2022

Reinforcement Learning Approaches to Improve Spatial Reuse in Wireless Local Area Networks

Yiwei Huang

Reinforcement Learning Approaches to Improve Spatial Reuse in Wireless Local Area Networks

A thesis submitted in partial fulfilment of the requirements for the award of the degree

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Yiwei Huang

Bachelor of Engineering (Telecommunications)

Master of Data Science

School of Electrical, Computer and Telecommunications Engineering

August 2022

Statement of Originality

I, Yiwei Huang, declare that this thesis, submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institutions.

Signed

Yiwei Huang

August 2022

Abstract

The ubiquitous deployment of IEEE 802.11 based Wireless Local Area Networks (WLANs) or WiFi networks has resulted in dense deployments of Access Points (APs) in an effort to provide wireless links with high data rates to users. This, however, causes APs and users/stations to experience a higher interference level. This is because of the limited spectrum in which WiFi networks operate, resulting in multiple APs operating on the same channel. This in turn affects the signal-to-noise-plus interference ratio (SINR) at APs and users, leading to low data rates that limit their quality of service (QoS).

To improve QoS, interference management is critical. To this end, a key metric of interest is spatial reuse. A high spatial reuse means multiple transmissions are able to transmit concurrently, which leads to a high network capacity. One approach to optimize spatial reuse is by tuning the clear channel access (CCA) threshold employed by the carrier sense multiple access with collision avoidance (CSMA/CA) medium access control (MAC) protocol. Specifically, the CCA threshold of a node determines whether it is allowed to transmit after sensing the channel. A node may increase its CCA threshold, causing it to transmit even when there are other ongoing transmissions. Another parameter to be tuned is transmit power. This helps a transmitting node lower its interference to neighboring cells, and thus allows nodes in these neighboring cells to transmit as well. Apart from that, channel bonding can

be applied to improve transmission rate. In particular, by combining/aggregating multiple channels together, the resulting channel has a proportionally higher data rate than the case without channel bonding. However, the issue of spatial reuse remains the same whereby the focus is to maximize the number of concurrent transmissions across multiple channels.

Dynamic CCA threshold, transmit power control and channel bonding raise several issues. First, although adjusting the CCA threshold of APs may improve spatial reuse, it may also increase the amount of interference; i.e., this is because there are more concurrent transmissions. Secondly, although transmit power control helps reduce interference, it may also result in low received signal strength at receivers and thereby, causes a low data rate or excessive interference that leads to a decoding failure. Lastly, an AP that uses a bonded channel may suffer/generate an increasing level of interference from/to neighboring APs. Therefore, the use of CCA threshold adjustment, transmit power control and channel bonding needs to be carefully considered in order to balance the trade-off between the gain in spatial reuse, equivalently network capacity, and interference, which reduces data rates.

Henceforth, this thesis presents Reinforcement Learning (RL) based approaches to address the said issues. It first studies how to determine the CCA threshold of each device in densely deployed WiFi networks. This thesis presents a Markov Decision Process (MDP) formulation. Then, it outlines a Deep Q-Network (DQN) based approach that runs on each AP independently, where each AP adjusts the CCA threshold of each associated device. The proposed approach relies only on historical interference information at each device.

This thesis then presents an RL approach that jointly optimizes the CCA threshold and transmit power of an AP with random traffic arrivals. It presents the problem as a Semi-Markov Decision Process (Semi-MDP). Then, it shows how an AP can be equipped with a Hierarchical Reinforcement Learning (HRL) based approach that adopts DQN. This approach then allows an AP to learn the optimal CCA threshold and transmit power for different system states. Advantageously, the

approach learns using only locally observed information, such as interference and current queue length.

Lastly, this thesis outlines a novel approach to optimize the following quantities jointly: CCA threshold, transmit power and bonded channels with random traffic arrivals. Specifically, it considers bonding adjacent and non-adjacent channels. It formulates a three-layer MDP to jointly optimize the said quantities. Further, it outlines a three-tier learning approach that is run by an AP to determine a set of transmission channels, transmit power allocation and CCA threshold on each selected channel. The aim is to train an AP to adapt to interference and random traffic arrivals in order to improve spatial reuse on multiple channels and also to minimize its queue length.

Acknowledgments

First and foremost, words are not enough to express my deepest and sincerest gratitude to my supervisor, Associate Professor Kwan-Wu Chin, for his invaluable patience, guidance and inspiration throughout the entire time as a Ph.D. student. He is the best mentor I have ever met. He has profoundly influenced me to think and express critically and logically, which not only benefits my research but also my future life.

Special thanks go to the School of Electrical, Computer and Telecommunications Engineering, University of Wollongong for providing me with an excellent and extraordinary environment to study and research. Sincere thanks to all my friends, especially Dr. Dawei Gao and Dr. Shuming Seng, for all the encouragement, insightful discussions and joyful time. Wish them all the best in their future work and life.

I am particularly grateful to my girlfriend Dr. Jiaman Li. Her love and belief in me have encouraged and motivated me through every moment during my Ph.D. studies. It would be impossible for me to accomplish this journey without her being by my side. The memory with her at UOW is a treasure of mine, and It's fortunate to start a new page with her for our ongoing story.

Finally, I would like to thank my parents for their sustained love, support, encouragement and inspiration throughout my entire life. They always back me up so

that I can focus on my study. They are my heroes.

Contents

List of Figures

List of Tables

Abbreviations

WLAN	Wireless Local Area Network
AP	Access Point
BS	Base Station
BSS	Basic Service Set
OBSS	Overlapping Basic Service Set
V2V	Vehicle-to-Vehicle
IoT	Internet of Thing
MAC	Medium Access Control
DCF	Distributed Coordination Function
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
OFDMA	Orthogonal Frequency-Division Multiple access
RSSI	Received Signal Strength Indicator
PER	Packet Error Rate
CCA	Clear Channel Assessment
DSC	Dynamic Sensitivity Control
CB	Channel Bonding
TPC	Transmit Power Control
SINR	Signal to Interference plus Noise Ratio
MCS	Modulation Coding Scheme

MDP	Markov Decision Process
PPP	Poisson Point Process
Semi-MDP	Semi-Markov Decision Process
LSTM	Long Short Term Memory
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
HRL	Hierarchical Reinforcement Learning
DQN	Deep Q-Network
DDPG	Deep Deterministic Policy Gradient

Chapter 1

Introduction

1.1 Background

IEEE 802.11 based Wireless Local Area Networks (WLANs), aka WiFi networks, are one of the most popular and essential technologies that permeate almost every aspect of people's daily activities. Fig. 1.1 depicts an example WiFi network, where an Access Point (AP) serves multiple users in a Basic Service Set (BSS) or a cell. The AP is responsible for delivering/collecting data to/from its associated users. To date, WiFi networks are widely deployed in places such as offices, shopping centers, airports and hospitals to provide Internet connectivity to users [7]. As per Cisco's annual report [8], there will be more than 628 million public APs in operation by 2023. As a result, WiFi devices generate a significant amount of traffic, where it is predicted that 51% of the global Internet traffic will traverse WiFi networks by 2022 [9]. This traffic is mainly attributed to the proliferation of emerging applications, such as online streaming, video sharing, Virtual Reality (VR) and 4K resolution videos [10, 11]. These new applications require a network that offers high capacity that meets various Quality of Service (QoS) requirements.

To date, there are a number of approaches and revisions to the IEEE 802.11 standards, aiming to improve the capacity of WiFi networks; see [10, 12, 13]. These

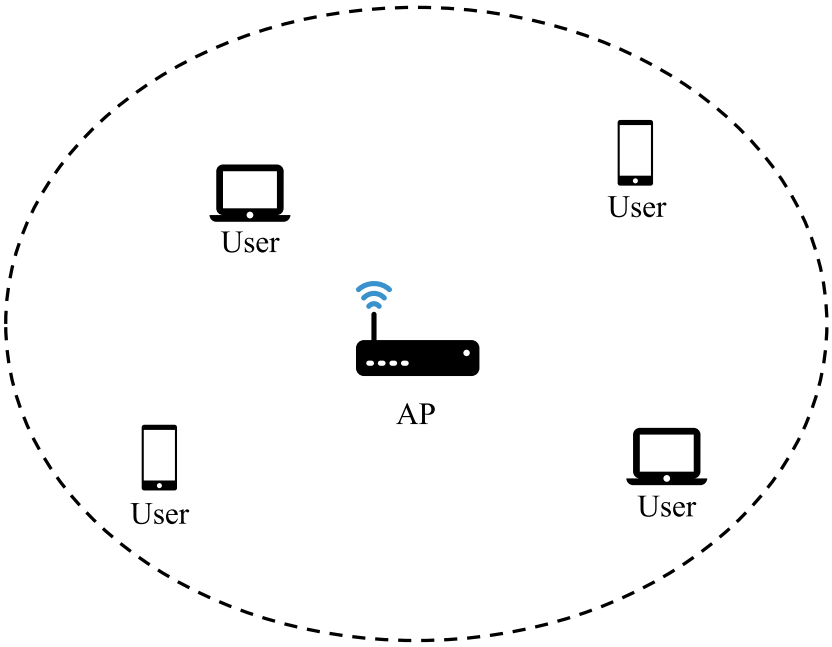

Figure 1.1: An example WiFi network.

revisions aim to increase bandwidth or data rates. To this end, IEEE 802.11ad [14] supports the 60 GHz band and has a wide bandwidth of 2.16 GHz for each channel. In addition, the IEEE has also standardized WiFi for use in spectrum normally occupied by other communication systems. For example, IEEE 802.11af [15] operates in the television white space spectrum, and IEEE 802.11ah [16] uses sub 1 GHz bands. Apart from expanding bandwidth resources, WiFi now also supports channel bonding [17] and channel aggregation [18]. Both of these features allow devices to combine multiple channels together in order to form a wide bandwidth for transmissions. In particular, channel bonding aims to bond multiple adjacent channels, and channel aggregation focuses on non-adjacent channels [17].

Different from improving transmission capacity, a number of works have recently emphasized the need to improve the performance of densely deployed APs, e.g., [10, 12, 19–22]. One reason is that the proliferation of WiFi devices inevitably creates dense WLANs or deployment scenarios [12]. The second reason is the need to support high data rates. Reference [23] notes that the performance of WLANs is limited by the number of users associated with an AP. Therefore, the authors of [23] suggest that future wireless networks should focus on cells with a small number

of users or densely deployed APs. This helps reduce the distance between users and APs [24], which improves the Signal-to-Noise plus Interference Ratio (SINR) of received signals and hence, leads to increased data rates. Apart from that, dense deployment of APs also improves the coverage of wireless signals [24].

A key consideration in densely deployed WiFi networks is to improve spatial reuse [19]. This is because increasing spatial reuse means multiple concurrent transmissions can coexist together [25]. Hence, high spatial reuse improves network capacity [26]. The level of spatial reuse is determined by the adopted Medium Access Control (MAC) scheme. In particular, current WiFi standards use Distributed Coordination Function (DCF) for channel access and interference management [27]. The operating principle of DCF is based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). A device is required to sense the channel before initiating any transmission. Specifically, only when the channel is sensed idle, a device is allowed to transmit. In this respect, each device uses a Clear Channel Assessment (CCA) threshold to determine when a channel is deemed idle. If the sensed energy level of a channel exceeds a predefined CCA threshold, the channel is considered as busy. Otherwise, the channel is idle. To date, the value of the CCA threshold is conservative. For example, it is -82 dBm for a 20 MHz channel [27]. This threshold value is suitable when cells do not overlap as it helps reduce the interference and probability of collisions. However, in dense WiFi networks, this conservative CCA threshold degrades network performance. In particular, as there are a high number of overlapping cells operating on the same channel, a device will receive signals from nearby cells, i.e., interference from neighboring cells. If the CCA threshold is set to a low value, devices will always judge the channel to be busy and defer their transmissions. This problem is known as the exposed node problem [28], which reduces the spatial reuse or network capacity of a wireless network.

Various methods can be used to address the exposed node problem. For example, equipping senders with a directional antenna [29] helps focus transmitted signals towards intended receiver(s), and thereby reducing the interference to other devices.

Apart from that, this thesis emphasizes the use of CCA threshold adjustment and transmit power control on improving spatial reuse [19]. Adjusting the CCA threshold of devices or stations help increase transmission opportunities, especially when they experience interference. Advantageously, with a higher CCA threshold, a transmitter is more likely to transmit even though there may be ongoing transmissions in neighboring cells. In this way, the number of concurrent transmissions can be increased, which leads to an improvement in network throughput [30, 31]. In terms of transmit power control, the primary goal is to avoid causing excessive interference to neighbors. This is because a transmitter's power level determines the interference level to its neighbors [32]. Therefore, a transmitter with an effective transmit power control strategy is able to minimize the interference generated to neighbor cells to improve spatial reuse [21]. As an example, consider the scenario in Fig. 1.2. When AP-2 is transmitting to User-2 with a power of 25 dBm, User-1 has to defer its transmission if it uses a CCA threshold of -82 dBm. This is because the interference or received power from AP-2 is -62 dBm. However, User-1 may transmit if the Signal-to-Interference plus Noise (SINR) ratio at both AP-1 and User-2 exceeds a given threshold. To allow for this, we can increase the CCA threshold of User-1 to -60 dBm or reduce the transmit power of AP-2 to 20 dBm. As a result, even if AP-2 transmits, User-1 will conclude the channel is idle and initiate its transmission to AP-1. This results in two concurrent transmissions; namely, User-1 to AP-1 and AP-2 to User-2, which improves spatial reuse or network capacity.

Spatial reuse further improves network capacity when incorporated with channel bonding [33]. This is because a higher spatial reuse means devices are more likely for a device to access channels, which increases the probability to bond multiple channels. Consider an example as shown in Fig. 1.3, where an AP serves four users. The AP is able to bond up to three channels for transmissions, and all users and the AP experience inter-cell interference on different channels. When the AP has packets to User-1, assume it experiences high interference on Channel-2, and low interference on Channel 1 and 3. In addition, consider User-1 experiences high interference on

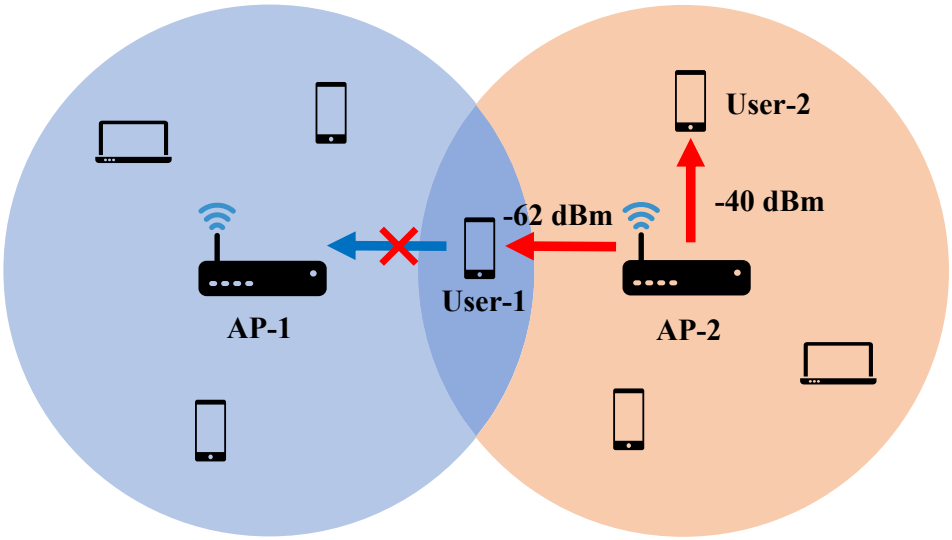

Figure 1.2: An example of limited spatial reuse. The coverage of AP-1 and AP-2 overlaps each other. User-2 is associated with AP-2. User-1 is associated with AP-1. When AP-2 transmits with a power of 25 dBm, the received power or interference at User-1 is -62 dBm.

Channel-3, and low interference on Channel-1 and Channel-2. In this regard, the AP can choose to bond Channel-1 and Channel-3, and increase the CCA threshold on these channels to gain more opportunities to transmit. Moreover, the AP can allocate a higher transmit power on Channel-3 than on Channel-1 if the Signal to Interference plus Noise Ratio (SINR) on Channel-1 exceeds a given threshold. This helps to ensure the SINR on both channels exceeds a given value, and reduce the interference to neighboring cells. Consequently, the AP is able to transmit packets to User-1 with a high data rate.

1.2 Problem space and Motivation

The use of CCA threshold adjustment, transmit power control and channel bonding raises several problems. First, as noted above, an AP can increase its CCA threshold to gain more transmission opportunities. However, doing so may cause excessive interference to neighboring cells as the AP will aggressively use the channel. This

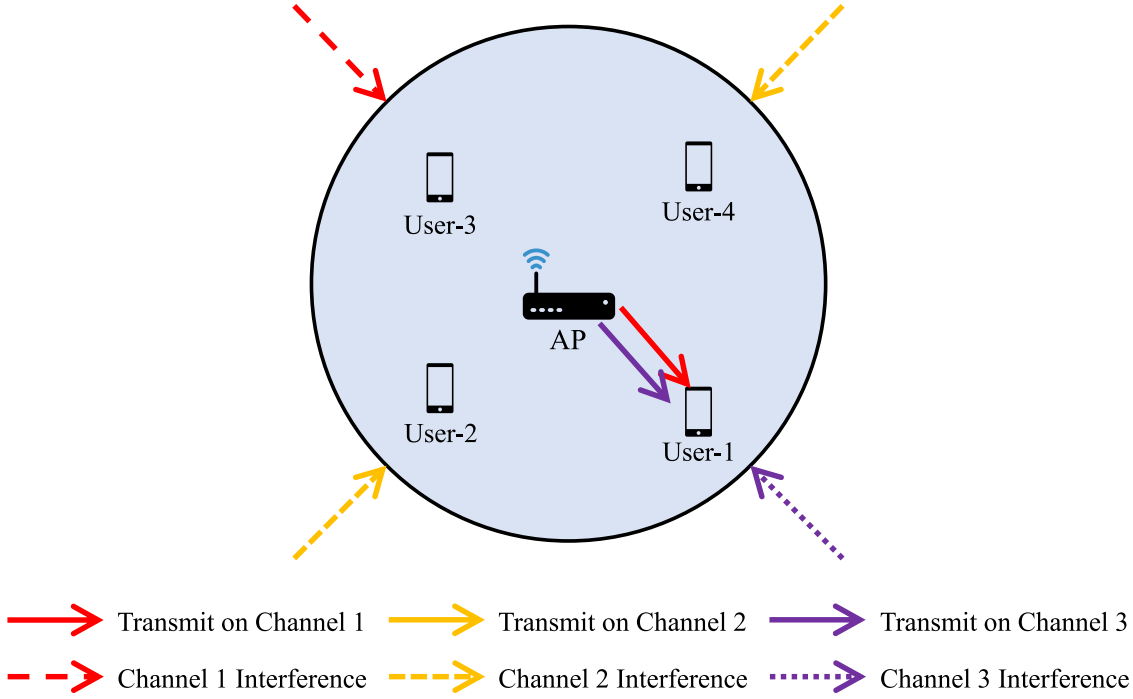

Figure 1.3: An example WLAN where an AP serves four users. The AP is able to bond up to three channels and use different transmit power on each channel. Both AP and users experience interference from neighboring cells on each channel.

will degrade the capacity of neighboring cells.

Secondly, although transmit power control helps decrease interference, it may result in low received signal strength at intended receiver(s). As a result, an AP is forced to use a robust Modulation Coding Scheme (MCS) that has a low data rate.

Third, although an AP using a bonded channel is able to transmit with a high data rate, it may also suffer a significant amount of interference from neighboring cells [34]. This is because the resulting wider bandwidth is more likely to overlap with channels used by these cells. In addition, an AP using a wider channel leads to a lower power density [35], which may result in low SINR at users. Further, devices in neighboring cells may not hear its transmission on some channels and thereby start to transmit [36]. This may cause severe interference to ongoing transmissions or even collisions.

Lastly, there are a number of factors when optimizing CCA threshold, transmit power and channel bonding in WiFi networks. First, the channel condition to each

AP/user is random and time-varying. This means the channel gain for both transmission and interference varies over time. Secondly, the transmissions and assigned channels of devices in neighboring cells are also time-varying, which means the presence of interference on a given channel changes over time. Thirdly, the amount traffic at each AP is random, which means an AP needs to adapt its channels, CCA threshold and transmit power to ensure it has sufficient capacity to deliver all queued packets. Otherwise, it may lead to excessive delays or packet loss.

Consequently, the use of channel bonding, transmit power control and CCA threshold adjustment of an AP needs to be carefully optimized in a manner that balances spatial reuse and interference. As discussed in Chapter 2, prior works related to CCA threshold adjustment and transmit power control mainly focus on heuristic methods or fixed rules. Most of them assume a fixed relationship between the CCA threshold and transmit power of a device, e.g., inversely proportional or linear. By determining a CCA threshold/transmit power, a corresponding transmit power/CCA threshold is obtained based on the said relationship. They do not consider allowing a device to *learn* and adapt to its surrounding environment. Moreover, they do not consider random traffic load. As for channel bonding, most prior works focus on centralized methods or cooperation among APs and the aim is to avoid or minimize interference. In addition, they consider bonding adjacent or partially overlapping channels.

Henceforth, this thesis aims to contribute by applying machine learning based solutions to optimize CCA threshold, transmit power or channel bonding in order to improve spatial reuse in WiFi networks. In this respect, this thesis studies the following research questions:

- How to effectively adjust the CCA threshold of devices? A modified CCA threshold has a direct impact on network performance. Specifically, an incorrect CCA threshold leads to severe network performance degradation [30]. Therefore, how to precisely adjust CCA threshold to achieve network performance improvement is a critical problem. The main challenge is the afore-

mentioned random interference and channel gain. Therefore, the first research question of interest in this thesis is to adjust the CCA threshold for both APs and users to effectively improve network capacity.

- How to jointly optimize the CCA threshold and transmit power for an AP with random arrival traffic? Apart from CCA threshold adjustment, effective transmit power control reduces interference to neighbor cells and hence, improves spatial reuse and network capacity [23]. However, transmit power adjustment is a complex problem [32]. In addition, the traffic load of each AP is random and varies over time. Further, considering random interference and channel gains, an AP needs to adaptively determine its CCA threshold and transmit power in order to achieve a high transmission capacity in order to avoid packet loss or queue overflow. A key challenge is that an AP is only aware of its local observed information, e.g., interference, current queue length and arrival traffic, within its managed cell; i.e., it has no such information for neighboring cells that are managed by a different operator.

- How to improve spatial reuse when channel bonding is considered given random arrival traffic? A wide channel provides an increasing data rate for transmission. However, an incorrectly bonded channel may cause an AP to generate/experience an increasing level of interference [37]. This in turn reduces the capacity or throughput of all APs and users that share the bonded channel. A key challenge is the channel condition and interference on each channel is unknown and random. An AP may bond different channels over time, and the transmissions in neighboring cells on each channel are also random. This means the aggregated interference on each channel changes over time. To this end, one approach is to apply CCA threshold adjustment on each channel. An AP adapts the interference on each channel and thereby determines an optimal channel bonding policy. Alternatively, transmit power control helps an AP to allocate different transmit power level on each bonded channel. This helps to

minimize interference, or to increase the SINR on some channels. Lastly, an AP needs to monitor its queue length to determine its capacity by optimizing the following parameters in order to minimize an AP's queue length: CCA threshold, transmit power control and channel bonding policy.

1.3 Contributions

This thesis outlines machine learning based resources allocation algorithms to address the aforementioned problems. Specifically, it proposes to use Reinforcement Learning (RL) [38] to optimize for CCA threshold adjustment, transmit power control or channel bonding. Briefly, RL is a branch of machine learning algorithms that aims to optimize for decision-making problems [38]. To date, RL based applications are widely adopted in wireless networks [39] to optimize for throughput or energy efficiency; these applications will be elaborated later in Chapter 2.3. In RL, a learning entity, aka an agent, learns from its surrounding environment and makes decisions to achieve an objective. Specifically, an agent observes the current *state* and then selects an *action* to obtain a *reward*. Here, a *state* includes quantitative values of a wireless environment, such as interference, channel gain, and experienced delay. An *action* corresponds to the allocation of wireless resources such as transmitting channels or power. The *reward* is a metric to be optimized, e.g., throughput or energy efficiency. The goal for an agent is to determine an action selection policy that maximizes the cumulative long-term *reward*. In addition, by exploiting deep learning [40] or Deep Reinforcement Learning (DRL), an agent is able to provide better performance than conventional RL [41].

This thesis proposes to equip APs with a RL-based solution. This would then allow an AP to learn a CCA threshold, transmit power control or channel bonding policy given random channel conditions, interference and arrival traffic. Critically, the RL-based solution is so called model-free as it relies only on locally measured information, such as interference level and queue length. That is, it does not require

any probability distribution for the said parameters. The details of the proposed solutions are listed below.

1.3.1 Deep Q-Network based spatial reuse optimization

This thesis first addresses the problem of adjusting the CCA threshold of each device in densely deployed WiFi networks, where each device experiences interference from neighboring cells. Each AP is responsible for adjusting the CCA threshold of each associated user, including itself, to maximize spatial reuse. This is in contrast to prior works that use fixed rules to adjust the CCA threshold of nodes, e.g., [42–50]. A key challenge is that the presence of interference and channel conditions change over time. Therefore, this thesis models the CCA threshold adjustment problem as a Markov Decision Process (MDP) [51], and outlines a Deep Q-Network (DQN) based learning approach running on each AP. The proposed learning approach relies on historical interference data measured by both APs and users. Note that an AP is able to obtain the interference data of its associated users using IEEE 802.11k [52]. The approach then learns to adjust the CCA threshold of each device within a cell.

1.3.2 Hierarchical CCA threshold and transmit power optimization

This thesis addresses the problem of jointly optimizing the CCA threshold and transmit power of an AP with DRL. It allows an AP to learn the optimal policy for setting its CCA threshold and transmit power given its queue length and interference level. There are three main challenges. First, the channel condition to devices varies over time. This means the channel gain for data transmission and interference varies over time. Secondly, the transmission in a neighboring cell is also time-varying, which means the resulting interference from neighboring cells changes over time. Lastly, the traffic arrival is random, which means the queue length of an AP also varies over time. To reduce the learning complexity, this thesis outlines a Semi-

Markov Decision Process (MDP) for the problem and solves it using a decentralized deep Hierarchical Reinforcement Learning (HRL) approach. We then evaluate its performance using traffic generated via a Poisson distribution and from a trace file.

1.3.3 Joint spatial reuse and channel bonding optimization

This thesis addresses a novel problem of learning the optimal channel bonding, CCA threshold adjustment and transmit power control policy with DRL. Specifically, we focus on both adjacent and non-adjacent channel bonding in WiFi networks; the non-adjacent channel bonding is now supported in IEEE 802.11ax [18]. The aim is to improve spatial reuse on multiple channels and minimize the queue length of an AP with random arrival traffic. This thesis first formulates the optimization problem as a three-layer MDP. After that, it outlines a three-tier learning approach based on DRL algorithms that runs on an AP. The proposed approach learns the optimal policy for an AP that determines the transmitting channels, transmit power and CCA threshold on each channel over time.

1.4 Publications

The aforementioned contributions have appeared in the following venues:

1. **Y.W Huang**, K-W Chin, "A Deep Q-Network Approach to Optimize Spatial Reuse in WiFi Networks", *IEEE Transactions on Vehicular Technology*, vol. 71, no. 6, pp. 6636 - 6646, June 2022.

2. **Y.W Huang**, K-W Chin, "A Hierarchical Deep Learning Approach for Optimizing CCA Threshold and Transmit Power in WiFi Networks", *IEEE Transactions on Cognitive Communications and Networking*, 2022. *Under Review*

3. **Y.W Huang**, K-W Chin, "A Three-Tier Deep Learning Based Channel Access Method for WiFi Networks", *IEEE Transactions on Machine Learning in Communications and Networking*, 2022. *Under Review*

1.5 Thesis Structure

1. *Chapter 2.* This chapter provides a comprehensive survey on prior works that consider CCA threshold adjustment, channel bonding, transmit power control and applications of RL in wireless networks.

2. *Chapter 3.* This chapter proposes a DQN based approach that adjusts the CCA threshold of devices to improve spatial reuse in WiFi networks.

