train a single policy that can handle different arrival rates, the centralized algorithm randomly samples a mini-batch with half of the experiences coming from each memories. Hence, as training evolves and the arrival rate increases over episodes, the policy has still some influence from the past episodes, and this keeps performance for lower arrival rates more stable and avoids drastic policy steps that may cause performance regression for lower arrival rates.

Within an episode, at the beginning of time slot t, the episodic training first runs decentralized execution with the current version of policy parameters, i.e., ψ_{agent}. The decentralized execution returns policy's resource allocation decision to be used during time slot t, i.e., $\boldsymbol{p}_m^{(t)}$, $\forall m \in \mathcal{M}$, and state-action pairs $\left(s_n^{(t)}, a_n^{(t)}\right) \forall n \in \mathcal{N}$. With the resource allocation set by the decentralized execution, the episodic training scheme waits until the end of time slot t. Next, it observe the new queue states and determine reward $r_n^{(t+1)}, \forall n \in \mathcal{N}$. For each agent n, it later forms experiences in the form of $\left(s_n^{(t)}, a_n^{(t)}, r_n^{(t+1)}, s_n^{(t+1)}\right)$ and add these newly formed experiences to the experience-replay memory $\mathcal{D}$. Finally, the policy parameters are updated according to the centralized training algorithm with its inputs set to current version of policy parameters and memories.

If the queues remain stable through the episode, the execution and training interaction that we described above is repeated for $T_{\max}$ time slots. After $T_{\max}$ time slots before moving to the next episode, the resulting ψ is stored to be returned later as episodic training's final result. In addition, at the end of each time slot we also check for the stability of the queues, and if the queues become unstable, i.e., if one of the links has a queue with number of waiting packets above a certain threshold, the episode is labeled as unsuccessful and terminated. The episodic training process ends after E_{fail} consecutive unsuccessful episodes and most-recent deep Q-network parameters ψ from the last successful episode becomes the final outcome of training.

5.5. Simulation Results

In this section, we evaluate the performance of the proposed traffic-driven resource allocation scheme through several numerical results.

5.5.1. Simulation Setup

We consider N users that are randomly deployed on K homogeneous cells of 500 meters radius with each cell having a stationary base station located at its center. We vary the number of links and base stations in order to examine the scalability of the proposed resource management approach. In our simulations, we use channel parameters that are in compliance with the LTE standard. The distance dependent base station to user path-loss is simulated by $128.1+37.6\log_{10}(d)$ (in dB), where d is base station-to-user distance in km. Additionally, we set the log-normal shadowing standard deviation to 8 dB. Given the spectrum scarcity in wireless cellular networks, it is typical to have much fewer available subbands than number of users, so we vary number of subbands M just from 1 to 4 in our simulations. The bandwidth of a subband is 10 MHz and we set $P_{\max}$ and σ^2 to 23 dBm and -114 dBm, respectively. Due to the typical limitations of finite-precision signal processing, we cap the spectral efficiency calculation at 30 dB received SINR. The duration of time slot T is set to 20 ms. Unless the user locations are unchanging for a simulation scenario, we set the maximum Doppler frequency to 10 Hz for all receivers.

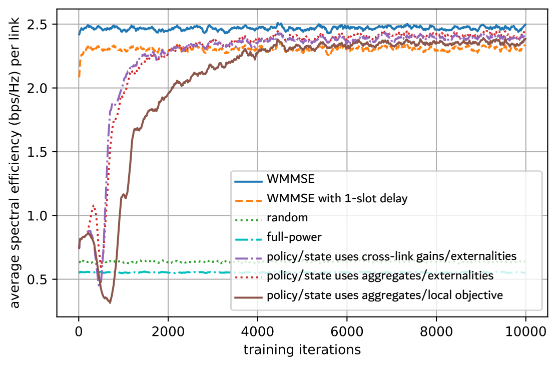
We perform three main categories of simulations: sum-rate maximization, proportionally fair scheduling, and traffic simulations where queues are enabled as described in Section 5.2. In the first two categories of simulations, we assume full-buffer traffic, since the network objective is the sum-rate and the sum of log average spectral efficiency, respectively. These categories are used for the proof-of-concept of the multi-agent reinforcement learning for resource management framework and showing the effect of recent modifications in the state set and reward function designs. Beside these categories, we carry through the traffic simulations with the purpose of proof-of-concept for the traffic-driven resource management with channel measurements and user priorities depending on their queue conditions.

In the traffic simulations, we set the packet length to 0.5 Mbits and vary average arrival rate (traffic load) per link from 1 to 70 packets per second. Packet arrivals follow a Poisson process with λ being the arrival rate in packets per second, specifically, $\Pr\left(A_n^{(t)} = k\right) \sim \frac{(\lambda T)^k}{k!} e^{-\lambda T}$. For each link, at the beginning of time slot t, if new packets arrived, we append these newly arrived packets at the end of links' queue. 'Packet' objects consist of an arrival timestamp and a parameter that denotes the number of remaining bits to be served. In addition to these, we also store two additional timestamps to record packet's wait time that is set after packet's first bit starts to be transmitted from the base station and total service time that is set after packet's last bit is transmitted.

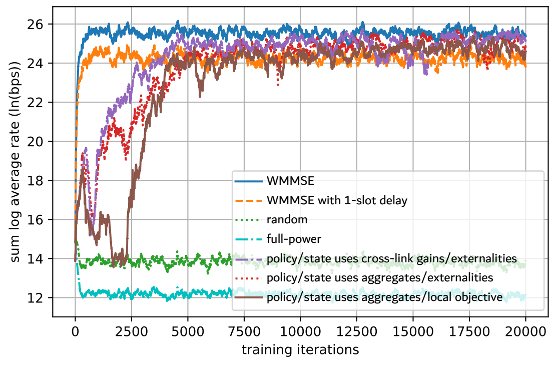
After loading the traffic, we determine the priority of each link based on the power allocation scheme, e.g., average spectral efficiency for proportionally fair scheduling [**8**,**34**] or queue lengths for a traffic-aware scheme as proposed in this chapter. Next, we run the specified power allocation scheme using user priorities and determine each link's allocated spectral efficiency for time slot t. At the end of the time slot, using the allocated spectral efficiency, the bandwidth, and the duration of a single time slot; we determine the total number of bits that can be transmitted within the time slot as explained in (5.8), and process the packets using FIFO rule.

5.5.2. Sum Rate Maximization and Proportionally Fair Scheduling

Before the simulation results for the traffic-driven approach, we first start with sum-rate maximization and proportionally fair scheduling. The user priorities for these objectives are described in Section 5.3. Since user priorities are not traffic-driven and we assume full-buffer traffic for these scenarios, we just execute training on a single episode similar to Chapter 2. We run policy training for 5 different training seeds and show the convergence of policy training (averaged over seeds within the plots) in Fig. 5.4. As we have pointed out before, in this chapter, we removed the individual interfering channel gain measurement assumption. Hence, compared to the work in Chapter 2, we replaced these interfering channel gain measurements with aggregated interference power measurements. Besides, these interfering channel gain measurements are also required for externality computation described in (5.21), so we replaced the reward function that uses externalities in (5.21) with the local objective approach in (5.18). As shown in Fig. 5.4, the change in the state set design causes smaller regression compared to the modification in the reward function. However, the performance regression in the network objective is still negligible,



(a) Sum-Rate Maximization



(b) Proportionally Fair Scheduling

Figure 5.4. Training convergence on the $N = 20$ links, $K = 10$ cells, $M = 1$ subband scenario with different state set and reward function approaches.

so we conclude this subsection by noting that the modifications in the state set and the reward function designs do not cause significant performance degradation.

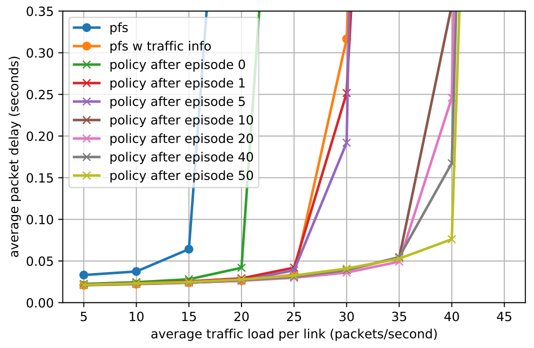
5.5.3. Performance of the Proposed Traffic-Driven Resource Management Scheme