3. *Chapter 4.* This chapter presents an HRL based approach to jointly optimize the CCA threshold and transmit power of an AP to improve the spatial reuse in order to minimize its queue length.

4. *Chapter 5.* This chapter outlines a three-tier learning approach based on DQN and DDPG to jointly optimize for channel bonding, transmit power and CCA threshold for an AP to minimize its queue length with random traffic.

5. *Chapter 6.* This chapter concludes the thesis with its main contributions. It also outlines potential future research directions.

Chapter 2

Literature Review

Fig. 2.1 shows the taxonomy of past works summarized in this chapter. Section 2.1 first reviews prior works on CCA threshold adjustment. After that, Section 2.2 discusses related works that focus on channel bonding, and Section 2.3 outlines previous works that apply reinforcement learning in wireless networks.

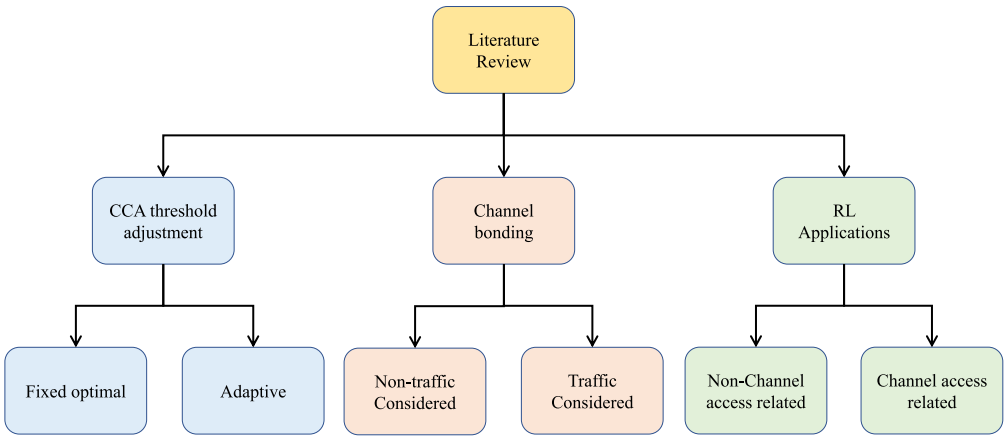

Figure 2.1: Categories of summarized prior works

2.1 CCA threshold adjustment

CCA threshold plays an essential role in improving spatial reuse in networks as it determines the availability of a channel for each device. A number of works have considered adjusting the CCA threshold of devices in order to improve the overall network capacity of a WiFi network. A key challenge is to balance the trade-off between improved spatial reuse and interference [26]. In this regard, Section 2.1.1 reviews prior works that aim to optimize a fixed and optimal CCA threshold for each device. Section 2.1.2 discusses works that adapt the CCA threshold of clients/stations.

2.1.1 Fixed CCA Threshold

The default CCA threshold, i.e., -82 dBm for a 20 MHz channel, used in the current WiFi standard is conservative, which reduces the network performance in a dense deployment environment [13]. The authors in [30] and [53] prove that varying the CCA threshold of devices has an impact on the overall network throughput. They consider optimizing the CCA threshold for APs by iterating through all possible values and selecting the one that provides the highest aggregated throughput. The authors of [54] focus on multi-hop mesh networks, and argue that for each station, its CCA threshold should be set to provide a carrier sensing range that covers its interference range, where the interference range is calculated based on a certain SINR threshold. Reference [55] derives an analytical model for optimizing the CCA threshold for all devices in a wireless ad-hoc network. The model is developed based on a Markov chain with four states, which is then used to determine the aggregated network throughput. The authors propose to iterate through all possible CCA thresholds and select the one that yields the highest aggregated throughput.

2.1.2 Dynamic Sensitivity Control (DSC)

Reference [56] raises a number of issues when devices use a fixed CCA threshold. In particular, using a fixed CCA threshold causes unfairness to devices that are located

at the edge of BSSs. These devices may experience high aggregated interference from neighboring BSSs due to concurrent transmissions and thus are not able to transmit. To this end, DSC algorithms that adaptively adjust the CCA threshold of devices is a promising solution to alleviate this problem. These works can be grouped based on the *metric* used to adjust the CCA threshold of devices; namely, Received Signal Strength Indicator (RSSI), Packet Error Rate (PER), and channel status duration. Table 2.1.2 presents a comparison of these works.

2.1.2.1 Received Signal Strength Indicator (RSSI)

RSSI measures the energy level of a received packet [27]. Works that use RSSI to adjust the CCA threshold nodes can be classified as users or APs oriented.

For users oriented works, the work in [42] and [43] considers adjusting CCA threshold in a distributed manner. Each user uses a *margin* value to determine its CCA threshold. Specifically, in [42], a user records the average RSSI received from its associated AP. On the other hand, a user in [43] only records the lowest detectable RSSI. At the end of each predefined interval, each user subtracts the *margin* value from the recorded RSSI; this is to ensure an adequate SINR and allow for fading. The resulting value is then set as the CCA threshold for the following interval. The authors in [57] outline an extension, where they use deep learning to select the *margin* value. The aim is to reduce interference.

In contrast to focusing only on users, the work in [44, 58] and [59] considers adjusting the CCA threshold for APs. In reference [44], each AP records the lowest RSSI received from its associated users as well as the highest RSSI received from neighboring BSSs. At the end of a predefined interval, each AP subtracts a *margin* value from the lowest recorded RSSI, and sets the result as the CCA threshold. In [58] and [59], a centralized controller first collects all RSSI information recorded by APs and users. Then, the centralized controller in [58] applies a heuristic search for each AP such that each AP will use a different CCA threshold when initiating transmissions to different users. On the other hand, in [59], for each AP, the centralized

Table 2.1: Works that adaptively adjust CCA threshold

Paper	Consideration	Adjust entity	Aim	Approach
[42]	RSSI	User	Maximize aggregated throughput	Select a CCA threshold based on a margin value
[43]		User	Maximize average throughput of users	Select a CCA threshold based on a margin value
[57]		AP/User	Reduce interference and improve fairness	Use deep learning to find the optimal margin value
[44]		AP	Maximize aggregated throughput	Select a CCA threshold based on a margin value
[58]		AP	Maximize aggregated throughput	A centralized controller applies a heuristic search of CCA threshold
[59]		AP	Maximize aggregated throughput	A centralized controller adjusts the CCA threshold with pre-defined step size
[45, 60]	PER	AP/User	Maximize aggregated throughput	Adjust CCA threshold if PER changes
[46]		AP	Reduce interference and maximize aggregated throughput	Adjust CCA threshold based on pre-defined PER threshold
[47]		User	Improve fairness and maximize average throughput	Adjust CCA threshold based on pre-defined PER threshold
[61, 62]		AP/User	Maximize aggregated throughput	A centralized controller adjusts CCA threshold based on highest PER
[63]		User	Minimize interference and maximize successful transmissions	Select CCA threshold based on a heuristic method
[50, 64]	Channel occupancy and PER	User	Ensure QoS of prioritized services	Adjust CCA threshold based on heuristic method using channel occupancy information
[48, 49]	Channel occupancy	User	Minimize interference to neighboring cells	Select the CCA threshold to minimize a cost function that measures interference to neighboring cells
[65, 66]		User	Minimize number of hidden and exposed devices	Select CCA threshold based on channel occupancy information shared by AP
[67]	PER	AP	Maximize aggregated throughput and improve fairness	Set the CCA threshold first and then transmit power based on an inverse relationship
[68]	RSSI	AP	Maximize the number of successful transmissions	Set the CCA threshold first and then transmit power based on an inverse relationship
[69, 70]	PER/RSSI	User	Maximize average throughput of users	Set transmit power first and then CCA threshold based on an inverse relationship
[71]	SINR	User	Minimize interference and ensure a minimum data rate	Select CCA threshold, transmit power and data rate based on an SINR threshold
[72]	Interference	AP	Reduce collision and maximize aggregated throughput	Construct a interferer table and set CCA threshold and transmit power accordingly
[73]	CCA threshold and transmit power	AP/User	Maximize aggregated throughput	Use deep learning to learn the optimal transmit power and CCA threshold

controller calculates the expected worst SINR to each associated user. Then the AP is configured to update its CCA threshold based on the lowest SINR.

2.1.2.2 Packet Error Rate

Packet error rate (PER) measures the number of failed transmissions and the number of total attempted transmissions within a predefined interval. A high PER means devices are experiencing high interference or collision probability. Existing works use historical PER with pre-defined rules, e.g., using PER threshold or comparing PER of adjacent time slots, to determine a CCA threshold for future intervals.

The work in [60] and [45] focuses on the PER of each two adjacent intervals. At the end of a given interval, each user adjusts its CCA threshold if the PER of the current interval and previous interval differs. In particular, if the current PER is higher than the previous PER, a user reduces its CCA threshold with a fixed step size, and vice-versa. To prevent the CCA threshold from being adjusted frequently, the authors in [45] adjust the CCA threshold of users only if the difference in PER exceeds a pre-defined threshold.

References [46] and [47] consider setting a PER threshold for each AP and user, respectively. If the PER over an interval exceeds a predefined threshold, a device reduces its CCA threshold by a fixed step size, and vice-versa. In addition, the work in [47] further considers a *fairness* ratio for each user that is calculated based on the number of neighboring users. Then, each user applies this *fairness* ratio to determine whether to adjust its CCA threshold.

Lastly, some works consider adjusting the CCA threshold of devices in a multi-hop mesh network. For example, the authors of [61] and [62] require each device to measure and report its per-link PER information to a centralized controller. The centralized controller will then use the highest PER to determine a CCA threshold for all devices. Apart from that, the authors in [63] propose to use both PER and the current CCA threshold in a distributed manner. They derive a model that each device periodically measures its PER. At the end of a predefined interval, the

authors derive a pricing function and a utility function based on the PER and CCA threshold. The aim of the penalty function is to prevent a user from aggressively increasing its CCA threshold. On the other hand, the utility function uses the PER to indicate the QoS level of each user; the utility function returns a maximized value when the PER reaches a predefined threshold. Each device then sets its own CCA threshold that minimizes the penalty function whilst maximizing the utility function.

2.1.2.3 Channel Occupancy

The works in this section use channel occupancy information to adjust the CCA threshold of nodes. In particular, channel occupancy information can be characterized by different channel statuses, e.g., *busy* or *idle*. Prior works use the duration of these statues to derive function to evaluate throughput or identify the presence of hidden or exposed devices.

References [48, 49] and [50] share a similar strategy, where they periodically adjust the CCA threshold of each user. Their method is to derive a penalty function for a given CCA threshold that quantifies the influence of a user's transmission to its neighboring users; the function is derived by each user in a distributed manner by observing the channel statuses for a given interval. Each user then selects the CCA threshold that minimizes the penalty function. In addition, the authors in [49] extended their algorithm in [50] and [64] to consider frame error rate in order to ensure a certain QoS level. The algorithm maintains a fixed CCA threshold until the frame error rate reaches a predefined value.

In a different work, reference [74] uses a channel utilization ratio to adjust the CCA threshold for each user, where the channel utilization ratio represents the fraction of period when channel status is *busy*. The authors divide the channel utilization ratio into three categories, namely low, intermediate and high utilization. Each user periodically monitors a channel utilization ratio and adjusts its own CCA threshold accordingly. In particular, a user increases its CCA threshold when the

channel utilization ratio is low, and vice-versa. Further, the authors propose a contention window size adjustment scheme to reduce interference and collisions.

In [65] and [66], the authors notice that the presence of hidden and exposed users leads to different observations of channel occupancy at AP and users. Therefore, they propose to let each AP periodically broadcast its collected channel occupancy information to each associated user. Each user then compares this information with its locally measured channel occupancy information to identify hidden and expose users, and set a CCA threshold that minimizes the total number of hidden and exposed devices.

2.1.3 CCA threshold with transmit power control

Some past works consider jointly optimizing the transmit power and CCA threshold of nodes, e.g., [67–73]. This is because transmit power control helps reduce interference and hence improve spatial reuse [32].

The authors of [67] and [68] assume that there is an inversely proportional relationship between the CCA threshold and transmit power. The work in [67] first selects a CCA threshold by monitoring the PER of an access point. On the other hand, the work in [68] first adjusts the CCA threshold of an access point based on the received signal strength from its associated user. Then, both [67] and [68] derive a transmit power according to the said inversely proportional relationship with CCA threshold.

The work in [69] and [70] investigates whether a user with a high transmit power should use a low CCA threshold, and vice-versa. The approach taken in [69] is to first set the transmit power of users using the number of successful transmissions before assigning them an appropriate CCA threshold. In [71], the authors adjust the transmit power and CCA threshold of users to minimize interference experienced by neighboring cells. Their aim is to balance the trade-off between spatial reuse and data rate. The work in [72] assumes both access points and users are able to identify

the source of interference from neighboring cells. This interference information is then used to construct a CCA threshold to transmit power table that maps a CCA threshold to a transmit power for each interfering source.

The authors in [73] use a neural network in a centralized manner to jointly adjust the CCA threshold and transmit power for each device. Their method requires labeled data that is periodically collected from networks. The input to the neural network consists of the CCA threshold and transmit power of each device, and the output corresponds to the achieved throughput of each device. The authors aim to find the optimal combination of CCA threshold and transmit power for each device after efficient training.

2.2 Channel bonding

The key challenge when using channel bonding is to determine a bonding strategy in a dynamic wireless environment. This is important as the amount of interference that an AP experiences on each channel varies over time. Moreover, an AP that bonds multiple channels generates increasing level of interference to neighboring cells [34]. Further, an AP using a wider channel leads to a lower power density, i.e., Watts/Hz [35], which may cause a low data rate. All of these issues impact the throughput of the wireless networks. Therefore, the level of channel bonding needs to be carefully considered. As shown in Table 2.2, past works that adopt channel bonding can be grouped into two categories, namely (i) works that consider traffic, and (ii) works that *do not* consider traffic. The former mainly aims to improve the network capacity or reduce interference [17], and the latter considers arrival traffic and aims to satisfy the traffic demands of APs or users.

2.2.1 Non-Traffic-Considered channel bonding

A number of prior works use channel bonding to maximize the throughput of APs. These works assume saturated or fixed traffic at APs or users, meaning they always

Table 2.2: Works related to channel bonding in wireless networks

Paper	System	Traffic	Aim	Approach	Consideration	Channels	Method
[75]	WiFi	No	Avoid interference between adjacent cells	Each AP selects users and bonds a set of channels	Packet error rate	Non-adjacent	Heuristic method
[76, 77]	WiFi		Maximize aggregated throughput	Each AP selects a set of channels	SINR	Partially overlapping	Game theory
[36]	WiFi		Maximize the throughput of each AP	Each AP selects a set of channels	Interference on each channel	Adjacent	RL
[78]	Cognitive		Maximize the number of successful transmissions	Each secondary user selects a set of channels to sense and bond	Interference on each channel	Non-adjacent	DQN
[79]	WiFi		Improve fairness and avoid collisions	Selects the CCA threshold of the primary channel for each user	SINR	Adjacent	Heuristic method
[33]	WiFi		Maximize the overall network throughput	Each AP determines the CCA threshold of secondary channels for itself	SINR and distance	Adjacent	Heuristic method
[80]	Femtocell		Avoid interference to macrocell	Each femtocell base station selects a set of channels and transmit power	Interference experienced by macrocell	Non-adjacent	Q-learning and gradient decent method
[81]	WiFi		Avoid collisions in networks	Each AP determines a channel bonding level	Interference and theoretical throughput	Partially overlapping	Multi-armed bandit
[82]	WiFi		Ensure QoS for video traffic and reduce collision	Each AP assigns sub-channels to each user	SNR	Adjacent	Heuristic method
[83]	OFDMA		Maximize the energy efficiency of the overall network	Each base station assigns subcarriers and transmit power for transmissions	Maximum transmit power and minimum required data rate	Non-adjacent	Heuristic method
[84]	WiFi		Maximize the throughput of each AP and reduce collisions	Each AP selects the bandwidth and data rate for transmissions	Distance	Adjacent	Heuristic method
[85]	WiFi		Minimize interference and maximize SINR	Each AP selects a central frequency and bandwidth for transmissions	SNR and interference	Partially overlapping	Heuristic method
[18]	WiFi	Yes	Improve fairness among users	Each user selects a probability to sense each channel to bond	Traffic load on each channel	Non-adjacent	PPO
[86]	WiFi		Mitigate interference and maximize average throughput	A centralized controller determines a set of channels for each AP	SINR, interference and traffic demand of each AP	Partially overlapping	Graph theory and heuristic method
[87]	WiFi		Minimize delay in networks	A centralized controller determines whether to allow channel bonding or not	Traffic load on each channel and delay of each user	Adjacent	Heuristic method
[88]	WiFi		Minimize average delay of all APs	Each AP determines a set of channels for transmissions	Traffic load of each AP	Adjacent	Multi-agent RL
[89]	WiFi		Satisfy the traffic demand of each AP	A centralized controller group APs and assign channels	SINR, interference and collision probability	Adjacent	Heuristic method
[90]	OFDMA		Satisfy the traffic demand of each user	A centralized controller schedules transmissions and assign a set of channels of each AP	Traffic demand of each AP	Non-adjacent	Heuristic method
[91]	WiFi		Assign non-overlapping channels to avoid interference	A centralized controller assigns a set of channels for each AP	Random traffic demand of each AP	Adjacent	Integer linear programming
[92]	WiFi		Reduce interference and improve fairness among APs	A centralized controller assigns a set of channels for each AP	Interference, channel bonding and past traffic load of each AP	Adjacent	Actor-Critic
[93]	WiFi		Improve fairness among APs	A centralized controller assigns non-overlapping channels for APs	Graph theory, traffic load	Adjacent	Integer linear programming
[94]	WiFi		Avoid interference on bonded channels	Predict benefit for using a different bandwidth	Dynamic Channel Point Process	Adjacent	Deep learning

have packets to transmit. Therefore, the primary goal is to maximize the throughput or minimize interference. For example, the work presented in [75] uses the PER on each channel to determine a set of channels to use for each AP. The aim is to maximize the aggregated throughput of all APs. References [76, 77] use game theory to determine a channel bonding strategy for each AP, where each AP is a player. The authors of [76, 77] assume each AP monitors the channels used by its neighboring APs and then determines its channel bonding strategy. The aim is to improve the overall network throughput. The authors in [36] consider the interference on bonded channels. They proposed a *Post-CCA* mechanism, where each AP senses the used channels again after each transmission. This is to identify the interference on each bonded channel. Then the authors use reinforcement learning to select a set of channels for each AP to bond in the following time slot. The goal is to maximize the throughput of each AP. Reference [78] uses reinforcement learning to bond non-adjacent channels for a secondary user in cognitive networks. The state is the status of each channel, and the action is to sense and bond a set of channels. The reward is a binary value that indicates whether the transmission is successful. The aim is to maximize the number of successful transmissions over time.

The work in [33, 79] adjust the CCA threshold of APs first. The authors in [79] assume each user evaluates the signal strength and average level of interference, and reports theses information to its associated AP, with which each AP determines a CCA threshold for the primary channel of each user. After that, the CCA threshold on each secondary channel is obtained by adding a fixed value to the CCA threshold on the primary channel. The work in [33] adjusts the CCA threshold for secondary channels only. The CCA threshold on each secondary channel is the same, and is calculated based on a Signal to Interference plus Noise Ratio (SINR) threshold and the distance between AP and users. Then, the works in [33, 79] bond channels for each AP according to the CCA results. Their aim is to maximize the throughput as well as the fairness of all APs.

Some works aim to use channel bonding to reduce the interference in networks.

For example, The work presented in [80] aims to maximize the throughput of femtocells as well as minimize the interference experienced by macrocell users. The authors assume femtocells have knowledge of the interference levels they generate to both macrocells and neighboring femtocells. Each femtocell uses reinforcement learning to select a set of channels to transmit. After that, it uses a gradient method to allocate transmit power on each selected channel. The authors in [81] first use a Markov chain to model the throughput for bonded channels. Then, they use the multi-armed bandit algorithm to find a channel bonding level. Another work [82] first uses OFDMA to bond a number of channels. Then, it uses a heuristic method to assign a set of sub-channels for each device. The method considers the SINR of each device on each sub-channel, and it aims to reduce collisions. The work [83] also uses OFDMA to assign a set of sub-channels and transmit power for each transmitter. However, the goal is to maximize the energy efficiency of each transmitter. In [84], the authors assume each AP uses the RTS/CTS control packets to measure the distance to an associated user. This distance is then used to determine the channels and data rate used for transmissions. The aim is to maximize the throughput and minimize collisions. The authors in [85] propose that each AP selects a set of channels to minimize a cost function. The cost function is defined as a function of experienced interference on each bonded channel and SINR to each associated user.

2.2.2 Traffic aware channel bonding

Another group of works considers random arrival traffic at each AP when designing a channel bonding strategy. The main challenge is to determine a bandwidth that satisfies the demand of given arrival traffic. Existing works also consider to minimize transmission delays, maximize throughput, avoid interference and ensure APs/users can fairly use channels.

There are works that consider maximizing the throughput of APs/users given random traffic. The work in [18] uses reinforcement learning to select non-adjacent

channels for each user in IEEE 802.11ax networks. The authors assume APs exchange traffic information on each channel, and learn to set a sensing probability on each channel for all associated users. Each user then senses each channel with the said probability, and bond a channel if it is sensed idle. The goal is to maximize the total aggregated throughput of all users. The authors further consider the fairness issue among users, i.e., ensuring the throughput of each user exceeds a pre-defined threshold over time. In a different work [86], the authors jointly consider the SINR, interference and traffic demand of each AP to build a conflict graph. Then, a centralized controller uses the conflict graph and a heuristic algorithm to determine the central frequency and bandwidth used by each AP.

References [87, 88] aim to minimize transmission delay. In [87], the authors derive an analytical model to evaluate transmission delay, where the model jointly considers the traffic load on each channel and current delay experienced by users. They use the analytical model to determine whether to allow users to bond channels. The goal is to satisfy the delay requirement of delay sensitive users. Reference [88] uses multi-agent reinforcement learning [95], where each AP is an agent. Each AP selects a set of channels for transmission by monitoring the arrival traffic, and current queue length in order to minimize the transmission delay of the network.

Some other works focus on satisfying the traffic demand of APs. Reference [89] provides an analytical model that considers the effect of SINR, interference and collision probability. The authors propose a heuristic algorithm that groups APs based on their experienced interference and then determines a set of transmission channels to satisfy their traffic demand. The authors of [90] derive a greedy algorithm that considers the traffic demand of each AP and the interference on each sub-channel. The aim is to schedule the transmission of each AP over a set of non-adjacent sub-channels. The objective of the work in [91] is to satisfy the traffic demand and avoid interference among APs. The authors apply integer linear programming to determine the central frequency and bandwidth used by each AP. Further, the authors aim to minimize the total bandwidth used by all APs.

There are works that use channel bonding to ensure each device is able to *fairly* use the channels in networks, where the term *fairly* means the throughput or transmitting bandwidth of each devices is not less than a given threshold. Reference [92] uses reinforcement learning on a centralized controller to manage all APs. The state consists of channel configuration, traffic arrival rate and interference relationship among all APs. The action is to select a set of channels for each AP, and the reward is calculated based on the experienced interference, achieved throughput and traffic load. In particular, an AP receives high reward if it experienced high interference or high throughput, and the authors use reinforcement learning to minimize the cumulative reward of each AP over time. The aim is to reduce interference and balance the number of channels used by each AP. The authors of [93] build a conflict graph based on the coverage area of APs. Then, the authors use integer linear program to assign non-overlapping channels for each AP according to its traffic demand. The goal is to ensure each AP has a fair share of the channels, i.e., the bandwidth assigned to an AP exceeds a threshold that is calculated based on the AP's traffic load.

Lastly, the work in [94] outlines a Dynamic Channel Point Process (DCPP) based on Generative Adversarial Network. The DCPP captures the effects of channel bonding and interference on the throughput of APs. Then, each AP uses DCPP to predict the activity of neighboring APs and bond channels if there is a throughput gain. The goal is to avoid interference and maximize the throughput of each AP.

2.3 Applications of Reinforcement Learning in wireless networks

To date, a large number of works have applied reinforcement learning in wireless networks [41]. Considering their objectives, they can be grouped into two categories, namely works that *do not* consider channel access and works that consider channel

access. Table 2.3 presents a summary of works that use reinforcement learning in wireless networks.

2.3.1 Non-Channel Access related

A number of prior works implement reinforcement learning to maximize throughput and energy efficiency, or minimize interference. These works assume channels are always available and *do not* consider channel access issues.

Some works aim to minimize interference in some networks. For example, references [96, 98, 99] and [97] focus on adjusting the transmit power of base stations in femtocell networks. The authors use Q-learning, a reinforcement learning algorithm, in a distributed manner, and assume macrocell base stations share interference information with femtocell base stations. The state is a tuple comprising of the transmit power of a femtocell base station and the aggregated interference measured at macro users. The action is to assign a transmit power on each subcarrier. The reward that a femtocell in [96] and [97] receives is high if the macrocell capacity is close to a predefined threshold, and low otherwise. Therefore, the work in [96] and [97] only considers to maintain the capacity of macrocells at a certain level. In contrast, in [98] and [99], the authors consider to preserve capacity for both macrocells and femtocells. Thus, they propose a penalty function to further adjust reward. The reward for a femtocell is low if it achieves a low data rate, and vice-versa. Moreover, the authors of [96, 98, 99] apply transfer learning, where adjacent femtocell base stations share their Q-table to accelerate the learning process.

In [100, 101] and [102], the authors propose to apply Q-learning in cognitive radio networks for transmit power control, where each secondary user runs Q-learning. The state for a secondary user consists of the interference generated to primary users, the distance and current transmit power, and the action is to select a power level. The reward for an agent is the SINR measured at a primary user side. The goal is to ensure the aggregated interference caused by all secondary users is below

Table 2.3: Works apply RL to wireless networks

Paper	System	Channel Access	Aim	Approach	Algorithm
[96, 97]	Femtocell	NO	Minimize interference and maximize capacity of macrocell users	Assign transmit power on a resource block in OFDMA	Q-learning
[98, 99]	Femtocell		Maximize data rate for both femtocell and macrocell users	Assign a user and transmit power on each subcarrier	Q-learning
[100–102]	Cognitive Radio		Minimize interference to primary users	Each secondary user selects a transmit power	Q-learning
[103]	WiFi		Maximize number of successful transmissions	Select data rate, bandwidth, frame aggregation level and guard interval	Multi-armed Bandit
[104]	WiFi		Maximize number of successful transmissions	Configure link parameter based on traffic load	SARSA
[105]	WiFi		Maximize number of successful transmissions	Select data rate for transmissions	Q-learning
[106]	WiFi		Maximize the aggregated downlink throughput and ensure fairness	Users selection and data rate control	Multi-armed Bandit
[107]	Satellite		Maximize energy efficiency and throughput	Select transmit power and data rate	DQN
[108]	Multi-hop		Maximize network energy efficiency and minimize bit error rate	Select probability and transmit power to relay a packet	Q-learning
[109]	WiMAX		Maximize energy efficiency and delay requirement	Select data rate for transmissions	RL
[110]	V2V networks	Yes	Maximize throughput and minimize latency	Select a channel and transmit power	DQN
[111]	IoT		Avoid interference and maximize throughput	Select a channel	DQN
[112]	IoT		Maximize energy efficiency	Select a set of channels, data rate and volume of transmitted data	DQN
[113]	IoT		Maximize aggregated uplink throughput	Select a channel and schedule transmissions for multiple sensors	DQN with LSTM
[114]	Cellular		Maximize downlink throughput	Select channel access probability for each channel	DQN with LSTM
[115]	WiFi		Minimize MAC service time	Each AP selects whether to access channel or not, and transmit power control	Q-learning
[116]	Cognitive Radio		Minimize queue length for primary users	Secondary users select energy detection threshold to access channel	Q-learning
[117]	WiFi		Maximize throughput of each AP	Each AP selects a channel and transmit power	Multi-armed Bandit
[118]	Cognitive Radio		Avoid interference between secondary users	Each secondary predicts channel status of each channel for access	Multi-armed Bandit
[119]	WiFi		Minimize interference between APs and satisfy traffic demand	Select a set of channels to bond	DQN
[120]	WiFi		Maximize energy efficiency of each AP with random traffic	Select a set of channels to bond and assign transmit power	DDPG
[121]	WiFi		Maximize network throughput	Select a CCA threshold for all devices	Q-learning
[122]	WiFi		Reduce collision rate	Set a contention window size	RL
[123]	WiFi		Ensure prioritized traffic has high throughput	Set a contention window size based on traffic type	Q-learning
[124]	WiFi		Maximize network throughput	Each AP sets a contention window for each user	DQN & DDPG
[125]	WiFi		Maximize throughput of each AP	Select a channel, transmit power or CCA threshold to each AP	Multi-armed Bandit

a certain level.