In this subsection, we compare the performance of the traffic-driven approach with two benchmarks: proportionally fair scheduling (pfs) and pfs with traffic information. The first benchmark uses WMMSE algorithm that maximizes a weighted sum-rate objective. The user weights, i.e., priorities, follows (5.11) to achieve proportional fairness. Also, it is important to keep in mind that, WMMSE algorithm is centralized, requires full CSI, and assumes no backhaul delay while receiving CSI information or sending resource management decisions back to base stations. Therefore, WMMSE algorithm does not follow the practicality constraints introduced in Section 5.4.1, whereas the proposed traffic-driven resource management is able to operate with those constraints. The second benchmark also follows the WMMSE algorithm, but enhances the user priority assignment described in (5.11) by also setting user priority to zero if user's queue is empty at the beginning of time slot. Although this enhancement is not an elegant solution, it is quite an effective way to avoid utilization to the links that have no packet to send during the time-slot and reserve more resources to the links that have traffic. We observe that for lower traffic loads, this enhancement significantly improves the average packet latency performance compared to the first benchmark, but as the traffic load increases the second benchmark essentially becomes identical to regular pfs, because it gets more likely to have majority of links with unempty queues as traffic load increases. Similarly, for the traffic-driven resource management, an agent does not execute the policy to determine an action when its corresponding link has an empty queue, and the agent simply set transmitter power to zero. This is because, intuitively, the optimal action is zero-power for an empty queue case, i.e., zero-priority case, so there is no need to explore for the zero-priority case.

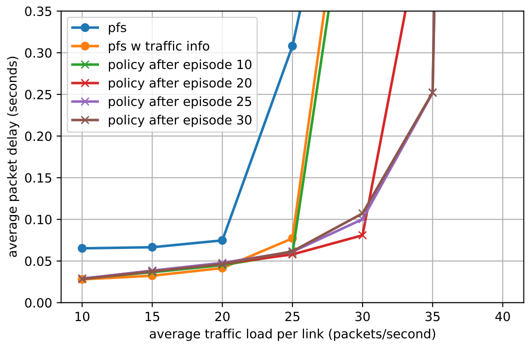
For training, we set $T_{\max}$ and $E_{\max}$ to 2500 time slots and 10 episodes, respectively. The discount factor is set to 0.8. In this subsection, each testing execution runs for 5000 time slots and we label an execution as unstable, if the number of packets waiting for a queue is above 100 packets.

We start the simulations in this subsection by comparing the average packet delay performance achieved by the proposed traffic-driven resource management scheme (which is indicated as 'policy' in Fig. 5.5) as training proceeds from episode to episode. In Fig. 5.5a, we assume a single subband and a single user per base station scenario. We start to train the policy on a $(N = 5 \text{ users}, K = 5 \text{ cells})$ cell scenario, and after some certain episodes we test the current version of the policy parameters on an independent and larger $(N = 20 \text{ users}, K = 20 \text{ cells})$ scenario. For this first experiment, pfs and pfs with traffic information becomes unstable after an average traffic load per link value of 15 and 25 packets/second. We start to test the policy at the beginning of episodic training, i.e., episode 0, which employs the most simple approach where each agent just picks a random transmit power when it has a non-empty queue, and a zero-transmit power level when its queue is empty. At episode 0, the policy becomes unstable after 20 packets/second. Considering packet size being 500 Kbits and total bandwidth being, this level is attainable by maintaining an average spectral efficiency of 1 bit/second/Hz. It is important to note that pfs can not maintain even this level of average spectral efficiency for all links and become unstable much sooner than 20 packets/second for this experiment because of at least one link with a significantly worse channel conditions than its neighboring links. Next, after multiple episodes, we observe that the delay performance of the policy is consistently improved through the training. The resulting policy after 50 episodes becomes unstable after 40 packets/second which shows that the traffic-driven approach can carry almost the double traffic of what policy with traffic information can handle for this scenario. Furthermore, we also carry out another similar experiment for a multiple links per cell scenario to show that the auction scheme shown in Fig. 5.2 works as expected. In Fig. 5.5b, the proposed policy can carry more than 10 packets/second on average than the pfs with traffic information before becoming unstable.

We further show the average delay performance of a pre-trained policy on various total number of subbands configurations in Fig. 5.6. The policy can carry the double the amount of policy with traffic information for all subband configurations.



(a) Policy is trained on $N = 5$ users on $K = 5$ cells and is tested on a larger deployment with $N = 20$ users on $K = 20$ cells.



(b) Policy is trained and tested on $N = 10$ users on $K = 5$ cells.

Figure 5.5. Testing the policy along the episodic training. The number of subbands $M = 1$.

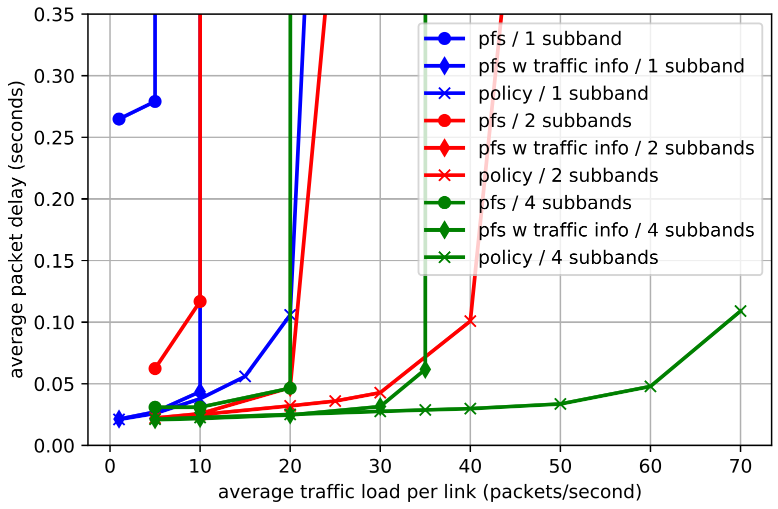


Figure 5.6. Test a converged policy on a ($N = 20$ users, $K = 10$ cells) scenario for total number of subbands $M \in \{1, 2, 4\}$.

After the proof-of-concept experiments that we have shown above, we extend the simulations over multiple testing seeds. In Fig. 5.7, we draw the confidence regions representing average packet delays for the three resource allocation schemes for the ($N =

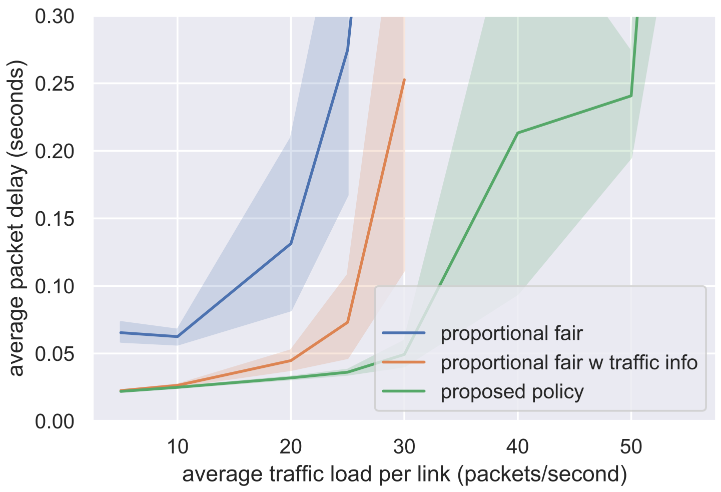
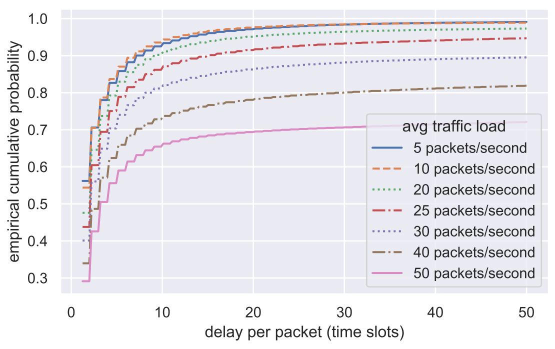


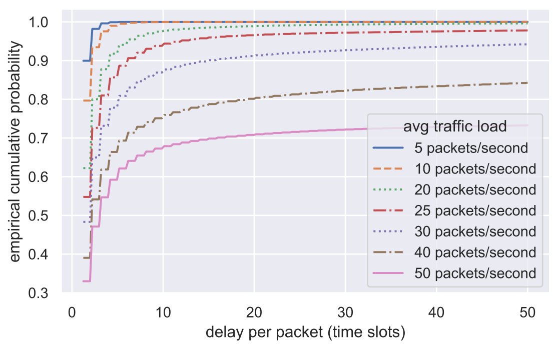
Figure 5.7. Testing a pre-trained policy on 10 different testing seeds. $N =$ 20 links, $K = 10$ cells, $M = 2$ subbands.