Other works aim to improve the throughput of a transmitter by adjusting wireless transmission parameters such as guard intervals, modulation and coding scheme and level of frame aggregation [103]. The work in [103] and [104] aims to find the optimal set of wireless transmission parameters for each transmitter that provides the highest throughput. The authors use distributed reinforcement learning algorithms on each device to adjust wireless transmission parameters, such as guard intervals, modulation and coding scheme (MCS) and level of frame aggregation, based on observed SINR and PER. The objective is to maximize the throughput of each transmission. Another work presented in [105] uses the contention window size as the state, with which a device selects a data rate for each transmission. The reward is the number of successful transmissions measured within a pre-defined interval.

The authors of [106] apply reinforcement learning to select users for downlink MU-MIMO transmissions as well as optimize the data rate for each link in MU-MIMO transmissions. The authors assume a centralized controller periodically monitors channel state information between each user and its associated AP. Based on collected channel state information, the centralized controller adaptively selects a group of users to initialize downlink MU-MIMO transmissions. Further, the centralized controller determines a data rate for each user to maximize aggregated throughput.

The work in [107] proposes an approach for rate control in a complex satellite communication system, where a transmitter is an agent. The state observed by an agent consists of transmission parameters in a previous time slot, such as the MCS level and bit error rate. The agent learns to select a set of actions, including symbol rate, MCS and transmit power for the following time slot. The objective is to maximize the energy efficiency and throughput as well as ensure a low bit error rate.

The work in [108] and [109] aims to maximize the energy efficiency in networks.

In [108], the authors focus on a cooperative re-transmission problem where a device is able to relay packets from its neighbors. Each device learns a transmit probability that either transmits its own packet or re-transmits/relays a packet heard from a neighboring device. It also learns a suitable transmit power if it decides to transmit. The aim is to maximize network energy efficiency. In [109], the authors run a reinforcement learning algorithm on each user. The state consists of the current channel status and the queue length of the user itself, and the action is to select an MCS for transmissions. The aim is to improve the energy efficiency and meet the delay requirement of each user.

2.3.2 Channel Access Related

There are also works that employ reinforcement learning on channel access. Past works consider channel assignment, channel bonding, transmission scheduling, transmit power control and contention windows adjustment. The objective is to use reinforcement learning to control channel access of devices in order to avoid interference or maximize throughput and energy efficiency.

In [110], the authors consider channel selection in Vehicle-to-Vehicle (V2V) networks in a decentralized manner, meaning each vehicle is an agent. Each agent observes the channel gain, interference, channels used by neighboring agents and experienced latency, and learns to select a channel and transmit power for transmission. The aim is to maximize the reward obtained from a function of throughput and latency. The work presented in [111, 112] studies the channel access problem in Internet of Thing (IoT) networks. In[111], the authors assume the channel condition is random and has two states, i.e., good or bad. A sensor runs DQN and learns to select a channel for uploading data. The aim is to avoid interference and maximize the number of successful transmissions. The authors further consider to re-train DQN when the moving average reward obtained by the agent is lower than a pre-defined threshold. In [112], a sensor that relays data from other sensors runs

a DQN, and learns to select a set of channels, data rate and the volume of data to transmit. The aim is to maximize energy efficiency. The work in [117] proposes a stateless decentralized reinforcement learning algorithm in a wireless network. The authors assume each AP serves one user, and learn to select a channel and transmit power. The goal is to maximize the throughput to its associated user. In [118], the authors formulate a multi-armed bandit model that is used to predict whether a channel is idle in order for secondary users to transmit.

The authors of [113] consider a centralized method in IoT networks, where a base station allocates channels and schedules transmissions for multiple sensors. The key challenge is that the energy level of each sensor is random. Therefore, the authors propose that a base station runs DQN with multiple Long Short-Term Memory (LSTM) layers [126], and learns to select a sensor to transmit in each time slot. The aim is to maximize the total uplink throughput of all sensors. Reference [114] also runs a DQN with LSTM on a base station in cellular networks, where the aim is to train the base station to access channels with a probability that maximizes the throughput of downlink transmissions. Another work in [115] trains each AP to transmit or defer after having detected a transmission in neighboring cells. If an AP chooses to transmit, it sets an appropriate data rate. The authors aim to minimize the channel access service time, i.e., the total elapsed time to successfully transmit one packet.

The authors of [119, 120] use deep reinforcement learning to determine a set of channels for transmissions. The authors assume channels are always available. In [119], the authors use reinforcement learning to learn the optimal channel bonding policy, whereas the aim is to minimize interference between APs and satisfy the time varying traffic demands at each AP. In [120], the authors jointly optimize channel bonding and transmit power for an AP. The goal is to maximize the energy efficiency of the AP with random arrival traffic.

The authors in [121] use centralized reinforcement learning to dynamically select an optimal CCA threshold for all devices in WiFi networks. The authors assume

that the deployment of access points and stations follows a Poisson point process. A device that intends to transmit is an active device. The number of active devices varies over time in order to simulate realistic network scenarios. A centralized controller observes the number of active devices and selects a CCA threshold. The aim is to maximize the normalized network throughput. In [116], the work assumes each secondary user observes the queue length of a primary user and learns the optimal energy detection threshold for primary channel access. Their aim is to minimize the queue length of a primary user. The work in [125] uses multi-armed bandit to assign a transmitting channel, transmit power and CCA threshold to each AP. Their aim is to maximize the throughput of each AP.

The work in [122–124] considers to use reinforcement learning to adjust the contention windows size for each device. The approach proposed in [122] is decentralized. Each device observes the arrival traffic and PER after each transmission, and determines whether to double or halve current contention window size or keep it unchanged. The goal is to reduce collision. In [124], the authors propose a centralized method, where they assume each AP optimizes the contention window size of its associated users, including the AP itself. The state is the PER of all devices within a cell, and the action is to set a value for the contention window size. The aim is to maximize the whole network throughput. In [123], the authors consider different priorities for different types of traffic. The contention window size for services, e.g., video and audio, is smaller than other services. The aim is to ensure services with a high priority have high throughput.

2.4 Summary

In summary, this chapter reviews prior works that focus on CCA threshold adjustment, channel bonding and reinforcement learning in wireless networks. Prior works show following gaps:

- Past works on dynamic CCA threshold adjustment mainly focus on heuris-

tic methods, and they do not have the learning ability, e.g., [42, 43, 45–50]. An AP/user or a centralized controller monitors the metrics of the network and adjusts CCA threshold following predefined rules. These methods provide optimal solutions as presented in literature. However, one drawback is that these rule-based algorithms lack of robustness. A minor modification on the topology can significantly degrade the performances of methods. Moreover, these works only monitor the network metrics such as RSSI, PER and channel occupancy to maximize throughput of energy efficiency. They ignore the influence of random traffic load. In addition, some works such as [42, 57, 115] does not consider the impact of random channel gains, and assume the data rate on a given channel is fixed. Some other jointly optimize the CCA threshold and transmit power for devices, e.g., [67–73, 125]. They assume the CCA threshold and transmit power have a fixed relationship, i.e., inversely proportional or linear. Therefore, by determining the CCA threshold/transmit power, a device obtains a corresponding value for transmit power/CCA threshold. The work in[73] uses deep supervised learning for CCA threshold adjustment or transmit power control. However, this work requires data from multiple APs. In other words, the authors assume APs managed by a centralized controller. In addition, the work in [125] only consider a fixed combination of CCA threshold and transmit power for each AP to choose, which means the RL agent may not be able to find the optimal combination of CCA threshold and transmit power if it is not included in the action space. Different from prior works, this thesis considers the impact of random channel conditions and random arrival traffic, and independently and adaptively adjusts the CCA threshold/transmit power for each device using reinforcement learning. Each agent only uses its locally observed metrics and do not cooperate with each other.

- Prior works on channel bonding aim to improve capacity or avoid interference in WiFi networks, e.g., [18, 36, 75–77, 81, 82, 84–88, 91, 93, 94]. They do

not consider to adjust CCA threshold on each channel to improve spatial reuse of channels. Works such as [86, 91, 93] use a centralized method to optimize channel bonding. These methods require global information and cooperation between APs. In addition, previous works, such as [84, 89–91, 93], only bond adjacent channels. However, non-adjacent channels have the potential to provide more flexibility than bonding adjacent channels and is now supported in IEEE 802.11ax [21]. Although works in [33, 79] adjust the CCA threshold of channels for channel bonding, they consider to set a same CCA threshold for all bonded channels. Further, they do not consider random traffic.

- A large number of past works apply reinforcement learning in WiFi networks to optimize the throughput and energy efficiency, e.g., [96–109]. These works do not consider channel access and assume channels are always available. Other works, such as [110–115, 117–120, 122–124] investigate channel access. However, they consider to set an appropriate data rate, contention window, transmit power or channel to avoid interference, and do not consider to optimize CCA threshold to improve spatial reuse. Although the work [121] uses Q-learning to adjust the CCA threshold, it assumes all devices are managed by a a centralized controller and share the same CCA threshold. Lastly, no prior works consider to use reinforcement learning to improve spatial reuse on multiple channels.

Chapter 3

Maximizing Spatial Reuse Using Deep Q-Network

As discussed in Chapter 2, a number of prior works aim to improve spatial reuse through CCA threshold adjustment. Specifically, the majority of these works periodically monitor network metrics and use heuristic methods or pre-defined rules to adapt the CCA threshold of APs or users. They do not consider learning and adapting the wireless environment of APs or users. In addition, some prior works require global information which may cause overhead issue. In contrast, this thesis first formulates the problem for optimizing CCA threshold as an MDP. It then proposes a DQN based learning approach running on each AP. Each AP is responsible to assign the optimal CCA threshold for each device within its cell given the historical locally measured interference data.

The rest of this chapter is organized as follows. Section 3.1 presents the system model. In Section 3.2, it models the CCA threshold adjustment problem as a Markov Decision Process (MDP). Section 3.3 and Section 3.4 outline a DQN-based approach and results, respectively. Finally, Section 3.5 concludes this chapter.

Table 3.1: Notations and Explanation

Notations	Explanation
A	The set of APs
U	The set of users
$\mathcal{M}$	Area for placing APs and users
λ_u	User density
$d_{i,j}$	Distance between node-i to node-j
A_i	AP of user-i
C_j	Cell j
γ_x^t	CCA threshold of transmitter-x in time slot t
Γ^t	A vector of CCA thresholds used by all transmitters in time slot t
$l_{x,y}$	A directed link from transmitter-x to receiver-y
P_x^t	Transmit power of transmitter-x
$P_{x,y}^t$	Received power from transmitter-x to receiver-y (dBm)
$\hat{P}_{x,y}^t$	Received power from transmitter-x to receiver-y (Watts)
$PL(d_{x,y})$	Path-loss from transmitter-x to receiver-y
$PL(d_0)$	Path-loss at reference distance d_0
ω	Path-loss exponent
$\beta_{x,y}^t$	SINR of link $l_{x,y}$ in time slot t
$r_{x,y}(\beta_{x,y}^t)$	Data rate of link $l_{x,y}$ with SINR $\beta_{x,y}^t$
Z^t	The set of active transmitters in time slot t
I_y^t	Interference at node y in Watts
$I(x,t)$	Indicator function for transmitting

3.1 System Model and Problem

This chapter considers a set of APs, denoted as A, and a set of users, denoted as U, that operate over the same wireless channel. For practical reasons, APs do not cooperate with one another; see Section 3.3.5 for more information. Each user is indexed by i and each AP is indexed by j. APs are placed on an area with size $\mathcal{M}$ (in m^2). In this area, users are distributed as per a Poisson Point Process (PPP) with density λ_u. The Euclidean distance between user i and an AP j is $d_{i,j}$. Each user i is associated to the closest AP j, which is denoted by $A_i = \arg\min_{j \in A} d_{i,j}$. Let $\{C_1, C_2, C_3, \ldots, C_{|A|}\}$ be a set of cells. Each cell j contains users associated to AP j. Specifically, a cell j is given by $C_j = \{i \mid i \in U \wedge A_i = j\}$. All devices operate

over discrete time slots indexed by t; each time slot has fixed duration.

Nodes use CSMA/CA for channel access. Each AP is responsible for adjusting the CCA threshold and transmit power of its own and its associated users via the beacon frames over time. Denote the CCA threshold for transmitter x in time slot t as γ_x^t, where $x \in A \cup U$. Each γ_x^t has range $[\gamma_{min}, \gamma_{max}]$. Let $\Gamma^t = \{\gamma_1^t, \gamma_2^t, \gamma_3^t, \ldots, \gamma_M^t\} \in [\gamma_{min}, \gamma_{max}]^M$, where $M = |A \cup U|$, be a vector that records the CCA threshold of all nodes at time slot t.

A directed link is denoted as $l_{x,y}$, where x is the transmitter and y is the receiver. This chapter considers both uplink and downlink transmissions. Therefore, a transmitter can be an AP or a user, and the corresponding receiver can be a user and an AP within the same cell. All devices always have data to transmit.

Denote Z^t as the set of transmitters in time slot t, where Z^t will be defined formally later. The received power $P_{x,y}$ from a transmitter $x \in Z^t$ with transmit power P_x^t to a receiver y is calculated using the Log-distance path loss model. Formally, the received power (in dBm) is

$$P_{x,y}^t = P_x^t - PL(d_{x,y}), \tag{3.1}$$

where

$$PL(d_{x,y}) = 10\omega log_{10}\left(\frac{d_{x,y}}{d_0}\right) + PL(d_0) + X_g. \tag{3.2}$$

Here, $PL(d_0)$ is the path loss at reference distance is d_0, ω is the path loss exponent, and X_g (in dB) is a random variable drawn from a zero-mean Gaussian distribution, which models shadowing effects.

Denote the Signal-to-Interference-plus-Noise Ratio (SINR) of a link $l_{x,y}$ in time slot t as $\beta_{x,y}^t$. Formally, for transmitter x, the SINR at its receiver y is represented as

$$\beta_{x,y}^t = \frac{\hat{P}_{x,y}^t}{I_y^t + N_0}, \tag{3.3}$$

where $\hat{P}_{x,y}^t$ is the received power in Watts, i.e., $\hat{P}_{x,y} = 10^{(P_{x,y}^t/10)}/1000$, N_0 is the

ambient background noise, and I_y^t represents the aggregated interference at node y in time slot t. Here, I_y^t is equal to the *sum of received power* from all neighboring active transmissions, which is given by

$$I_y^t = \sum_{z \in Z^t, A_z \neq A_y} \hat{P}_{z,y}^t. \tag{3.4}$$

Let $r_{x,y}(\beta_{x,y}^t)$ denote the data rate for SINR $\beta_{x,y}^t$. In practice, the corresponding data rate for SINR $\beta_{x,y}^t$ can be obtained from the datasheet of a radio; e.g., [4]. If the SINR $\beta_{x,y}^t$ exceeds the SINR threshold of a given MCS index, node x is able to transmit with the corresponding data rate of the MCS index.

Now, to formally define Z^t, recall that node x is using the CCA threshold γ_x^t. Let V_x denote the set $C_{A_x} \cup \{A_x\} \setminus \{x\}$; this set contains the users in cell C_{A_x} plus its AP minus transmitter x. Let $I(x,t)$ indicate whether node x transmits at the beginning of a time slot t. Formally,

$$I(x,t) = \begin{cases} 1, & \text{if } I_x^t < \gamma_x^t \ \wedge \ \sum_{u \in V_x} I(u,t) = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

In other words, a user/AP can transmit at time t if I_x^t is less than its CCA threshold and it is the only transmitting node. Hence, the set of transmitters Z^t in given by $Z^t = \{x \mid x \in A \ \cup \ U \ \wedge \ I(x,t) = 1\}$. Observe that the size of Z^t, i.e., $|Z^t|$, is affected by γ_x^t or Γ^t and also the transmit power P_x^t of transmitter x, which has an impact on I_x^t.

The sum rate of all transmitters in Z^t is denoted as

$$\mathcal{D}(\Gamma^t) = \sum_{x \in Z^t} r_{x,y}(\beta_{x,y}^t). \tag{3.6}$$

The problem is to find the optimal CCA threshold $\Gamma^{t*} \in [\gamma_{min}, \gamma_{max}]^M$ for each

time t that maximizes the following expected sum-rate:

$$\mathcal{R} = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\Big[\sum_{t=1}^{\infty} \mathcal{D}(\Gamma^{t*})\Big], \tag{3.7}$$

where the expectation is taken with respect to the joint probability distribution of received power or interference over T time slots.

3.2 A Markov Decision Process Model

To solve (3.7), this chapter formulates a Markov Decision Process (MDP). An MDP provides a mathematical framework for optimizing decision-making problems [127]. An MDP is denoted as a tuple of $[\mathcal{S}, \mathcal{A}, \mathcal{P}(s_{t+1}|s_t, a_t), \mathcal{R}(s_t, a_t)]$, where $\mathcal{S}$ is a set of states and $\mathcal{A}$ is a set of actions, $\mathcal{P}(s_{t+1}|s_t, a_t)$ is a transition probability function that yields the probability of transitioning from state $s_t \in \mathcal{S}$ to state $s_{t+1} \in \mathcal{S}$ after taking an action of $a_t \in \mathcal{A}$, and $\mathcal{R}(s_t, a_t)$ is the reward for taking action a_t at state s_t. Denote π as a policy that maps a state to an action, represented as $a_t = \pi(s_t)$.

Let $V^{\pi}(s)$ be a value function that measures the expected long-term reward for a state s. This expected long-term reward is obtained by calculating the cumulative discounted reward if a decision-maker or an agent starts from the state s and chooses its actions based on policy π in subsequent states. Formally,it is defined as,

$$V^{\pi}(s) = E_{\pi}\Big[\sum_{k=0}^{\infty} \gamma^k \mathcal{R}(s_{t+k}, \pi(s_{t+k}))|s_t = s\Big], \tag{3.8}$$

where $\gamma \in [0, 1]$ is the discount factor that weighs the importance of future rewards. The goal of an MDP is to find the optimal policy π^* that maximizes the value function for each state s. This optimal policy π^* can be obtained by first solving the optimal value of Equ. (3.8), denoted as V^*, via the Bellman equation. Formally,

for a given state $s_t \in \mathcal{S}$, its optimal value is given by

$$V^*(s_t) = \max_{a_t \in \mathcal{A}} \Bigg[\mathcal{R}(s_t, a_t) + \gamma \sum_{s_{t+1} \in \mathcal{S}} \mathcal{P}(s_{t+1}|s_t, a_t) V^*(s_{t+1}) \Bigg], \tag{3.9}$$

where $a_t = \pi(s_t)$. Then, the optimal policy π^* for each state $s_t \in \mathcal{S}$ can be obtained as

$$\pi^*(s_t) = \arg\max_{a_t \in \mathcal{A}} \Bigg[\mathcal{R}(s_t, a_t) + \gamma \sum_{s_{t+1} \in \mathcal{S}} \mathcal{P}(s_{t+1}|s_t, a_t) V^*(s_{t+1}) \Bigg]. \tag{3.10}$$

For the proposed problem, let **state** s_t^x be a vector, where its elements represent the interference observed by a transmitter x at time step t in the past T time slots, i.e., $s_t^x = [I_x^{t-T}, I_x^{t-T+1}, I_x^{t-T+2}, \ldots, , I_x^{t-1}]$. The **action** a_t^x for a given state s_t^x corresponds to a CCA threshold $\gamma_x \in [\gamma_{min}, \gamma_{max}]$. The **transition probability function** $\mathcal{P}(s_{t+1}^x \mid s_t^x, a_t^x)$ is unknown as this chapter considers a model-free approach. This means the proposed solutions are practical because they do not assume any specific channel gain model between nodes. In addition, the solutions need to be trained in an online manner where agents/APs interact with the environment to learn the best action for each state. The **reward** $\mathcal{R}(s_t^x, a_t^x)$ for a transmitter is the throughput over the next T time slot that is calculated as,

$$\mathcal{R}(s_t^x, a_t^x) = \frac{1}{T} \sum_{k=1}^{T} r_{x,y}(\beta_{x,y}^t) I(x,t). \tag{3.11}$$

Recall that the proposed approach is model-free, the transition probability function $\mathcal{P}(.)$ is unknown. Hence, the MDP cannot be solved using standard algorithms such as Value Iteration [128]. In addition, the interference I_x in each time slot is a continuous number, which results in a large state space $|\mathcal{S}|$. Therefore, this chapter proposes to use a model-free reinforcement learning algorithm to solve the

formulated MDP. Specifically, it will employ Deep Q-Network (DQN) [129].

3.3 Deep Q-Network based approaches

The DQN algorithm aims to solve MDP problems by combining the Q-learning algorithm [130] with neural networks. This section introduces the key concepts leveraged by DQN.

3.3.1 Q-learning

Q-learning is a model-free algorithm as it does not require the transition probability between states [130]. It maintains and updates a Q-table that records the Q-value of each state-action pair, denoted as $Q(s_t, a_t)$. For a given state, a high Q-value for an action represents a high potential reward. The Q-learning algorithm updates the Q-value of each state-action pair (s_t, a_t) as follows:

$$\begin{aligned} Q(s_t, a_t) = Q(s_t, a_t) + \alpha \, [\mathcal{R}(s_t, a_t) \\ + \gamma \times \max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \, , \end{aligned} \tag{3.12}$$

where α is the learning rate. To balance the relationship between exploitation and exploration, Q-learning follows an ϵ-greedy policy, which is for a given probability, the agent will choose the action with highest Q-value; otherwise, the agent will randomly choose an action. Q-learning has been proven to converge to the optimal Q-table when each state-action pair is visited infinitely often [130]. This means after adequate training, an agent is able to select the optimal action for each encountered state by simply checking the Q-value for each action.

3.3.2 DQN

Traditional Q-learning algorithm works effectively when the state-action space $|\mathcal{S}| \times |\mathcal{A}|$ is small. However, it becomes impractical when the Q-table size is large. The

reason is that each state is visited less frequently when the size of Q-table is large, meaning the Q-value is updated rarely. Hence, it will take a much longer time for Q-values to converge to the optimal Q-value. In addition, recall that the state defined in the proposed problem contains continuous numbers, traditional Q-learning algorithm is, thus, not practical.

DQN alleviates the said convergence problems by combining Deep Neural Networks (DNN) with the traditional Q-learning algorithm. It uses a DNN to approximate the Q-table, aka a Q-network [129], denoted as $\mathcal{Q}(s_t, a_t, \theta)$, where θ is a weight that maps the input to the output of a DNN. The input to the Q-network is the state, and the output corresponds to the Q-value for each action. Specifically, a DQN consists of two Q-networks with the same topology, namely the prediction Q-network $\mathcal{Q}(s_t, a_t, \theta)$ and the target Q-network $\mathcal{Q}(s_t, a_t, \theta')$. For each iteration, DQN updates the prediction Q-network weight θ to minimize the following loss function:

$$\mathcal{L}(\theta) = \mathbb{E}[(\mathbf{y} - \mathcal{Q}(s_t, a_t, \theta))^2], \tag{3.13}$$

where

$$\mathbf{y} = \mathcal{R}(s_t, a_t) + \gamma \max_{a \in \mathcal{A}} \mathcal{Q}(s_{t+1}, a, \theta'). \tag{3.14}$$

On the other hand, the target Q-network weight θ' is replaced with θ every K iterations, where K is a pre-defined integer.

DQN adopts experience replay [131]. Each combination of $(s_t, a_t, \mathcal{R}(s_t, a_t), s_{t+1})$ is called experience and stored as an element in the dataset called *memory*. When updating the weight θ, DQN randomly samples a mini-batch from *memory* and uses it to update θ. In this way, DQN mitigates the correlation among observed states and alleviates instabilities during training [129].

3.3.3 DQN based approaches

This chapter solves the proposed problem with DQN in a distributed manner, i.e., each AP runs a DQN and learns its optimal policy independently. In particular, it considers a multi-agent RL approach whereby APs/agents are independent learners [132]; i.e., an AP sees other APs as part of the environment. The definition of state space $\mathcal{S}$, action space $\mathcal{A}$ and reward function $\mathcal{R}$ are specified in Section 3.2. Each AP uses the ϵ-greedy to select an action. Briefly, an agent randomly selects an action $a_t \in \mathcal{A}$ with probability ϵ. Otherwise, the agent selects an action with highest Q-value $a_t = \arg\max_a \mathcal{Q}(s_t, a, \theta)$. The value of ϵ decays during learning process to ensure convergence. Two DQN based learning patterns are studied in this chapter, namely Episodic-Learning (EpL) and Instant-Learning (InstL). The details of both learning patterns are as follows:

- *Episodic-Learning (EpL).* As shown in Algorithm-1, aims to train a DQN after the completion of an episode. Here, an episode is defined as a round in which an AP adjusts the CCA threshold of devices in its cell one by one. To be specific, as shown in Algorithm-1, an AP first initializes the DQN with weights θ and θ'. Next, for each transmitter x in the AP's cell, the AP selects an action a_t^x based on the observed state s_t^x as per ϵ-greedy. The corresponding reward $\mathcal{R}(s_t^x, a_t^x)$ and next state s_{t+1}^x are then stored together with s_t^x and a_t^x into *memory*. Upon the completion of one episode, the AP samples a mini-batch from *memory* and uses it to updates network weights, see Section 3.3.2.

- *Instant-Learning (InstL).* It first initializes DQN with weights θ and θ'. The process of selecting action and storing experience into memory is the same as EpL. However, *InstL* updates the neural network weights instantly after a reward is obtained, as shown in Algorithm-2.

Algorithm 1: Pseudocode for Episodic-Learning

```
Initialize: 𝒜, 𝒰,{C_1, C_2, C_3, ..., C_|𝒜|}, θ, θ'
for episode=1, M do
    for each transmitter x do
        Observe a state s_t^x
        Select an action a_t as per ϵ-greedy
        Get reward ℛ(s_t^x, a_t^x), and observe next state s_{t+1}^x
        Store data (s_t^x, a_t^x, R(s_t^x, a_t^x), s_{t+1}^x) into memory
    end
    Sample a mini-batch of data from memory
    Use mini-batch data to update θ of DQN as per Equ. 3.13
    Decrease ϵ
    Replace θ' with θ every K iterations
end
```

Algorithm 2: Pseudocode for Instant-Learning

```
Initialize: 𝒜, 𝒰,{C_1, C_2, C_3, ..., C_|𝒜|}, θ, θ'
for episode=1, M do
    for each transmitter x do
        Observe a state s_t^x
        Select an action a_t as per ϵ-greedy
        Get reward ℛ(s_t^x, a_t^x), and observe next state s_{t+1}^x
        Store data (s_t^x, a_t^x, R(s_t^x, a_t^x), s_{t+1}^x) into memory
        Sample a mini-batch of data from memory
        Use mini-batch data to update θ of DQN as per Equ. 3.13
        Decrease ϵ
        Replace θ' with θ every K iterations
    end
end
```

3.3.4 Extension with Transmit Power Control (TPC)

This chapter further considers combining Transmit Power Control (TPC) with CCA threshold adjustment. This accomplished by modifying the action selection of an agent. Specifically, upon seeing a state s of transmitter x, an agent will assign a CCA threshold γ_x and a value of transmit power P_x^t simultaneously to the transmitter x. In this respect, a change in the definition of reward is needed, see Section 3.4.3.1 for reasons. Here, this chapter introduces a penalty coefficient η, which balances the weighting between throughput and transmit power in the reward. As a result, the

reward for a transmitter x that adopts TPC with penalty is given as

$$\mathcal{R}_p(s_t^x, a_t^x) = \eta\mathcal{R}(s_t^x, a_t^x) - (1-\eta)P_x^t. \tag{3.15}$$

3.3.5 Discussion

The proposed solutions have a number of practical advantages. First, they do not require cooperation between APs, which helps save on signaling overheads. Specifically, they do not require information from neighboring APs. Instead, they use information local to an AP, which can be obtained using IEEE 802.11k[1]. Note also that cooperation between APs is impossible if APs are operated by different organizations. Second, they are model-free, meaning they *do not* require as input the probability distribution of interference experienced by an AP and its associated users. Consequently, they are able to operate in any WiFi environment without first conducting an extensive measurement campaign. Instead, an AP learns over time the optimal policy or CCA threshold that maximizes its throughput for a given WiFi network.