20 users, $K = 10$ cells, $M = 2$ subbands) scenario. These regions show that a pre-trained policy is able to outperform pfs with traffic information scenario by some margin. Furthermore, we also examine the empirical Cumulative Distribution Function (eCDF) of all packet delays and the average delay that user observes (user delay) in Figs. 5.8 and 5.9, respectively. Figs. 5.8 and 5.9 clearly depict the significant packet delay difference between pfs with and without traffic information. Additionally, we see that average packet delay and average user delay is signifanctly better for the traffic-driven approach. Below 25 packets/second load, the proposed policy ensures that about 95% of packets and users experience a delay that is less than 5 time slots and 0.1 seconds, respectively, whereas 95-th percentile values for the pfs with traffic information are about 10 time slots and 0.2 seconds.

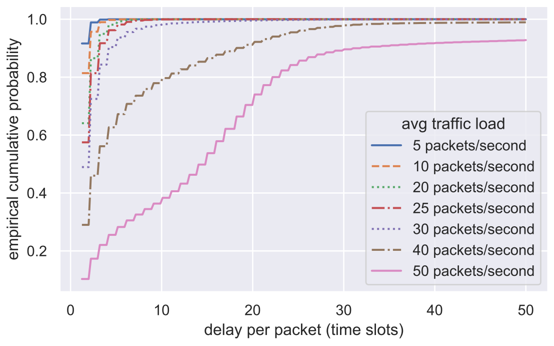
As we increase the average traffic load, this performance gap between the proposed policy and pfs with traffic information becomes even higher. For example, for the 30



(a) proportional fair



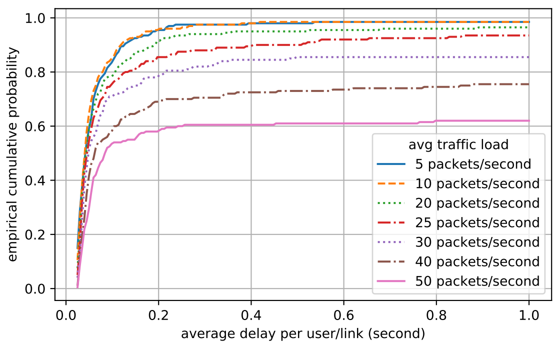
(b) proportional fair with traffic info



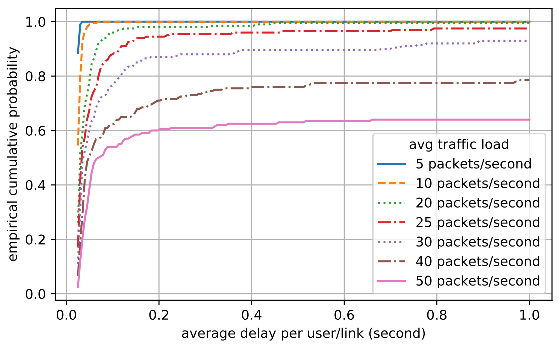
(c) proposed policy

Figure 5.8. Empirical CDF (up to 50 time slots) of all packet delays for testing on $N = 20$ links, $K = 10$ cells, $M = 2$ subbands.

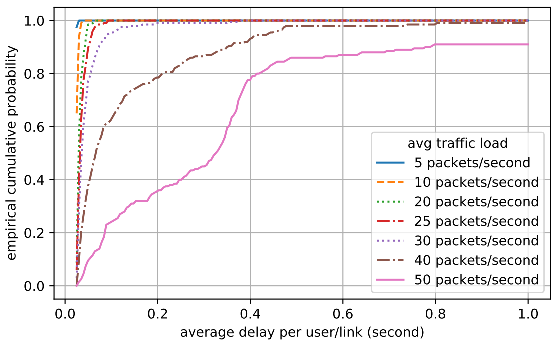
packets/second load, the proposed policy ensures that almost all packets are transmitted within 15 time slots. On the other hand, almost 10% of the packets have a delay value that is higher than 50 time slots which essentially means that those packets will be dropped. This situation is caused by the constant accumulation of packets and instability of the pfs after a certain average traffic load that is significantly less than what the proposed policy can handle without having unstable queues. After 50 packets/second, the proposed policy also observes instability and fails to pass almost 10% of the packets. Meanwhile, for both pfs scenarios the ratio of the dropped packets becomes almost 30% for 50 oackets/second.



(a) proportional fair



(b) proportional fair with traffic info



(c) proposed policy

Figure 5.9. Empirical CDF (up to 1 second) of average delay that user observes for testing on $N = 20$ links, $K = 10$ cells, $M = 2$ subbands.

Note that as traffic load increases and queues tend to remain unempty, pfs with traffic information actually becomes indifferent to the regular pfs scenario.

5.6. Conclusion

In this chapter, we have developed a traffic-driven resource management scheme that effectively stabilizes queue lengths and minimizes average packet delay. We compared the proposed traffic-driven approach with WMMSE algorithm that maximizes a weighted sum-rate objective with weights are particularized to achieve proportionally fair scheduling. In our comparisons, we have slightly enhanced proportional fairness by setting weights to zero if link's queue is empty to avoid unnecessary utilization of the channel in case of no traffic. Besides this modification, proportional fairness does not use any traffic information. The main goal of the comparison between proportional fairness and the proposed traffic-driven approach was to observe how much additional traffic can be passed through the network without causing any instability with integrating queue length information to the resource management decisions. In certain scenarios, the traffic-driven approach can pass twice amount of traffic load what proportional fairness can manage without suffering from queue instability.

CHAPTER 6

Conclusion and Future work

We have developed novel deep reinforcement learning schemes on various wireless network resource management problems that involves power control and spectrum allocation.

Until Chapter 5, we assumed full-buffer and all transmitters had something to transmit at the beginning of each time slot. In Chapter 5, we have developed a reinforcement learning based solution for the dynamic resource management problem in a cellular wireless network over time-varying multi-channel and traffic conditions. The fundamental resource management problem aims to stabilize all queue lengths and maximize users' long-term quality of service. As described in Chapter 5, this problem resembles a Markov decision process where the goal is to maximize a long-term utility function. This kind of Markov decision process problems are effectively solvable by reinforcement learning approaches. We think that reinforcement learning based solutions are essential for future generation wireless networks. Our study shows that a control policy parameterized as a neural network which is trained by a deep reinforcement learning algorithm is able to effectively turn channel and traffic conditions into resource allocation decisions. As of now, the decision parameters are related to subband selection and power control. However, we believe that with the same approach to the fundamental problem, the results can be extended to deciding on other wireless network resource management control parameters such as user association and multiple-input multiple-output (MIMO) beamforming.

For first four chapters, when the instantaneous global CSI is available, the sum-rate maximization problem becomes deterministic, and its solution depends only on the current channel gains. Although we have limited delayed local information and varying weights in the proportional fairness scheme, the underlying Markov decision process is still limited and not well-explained. This causes a lack of motivation to apply reinforcement learning in the full-buffer traffic setup. Introducing a traffic model in Chapter 5, and as a result queues, makes the underlying Markov decision process more clear and much easier to explain along with the fundamental radio resource management problem. The deep reinforcement learning makes better sense of the discount factor to maximize a long-term objective. In a system model that involves both time-varying channel and queue lengths, we have exploited the full potential of deep reinforcement learning.

The current repositories for Chapters 3 and 4 are well-documented, seeded, and published for easy reproducibility [**64**, **65**].

In addition to the proposed future work above, we have some side future-work problems in mind. First, we are looking into better and easily tunable training and exploration schemes to better adapt to the environment non-stationarity of the multi-agent setting. The multi-agent learning scheme violates the stationary environment assumption of common reinforcement learning algorithms. This causes instability which is often handled by fine-tuning of exploration and learning hyper-parameters and assumptions like using global policy parameters across all agents. We will investigate this problem and try to improve our multi-agent framework. Secondly, even though we significantly simplified the state set design in Chapter 5 by considering aggregated interference levels instead of direct interfering channel gains, it is still rather intuitive than systematic. We think that this can be done in a systematic manner as well by analyzing the hidden-layer weights of a trained policy. Suppose we assume that the global CSI is available at each learning agent, we can train a policy that takes the full CSI as its input. Later, we can analyze the trained policy and look deep into the trained network weights at the hidden-layers. To pick the features to be used in the state set design, we can analyze which state inputs are associated with higher network weights and analyze which environment features impact the decision strategy most strongly.