There are several improved deep Q-learning algorithms such as Double Deep Q-Learning (Double-DQL) [133] and Dueling Deep Q-Learning (Dueling-DQL) [134]. Both of them provide the same results as traditional DQN. Therefore, in the following section, this chapter only reports the results obtained using traditional DQN.

3.4 Evaluation

The experiments are conducted using Python 3.7.7. Multiple APs and users are placed on a square area of dimension 100 $m \times 100$ m. The number and location of APs are fixed in each experiment; the number and location of users are randomly generated using a Poisson Point Process (PPP) with a density of λ_u. Note that the

[1]IEEE 802.11k standardizes radio resource measurement interfaces, through which devices are able to measure their link quality to neighboring devices, interference level and channel load statistics [52]. This amendment also allows devices to share measured radio information.

Table 3.2: Parameter values used for experiment

Parameter	**Value(s)**
Area size (m^2)	100 x 100
$\|A\|$ Number of AP	2,4,6,8,10
λ_u user density (user/m^2)	0.001,0.002,0.003,0.004
Default transmit power for AP (dBm)	20 [135]
Default transmit power for user (dBm)	15 [135]
Path loss exponent	2 [136]
P_n Environment Noise (dBm)	-90
Default CCA threshold (dBm)	-82 [135]
γ_{min} (dBm)	-82
γ_{max} (dBm)	-10
Available transmit power (dBm)	-20 to 20
Number of evaluation time slot T	100

Table 3.3: Data rate look-up table [4]

SINR (dB)	**Data Rate (Mbps)**	**SINR (dB)**	**Data Rate (Mbps)**
2	7.2	18	57.8
5	14.4	20	65
9	21.7	25	72.2
11	28.9	29	86.7
15	43.3		

proposed learning approach is able to be generalized to other deployments as it only relies on the locally obtained information, i.e., the historical data of interference. Therefore, the learning approach is independent of the deployment of APs and users or the size of networks. All devices operate over a 20 MHz channel, and the achievable data rate of a link with a given SINR is as per Table 3.3. Note that if the SINR of a transmission falls below two dB or a required SINR threshold, it is considered as failed and receives a data rate of zero. Table 3.2 lists all parameter values. The path loss is set to two to ensure the transmission range of each device covers the whole area. Therefore, each device is able to hear the transmission of all other devices, which creates a dense network. The neural network used by a DQN agent is constructed using Tensorflow 1.14.0 [138] and Keras 2.2.5 [139] with

Table 3.4: parameters for DQN agent

Parameter	**Value(s)**
$\|\mathcal{S}\|$	100
$\|\mathcal{A}\|$	36 (DSC only) 324 (DSC with TPC)
Number of hidden-layers	2
Hidden-layer size	40 (DSC only) 330 (DSC with TPC)
Optimizer	Adam [137]
α Learning rate	0.001
γ Discount factor	0.95
ε initial value	1
ε decay factor	0.004
Minimal ε	0.001
Batch size for training	32
Memory buffer size	200
Activation function	ReLU
Target network update frequency	100

parameters listed in Table 3.4. Specifically, a neural network has two hidden-layers, which is sufficient to approximate Q-values [140]. Each AP runs a DQN agent that assigns a CCA threshold for each device in its cell. The RTS/CTS mechanism is disabled as APs are within carrier sense range of one another; i.e., there are no hidden terminals.

The proposed solutions are compared with the legacy Dynamic Sensitivity Control (DSC) algorithm reported in [42], which is denoted as Le-DSC. Briefly, Le-DSC adjusts its CCA threshold of users based on the received power from their associated AP. For each AP, its CCA threshold is based on the received power of neighboring APs as well as its associated users. The network environment used to compare the performance of *InstL*, *EpL* and Le-DSC is the same for each experiment. The number and locations of APs are fixed. In addition, the channel model and transmit power of each device is also identical. The distribution of users is generated with the same *seed* to ensure the number and location of users are identical. Two metrics are

collected at receiver side: aggregated throughput and average throughput. The aggregated throughput is the total throughput of all cells, and the average throughput specifies the average throughput per device. For each experiment in Section 3.4.2 and Section 3.4.3.3, The agent running on each AP is trained until its performance converges; this stage is called the *Training stage*. Then, the well-trained agents are tested to record the performance. The following factors are studied:

1. *User density* λ_u. This parameter impacts the number of users. It increases λ_u from 0.001 to 0.004 with an interval of 0.001. Note that this simulation does not include user density higher than 0.004 as the intra-cell competition among devices will be exacerbated.

2. *Number of APs*. This parameter impacts the number of cells and the number of concurrent transmissions. The number of APs varies from two to fourteen with an interval of two.

3. *Transmit Power Control (TPC)*. It considers to combine TPC with CCA threshold selection for each device. To this end, the action of a DQN agent is redefined as assigning a CCA threshold and transmit power to a device simultaneously.

3.4.1 Learning phase

EpL and *InstL* are first trained, and compared against Le-DSC. The network under consideration has four APs and a user density of $\lambda_u = 0.001$. Training is carried out for 500 episodes.

From Fig. 3.1, both *InstL* and *EpL* have better performance than Le-DSC. Le-DSC has an average throughput of 3.61 Mbps, while *InstL* and *EpL* reach a throughput of 5.86 Mbps, i.e., *InstL* and *EpL* outperform Le-DSC by 62.40%. The reason is that the reward of proposed solutions corresponds to the throughput of a device. If a device cannot transmit, it receives a reward of zero. Therefore, DQN agents will

ensure each device can access the channel in each time slot such that devices have the highest throughput over time. However, on the other hand, Le-DSC adjusts the CCA threshold of a device based on the received power of associated devices. For users that are close to their associated AP, the CCA threshold is high. Therefore, these users have a high probability to transmit in each time slot. However, for users that are far away from their associated AP, their CCA threshold is low, which means they cannot always transmit. Consequently, devices that are controlled by Le-DSC have a lower average throughput.

It is also noticed that the average throughput of *InstL* and *EpL* is lower than the minimum data rate of 7.2 Mbps. This is because although *InstL* and *EpL* are able to ensure a device transmitting in each time slot, a device may receive an SINR lower than 2 dB due to the dynamic wireless environment. This means the transmission is failed and thus has a data rate of zero. Consequently, the average throughput falls below the minimum data rate of 7.2 Mbps over time.

It also shows in Fig. 3.1 that *InstL* converges significantly faster, i.e., it converges to 5.86 Mbps within 20 episodes while *EpL* converges at around 250 episodes. This is because the value of ε reduces at a different rate in *InstL* and *EpL*. The probability that an agent explores the action space is controlled by ε, where a high ε value means a high probability of random exploration. In *InstL*, a DQN agent updates its neural network after assigning a CCA threshold, while in *EpL*, a DQN agent only updates its neural network at the end of an episode. As a result, the ε value of *InstL* and *EpL* reduces at a different rate, which results in different convergence speed.

3.4.2 Network environment

3.4.2.1 User density

In this section, the number of APs is fixed to four and the user density λ_u increases from 0.001 to 0.004 with an interval of 0.001. Referring to Fig. 3.2 and Fig. 3.3, one can find that both *InstL* and *EpL* have better performance than Le-DSC. This

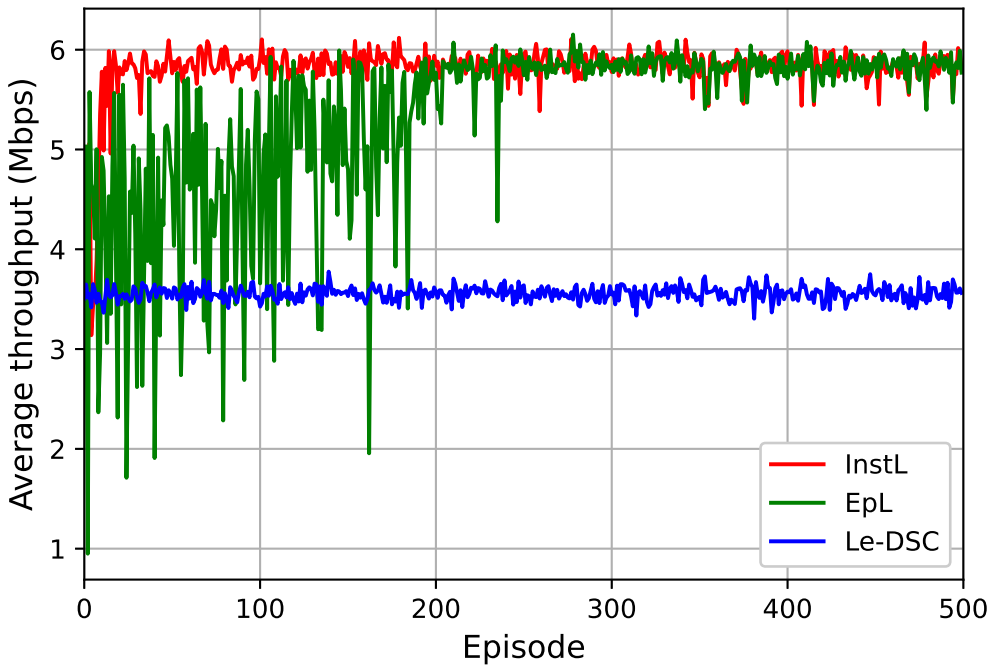

Figure 3.1: Results for Instant-Learning, Episodic-Learning and Le-DSC in fixed network environment.

is because both DQN based algorithms have developed a CCA threshold selection policy that selects CCA threshold with the maximum Q-value. One can also find that both of the aggregated throughput and the average throughput decrease as user density increases. This is because with increasing user density, the number of users in each cell increases. As there is only one transmission allowed in each cell, each user has decreasing probability to transmit along with increasing number of users in each cell. Furthermore, with increasing user density, it is more likely that there are many users that are placed in the overlapping area of adjacent cells. These users experience high interference and will have a low throughput when they are receiving data from their associated AP. These users also produce significant interference to neighboring cells. Therefore, both of the aggregated throughput and average throughput degrade.

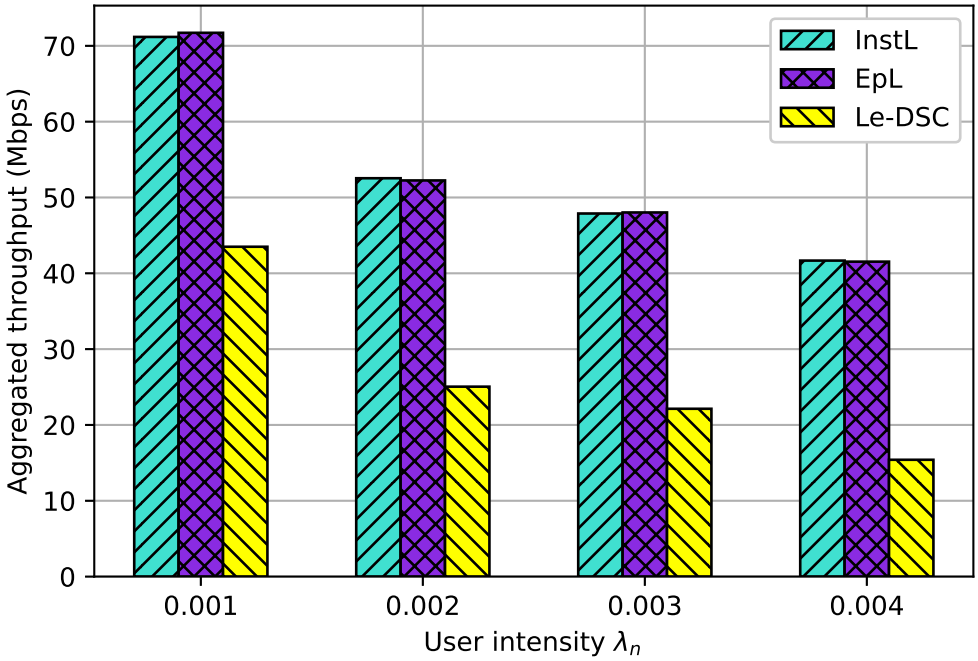

Figure 3.2: Results for aggregated throughput with different user density.

3.4.2.2 AP Numbers

This section studies the impact of different number of APs when the number and location of users are fixed. The user density λ_u is fixed to 0.004 and the number of APs increases from four to fourteen with an interval of two.

In Fig. 3.4 and Fig. 3.5, the aggregated throughput and average throughput have the same trend. The reason is that with an increasing number of APs, the distance between an AP and the number of associated users decreases. As a result, the received signal is high, which means devices are able to receive a high SINR and thus, have high throughput. In addition, with an increasing number of APs, the maximum number of concurrent transmissions increases. With more links activated simultaneously, both of the average throughput and the aggregated throughput increases. However, when the number of APs continues to increase, the increasing number of concurrent transmissions produces significant amount of interference. Therefore, both of the aggregated throughput and average throughput decrease when the number of APs is larger than eight.

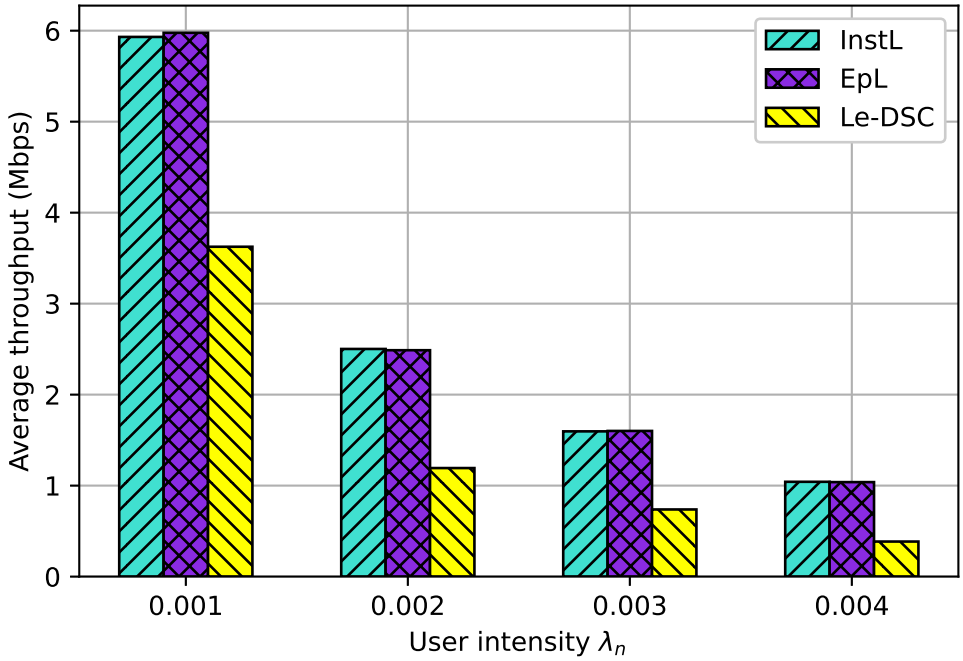

Figure 3.3: Results for average throughput with different user density.

3.4.3 Transmit power control

This section evaluates the impact of TPC. The number of APs is set to four and the user density λ_u is set to 0.001. TPC with and without penalty are both considered. The result is compared with those obtained in Section 3.4.1. After that, this section further studies the impact of distance between APs when using TPC.

3.4.3.1 TPC without penalty

The performance of TPC without penalty is first studied. Fig. 3.6 shows the average throughput of *InstL* and *EpL* with TPC has no significant difference with those without TPC. All algorithms converge to 5.85 Mbps. The reason is that the reward is set as the throughput of a device. This means the best option for a DQN agent is to transmit with the highest transmit power. As a result, devices are able to transmit with the highest data rate and thus, have the highest throughput. However, for TPC, the highest transmit power that a DQN agent can choose is 20 dBm. This value is equal to the transmit power of devices when TPC is not leveraged. Therefore, the

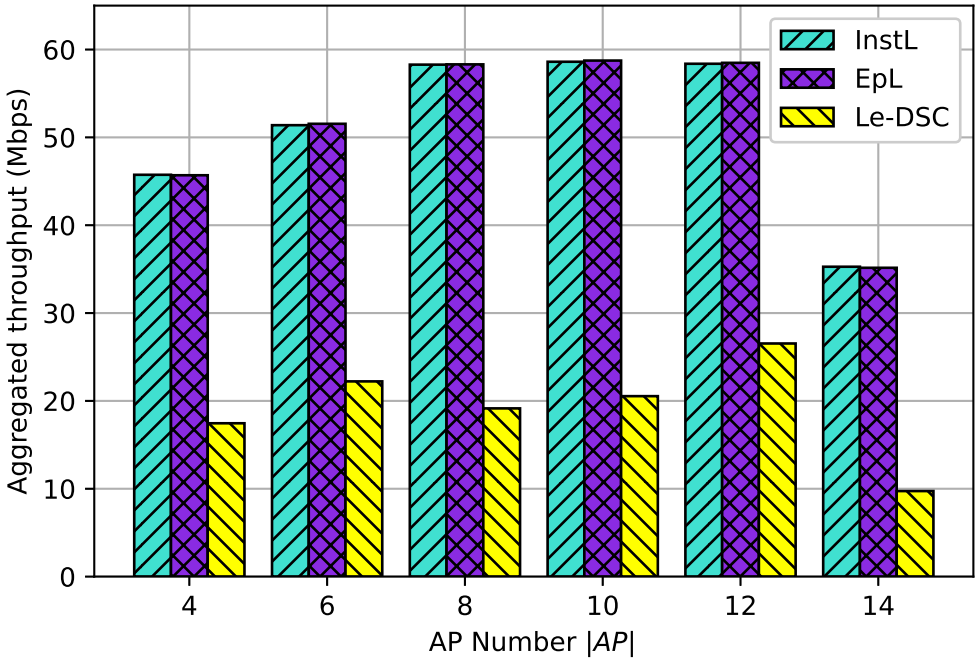

Figure 3.4: Results for aggregated throughput with different AP numbers.

average throughput of TPC and non-TPC converges to same value, which means using TPC without penalty on reward has negligible improvement on the average throughput. The convergence time for TPC is much slower than that of Non-TPC. The reason is that the action size for TPC and Non-TPC is different. The action size for Non-TPC is 36, while that of TPC is 324. The large action space requires a DQN agent to explore more at each state which increases convergence time.

3.4.3.2 TPC with penalty

To study TPC with penalty, this section sets all other parameters to the same value as Section 3.4.3.1. Fig. 3.7 shows that the average throughput of $\eta = 90\%$ is 41.8% higher than that of Non-TPC. This is because a DQN agent receives high penalty on reward if it chooses to transmit with high transmit power. As a result, a DQN agent will choose low transmit power if it can still provide high data rate. In this respect, a device produces less interference to neighboring cells, which means all devices can transmit with high data rates and have a high throughput.

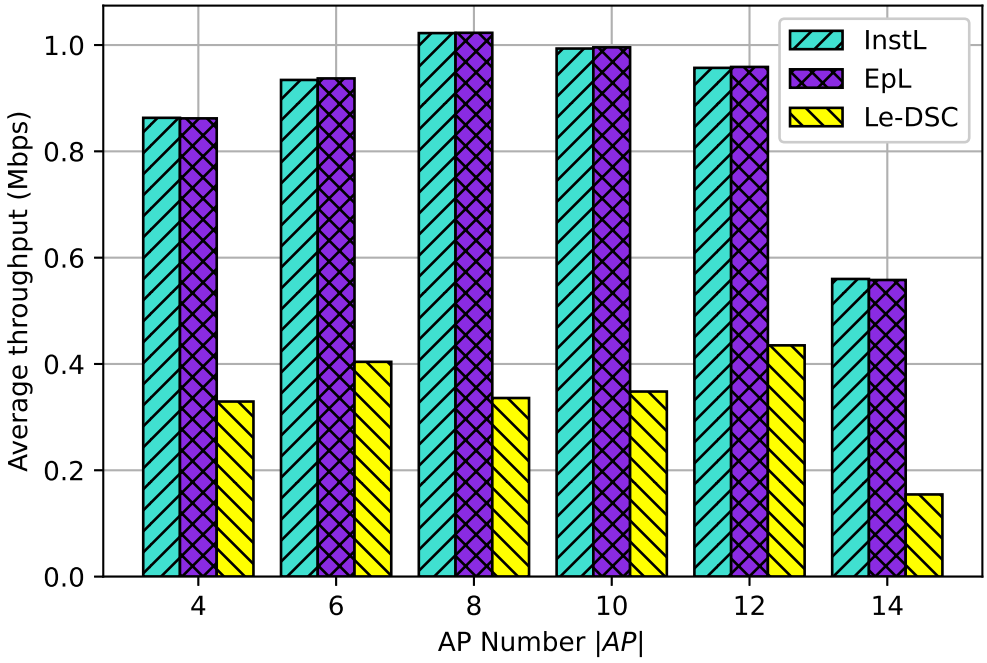

Figure 3.5: Results for average throughput with different AP numbers.

When $\eta = 10\%$, the average throughput of TPC is worse than that of Non-TPC, i.e., the average throughput of TPC with $\eta = 10\%$ is 92.5% of that of Non-TPC. This is because when $\eta = 10\%$, throughput occupies 10% in reward whereas transmit power occupies 90% in reward as penalty. This high percentage of penalty leads to significant degradation on reward. As a result, a DQN agent will choose the lowest transmit power for each device, which degrades the SINR of transmissions. Consequently, the average throughput is low.

3.4.3.3 Distance between APs

This experiment studies the impact of distance between APs. Further, DQN agents use TPC with penalty. Two APs are considered, and each AP has five users within the range of five meters. The distance between the two APs increases from five to 40 meters to simulate different levels of interference. The value of η is respectively set to 10% and 90% to train DQN agents with *InstL* and *EpL* separately. Then the results are compared against DQN algorithm with Non-TPC, i.e., the DQN agent

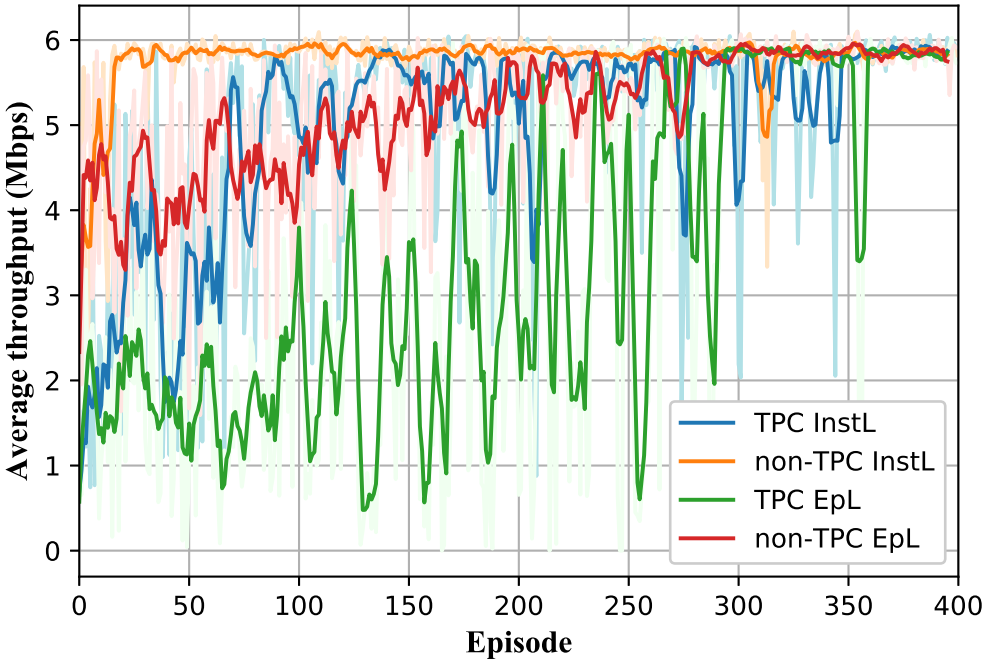

Figure 3.6: Results for TPC without penalty.

only assigns the CCA threshold and will not adjust the transmit power for each device.

Fig. 3.8 shows the average throughput per device for different inter-AP distance. The average throughput of all algorithms increases with increasing inter-AP distance. The average throughput of TPC with $\eta = 10\%$ and $\eta = 90\%$ increases from 0.73 Mbps to 4.35 Mbps and 2.19 Mbps to 7.08 Mbps as the AP distance increases from 5 m to 40 m. Also, the performance of Non-TPC increased from 1.25 Mbps to 7.36 Mbps. This is because the interference from a neighboring cell decreases as the distance between APs increases. Therefore, all devices are able to transmit with a high data rate and thus, have a high throughput. Further, TPC with $\eta = 90\%$ has better performance than Non-TPC when the distance is small. DQN agents that use TPC with $\eta = 90\%$ outperform Non-TPC by 76.3% and 37.1% when the distance between APs is 5 m and 10 m, respectively. This is because devices that use TPC with $\eta = 90\%$ learn to use a low transmit power. To be specific, when APs are close to each other, devices experience excessive amount of interference from neighboring

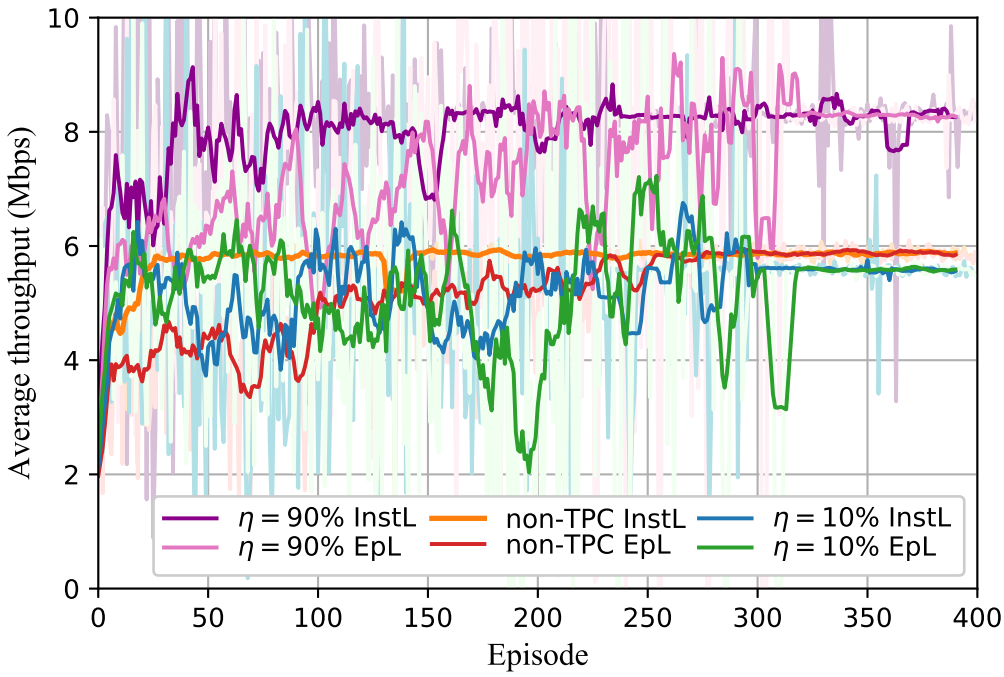

Figure 3.7: Non-TPC and TPC with $\eta = 10\%$ and $\eta = 90\%$.

cells. If a device uses a low transmit power to transmit, it significantly reduces interference to a neighboring cell. This means all devices are able to transmit with a high data rate. In this respect, the average throughput is high. In addition, as the distance between APs increases, TPC with $\eta = 90\%$ has a similar performance with Non-TPC. The average throughput of TPC with $\eta = 90\%$ is 4.40 Mbps and 7.05 Mbps when the AP distance is 20 m and 40 m, respectively. By contrast, the performance of Non-TPC is 4.23 Mbps and 7.36 Mbps. In other words, the difference between TPC with $\eta = 90\%$ and Non-TPC is only 4.15% on average. This is because TPC with $\eta = 90\%$ learns to increase the transmit power to achieve a high data rate when the distance between APs is large. On the other hand, Non-TPC agents are only able to adjust their CCA threshold to a value in Table 3.2. Moreover, they use the highest transmit power of 20 dBm. Also, agents using TPC with $\eta = 90\%$ use the same transmit power as Non-TPC agents. Consequently, they have similar performance. Moreover, TPC with $\eta = 10\%$ has the worst performance. The average throughput of TPC with $\eta = 10\%$ is inferior to both TPC with $\eta = 10\%$

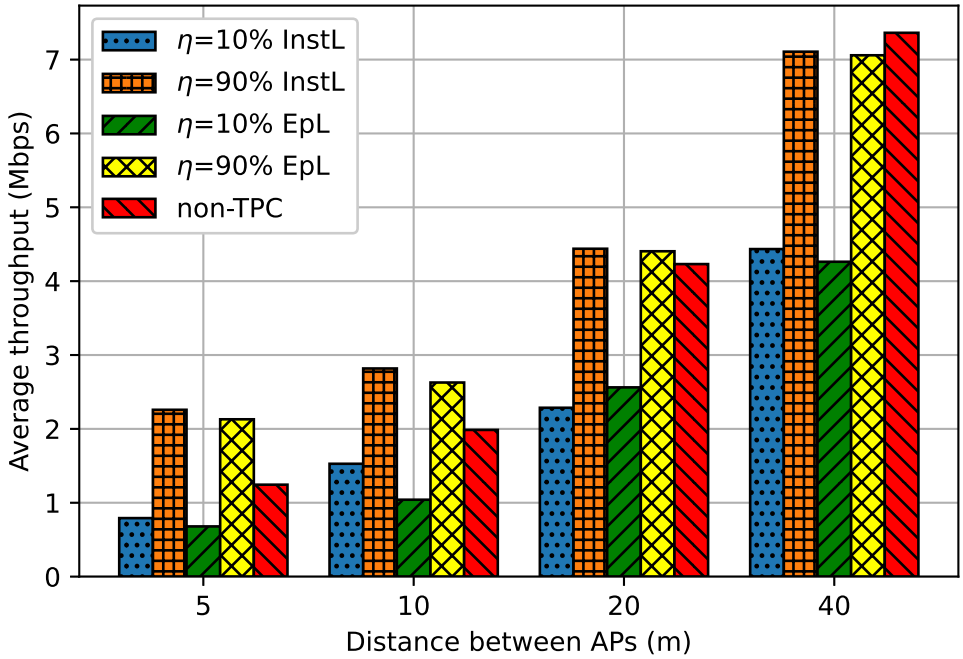

Figure 3.8: A comparison between Non-TPC and TPC with $\eta = 10\%$ and $\eta = 90\%$ under different AP distances.

and Non-TPC. The average throughput for TPC with $\eta = 10\%$ is 49.22% of the throughput of TPC with $\eta = 90\%$ and 59.93% of the throughput of Non-TPC for all tested AP distances, respectively. This is because when $\eta = 10\%$, the transmit power degrades the reward by 90% of its value, see Equ. 3.15. Therefore, an agent will consider using the lowest transmit power to get a high reward over time. As a result, the average throughput is low.

3.5 Conclusion

This chapter considers a dense WiFi network where a number of APs are closely placed in a given area. Each AP runs a DQN to learn assign the optimal CCA threshold for each device in its cell. It proposes two learning patterns, namely *EpL* and *InstL*. The simulation results indicate that with sufficient learning, both *EpL* and *InstL* converge to the same optimal value. *InstL* converges faster than *EpL* in a fixed network topology, i.e., 20 episodes versus 250 episodes, as it has more times of

learning within one episode. In addition, both *EpL* and *InstL* outperform Le-DSC by 62.4%. After that, it also finds that the average throughput is further improved when TPC is enabled. With penalty coefficient $\eta = 90\%$, the average throughput is 41.8% higher than that of non-TPC.

A problem with jointly optimizing CCA threshold and transmit power is that it incurs a long training time. This is because the action space size increases significantly when transmit power control is considered. As a result, a DQN agent needs increasing time to learn the optimal action and achieves performance convergence. To this end, next chapter considers to use a hierarchical control of CCA threshold and transmit power to address this issue.

Chapter 4

Hierarchical CCA threshold and transmit power optimization

This chapter address the problem of jointly optimizing CCA threshold and transmit power, which is a future research direction mentioned in Chapter 3. Naively jointly optimizing CCA threshold and transmit power takes extra training time as shown in Chapter 3. This is because the action size increases dramatically, and thus a DQN agent needs long time to explore in order to learn the optimal policy. In addition, considering the traffic arrives at an AP may vary over time, an AP needs to carefully select its transmit power and CCA threshold to ensure it has sufficient capacity to avoid queue overflow. To this end, this chapter proposes a hierarchical approach to address the said issue when jointly optimizing the CCA threshold and transmit power. It first formulates the joint optimization problem at hand as a Semi-Markov Decision Process (Semi-MDP) [141]. Then, it proposes a Hierarchical Reinforcement Learning (HRL) based approach, using DQN as learning algorithm, to solve the Semi-MDP. The aim is to minimize the queue length of an AP given random traffic arrival.

The rest of this chapter is organized as follows. Section 4.1 presents the system

Table 4.1: Notations and explanation.

Notations	Explanation
$\bar{\mathcal{A}}$	The set of APs
$\mathcal{U}_i$	The set of users served by AP i
d_{ij}	The Euclidean between AP i and user j
γ_i^t	The CCA threshold of AP i
P_i^t	The transmit power of AP i
q_i^t	The length of queue of AP i
L	Packet size (in bits)
δ	Time slot duration
ω	The path-loss exponent
β_{ij}^t	The SINR from i to j
$r_{ij}(\beta_{ij}^t)$	Data rate given SINR β_{ij}
I_i^t	The aggregated interference to i
$\mathcal{N}_i^t$	The set of interfering APs of i
λ_i^t	Number of packets that arrive at queue q_i
γ	A discount factor

model. Section 4.2 presents a brief background of Semi-MDP and HRL. After that, Section 4.3 provides details of the proposed HRL approach. Lastly, Section 4.4 presents the simulation results, and Section 4.5 concludes the chapter.

4.1 System Model and Problem

Time is divided into a set $\mathcal{T}$ of time slots; each time slot has a fixed duration of δ and is indexed by t. The set of APs is denoted as $\bar{\mathcal{A}}$, and the set of users served by AP $i \in \bar{\mathcal{A}}$ is denoted as $\mathcal{U}_i$. The Euclidean distance between AP i and user j is d_{ij}. Each AP i has a queue of packets to transmit, and the length of the queue at the end of time slot t is q_i^t, where $0 \leq q_i^t \leq q_{max}$. Let λ_i^t be the number of packets that arrive at the queue of AP i at the beginning of time slot t. The value of λ_i^t is randomly sampled from a probability distribution. Further, each packet has a fixed size of L bits.

All APs operate over the same wireless channel, and they use CSMA/CA for

channel access. An AP i uses the CCA threshold γ_i^t to determine whether it is allowed to transmit at the beginning of a given time slot t. Denote by $\bar{\mathcal{A}}_t$ the set of APs that are able to transmit in time slot t. Therefore, the set of neighboring APs of AP i that are transmitting in time slot t is given by $\mathcal{N}_i^t = \{k \mid k \in \bar{\mathcal{A}}_t, k \neq i\}$.

Block fading channels are considered. The channel gain remains fixed within one time slot, and varies across different time slots. The channel gain from AP i to user j in time slot t is g_{ij}^t. The radio propagation between APs and users is modeled using the well-known Log-distance path loss model [142]:

$$PL(d_{ij})[\text{dB}] = PL(d_0) + 10\omega log_{10}\left(\frac{d_{ij}}{d_0}\right) + X_g, \tag{4.1}$$

where the first two terms on the right hand side model large-scale fading, and the last term models random attenuation. Specifically, the term $PL(d_0)$ (in dB) is the path loss at reference distance d_0, and ω is the path loss exponent. The last term X_g (in dB) is a random variable drawn from a zero-mean Gaussian distribution that represents shadowing effect. Denote $g_{ij}^t = \frac{1}{10^{PL(d_{ij})/10}}$.

The Signal-to-Interference-plus-Noise ratio (SINR) of a transmission from AP i to user j in time slot t is denoted as β_{ij}^t, which is calculated as

$$\beta_{ij}^t = \frac{P_i^t g_{ij}^t}{I_j^t + N_0}, \tag{4.2}$$

where P_i^t is the transmit power (in Watts) of AP i in time slot t, and N_0 is the ambient noise power (in Watts). The term I_j^t denotes the aggregated interference to user j in time slot t, which is given by

$$I_j^t = \sum_{k \in \mathcal{N}_i^t} P_k^t g_{kj}^t. \tag{4.3}$$

The SINR determines the data rate of a transmission. In particular, each value of SINR β_{ij}^t corresponds to a data rate, denoted as $r_{ij}(\beta_{ij}^t)$. Note that the proposed solution does not require interference information from interfering APs.

Recall that each AP has a queue of packets. As a result, the length of its queue (in packets), at the end of a given time slot t, evolves as per

$$q_i^t = \begin{cases} \min(\max(0, q_i^{t-1} + \lambda_i^t - \frac{r_{ij}(\beta_{ij}^t)\delta}{L}), q_{max}), & \text{if } i \in \bar{\mathcal{A}}_t \\ \min(q_i^{t-1} + \lambda_i^t, q_{max}), & \text{Otherwise.} \end{cases} \tag{4.4}$$

Let Γ and $\mathcal{P}$ be the set of valid CCA thresholds and transmit powers, respectively. Denote by π a policy, which is used by an AP to select a CCA threshold $\gamma^t \in \Gamma$ and transmit power $P^t \in \mathcal{P}$ for each time slot t. Let Ω be a collection of policies. The AP's goal is to optimize its reward or minimize its average queue length. Formally, for a given policy π running on an AP i, the expected reward is

$$R(\pi) = \lim_{|\mathcal{T}| \to \infty} \frac{1}{|\mathcal{T}|} \mathbb{E}^{\pi} \left[\sum_{t=1}^{|\mathcal{T}|} q_i^t \right]. \tag{4.5}$$

The problem is to find the optimal policy π^* that minimizes the objective $R(\pi)$. Mathematically,

$$\pi^* = \arg\min_{\pi \in \Omega} R(\pi). \tag{4.6}$$

4.2 A Semi-Markov Decision Process Model

This section first presents a brief background on Markov Decision Process (MDP) [51] and Semi-Markov Decision Process (Semi-MDP) [143]. Then, it will introduce the Hierarchical Reinforcement Learning (HRL) framework [141] and the Deep Q-Network algorithm [129].

4.2.1 MDP and Semi-MDP

An MDP provides a mathematical framework for decision making problems [51]. At a given time step t, a controller/agent observes its environment to obtain the current state $s_t \in \mathcal{S}$, where $\mathcal{S}$ is the set of states. Then, the agent chooses an action a_t from

the set of available actions $\mathcal{A}$ following a policy π, denoted as $a_t = \pi(s_t)$. The agent executes the action a_t to receive an instant reward $\mathcal{R}(s_t, a_t)$. Action a_t also causes a transition from state s_t to s_{t+1} according to the transition probability $\mathcal{P}(s_{t+1}|s_t, a_t)$. Define $V^\pi(s)$ as the value function that measures the expected long-term reward for a state s under policy π. It represents the cumulative discounted reward obtained by an agent that starts from state s and chooses actions as per policy π thereafter. Formally, $V^\pi(s)$ is given by

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{k=0}^{\infty} \gamma^k \mathcal{R}(s_{t+k}, \pi(s_{t+k})) \mid s_t = s\right], \tag{4.7}$$

where $\gamma \in [0, 1]$ is a discount factor.

An agent's objective is to find the optimal policy π^* that maximizes the value function for all states, denoted as V^*. This optimal policy π^* can be obtained by first solving to optimality Eq. (4.7) using Bellman equation [51]. For a given state $s_t \in \mathcal{S}$, V^* is given by

$$\begin{aligned} V^*(s_t) = \max_{a_t \in \mathcal{A}} \Big[& \mathcal{R}(s_t, a_t) \\ & + \gamma \sum_{s_{t+1} \in \mathcal{S}} \mathcal{P}(s_{t+1}|s_t, a_t) V^*(s_{t+1}) \Big]. \end{aligned} \tag{4.8}$$

Then, the optimal policy π^* for each state $s_t \in \mathcal{S}$ is given as

$$\begin{aligned} \pi^*(s_t) = \arg\max_{a_t \in \mathcal{A}} \Big[& \mathcal{R}(s_t, a_t) \\ & + \gamma \sum_{s_{t+1} \in \mathcal{S}} \mathcal{P}(s_{t+1}|s_t, a_t) V^*(s_{t+1}) \Big]. \end{aligned} \tag{4.9}$$

A key assumption of MDP is that the elapsed time to transition from the current to the next state is negligible [143]. However, for some problem instances, the transition time between states is a random variable. These problems can be represented as a Semi-MDP [143]. Specifically, Semi-MDP captures the fact that the current state s_t transitions to the next state $s_{t+\tau}$ after τ time steps. Then, the

transition probability is re-written as $\mathcal{P}(s_{t+\tau}, \tau | s_t, a_t)$, and the reward $\mathcal{R}(s_t, a_t, \tau)$ is the cumulative discounted reward over the transition time, which is given by

$$\mathcal{R}(s_t, a_t, \tau) = \sum_{k=t}^{t+\tau-1} \gamma^{k-t} \mathcal{R}(s_k, a_k). \tag{4.10}$$

As a result, the Bellman equation for solving optimal value function of Semi-MDP is

$$\begin{aligned} V^*(s_t) = \max_{a_t \in \mathcal{A}} \Big[& \mathcal{R}(s_t, a_t, \tau) \\ & + \sum_{\tau, s_{t+\tau} \in \mathcal{S}} \gamma^\tau \mathcal{P}(s_{t+\tau}, \tau | s_t, a_t) V^*(s_{t+\tau}) \Big], \end{aligned} \tag{4.11}$$

The optimal policy for Semi-MDP is thus, given by

$$\begin{aligned} \pi^*(s_t) = \arg\max_{a_t \in \mathcal{A}} \Big[& \mathcal{R}(s_t, a_t, \tau) \\ & + \sum_{\tau, s_{t+\tau} \in \mathcal{S}} \gamma^\tau \mathcal{P}(s_{t+\tau}, \tau | s_t, a_t) V^*(s_{t+\tau}) \Big]. \end{aligned} \tag{4.12}$$

4.2.2 Hierarchical Reinforcement Learning

The concept of *Option*, denoted as o, is introduced in [141] as a type of HRL framework to solve Semi-MDP problems. An Option o consists of a tuple $< I_o, \pi_o, \beta_o(s) >$, where $I_o \subseteq \mathcal{S}$ is a set of states in which an agent has available to an Option if its state s belongs to I_o, i.e., $s \in I_o$. Define π_o as the policy used for choosing actions if Option o is taken, and $\beta_o(s)$ is an indicator function that determines if an Option o is terminated in state s. An agent selects an Option o in a given state s_t, and after entering Option o, it selects an action following the policy π_o until Option o terminates according to $\beta_o(s_{t+\tau})$ after τ time steps. Next, the agent selects the next Option and continues until the Semi-MDP terminates.

The total reward for Option o is given by

$$\mathcal{R}(s_t, o) = \mathbb{E}^{\pi_o}\left[\sum_{k=t}^{t+\tau-1} \gamma^{k-t}\mathcal{R}(s_k, a_k)\right]. \tag{4.13}$$

Denote by $\mathcal{P}(s_{t+\tau}|s_t, o)$ the state transition probability from state s_t to $s_{t+\tau}$ after taking Option o. This probability is formally defined as

$$\mathcal{P}(s_{t+\tau}|s_t, o) = \sum_{\tau=1}^{\infty} \gamma^{\tau} p(s_{t+\tau}, \tau), \tag{4.14}$$

where the $p(s_{t+\tau}, \tau)$ is the probability that the Option o initiated at state s_t and terminates at $s_{t+\tau}$ after τ time steps.

Let μ be the policy that selects an Option for an agent in a given state, denoted as $o = \mu(s_t)$. Let $V^*_{\mathcal{O}}$ be the optimal value function over a set of available Options $\mathcal{O}$. This optimal value function is solved via Bellman equation as

$$V^*_{\mathcal{O}}(s_t) = \max_{o\in\mathcal{O}}\left[\mathcal{R}(s_t, o) + \sum_{s_{t+\tau}\in\mathcal{S}} \mathcal{P}(s_{t+\tau}|s_t, o)V^*_{\mathcal{O}}(s_{t+\tau})\right]. \tag{4.15}$$

The objective is to find the optimal policy $\mu^*(s_t)$ over $\mathcal{O}$ for each state, which is formalized as

$$\mu^*(s_t) = \arg\max_{o\in\mathcal{O}}\left[\mathcal{R}(s_t, o) + \sum_{s_{t+\tau}\in\mathcal{S}} \mathcal{P}(s_{t+\tau}|s_t, o)V^*_{\mathcal{O}}(s_{t+\tau})\right]. \tag{4.16}$$

4.2.3 Deep Q-Network

Recall that the transition probability between states is unknown. Therefore, the Deep Q-Network (DQN) [129] is used to learn both the optimal policy over Option μ and the policy of each Option π_o.

A DQN is able to learn the optimal policy by approximating the optimal state-action value (also known as the Q-value) by interacting with the environment [129]. Specifically, define Q-network $\mathcal{Q}(s_t, a_t, \theta)$, where θ represents the parameter of a

Deep Neural Network (DNN). The input to a Q-network is the state, and the output corresponds the Q-value of each action. A DQN consists of two Q-networks, namely the evaluation Q-network $\mathcal{Q}(s_t, a_t, \theta)$ and the target Q-network $\mathcal{Q}(s_t, a_t, \theta')$. The data used for updating the said Q-networks is randomly sampled from a *memory* buffer $\mathcal{M}$ that stores the historical pairs of state, action, reward and next state. For each iteration, a DQN uses the Bellman equation to update its θ to minimize a loss function as per

$$\mathcal{L}(\theta) = \mathbb{E}[(\mathbf{y} - \mathcal{Q}(s_t, a_t, \theta)^2], \tag{4.17}$$

where

$$\mathbf{y} = \mathcal{R}(s_t, a_t) + \gamma \max_{a \in \mathcal{A}} \mathcal{Q}(s_{t+1}, a, \theta'). \tag{4.18}$$

For every K iterations, a DQN replaces θ' with θ, where K is a pre-defined integer.

4.3 An HRL based approach

This section first instantiates a Semi-MDP model based on the Option framework [141]. Next, it proposes an HRL based approach that learns to optimize the transmit power and CCA threshold of an AP. The use of an HRL based approach is critical, especially when there is a large action space, i.e., $|\mathcal{A}| = |\Gamma| \cdot |\mathcal{P}|$ where Γ and $\mathcal{P}$ is the set of CCA thresholds and transmit powers, respectively. Therefore, standard DRL algorithm, such as DQN, may have a poor performance as it cannot efficiently learn the optimal policy from a large action space [144]. To this end, an HRL based approach improves the learning efficiency of an agent or AP by dividing the said optimization problem into two layers and learning the optimal policy for each layer [143].

4.3.1 A Semi-MDP model

The following Semi-MDP model considers an AP i as an HRL agent. As per Fig. 4.1, AP i observes the state s_t from the environment, and chooses an HRL action. The

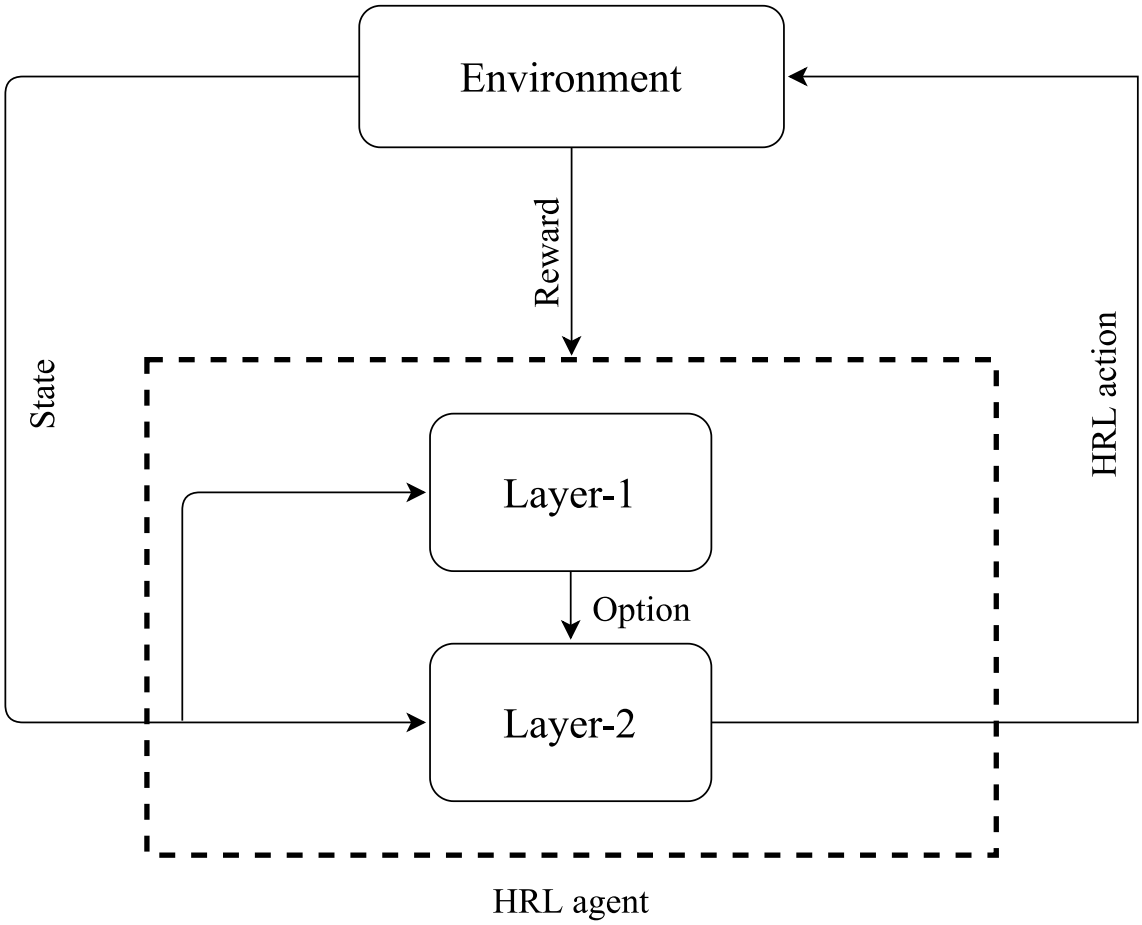

Figure 4.1: A flowchart of the proposed HRL model. In each time slot, the environment state is fed to both Layer-1 and Layer-2. In Layer-1, the HRL agent uses policy μ to select an Option o for the input state if there is currently no selected Option. Otherwise, the previously selected Option remains until terminated. Next in Layer-2, the HRL agent uses policy π_o, which is provided by the selected Option o, to select an action for the input state. The selected Option action are then executed, which then yields a reward and a new state.

HRL action corresponds to **Option** o and **action** a_t. Specifically, AP i will first use policy μ to select and initiate Option o in Layer-1. Option o remains fixed until it is terminated according to $\beta_o(.)$. Then, AP i uses policy π_o to select action a_t in Layer-2. Note, policy π_o is provided by Option o as specified in Section 4.2.2. The selected Option o and action a_t are the output of an HRL action, which are then executed to obtain a new state and reward.

For AP i and user j in time slot t, the state, option, action, reward and transition probability are defined as follows:

- **State**: The state s_t is a tuple that contains the queue length and the in-

terference experienced by AP i and user j in the last time slot, i.e., $s_t = (q_i^{t-1}, I_i^{t-1}, I_j^{t-1})$. Note, the interference I_j^{t-1} experienced by user j can be collected using IEEE 802.11k [52].

- **Option**: An Option o represents a transmit power $P_i^t \in [P_{min}, P_{max}]$ for AP i. Each Option o terminates when the AP has transmitted P packets since the Option is initiated, where P is an integer that is used to terminate Option o. The value of P is obtained through experimentation. The indicator function $\beta_o(.)$ is given by

$$\beta_o(s_{t+\tau}) = \begin{cases} 1, & \text{if} \sum_{k=t}^{t+\tau-1} \frac{r_{ij}(\beta_{ij}^k)\delta}{L} \geq P, \\ 0, & \text{Otherwise}, \end{cases} \tag{4.19}$$

  where τ is the number of time slots that have elapsed since the initiation of Option o.

- **Action**: An action a_t corresponds to a value of CCA threshold $\gamma_i^t \in [\gamma_{min}, \gamma_{max}]$. Each action a_t terminates at the end of time slot t.

- **Reward**: The reward for an action $\mathcal{R}(s_t, a_t)$ is the number of packets transmitted in time slot t. Further, the reward is set to -10 if the queue overflows, i.e., $q_i^t \geq q_{max}$. The reward for an Option $\mathcal{R}(s_t, o)$ corresponds to the cumulative discounted reward recorded during the execution of the Option o, and is calculated as per Eq. (4.13).

- **Transition probability function**:The transition probability function $\mathcal{P}(s_{t+1}|s_t, a_t)$ is unknown, which means the proposed approach is model-free. This is because the packets arrival and the interference experience by an AP and users are not available in practice.

4.3.2 HRL based approach

This chapter proposes to use DQNs to learn the optimal policy for both Layer-1 and Layer-2. Algorithm-3 shows the steps of the proposed HRL approach. First, in Layer-1, an AP observes the state s_t and selects an Option by calling the function **EpsilonGreedy**(.). Here, the function **EpsilonGreedy**(.) chooses an Option following the $\epsilon - greedy$ strategy. That is, AP i randomly chooses an Option $o \in \mathcal{O}$ with an exploration probability ϵ_μ. Otherwise, the Option with the highest Q-value $\arg\max_{o \in \mathcal{O}} \mathcal{Q}(s_t, o, \theta_\mu)$ is taken. Further, to ensure convergence, the value of ϵ_μ decreases after each time of calling **EpsilonGreedy**(.), see line 7, where ϵ_{dec} is the decay rate, and ϵ_{min} is the minimum exploration probability. AP i then enters Layer-2. In Layer-2, AP i observes state $s_{t+\tau}$ and calls the function **EpsilonGreedy**(.) to select an action $a_{t+\tau}$. Here, the symbol τ represents the number of time slots that have elapsed since an Option is initiated. The action $a_{t+\tau}$ is then executed to obtain the reward $\mathcal{R}(s_{t+\tau}, a_{t+\tau})$ and next state $s_{t+\tau+1}$. AP i collects the state-action-reward pair $(s_{t+\tau}, a_{t+\tau}, \mathcal{R}(s_{t+\tau}, a_{t+\tau}), s_{t+\tau+1})$ and uses it to update θ_o. AP i then goes back to line 10 and selects next action in Layer-2 until Option o terminates according to $\beta_o(.)$, where it returns a value of one if Option o is to be terminated. Upon the termination of Option o, AP i calculates the reward $\mathcal{R}(s_t, o)$ for Option o, and uses it to update θ_μ for policy μ. Then, AP i executes line 4 and selects another Option until the learning process ends.

4.4 Evaluation

Simulations are conducted using Python 3.7.7, and Keras 2.2.5 [139] with TensorFlow 1.14 [138] on an Intel i7-8700 computer with 16 GB RAM. All parameter values are listed in Table 4.2 and 4.3. The simulations consider a single AP, labeled as AP i, that runs the proposed HRL based approach. AP i has four associated users, and they are placed uniformly within 5 m range. Both AP i and its associated users experience interference from neighboring cells that is measured using Eq. (4.3), where

Algorithm 3: An HRL based learning approach.