References

[1] M. Chiang, P. Hande, T. Lan, and C. W. Tan, "Power control in wireless cellular networks," *Foundations and Trends in Networking*, vol. 2, no. 4, pp. 381–533, 2007.

[2] Z. Q. Luo and S. Zhang, "Dynamic spectrum management: Complexity and duality," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 1, pp. 57–73, Feb 2008.

[3] 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 Transactions on Signal Processing*, vol. 59, no. 9, pp. 4331–4340, Sept 2011.

[4] K. Shen and W. Yu, "Fractional programming for communication systems—part I: Power control and beamforming," *IEEE Transactions on Signal Processing*, vol. 66, no. 10, pp. 2616–2630, May 2018.

[5] I. Sohn, "Distributed downlink power control by message-passing for very large-scale networks," *International Journal of Distributed Sensor Networks*, vol. 11, no. 8, p. e902838, 2015.

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

[7] S. G. Kiani, G. E. Oien, and D. Gesbert, "Maximizing multicell capacity using distributed power allocation and scheduling," in *2007 IEEE Wireless Communications and Networking Conference*, March 2007, pp. 1690–1694.

[8] 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 Journal on Selected Areas in Communications*, vol. 29, no. 6, pp. 1214–1224, June 2011.

[9] Y. S. Nasir and D. Guo, "Deep reinforcement learning for distributed dynamic power allocation in wireless networks," *arXiv e-prints*, p. arXiv:1808.00490v1, Aug. 2018.

[10] 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 Transactions on Signal Processing*, vol. 66, no. 20, pp. 5438–5453, Oct 2018.

[11] W. Cui, K. Shen, and W. Yu, "Spatial deep learning for wireless scheduling," *IEEE Journal on Selected Areas in Communications*, pp. 1–1, 2019.

[12] F. Liang, C. Shen, W. Yu, and F. Wu, "Towards optimal power control via ensembling deep neural networks," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1760–1776, 2019.

[13] M. Eisen and A. Ribeiro, "Optimal wireless resource allocation with random edge graph neural networks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2977–2991, 2020.

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

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

[16] 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 Communications Letters*, vol. 6, no. 4, pp. 458–461, Aug 2017.

[17] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys Tutorials*, 2019, to be published. [Online]. Available: https://ieeexplore.ieee.org/document/8714026.

[18] H. Ye, G. Y. Li, and B. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, April 2019.

[19] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1277–1290, June 2019.

[20] R. Li, Z. Zhao, Q. Sun, C. I, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018.

[21] M. Bennis and D. Niyato, "A Q-learning based approach to interference avoidance in self-organized femtocell networks," in *2010 IEEE Globecom Workshops*, Dec 2010, pp. 706–710.

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

[23] 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 Transactions on Wireless Communications*, 2019, to be published. [Online]. Available: https://ieeexplore.ieee.org/document/8731967.

[24] E. Ghadimi, F. D. Calabrese, G. Peters, and P. Soldati, "A reinforcement learning approach to power control and rate adaptation in cellular networks," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–7.

[25] 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 Communications Magazine*, vol. 56, no. 9, pp. 138–145, Sep. 2018.

[26] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv e-prints*, p. arXiv:1509.02971, Sep. 2015.

[27] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," *CoRR*, vol. abs/1901.07159, 2019. [Online]. Available: http://arxiv.org/abs/1901.07159

[28] J. Tan, Y. C. Liang, L. Zhang, and G. Feng, "Deep reinforcement learning for joint channel selection and power control in D2D networks," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2020.

[29] Z. Qin, H. Ye, G. Y. Li, and B. F. Juang, "Deep learning in physical layer communications," *IEEE Wireless Communications*, vol. 26, no. 2, pp. 93–99, 2019.

[30] F. Meng, P. Chen, and L. Wu, "Power allocation in multi-user cellular networks with deep Q learning approach," in *2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[31] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6255–6267, 2020.

[32] Y. S. Nasir and D. Guo, "Deep actor-critic learning for distributed power control in wireless mobile networks," in *the 54th Asilomar Conference on Signals, Systems, and Computers*, 2020, pp. 398–402.

[33] Z. Zhou and D. Guo, "A centralized metropolitan-scale radio resource management scheme," *arXiv:1808.02582*, 2018.