Initialize: $\theta_\mu, \theta'_\mu, \mathcal{M}_\mu, \epsilon_\mu$ for policy μ in Layer-1
Initialize: $\{(\theta_o, \theta'_o, \mathcal{M}_o, \epsilon_o \mid o \in \mathcal{O}\}$ for each policy π_o in Layer-2
$t = 1$
while *not terminated* **do**
  /********** Layer-1 **********/
  $\tau = 0$
  Observe s_t
  $o = EpsilonGreedy(s_t, \mathcal{O}, \theta_\mu, \epsilon_\mu)$
  $\epsilon_\mu = \max[\epsilon_\mu \epsilon_{dec}, \epsilon_{min}]$
  Execute o
  /********** Layer-2 **********/
  while *o not terminated* **do**
    Observe $s_{t+\tau}$
    $a_{t+\tau} = EpsilonGreedy(s_{t+\tau}, \mathcal{A}, \theta_o, \epsilon_o)$
    $\epsilon_o = \max[\epsilon_o \epsilon_{dec}, \epsilon_{min}]$
    Execute $a_{t+\tau}$
    Observe $\mathcal{R}(s_{t+\tau}, a_{t+\tau})$ and $s_{t+\tau+1}$
    Store $(s_{t+\tau}, a_{t+\tau}, R(s_{t+\tau}, a_{t+\tau}), s_{t+\tau+1})$ into $\mathcal{M}_o$
    Update θ_o as per Eq. (4.17)
    Every K iterations, $\theta'_o = \theta_o$
    $\tau = \tau + 1$
  end
  /********** End of Layer-2 **********/
  Calculate $\mathcal{R}(s_t, o)$ as per Eq. (4.13)
  Observe $s_{t+\tau}$
  Store $(s_t, o, \mathcal{R}(s_t, o), s_{t+\tau})$ into $\mathcal{M}_\mu$
  Update θ_μ as per Eq. (4.17)
  Every K iterations, $\theta'_\mu = \theta_\mu$
  $t = t + \tau$
  /********** End of Layer-1 **********/
end

I use another AP to simulate and induce the said interference. The neighboring AP is able to induce the aggregated amount of interference from a dense network due to shadowing effect. The achievable data rate for a transmission with a given SINR is discretized into nine levels as per Table 4.4.

The following algorithms/rules are implemented and compared:

- **HRL-TPCCCA**: An HRL based algorithm. For AP i, the set of Options at Layer-1 corresponds to different transmit power levels, and the set of actions at Layer-2 corresponds to the set of CCA thresholds.

Table 4.2: Parameter values.

Environment Parameters	**Value(s)**
Number of APs	2
Number of users per AP	4
Distance between APs (meter)	20
Available CCA threshold (dBm)	-80 to -20
Available transmit power (dBm)	0 to 25
Carrier frequency (GHz)	2.4
Channel bandwidth (MHz)	20
Path loss exponent	3.5
Path loss reference distance d_0 (meter)	1 (dB) [136]
Path loss at reference distance $PL(d_0)$	40.05 (dB) [136]
Environment noise (dBm)	-90
Initial queue length of each AP (packets)	2000
Maximum queue length of each AP (packets)	16000
Packet size (byte)	2304
Duration of each time slot (seconds)	0.0045
RTS/CTS mechanism	Disabled

- **Fix Layer One (HRL-FLO)**: AP i adopts HRL-TPCCCA, but fixes the transmit power to 25 dBm at Layer-1, and only learns to select CCA threshold at Layer-2.

- **Fix Layer Two (HRL-FLT)**: AP i adopts HRL-TPCCCA, but fixes the CCA threshold to -82 dBm at Layer-2, and only learns to select transmit power at Layer-1.

- **HRL-CCATPC**: An HRL based algorithm. For AP i, the set of Options at Layer-1 corresponds to the set of CCA thresholds, and the set of actions at Layer-2 corresponds to the set of transmit power levels.

- **Dynamic Sensitivity Control (DSC)** [46]: AP i periodically monitors the successful transmission rate. If the successful transmission rate falls below a pre-defined threshold, AP i increases its CCA threshold with a pre-defined step size of 5 dB. Otherwise, it decreases its CCA threshold with the pre-defined step size.

Table 4.3: Parameter values used by HRL agents.

Parameters	**Value(s)**
Policy network	Deep Q-Network
Neural network	Fully connected
Number of hidden layers	2
Number of neurons in each layer	32
Learning rate α	10^{-3}
Reward discount factor γ	0.95
Initial exploration rate ϵ	1
Final exploration rate ϵ_{min}	0.01
Decay factor ϵ_{dec}	0.999
Memory size (samples)	1024
Batch size for training (samples)	32
Target network replacing frequency (time slots)	100
Option termination threshold (packets)	100
Activation function	ReLU
Optimizer	Adam [137]

- **Legacy**: AP i uses a fixed transmit power of 25 dBm and CCA threshold of -82 dBm.

The conducted simulation has three parts: *Training*, *Test* and *Multiple APs*. In *Training*, HRL agents are trained over a fixed network environment where each AP has saturated traffic, i.e., the queue length of an AP never drops to zero. Each episode is set to have 3000 time slots. For each episode, the following metrics are collected:

- **Average number of transmitted packets**: This is the average number of packets that are transmitted successfully per time slot.

- **Average success ratio**: This is defined as the average ratio of time slots where the AP successfully accesses the channel over the total number of time slots. This metric is measured when the queue of the AP is not empty.

In *Test*, the well-trained HRL agents are tested to study two factors: traffic model and interference intensity. For the traffic model, the number of packets arriving at

Table 4.4: Data rate as per SINR datasheet [5, 6]. The datasheet in [5] outlines the relationship between SNR and MCS index, while the author in [6] provides a table that maps a data rate to a given MCS index. By jointly using [5] and [6], the data rate in this table is obtained.

SINR (dB)	Data Rate (Mbps)	SINR (dB)	Data Rate (Mbps)
2	7.2	18	57.8
5	14.4	20	65
9	21.7	25	72.2
11	28.9	29	86.7
15	43.3		

each time slot is sampled from a Poisson distribution, where the arrival rate varies from five to 40. Then in Section 4.4.2.3, a traffic trace data from [145] is used. To study the impact of interference intensity, the distance between APs varies from ten to 30. The aim is to study the impact of varying amount of cumulative interference to AP i, which is equivalent to varying number of neighboring APs. The average queue length of AP i is collected for a test period with T time slots, and it is calculated as $\frac{1}{T}\sum_{t=1}^{T} q_i^t$.

Lastly, in *Multiple APs*, two interfering APs are placed 20 m apart; each runs the proposed HRL approach and has four associated users. The two APs are trained to compare their average throughput and success ratios.

The neural network used in each simulation consists of two fully connected hidden layer. Each layer has 32 neurons with ReLU as activation function; see Table 4.3 for parameter values.

4.4.1 Training stage

Referring to Fig. 4.2 and 4.3, HRL-TPCCCA, HRL-CCATPC and HRL-FLO have the best performance. From Fig. 4.2, the average number of transmitted packets for HRL-TPCCCA, HRL-CCATPC and HRL-FLO increases and converges to 20.98. These three algorithms are able to learn the optimal CCA threshold over time and transmit a high average number of packets. Indeed, the optimal CCA threshold

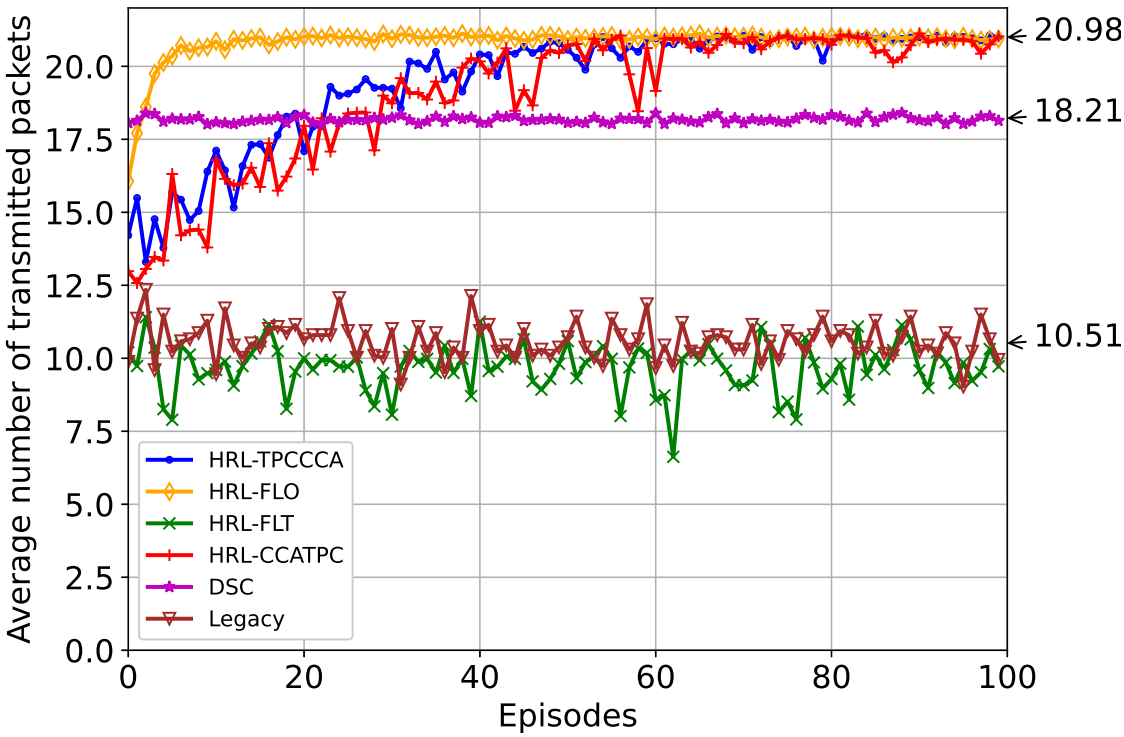

Figure 4.2: Elapsed time versus the number of transmitted packets.

allows APs to transmit in each time slot. This can be observed in Fig. 4.3, where the average success ratio for HRL-TPCCCA, HRL-CCATPC and HRL-FLO converges to one. In addition, HRL-FLO converges faster than HRL-TPCCCA and HRL-CCATPC, whereby it converges to 20.98 within 10 episodes while HRL-TPCCCA and HRL-CCATPC converge at around 40 episodes. The reason is because HRL-FLO uses a fixed transmit power of 25 dBm, and it only needs to learn to assign CCA thresholds. To this end, the optimal policy is to set a high CCA threshold in order to transmit in each time slot. By contrast, HRL-TPCCCA and HRL-CCATPC need to learn both the CCA threshold and transmit power of an AP, which require more episodes to converge.

The second best algorithm is DSC, where it transmits 18.91 packets on average, meaning HRL-TPCCCA, HRL-CCATPC and HRL-FLO outperform DSC by 17.14%. This is because HRL-TPCCCA and HRL-CCATPC assign the best trans-

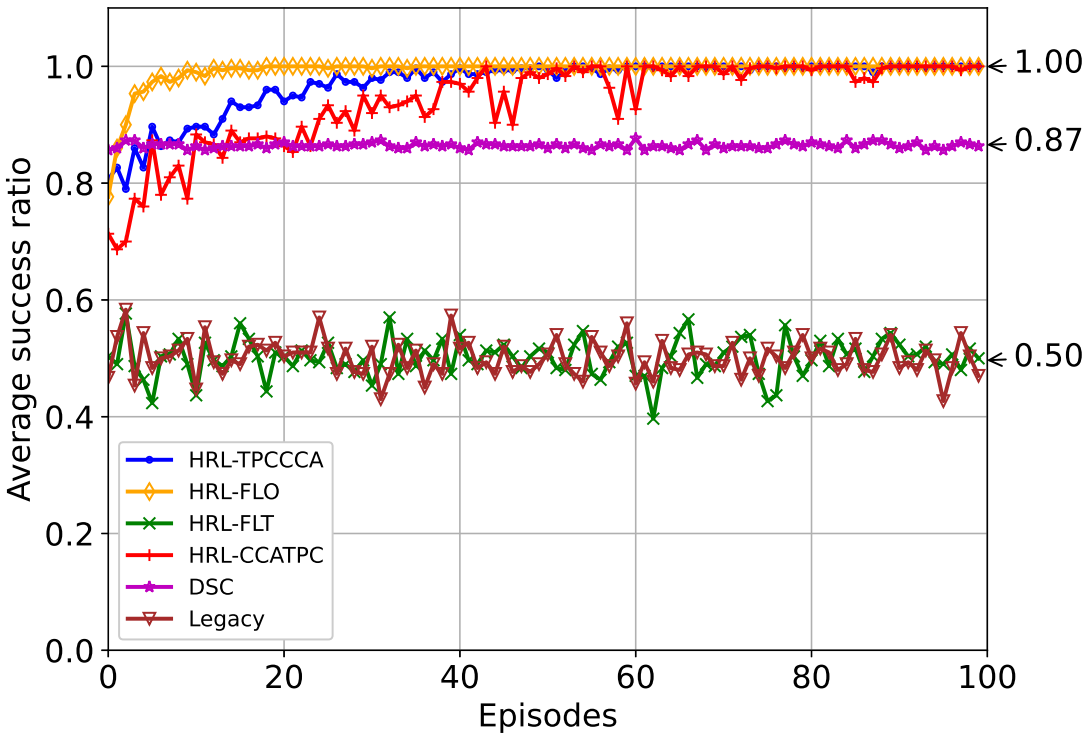

Figure 4.3: Elapsed time versus channel access success.

mit power and CCA threshold, and HRL-FLO uses the best CCA threshold. In contrast, DSC adjusts the CCA threshold of an AP based on historical channel access data. It will only increase an AP's CCA threshold when the ratio of successful channel access drops below a pre-defined threshold. Therefore, DSC has a lower average success ratio than HRL-TPCCCA, HRL-CCATPC and HRL-FLO. As shown in Fig. 4.3, HRL-TPCCCA, HRL-CCATPC and HRL-FLO have value of 1.0 for average success ratio and DSC has a value of 0.85. As a result, DSC has a lower number of transmitted packets than HRL-TPCCCA, HRL-CCATPC and HRL-FLO.

HRL-FLT and Legacy have the worst performance. Legacy has a performance of 10.51 packets per time slot. This is because Legacy has a fixed transmit power of 25 dBm and CCA threshold of -82 dBm. This CCA threshold is conservative and prevents AP i transmitting parallelly with its neighbor. This can be observed

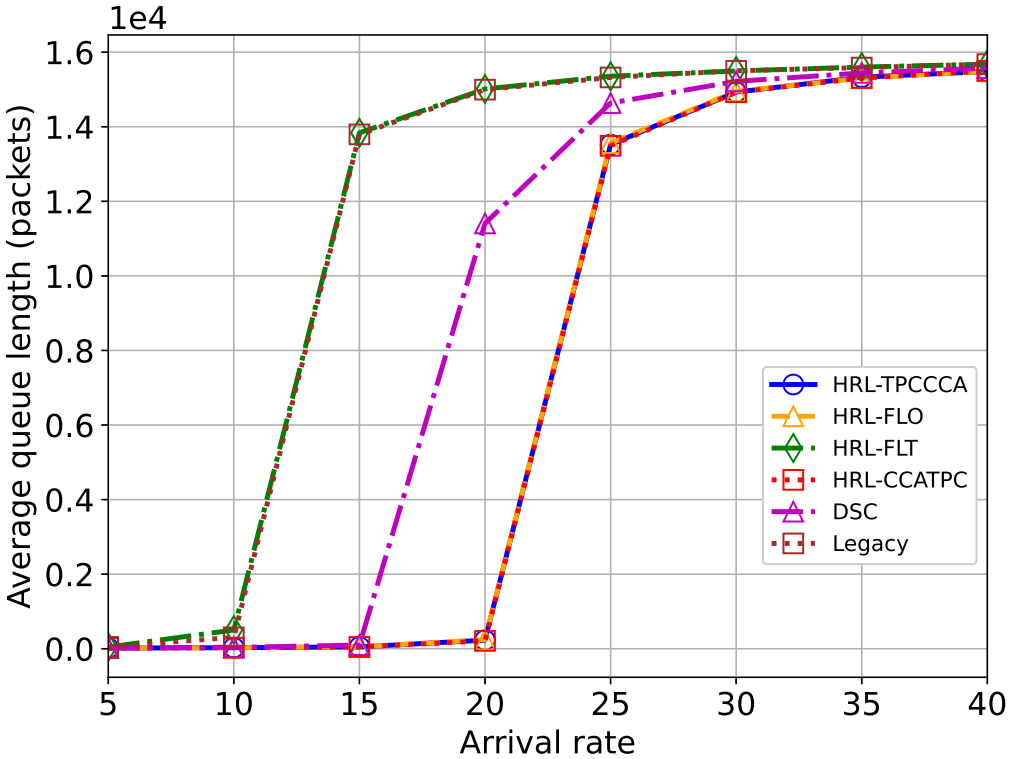

Figure 4.4: Impact of arrival rate on average queue length.

in Fig. 4.3 where the average success ratio of Legacy is 0.51. Consequently, the average number of transmitted packets is low. However, it shows that HRL-FLT has the same performance as Legacy. This is because HRL-FLT only learns to adjust transmit power, and it has a fixed CCA threshold of -82 dBm. Therefore, it is not able to increase its opportunity to transmit. Therefore, the optimal option for HRL-FLT is to transmit with the highest transmit power when it is able to transmit, which means HRL-FLT uses the same CCA threshold and transmit power as Legacy. Therefore, both HRL-FLT and Legacy have the lowest throughput.

4.4.2 Test stage

4.4.2.1 Poisson Traffic Model

In this experiment, Poisson traffic model is considered with arrival rate that increases from five to 40; it runs 10000 time slots for each arrival rate to collect the results.

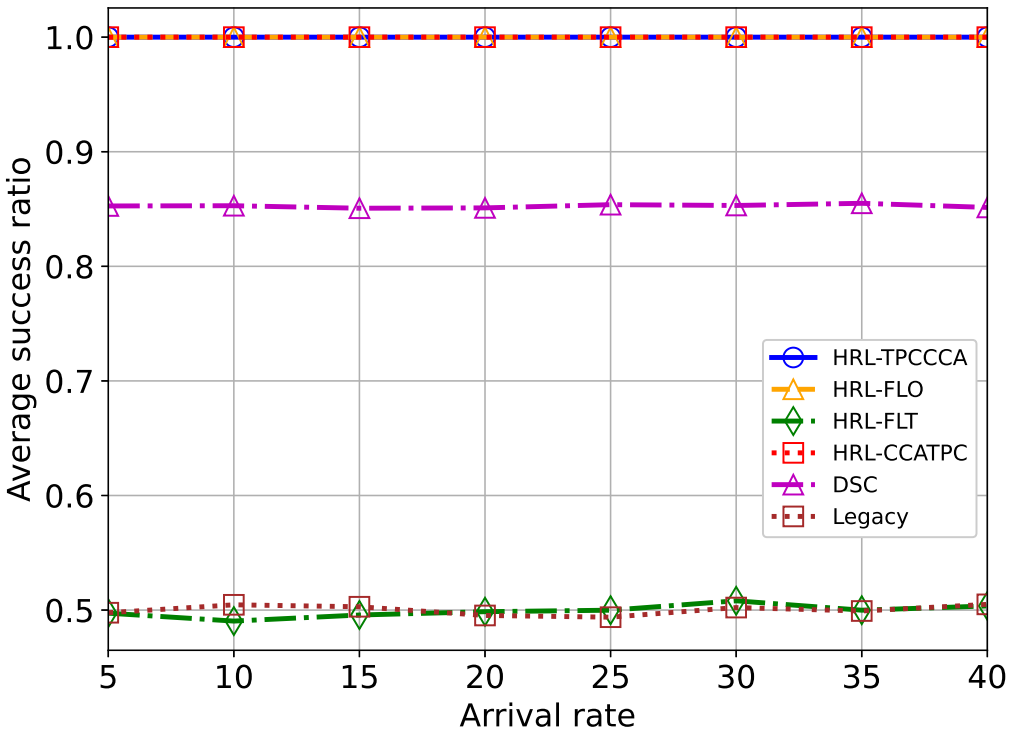

Figure 4.5: Impact of arrival rate on channel access success.

Fig. 4.4 and 4.5 show the impact of arrival rates. From Fig. 4.4, the average queue length for HRL-TPCCCA, HRL-CCATPC and HRL-FLO is around zero when the arrival rate is not larger than 20. As shown in Fig. 4.2, these three algorithms are able to achieve a throughput of 20.98 packets over time. Therefore, they are able to empty the queue of an AP when the traffic is low. However, as the arrival rate increases, APs do not have sufficient throughput to deliver arriving packets, which leads to a significant increase in queue length. This can be observed in Fig. 4.4, where the average queue length increases from 220 to 15490 when the arrival rate increases from 20 to 40. A similar trend for DSC is observed, where it has a throughput of 17.91 in Fig. 4.2. The average queue length for DSC reduces to zero when the arrival rate is below 17.91. On the other hand, for HRL-FLT and Legacy, they have a low throughput of 10.51. Therefore, they are not able to reduce their queue length when arrival rate exceeds 10.51. From Fig. 4.5, the average success ratio does not change

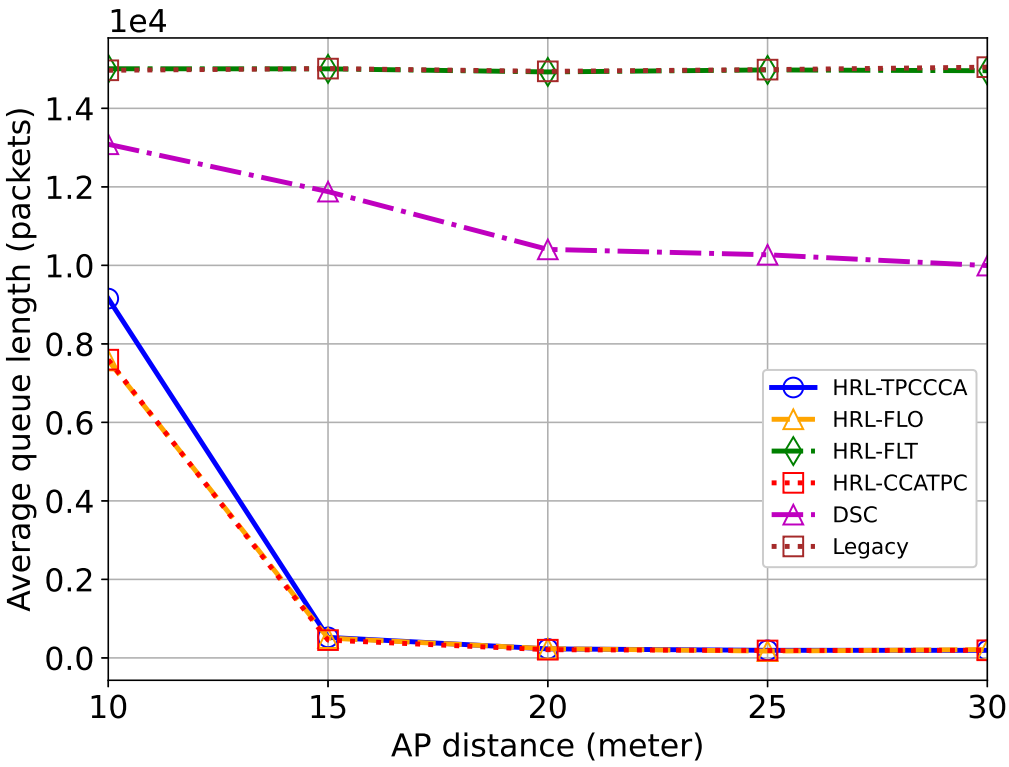

Figure 4.6: Increasing AP distance versus average queue length.

with different arrival rate. This is because the arrival rate has no impact on the interference intensity at APs, and thus, has no impact on spatial reuse.

4.4.2.2 Varying AP distance

This section studies the impact of interference intensity. The arrival rate is fixed to 20 packets, and the distance between APs varies. For each AP distance, all compared algorithms/rules run 10000 time slots to collect the average results. From Fig. 4.6 and 4.7, the average queue length decreases as the distance between APs increases. The average queue length for HRL-TPCCCA, HRL-FLO and HRL-CCATPC is around 8000 when the AP distance is 10 m. This is because the interference from the neighboring AP is high when the distance between APs is small. High interference leads to low SINR for each transmission, and causes a low data rate for the AP. In addition, high interference may exceed the CCA threshold on the AP, therefore

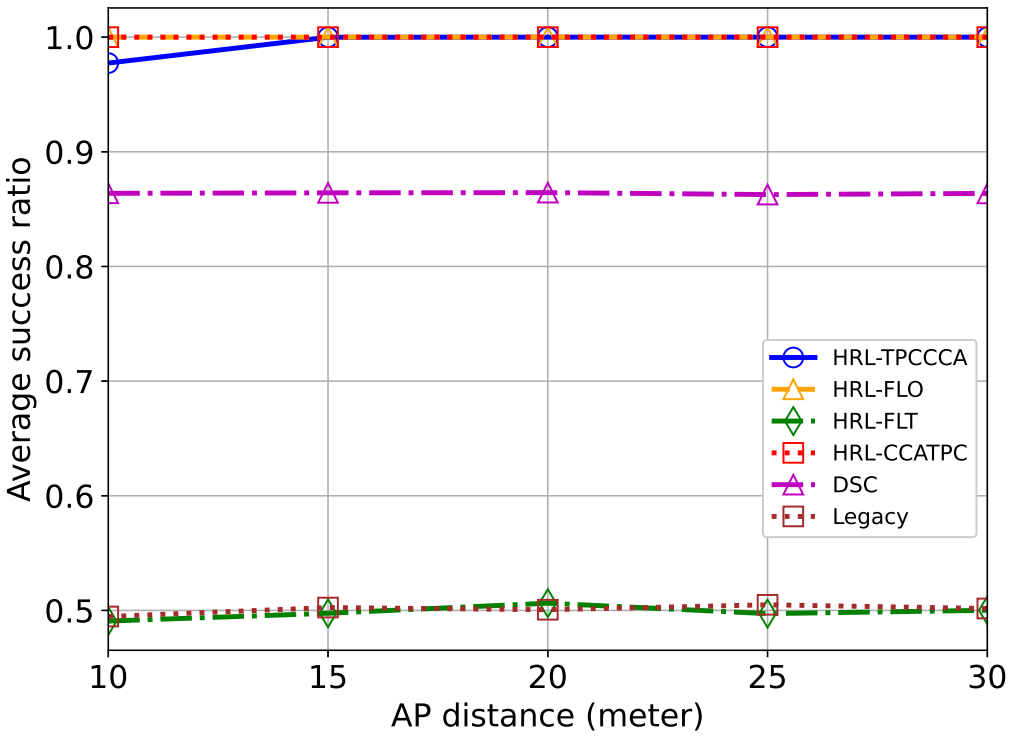

Figure 4.7: Increasing AP distance versus channel access success.

preventing the AP from transmitting. Hence, APs do not have sufficient capacity to transmit arriving packets, which results in a high average queue length. In Fig. 4.6, the queue length of HRL-TPCCCA, HRL-FLO and HRL-CCATPC decreases to around 200 packets when distance between APs increases to 20 m. The reason is because there is less mutual interference between APs as the distance between them becomes larger. Therefore, APs are able to transmit with a high data rate to empty their queue. From Fig. 4.7, the average success ratio for HRL-TPCCCA, HRL-FLO and HRL-CCATPC remains fixed. They all have a value of 1.0 for average success ratio, meaning they are always able to transmit. This is because these three algorithms learn to adjust the CCA threshold to increase transmission opportunity.

HRL-TPCCCA, HRL-CCATPC and HRL-FLO have better performance than DSC. The average queue length of DSC declines from 13134 packets to 10223 packets as the distance between APs increases from 10 m to 30 m. The average queue length

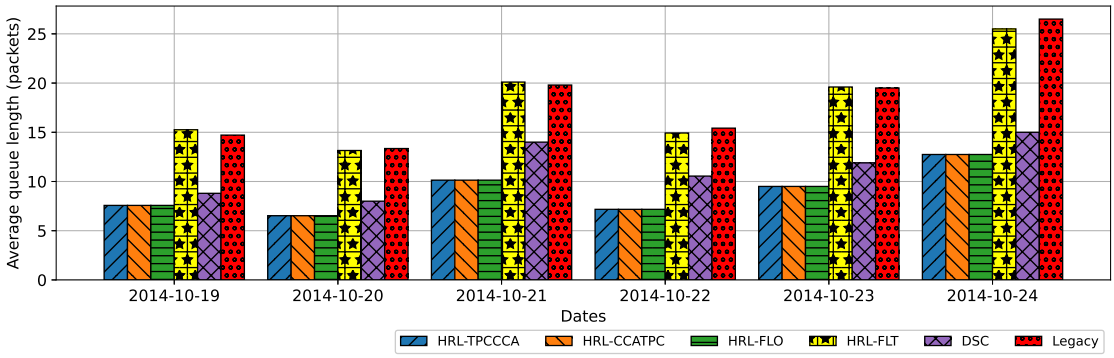

Figure 4.8: Average queue length with different trace data.

of the three HRL based algorithms is 9240 packets lower than DSC, which means the three HRL based algorithms outperform DSC by 85% on average. Therefore, an AP using an HRL based algorithm experiences lower queuing delays than DSC.

Fig. 4.6 further shows that the average queue length for Legacy and HRL-FLT is not affected by the distance between APs. The average queue length for both of them is near 16000. This is because these two algorithms are not able to adjust CCA threshold to improve spatial reuse. Although the distance between APs increases from 10 to 30 m, the resulting interference from a neighboring AP, with 25 dBm transmit power, decreases from -50.05 to -66.75 dBm, which is higher than -82 dBm. This means AP i is not able to transmit concurrently with a neighboring AP to improve spatial reuse. This can be observed from Fig. 4.7 where Legacy and HRL-FLT both have the lowest average success ratio of 0.5. Consequently, APs using Legacy and HRL-FLT are not able to efficiently deliver packets, which results in queue overflow.

4.4.2.3 Trace Data

This section considers the trace data obtained from [145]. Fig. 4.8 shows the average queue length of each algorithm with different traffic trace data. HRL-TPCCCA, HRL-CCATPC and HRL-FLO always have the lowest queue length. i.e., their average queue length is 8.9. The second-best algorithm is DSC, with an average queue length of 11.78; i.e., the three HRL-based algorithms outperform DSC by 24.4% on

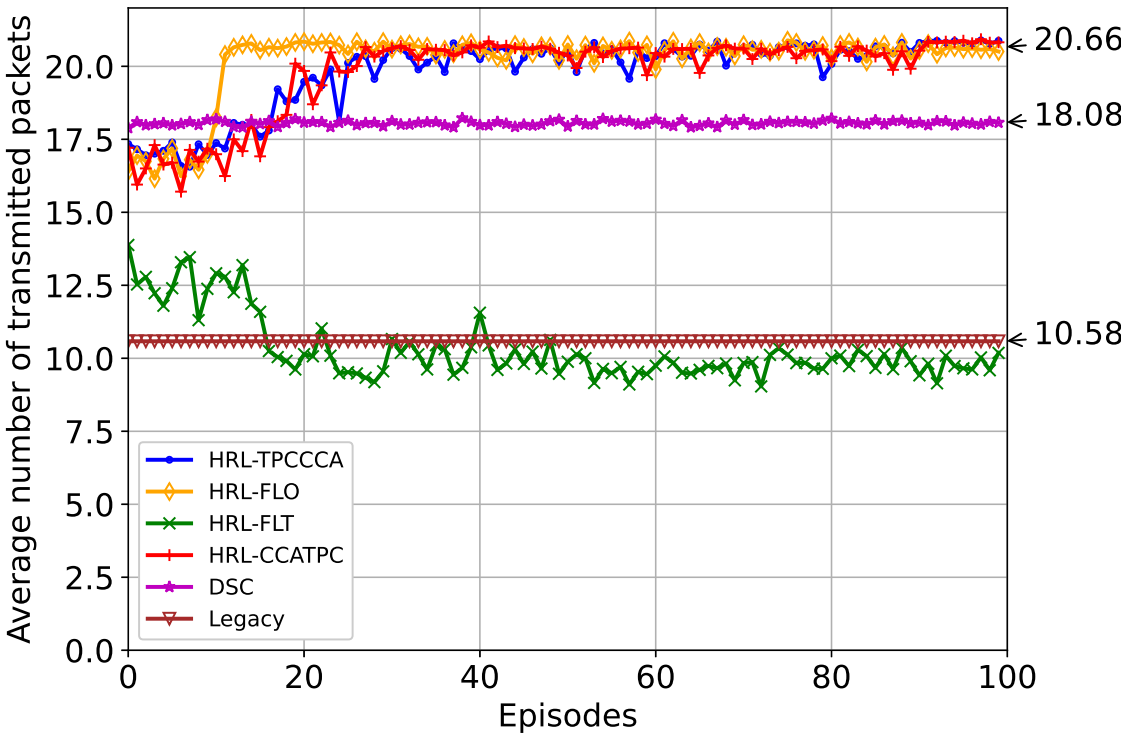

Figure 4.9: Elapsed time versus average number of transmitted packets of two APs.

average. HRL-FLT has a similar performance with Legacy. They all have an average queue length of 18.7. As shown in Fig. 4.8, HRL-TPCCCA and HRL-CCATPC always have similar performance. There is significant performance difference between HRL-FLO and HRL-FLT. Referring to Fig. 4.8, the average queue length of HRL-FLO is shorter than that of HRL-FLT by 52.4%. This means optimizing the CCA threshold of an AP as opposed to its transmit power delivers more packets.

4.4.3 Multiple APs

Here, a multi-agent scenario with interfering APs is studied. Note that the two APs *do not* cooperate with each other and learn their optimal policy independently using the proposed HRL based approach. Fig. 4.9 and 4.10 show the average number of transmitted packets and average success ratio of the two APs, respectively. From Fig. 4.9 and 4.10, HRL-TPCCCA, HRL-CCATPC and HRL-FLO are able to

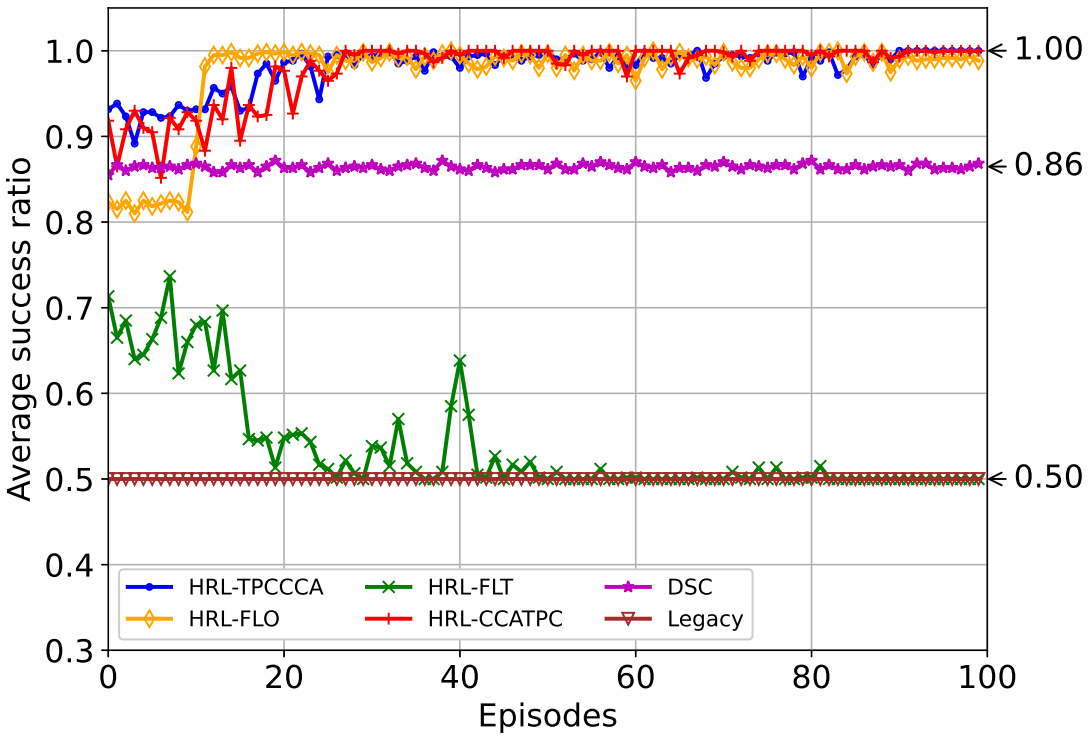

Figure 4.10: Elapsed time versus average success ratio of two APs.

achieve the highest success ratio and throughput over time. In Fig. 4.10, the average success ratio for HRL-TPCCCA, HRL-CCATPC and HRL-FLO converges to one after training. In Fig. 4.9, the average number of transmitted packets for HRL-TPCCCA, HRL-CCATPC and HRL-FLO increases and converges to 20.66 packets per time slot. Note that this value is the average number of transmitted packets for two APs, meaning both of them are able to transmit 20.66 packets on average in each time slot. This shows that an AP that runs the proposed HRL approach is able to learn the optimal policy in a multi-agent non-cooperative environment. The second best algorithm is DSC; it is able to transmit 18.08 packets per time slot and has a average success ratio of 0.86. Further, HRL-FLT has a decreasing throughput. The average number of transmitted packets for HRL-FLT decreases from 13.01 to 9.58 packets per time slot, and average access ratio decreases from around 0.7 to 0.5. Lastly, Legacy is able to transmit 10.58 packets per time slot and has a value of

0.5 for the average success ratio. This is because with a transmit power of 25 dBm and a CCA threshold of -82 dBm, only one AP is allowed to transmit in each time slot.

4.5 Conclusion

This chapter aims to improve spatial reuse and minimize the queue length of APs with random traffic arrival. Each AP runs an HRL agent that learns the optimal transmit power and CCA threshold. Advantageously, each AP only requires local information such as interference and queue length. The simulation results indicate that the proposed HRL based approach is able to learn the optimal transmit power and CCA threshold. Experiments over traffic trace data indicate that the average queue length of an HRL-equipped AP is shorter than DSC and Legacy by 24.4% and 52.4%, respectively.

Both Chapter 3 and 4 consider methods to improve spatial reuse. However, they only consider optimizing over one single channel. Current WiFi networks support channel bonding to improve the per transmission capacity. To this end, the next chapter considers optimizing spatial reuse over multiple channels.

Chapter 5

Joint spatial reuse and channel bonding optimization

As discussed in Chapter 1 and 2, current WiFi networks support channel bonding to boost per transmission capacity. A number of prior works have considered developing channel bonding policies to increase network capacity. However, the majority of them require global information and aim to reduce or avoid interference. Further, they only consider bonding adjacent channels. To this end, this chapter addresses the problem of optimizing channel bonding with the assistant of spatial reuse. It considers adjusting the CCA threshold and transmit power used on each bonded channel to improve the spatial reuse on multiple channels. Further, it considers bonding both adjacent and non-adjacent channel bonding for an AP; this is now supported in IEEE 802.11ax [18]. Lastly, it considers random traffic arrival.

This chapter outlines a distributed RL approach that runs on an AP. The AP is only aware of is local information and learns to select a set of channels, accompanied with the transmit power and CCA threshold allocated on each selected channel. The AP has random traffic arrival and its aim hence, is to maximize its throughput and minimize its queue length. The challenge is that the traffic arrival and inter-

ference from neighboring cells are both random, which impacts the queue length and throughput. Therefore, this chapter formulates the optimization problem as a three-layer MDP, which is solved by a three-tier learning approach.

The rest of this chapter is organized as follows. Section 5.1 presents the system model and problem, and Section 5.2 outlines a brief background of MDP, DQN and Deep Deterministic Policy Gradient (DDPG). After that, Section 5.3 provides details about the proposed three-tier learning approach. Lastly, Section 5.4 presents simulation results, and Section 5.5 concludes this chapter.

5.1 System Model

Table 5.1 lists the notations. Time is divided into T time slots; each time slot has a fixed length of δ seconds and it is indexed by t. An AP i has a set of users $\mathcal{U}$. The Euclidean distance between AP i and a user $u \in \mathcal{U}$ is d_{iu}. There are N channels and the set of channels is denoted as $\mathcal{C}$; each channel has a fixed bandwidth of B MHz. In each time slot t, AP i transmits to a user u over a set of channels $\mathcal{C}_i^t \subseteq \mathcal{C}$. The interference experienced by AP i and user u is caused by a set of neighboring APs, denoted as $\mathcal{N}_i$. Further, let $\mathcal{C}_j^t$ be the set of channels used by a neighboring AP $j \in \mathcal{N}_i$ in time slot t.

All APs use CSMA/CA for channel access. Specifically, AP i uses its CCA threshold γ_c^t to determine whether it is allowed to transmit on channel c in time slot t. Moreover, denote by $\mathcal{N}_i^t \subseteq \mathcal{N}_i$ as the set of neighboring APs that are transmitting at the beginning of time slot t. Therefore, on channel c, the set of neighboring APs that are transmitting at the beginning of time slot t is given by $\mathcal{N}_c^t = \{j | j \in \mathcal{N}_i^t \wedge c \in \mathcal{C}_j^t\}$.

This chapter considers block fading where the channel gain is fixed within one time slot and differs across time slots. The channel gain from AP i to user u in time slot t is denoted as g_{iu}^t, and it is calculated through the well-known Log-distance

Table 5.1: Parameter values.

Notation	Explanation
i	AP i
$\mathcal{U}$	The set of users associated with AP i
$\mathcal{C}$	The set of channels
$\mathcal{C}_i^t$	The set of channels used by AP i in time slot t
$\mathcal{N}_i$	The set of neighboring APs of AP i
$\mathcal{N}_c^t$	The set of neighboring APs that are transmitting on channel c in time slot t
B	The bandwidth of a single channel (in MHz)
T	Total number of time slots
δ	Length of each time slot (in seconds)
d_{iu}	The Euclidean between AP i and user u
g_{iu}^t	The channel gain from AP i to user u in time slot t
I_{uc}^t	The interference experienced by user u on channel c in time slot t
N_b	The ambient noise power density (in Watt/Hz)
β_{uc}^t	SINR from AP i to user u on channel c in time slot t
r_c^t	The data rate of AP i on channel c in time slot t
r_i^t	The aggregated data rate of AP i in time slot t
γ_c^t	The CCA threshold of AP i on channel c in time slot t
P_{ic}^t	The transmit power of AP i on channel c in time slot t
q_i^t	The queue length of AP i in time slot t
λ_i^t	The number of packets arriving at AP i in time slot t
L	Packet size (in bits)

path loss model (in dB) [142], that is given by

$$\mathrm{PL}(d_{iu}) = \mathrm{PL}(d_0) + 10\omega \log_{10}\left(\frac{d_{iu}}{d_0}\right) + X_g, \tag{5.1}$$

where $\mathrm{PL}(d_0)$ is the reference path loss (in dB) measured at reference distance d_0, and ω is the path loss exponent. The term X_g (in dB) is a random variable drawn from a zero-mean Gaussian distribution $\mathcal{N}(0, \sigma^2)$, representing the shadowing effect. Then, the channel gain is obtained as per $g_{iu}^t = \frac{1}{10^{\mathrm{PL}(d_{iu})/10}}$.

Let β_{uc}^t be the Signal to Interference plus Noise Ratio (SINR) of a transmission from AP i to user u over channel c in time slot t. Formally, the SINR is calculated

as

$$\beta_{uc}^t = \frac{P_{ic}^t g_{iu}^t}{I_{uc}^t + N_b B}, \tag{5.2}$$

where N_b is the ambient noise power density (in Watt/Hz), and P_{ic}^t is the transmit power (in Watt) used by AP i on channel c in time slot t. Note that the transmit power satisfies $0 \leq P_{ic}^t \leq P_{max}$ and $\sum_{c \in \mathcal{C}_i^t} P_{ic}^t = P_{max}$. The term I_{uc}^t is the aggregated interference experienced by user u on channel c in time slot t, which is calculated as

$$I_{uc}^t = \sum_{j \in \mathcal{N}_c^t} P_{jc}^t g_{ju}^t. \tag{5.3}$$

Denote by r_c^t the theoretical data rate of AP i on channel c in time slot t. This data rate is calculated using the Shannon-Hartley formula, which is given by

$$r_c^t = B \log_2(1 + \beta_{uc}^t). \tag{5.4}$$

Then, the aggregated data rate r_i^t of AP i over $\mathcal{C}_i^t$ channels in time slot t is given by,

$$r_i^t = \sum_{c \in \mathcal{C}_i^t} r_c^t. \tag{5.5}$$

AP i has a queue of packets to transmit. The length of the queue at the end of time slot t is q_i^t, where $0 \leq q_i^t \leq q_{max}$. At the beginning of each time slot t, denote by λ_i^t the number of packets arriving at AP i, where the value of λ_i^t is sampled from a probability distribution. Further, each packet has a fixed size of L bits. Therefore, the queue length of AP i evolves as per

$$q_i^t = \min\left(\max\left(q_i^{t-1} + \lambda_i^t - \frac{r_i^t \delta}{L}, 0\right), q_{max}\right). \tag{5.6}$$

Let π be a policy used by AP i that selects a set of channels $\mathcal{C}_i^t$, and assigns transmit power P_{ic}^t on each channel $c \in \mathcal{C}_i^t$ at the beginning of time slot t. Moreover, the policy π also adjusts the CCA threshold $\gamma_c^t \in [\gamma_{min}, \gamma_{max}]$ on each channel $c \in \mathcal{C}_i^t$.

Define by $R(\pi)$ an objective function for policy π. Formally,

$$R(\pi) = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}^{\pi}\left[\sum_{t=1}^{T}\left(\eta_1 r_i^t - \eta_2 q_i^t\right)\right], \tag{5.7}$$

where η_1 and η_2 are two weights that balance the data rate and queue length of AP i, and the term $\mathbb{E}^{\pi}[.]$ refers to the expectation over the objective value when using policy π.

Let Ω be a collection of policy π. The problem at hand is to find the optimal policy, denoted as π^*, that maximizes the objective function $R(\pi)$ over time. Mathematically, the optimal policy π^* is given by

$$\pi^* = \arg\max_{\pi\in\Omega} R(\pi). \tag{5.8}$$

5.2 A Markov Decision Process Model

This section first discusses Markov Decision Process (MDP) [51]. After that, it introduces Deep Q-Network (DQN) [129] and Deep Deterministic Policy Gradient (DDPG) [146] algorithms. Lastly, it introduces the simplex sampling method [1] that is used to determine the possible transmit power allocation over one or more channels.

5.2.1 Markov Decision Process

An MDP is defined as a tuple with four elements $(\mathcal{S}, \mathcal{A}, \mathcal{R}(s_t, a_t), \mathcal{P}(s_{t+1}|s_t, a_t))$, where $\mathcal{S}$ and $\mathcal{A}$ denote the set of states and actions, respectively. In each time slot t, an agent, i.e., AP i, observes a state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$. The environment then returns a reward $\mathcal{R}(s_t, a_t)$, and moves from state s_t to $s_{t+1} \in \mathcal{S}$ with probability $\mathcal{P}(s_{t+1}|s_t, a_t)$. Let $\pi(s_t)$ be a policy used by an agent, where the policy outputs an action a_t given state s_t, i.e., $a_t = \pi(s_t)$. Let $V^{\pi}(s)$ be a value function that measures the expected long-term reward that starts from state s using

policy π thereafter. Mathematically, it is given by

$$V^{\pi}(s) = \mathbb{E}^{\pi}\left[\sum_{k=0}^{\infty}\gamma^{t+k}\mathcal{R}(s_{t+k}, \pi(s_{t+k}))|s_t = s\right], \tag{5.9}$$

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

The goal of an agent is to find the *optimal policy* π^* that maximizes the value function for all states, denoted by V^*. This optimal value function V^* can be computed using Bellman's equation [51] as

$$\begin{aligned} V^*(s_t) = \max_{a_t \in \mathcal{A}} \Big[& \mathcal{R}(s_t, a_t) \\ & + \gamma \sum_{s_{t+1} \in \mathcal{S}} \mathcal{P}(s_{t+1}|s_t, a_t) V^*(s_{t+1}) \Big], \end{aligned} \tag{5.10}$$

where γ is the discount factor. The optimal policy π^* is then given by

$$\begin{aligned} \pi^*(s_t) = \arg\max_{a_t \in \mathcal{A}} \Big[& \mathcal{R}(s_t, a_t) \\ & + \gamma \sum_{s_{t+1} \in \mathcal{S}} \mathcal{P}(s_{t+1}|s_t, a_t) V^*(s_{t+1}) \Big]. \end{aligned} \tag{5.11}$$

5.2.2 Deep Q-Network

DQN is a value based reinforcement learning algorithm [129]. It learns the optimal policy by approximating the optimal Q-value for each state-action pair [129]. DQN consists of two neural networks, an evaluation network θ and a target network θ'. The two networks have the same structure, where the evaluation network θ outputs a Q-value for a given state-action pair, denoted as $\mathcal{Q}(s_t, a_t; \theta)$, and the target network θ' outputs the corresponding target Q-value $\mathcal{Q}(s_t, a_t; \theta')$.

DQN uses experience replay to update the weights of its neural networks. Specifically, each combination of state, action, reward and next state $(s_t, a_t, \mathcal{R}(s_t, a_t), s_{t+1})$ is called an *experience*. DQN will store an *experience* in each time slot into its memory buffer $\mathcal{M}$, which stores up to $|\mathcal{M}|$ experiences. For every K time slots,

DQN uniformly samples a batch of experiences from $\mathcal{M}$ to update the weights of its evaluation network θ. The goal is to minimize a loss function which is given by

$$\mathcal{L}(\theta) = \mathbb{E}[(\mathbf{y} - \mathcal{Q}(s_t, a_t; \theta))^2], \tag{5.12}$$

where

$$\mathbf{y} = \mathcal{R}(s_t, a_t) + \gamma \max_{a \in \mathcal{A}} \mathcal{Q}(s_{t+1}, a; \theta'). \tag{5.13}$$

In addition, to ensure stability, for every K' time slots, the weights of the target network θ' are replaced by the weights of the evaluation network θ.

5.2.3 Deep Deterministic Policy Gradient

A drawback of DQN is that it is not able to learn the optimal policy when the action space is continuous [146], e.g., the transmit power and CCA threshold of AP i. Therefore, this chapter proposes to use DDPG, an actor-critic based algorithm, to address the said issue [146]. DDPG has four neural networks, namely an actor network θ^{μ}, a target actor network $\theta^{\mu'}$, a critic network θ^{Q} and a target critic network $\theta^{Q'}$. The structure of a target actor network and target critic network is the same as the corresponding actor and critic network, respectively. The actor network chooses a deterministic action a_t at each state s_t, denoted as $a_t = \mu(s_t; \theta^{\mu})$, and the critic network evaluates the Q-value $\mathcal{Q}(s_t, a_t; \theta^{Q})$ for each selected action a_t at state s_t. Similarly, the target actor network selects a target action $\mu(s_t; \theta^{\mu'})$, and the target critic network outputs a target Q-value $\mathcal{Q}(s_t, a_t; \theta^{Q'})$.

DDPG also uses experience replay to update the weights of its networks. In particular, for every K time slots, DDPG first samples a batch of experiences from its memory buffer $\mathcal{M}$ to update the weights of its critic network θ^{Q}. The update follows a similar process as DQN, which aims to minimize the loss function as per

$$\mathcal{L}(\theta^{Q}) = \mathbb{E}[(\mathbf{y}^{Q} - \mathcal{Q}(s_t, a_t; \theta^{Q}))^2], \tag{5.14}$$

where

$$\mathbf{y}^Q = \mathcal{R}(s_t, a_t) + \gamma \mathcal{Q}(s_{t+1}, \mu(s_{t+1}; \theta^{\mu'}); \theta^{Q'}). \tag{5.15}$$

Next, the weights of the actor network are updated using the Q-values evaluated by the critic network. Specifically, DDPG first calculates the gradient of Q-values with respect to all actions in sampled batch, denoted as $\nabla_a \mathcal{Q}(s, a; \theta^Q)|_{s=s_t, a=\mu(s_t;\theta^\mu)}$. Further, DDPG calculates the gradient of all actions with respect to the weights of the actor network θ^μ, denoted as $\nabla_{\theta^\mu} \mu(s; \theta^\mu)|_{s=s_t}$. Then, by applying the chain rule, the weights of the actor network are updated using a policy gradient method [147] with the following approximation

$$\nabla_{\theta^\mu} \mathcal{Q} \approx \mathbb{E}\left[\nabla_a \mathcal{Q}(s, a; \theta^Q)|_{s=s_t, a=\mu(s_t;\theta^\mu)} \nabla_{\theta^\mu} \mu(s; \theta^\mu)|_{s=s_t}\right]. \tag{5.16}$$

Finally, DDPG applies a soft update on target networks. For every K time slots, the weights of corresponding target networks are updated as per

$$\theta^{Q'} = \tau \theta^Q + (1 - \tau)\theta^{Q\prime}, \tag{5.17}$$

$$\theta^{\mu'} = \tau \theta^\mu + (1 - \tau)\theta^{\mu'}, \tag{5.18}$$

where τ is a small positive number, representing the target network update rate.

5.2.4 Simplex sampling

A key issue for DDPG is that it needs to randomly select an action to explore the action space. Recall that for the set of channels $\mathcal{C}_i^t$ used by AP i in time slot t, the transmit power allocation is a $|\mathcal{C}_i^t|$-dimensional space. Further, any transmit power allocation must satisfy $0 \leq P_{ic}^t \leq P_{max}$ and $P_{max} = \sum_{c \in \mathcal{C}_i^t} P_{ic}^t$. To this end, the simplex sampling method from [1] is used to determine a transmit power allocation over the set of channels $\mathcal{C}_i^t$.

Let *Simplex*(.) return a vector $\mathbf{v}$ with $|\mathcal{C}_i^t|$ elements, i.e., $\mathbf{v} = \mathit{Simplex}(|\mathcal{C}_i^t|)$. Each element in $\mathbf{v}$ is in the range $[0, 1]$, representing a fraction of the maximum transmit

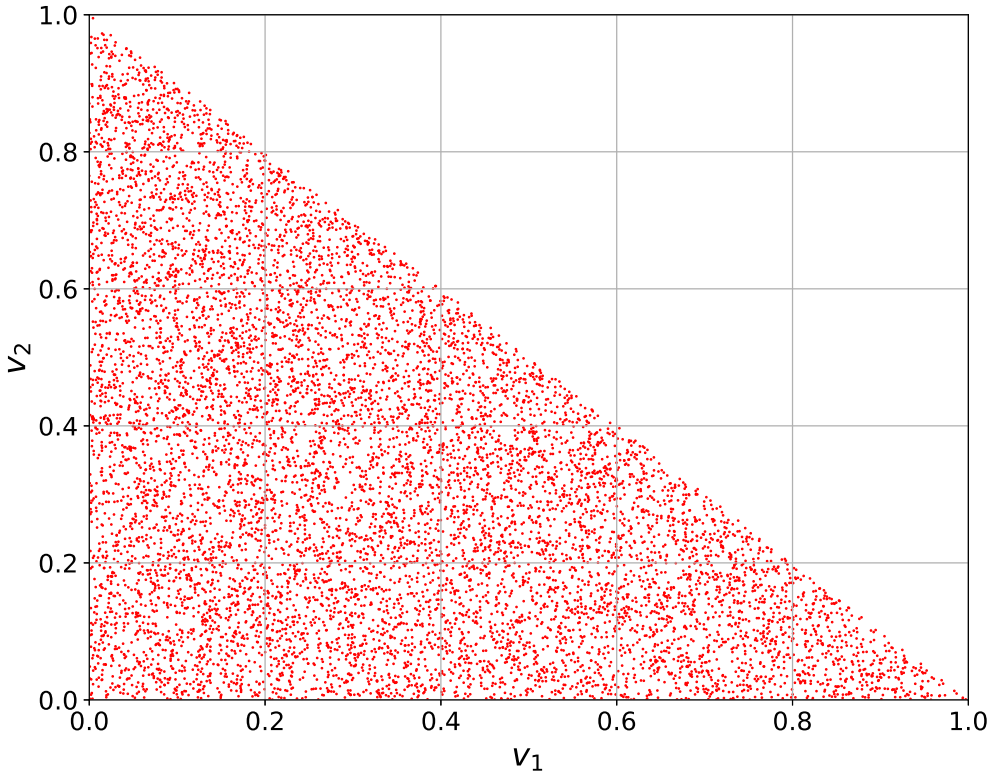

Figure 5.1: An example with 10,000 points that are sampled from a 3D space using simplex [1].

power P_{max} that is used on a certain channel. The function *Simplex*(.) first randomly generates a sequence of values, which it records in the vector $\mathbf{x} = \{x_1, x_2, \ldots, x_{|\mathcal{C}_i^t|-1}\}$; each element in $\mathbf{x}$ is uniformly sampled from the range $[0, 1]$. Then, it sorts the elements in $\mathbf{x}$ in an increasing order, and adds $x_0 = 0$ and $x_{|\mathcal{C}_i^t|} = 1$ to the beginning and the end of $\mathbf{x}$, respectively. After that, the vector $\mathbf{v}$ is obtained based on $\mathbf{x}$, where the i-th value v_i in vector $\mathbf{v}$ is calculated as $v_i = x_i - x_{i-1}$. As an example, assume there are three bonded channels. Then the corresponding vector $\mathbf{v} = [v_1, v_2, v_3]$ is sampled from a three-dimensional simplex space, i.e., $\mathbf{v} = \mathit{Simplex}(3)$. Fig. 5.1 shows the results of 10,000 samples generated by *Simplex*(3), where each red point (v_1, v_2) represents a sampled vector $\mathbf{v}$. Note that value v_3 is not shown as it can be calculated via $v_3 = 1 - v_1 - v_2$ [1].