[34] D. N. C. Tse and P. Viswanath, *Fundamentals of Wireless Communication.* Cambridge University Press, 2005.

[35] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

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

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

[38] A. Galindo-Serrano and L. Giupponi, "Distributed Q-Learning for interference control in OFDMA-Based femtocell networks," in *2010 IEEE 71st Vehicular Technology Conference*, May 2010, pp. 1–5.

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

[40] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436 EP –, May 2015.

[41] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Is multiagent deep reinforcement learning the answer or the question? A brief survey," *CoRR*, vol. abs/1810.05587, 2018. [Online]. Available: http://arxiv.org/abs/1810.05587

[42] J. Hu and M. P. Wellman, "Online learning about other agents in a dynamic multiagent system," in *International Conference on Autonomous Agents: Proceedings of the second international conference on Autonomous agents*, vol. 10, no. 13. Citeseer, 1998, pp. 239–246.

[43] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson, “Stabilising experience replay for deep multi-agent reinforcement learning,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1146–1155.

[44] L. Panait and S. Luke, “Cooperative multi-agent learning: The state of the art,” *Autonomous agents and multi-agent systems*, vol. 11, no. 3, pp. 387–434, 2005.

[45] M. L. Littman, “Value-function reinforcement learning in markov games,” *Cognitive Systems Research*, vol. 2, no. 1, pp. 55–66, 2001.

[46] L. Matignon, G. J. Laurent, and N. L. Fort-Piat, “Hysteretic q-learning : an algorithm for decentralized reinforcement learning in cooperative multi-agent teams,” in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2007, pp. 64–69.

[47] M. Tan, “Multi-agent reinforcement learning: Independent vs. cooperative agents,” in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[48] Y. Shoham and K. Leyton-Brown, *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[49] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, “Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems,” *The Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, 2012.

[50] A. Tampuu *et al.*, “Multiagent cooperation and competition with deep reinforcement learning,” *PloS one*, vol. 12, no. 4, p. e0172395, 2017.

[51] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, “Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications,” *IEEE Transactions on Cybernetics*, pp. 1–14, 2020.

[52] J. K. Gupta, M. Egorov, and M. Kochenderfer, “Cooperative multi-agent control using deep reinforcement learning,” in *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 66–83.

[53] J. Watt, R. Borhani, and A. K. Katsaggelos, *Machine learning refined: foundations, algorithms, and applications*. Cambridge University Press, 2016.

[54] “Radio Frequency (RF) system scenarios,” 3GPP TR 25.942 v.14.0.0, available at http://www.3gpp.org.

[55] B. Zhuang, D. Guo, and M. L. Honig, "Energy-efficient cell activation, user association, and spectrum allocation in heterogeneous networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 4, pp. 823–831, April 2016.

[56] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[57] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv e-prints*, p. arXiv:1609.04747, Sep 2016.

[58] V. François-Lavet, R. Fonteneau, and D. Ernst, "How to discount deep reinforcement learning: Towards new dynamic strategies," in *NIPS 2015 Workshop on Deep Reinforcement Learning*, 2015.

[59] "Study on channel model for frequencies from 0.5 to 100 GHz," 3GPP TR 38.901 v.14.0.0, available at http://www.etsi.org.

[60] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2239–2250, 2019.

[61] Z. J. Haas, "A new routing protocol for the reconfigurable wireless networks," in *Proceedings of ICUPC 97 - 6th International Conference on Universal Personal Communications*, vol. 2, Oct 1997, pp. 562–566.

[62] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[63] J. Achiam, "Spinning up in deep reinforcement learning," https://spinningup.openai.com, 2018.

[64] Y. S. Nasir and D. Guo, "TensorFlow code for deep actor-critic learning for distributed power control in wireless mobile networks," https://github.com/sinannasir/Power-Control-asilomar, 2020.

[65] ——, "TensorFlow code for deep reinforcement learning for joint spectrum and power allocation in cellular networks," https://github.com/sinannasir/Spectrum-Power-Allocation, 2020.

[66] Z. Lu and M. C. Gursoy, “Dynamic channel access and power control via deep reinforcement learning,” in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 2019, pp. 1–5.

[67] Y. S. Nasir and D. Guo, “Deep reinforcement learning for joint spectrum and power allocation in cellular networks,” in *2021 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2021.

[68] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA, USA: MIT press, 2018.