5.3 A Three-Tier Learning Approach

This section first model the optimization problem as a three-layer MDP. After that, show how an AP or agent uses a three-tier learning approach to determine its policy.

5.3.1 Three-layer MDP

An AP i is an agent, operating in an environment with three layers as shown in Fig. 5.2; each layer corresponds to a task for AP i, and it is modeled as an MDP. Briefly, AP i first selects a set of channels $\mathcal{C}_i^t$ in Layer-1. Then, it assigns a transmit power P_{ic}^t for each channel $c \in \mathcal{C}_i^t$ in Layer-2. After that, in Layer-3, AP i selects a CCA threshold γ_c^t for each channel $c \in \mathcal{C}_i^t$.

As per Fig. 5.2, AP i interacts with its environment as follows. In each time slot t, AP i observes a state in each layer. Specifically, the state of Layer-1 is observed from the environment, and the state of Layer-2 and Layer-3 is obtained from Layer-1 and Layer-2, respectively. Based on the observed states, the agent at each layer outputs an action. The AP executes the action of each layer, which yields a reward and a new state for Layer-1.

5.3.1.1 Definitions

In each time slot t, the state, action and reward in each layer are defined as follows:

- Channel Selection (Layer-1):
  - **State** s_t^1: The Layer-1 state s_t^1 consists of the current queue length q_i^{t-1}, and the interference experienced by AP i on each channel $\{I_{ic}^t | c \in \mathcal{C}\}$. Formally, $s_t^1 = \{I_{ic}^t | c \in \mathcal{C}\} \cup \{q_i^{t-1}\}$.
  - **Action** a_t^1: The action for Layer-1 is to select a set of channels for AP i to transmit, i.e., $a_t^1 = \mathcal{C}_i^t$, where $\mathcal{C}_i^t \subseteq \mathcal{C}$. For simplicity, let a_t^1 represent the set of selected channels $\mathcal{C}_i^t$ in the rest of this section.

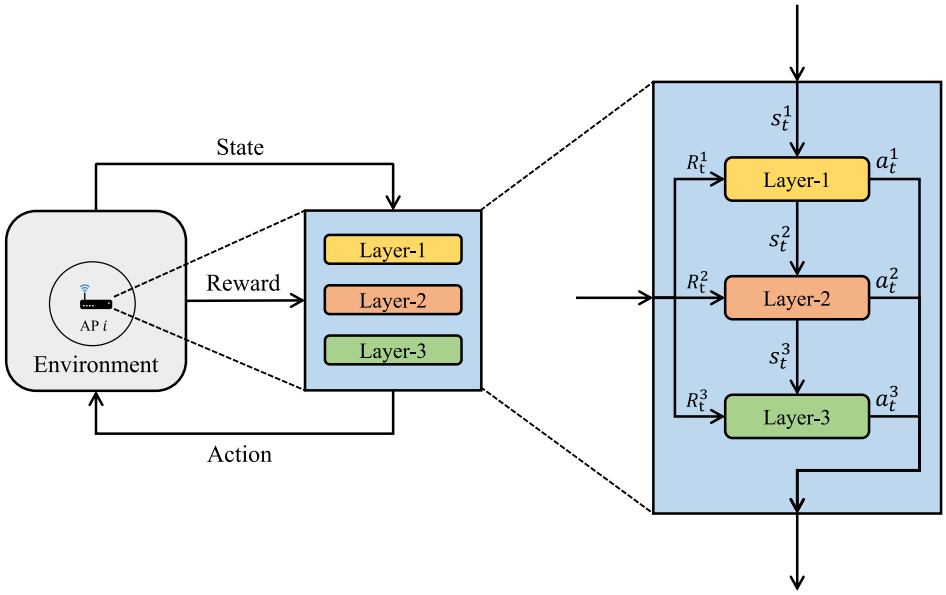

Figure 5.2: A flowchart of the proposed three-layer MDP model. In each time slot t, an agent observes a state in each layer. Specifically, the Layer-1 state s_t^1 is observed from the environment, and the Layer-2 state s_t^2 and Layer-3 state s_t^3 are obtained from their corresponding upper layer, i.e., Layer-1 and Layer-2, respectively. Based on the observed states, the agent outputs an action a_t^1, a_t^2 and a_t^3 in Layer-1, Layer-2 and Layer-3, respectively. The three actions a_t^1, a_t^2 and a_t^3 are then executed by the agent or AP, which yields the reward and a new state.

  - **Reward** R_t^1: The reward for Layer-1 is calculated based on the data rate r_i^t and queue length q_i^t of AP i, and is defined as $R_t^1 = \eta_1 r_i^t - \eta_2 q_i^t$, where η_1 and η_2 are two weights. Note that r_i^t is in Mbps and q_i^t is in number of packets.

- Transmit Power Allocation (Layer-2):

  - **State** s_t^2: The state for Layer-2 includes the interference on each channel and the set of channels $\mathcal{C}_i^t$ selected by Layer-1. Here, a binary indicator function $b_t^c \in \{0, 1\}$ is used to track whether channel c is selected in time slot t. That is, the binary indicator function b_t^c returns a value of one if channel c is selected by Layer-1; otherwise it returns zero. Formally, it is defined as

$$b_t^c = \begin{cases} 1, & \text{if } c \in a_t^1, \\ 0, & \text{Otherwise.} \end{cases} \tag{5.19}$$

The state for Layer-2 is thus defined as $s_t^2 = \{I_{ic}^t | c \in \mathcal{C}\} \cup \{b_t^c | c \in \mathcal{C}\}$.

- **Action** a_t^2: The action for Layer-2 is to assign a transmit power on each selected channel. Formally, $a_t^2 = \{P_{ic}^t | c \in a_t^1\}$. Note that the transmit power P_{ic}^t on each channel satisfies $0 \leq P_{ic}^t \leq P_{max}$ and $P_{max} = \sum_{c \in a_t^1} P_{ic}^t$.
- **Reward** R_t^2: The reward for Layer-2 is the same as Layer-1. Formally, $R_t^2 = \eta_1 r_i^t - \eta_2 . q_i^t$.

- CCA Threshold (Layer-3):
  - **State** s_t^3: The state for Layer-3 is the transmit power assigned on each channel $c \in a_t^1$. Formally, $s_t^3 = a_t^2$.
  - **Action** a_t^3: The corresponding action for Layer-3 is to select a CCA threshold for each channel. Formally, $a_t^3 = \{\gamma_c^t | c \in a_t^1\}$.
  - **Reward** R_t^3: The reward for Layer-3 is the achieved data rate of AP i, i.e., $R_t^3 = r_i^t$.

Lastly, as the propose approach is model-free for practicality reason, the transition probability $\mathcal{P}(.)$ in each layer is unknown.

5.3.2 Three-tier learning approach

AP i runs a three-tier learning approach. Specifically, it uses DQN to select a set of channels in Layer-1, and uses DDPG to assign transmit power over each selected channel in Layer-2. For Layer-3, each channel is managed by an agent using DDPG, where the agent on channel c assigns a CCA threshold for the channel. Note that these agents *do not* cooperate with each other as channels are orthogonal and the data rate achieved on a given channel has no impact on other channels.

Algorithm-4 shows the steps of the proposed approach. Initially, in Layer-1, AP i observes the state s_t^1 from its environment, and calls *Layer1SelectChannels*(.) to select an action a_t^1; see line 3. The algorithm *Layer1SelectChannels*(.) selects an action a_t^1 using the function ϵ-greedy(.), where it randomly selects an action with

Algorithm 4: Three-tier learning approach.

Initialize: $\theta_1, \theta_1', \mathcal{M}_1, \epsilon_1$ for DQN in Layer-1.
Initialize: $\theta_2^{\mu}, \theta_2^{\mu'}, \theta_2^{Q}, \theta_2^{Q'}, \mathcal{M}_2, \epsilon_2$ for DDPG in Layer-2.
Initialize: $\{\theta_c^{\mu}, \theta_c^{\mu'}, \theta_c^{Q}, \theta_c^{Q'}, \mathcal{M}_c, \epsilon_c | c \in \mathcal{C}\}$ for DDPG in Layer-3.

1 **while** t = *1,2,...,T* **do**
2 Observe s_t^1
3 $[a_t^1,\ s_t^2]$ = Layer1SelectChannels($s_t^1, \theta_1, \epsilon_1$)
4 $[a_t^2,\ s_t^3]$ = Layer2AssignPower($s_t^2, \theta_2^{\mu}, \epsilon_2$)
5 **for** *each* $c \in a_t^1$ **do**
6 Get transmit power P_{ic}^t on channel c
7 a_t^c = Layer3AdjustCCA($P_{ic}^t, \theta_c^{\mu}, \epsilon_c$)
8 **end**
9 $a_t^3 = \{a_t^c \mid c \in a_t^1\}$
10 Execute a_t^1, a_t^2, a_t^3 and observe R_t^1, R_t^2, R_t^3
11 **if** $t \geq 2$ **then**
12 Store $(s_{t-1}^1, a_{t-1}^1, R_{t-1}^1, s_t^1)$ into $\mathcal{M}_1$
13 Store $(s_{t-1}^2, a_{t-1}^2, R_{t-1}^2, s_t^2)$ into $\mathcal{M}_2$
14 **for** $c \in a_{t-1}^1$ **do**
15 Store $(P_{ic}^{t-1}, a_{t-1}^c, R_{t-1}^c, P_{ic}^t)$ into $\mathcal{M}_c$
16 **end**
17 **end**
18 **if** $t \mod K == 0$ **then**
19 Update θ_1 as per Eq. (5.12)
20 Update θ_2^Q, θ_2^{μ} and θ_c^Q, θ_c^{μ}, $\forall c \in \mathcal{C}$ as per Eq. (5.14) and (5.16)
21 Update $\theta_2^{Q'}$, $\theta_2^{\mu'}$ and $\theta_c^{Q'}$, $\theta_c^{\mu'}$, $\forall c \in \mathcal{C}$ as per Eq. (5.17) and (5.18)
22 Decrease ϵ_1, ϵ_2 and ϵ_c, $\forall c \in \mathcal{C}$
23 **end**
24 **if** $t \mod K' == 0$ **then**
25 $\theta_1' \leftarrow \theta_1$
26 **end**
27 **end**

probability ϵ_1. Otherwise, it selects the action with the highest Q-value; see line 1 in Algorithm-5. The value of ϵ_1 is reduced over time until a minimum value of ϵ_{min}. This is to ensure convergence. In addition, the algorithm *Layer1SelectChannels*(.) also outputs the Layer-2 state s_t^2.

Next, AP i enters Layer-2 with state s_t^2, and calls *Layer2AssignPower*(.); see line 4 in Algorithm-4. The algorithm first uses the function ϵ-greedy(.) to output a vector $\mathbf{v}$, where each element in vector $\mathbf{v}$ represents a certain fraction of the maximum transmit power P_{max}. Specifically, the function ϵ-greedy(.) in Layer-2

Algorithm 5: Layer1SelectChannels.

Input: $s_t^1, \theta_1, \epsilon_1$
Output: a_t^1, s_t^2
1 $a_t^1 = \epsilon$-greedy$(s_t^1, \theta_1, \epsilon_1)$
2 $s_t^2 = \{I_{ic}^t | c \in \mathcal{C}\} \cup \{b_t^c \mid c \in \mathcal{C}\}$
3 **Return** a_t^1, s_t^2

Algorithm 6: Layer2AssignPower.

Input: $s_t^2, \theta_2^\mu, \epsilon_2$
Output: a_t^2, s_t^3
1 $\mathbf{v} = \epsilon$-greedy$(s_t^2, \theta_2^\mu, \epsilon_2)$
2 $a_t^2 = P_{max}\mathbf{v}$
3 $s_t^3 = a_t^2$
4 **Return** a_t^2, s_t^3

will use *Simplex*$(|a_t^1|)$ to sample a random vector $\mathbf{v}$ with probability ϵ_2, where $|a_t^1|$ is the number of selected channels. Otherwise, it uses the output of $\mu(s_t^2, \theta_2^\mu)$ as vector $\mathbf{v}$. Note that the *Softmax* function is used as the activation function for the output layer of DDPG. This is to ensure the constraints for Layer-2 actions hold, i.e., $0 \leq P_{ic}^t \leq P_{max}$ and $P_{max} = \sum_{c \in \mathcal{C}_i^t} P_{ic}^t$. Then, *Layer2AssignPower*(.) scales the output vector $\mathbf{v}$ with the maximum transmit power P_{max} to obtain a Layer-2 action a_t^2, see line 2 in Algorithm-6. Lastly, the algorithm *Layer2AssignPower*(.) outputs the state s_t^3 for Layer-3.

In Layer-3, for each channel $c \in a_t^1$, the agent on channel c observes a state, i.e., transmit power $P_{ic}^t \in s_t^3$ and calls *Layer3AdjustCCA*(.); see line 5 to 8 in Algorithm-4. The algorithm first calls ϵ-greedy(.), where it uniformly samples a value v from the range $[0, 1]$ with probability ϵ_c. Otherwise, it uses the output of $\mu(P_{ic}^t; \theta_c^\mu)$ as value v. Note that each DDPG agent in Layer-3 uses the *Sigmoid* function as the activation function in the output layer as the CCA threshold on each channel c is a one-dimensional parameter. The value v is then scaled into the range of $[\gamma_{min}, \gamma_{max}]$ to obtain the action a_t^c on channel c; see line 2 in Algorithm-7. Finally, the action of Layer-3 is obtained as $a_t^3 = \{a_t^c \mid c \in a_t^1\}$.

Lastly, the three actions a_t^1, a_t^2 and a_t^3 are executed by AP i to obtain reward R_t^1, R_t^2 and R_t^3. Then, the agent at each layer stores its experience into its memory buffer

Algorithm 7: Layer3AdjustCCA.

Input: $P_{ic}^{t}, \theta_{c}^{\mu}, \epsilon_{c}$
Output: a_{t}^{c}
1 $v = \epsilon$-greedy$(P_{ic}^{t}, \theta_{c}^{\mu}, \epsilon_{c})$
2 $a_{t}^{c} = v(\gamma_{max} - \gamma_{min}) + \gamma_{min}$
3 **Return** a_{t}^{c}

Table 5.2: Parameter values.

Environment Parameters	**Value(s)**
Number of users per AP	4
Number of channels $\lvert\mathcal{C}\rvert$	3
Number of neighboring APs $\lvert\mathcal{N}_i\rvert$	3
Traffic arrival rate	180 packets per time slot
Available CCA threshold	-80 to -20 dBm
Total transmit power of each AP	20 dBm
Carrier frequency	5 GHz
Channel bandwidth B	20 MHz
Path loss exponent	3.5
Path loss reference distance d_0 [136]	1 m
Path loss at reference distance $PL(d_0)$ [136]	46.42 dB
Variance for Shadowing effect σ^2	3 dB
Environment noise power density N_b	5×10^{-17} mW/Hz
Initial queue length	8000 packets
Maximum queue length	16000 packets
Packet size L [148]	2304 Bytes
Length of each time slot δ [115]	4.5 ms
RTS/CTS mechanism	Disabled

starting from the second time slot; see line 11 to 16 in Algorithm-4. This is because the state for Layer-2 and Layer-3 depends on the action from their respective upper layer. Therefore, Layer-2 and Layer-3 obtain their respective next state s_{t+1}^{2} and s_{t+1}^{3} only after Layer-1 and Layer-2 select the action a_{t+1}^{1} and a_{t+1}^{2} in the following time slot. The stored experiences are then used by AP i to update the neural networks in each layer as shown in line 18 to 21 in Algorithm-4.

Table 5.3: Parameter values used by learning agents.

Parameters	**Value(s)**
Number of hidden layers	2
Number of neurons in each layer	32
Activation function for hidden layers	ReLU
Learning rate α	10^{-3}
Optimizer	Adam [137]
Reward discount factor γ	0.95
Initial exploration rate ϵ	1
Final exploration rate ϵ_{min}	0.1
Decay factor ϵ_d	0.9995
Memory size (samples)	5000
Batch size for training (samples)	32
Interval K to update θ	Every five time slots
Interval K' to replace DQN θ'	Every 500 time slots
DDPG target network update rate τ	0.005
Number of learning time slots N_L	40000

5.4 Evaluation

A simulator is implemented using Python 3.7 on a computer with i7-8700 CPU operating at 4.3 GHz and 16 GB RAM. It uses TensorFlow 1.14 [138] and Keras 2.2.5 [139] to build neural networks for learning agents. For the set of interfering APs $\mathcal{N}_i$; each AP is placed 20 m away from AP i, acting as the interference source to induce different interference states at AP i. There is at least one interfering AP operating on each channel. Each AP is associated with four users that are uniformly placed within 5 m distance. Unless otherwise stated, the parameters and values listed in Table 5.2 and 5.3 are used for each simulation run. The following algorithms/rules are implemented and compared:

- **DDPG**: AP i uses *DQN* to select a set of channels, and uses *DDPG* for both transmit power distribution and CCA threshold adjustment on each channel. This algorithm is the algorithm described in Section 5.3.2.

- **Mixed DDPG and DQN (MixDD)**: AP i uses *DQN* to select a set of

channels, and uses *DDPG* for transmit power distribution and uses *DQN* for CCA threshold adjustment on each channel. The CCA threshold is discretized into eleven levels, ranging from -80 to -30 dBm.

- **DQN**: AP i uses *DQN* to select a set of channels, and uses *DQN* for both transmit power distribution and CCA threshold adjustment on each channel. The transmit power is discretized into eight levels, ranging from zero to 100 Watts. The CCA threshold is discretized into 11 levels, ranging from -80 to -30 dBm.

- **All channels bonded (ACB)**: AP i will always use all channels for transmissions. The transmit power on each channel is evenly distributed. The CCA threshold for each channel is set to -82 dBm.

- **Random**: AP i will randomly select a set of channels for transmissions. The transmit power on each selected channel is evenly distributed. The CCA threshold for each channel is set to -82 dBm.

- **Primary channel only (PCO)**: AP i will randomly select a channel as its primary channel, and only use the primary channel for transmissions with the maximum transmit power. The CCA threshold for the primary channel is set to -82 dBm.

Each episode consists of 500 time slots. For each episode, the simulator collects the following metrics:

- **Average number of transmitted packets**: This is the average number of packets transmitted per time slot by AP i in an episode.

- **Average queue length of an AP**: This is the average queue length of AP i collected at the end of each time slot in each episode.

There are *three* stages in simulations, namely *Training*, *Test* and *Supplementary*. In the Training stage, AP i has three available channels; there is one interfering AP

on each channel and is placed 20 m away from AP i. AP i has saturated traffic, meaning it always has packets to transmit, where its queue has at least 200 packets in all time slots. Each packet has a fixed size of 2304 Bytes which is the maximum transmission unit specified in [148]. The simulation studies to follow first study the impact of ϵ decay rules on the performance of learning algorithms, i.e., DDPG, MixDD and DQN. Apart from that, learning agents are trained over a fixed network environment. Specifically, for the first 5000 time slots, each agent randomly selects actions; this ensures they collect sufficient data. Next, agents are trained for 40000 time slots. After that, their ϵ value is set to zero and they are run for another 5000 time slots to compare their convergence performance.

The Test stage considers three channels, and uses the trained agents from the Training stage to evaluate their performance under different network scenarios. Specifically, this stage studies the impact of traffic model, number of interfering APs and channel gain variance. There are two traffic models. The first is a Poisson traffic model, which is used to determine the number of packets arriving at AP i in each time slot. Its arrival rate increases from 30 to 240 packets per time slot. The second traffic model uses the tracefile or measurements from [145]. To study the impact of interfering APs on each channel, the number of APs interfering on each channel increases from one to six, which makes the total number of interfering APs increases from three to 18. Then, the impact of channel gain variance is evaluated.The variance of shadowing σ^2 increases from zero to 80 dB^2.

Next, in the Supplementary stage, this chapter studies the performance of DDPG, MixDD and DQN with different numbers of channels. The number of channels increases from two to eight, and there is one interfering AP on each channel. Lastly, this chapter evaluates the proposed algorithms using the topology and channel model provided by the IEEE 802.11ax task group [2].

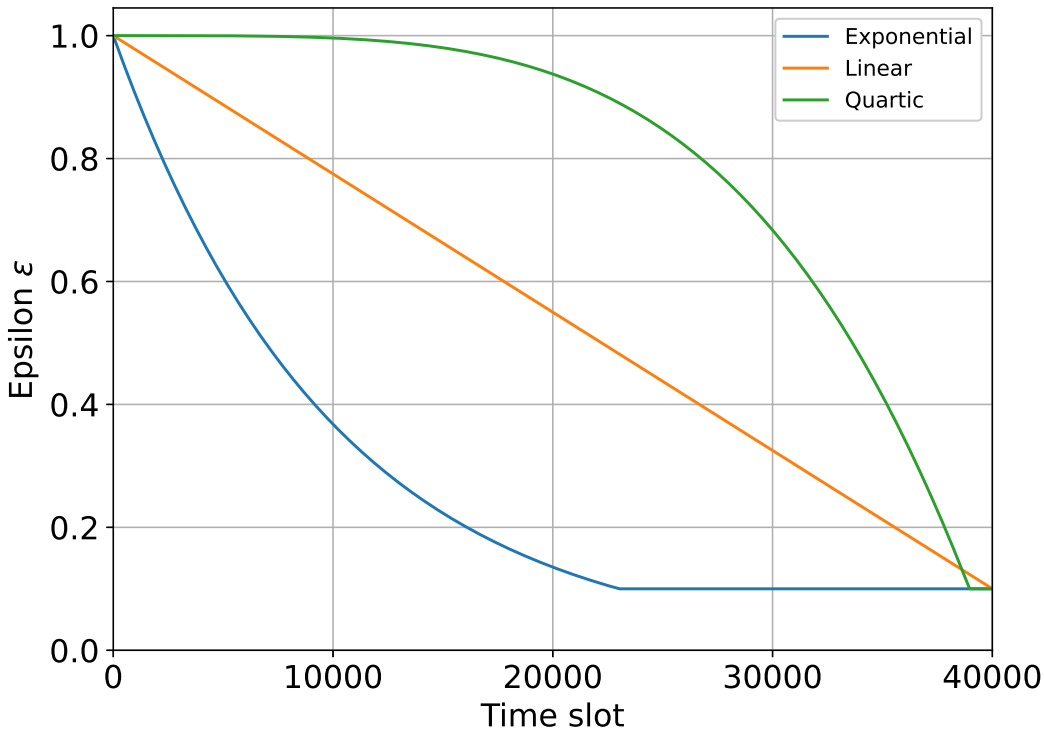

Figure 5.3: Elapsed time versus value of Epsilon.

5.4.1 Training Stage

5.4.1.1 ϵ decay rules

Three ϵ decay rules are evaluated, and the value of ϵ at each time slot t is calculated as follows:

- **Exponential**: $\epsilon^t = \max(\epsilon_d^{t/K}, \epsilon_{min})$.
- **Linear**: $\epsilon^t = \max(1 - \frac{(1-\epsilon_{min})t}{N_L/K}, \epsilon_{min})$.
- **Quartic**:$\epsilon^t = \max(1 - (\frac{t}{N_L/K})^4, \epsilon_{min})$.

The term K and N_L refer respectively to the learning frequency, and number of time slots for learning. Fig. 5.3 shows the evolution of ϵ over time. In the *Exponential* rule, the value of ϵ reduces at the fastest rate before 15000 time slots, and then it reduces at a slower rate than the *Quartic* and *Linear* rule. By contrast, the value

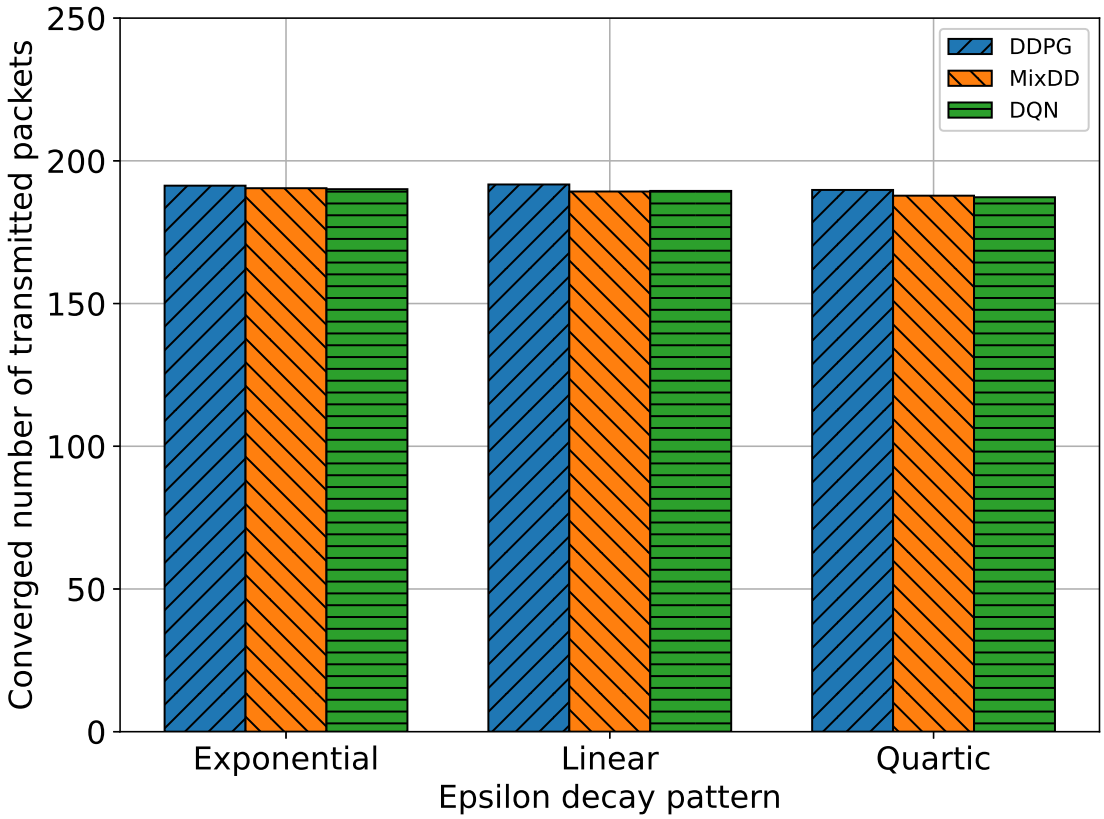

Figure 5.4: Converged throughput versus Epsilon decay pattern.

of ϵ for the *Quartic* rule decreases at the lowest rate at the beginning and starts to decrease faster after 10000 time slots. Note that the value of ϵ will not decrease below the minimum value of ϵ_{min}.

Referring to Fig. 5.4, DDPG, MixDD and DQN all converge to around 190 packets per time slot. In addition, Fig. 5.4 shows that different ϵ decay rules have no significant impact on the converged throughput for all testing algorithms. The largest difference of 4.5 packets per time slot is observed between DDPG with the *Linear* rule and DQN with the *Quartic* rule, which only differs by 2.39%. Fig. 5.5 shows the average throughput for different ϵ decay rules. The *Exponential* rule has the highest average throughput for all three learning algorithms. The average number of transmitted packets per time slots for DDPG, MixDD and DQN is 165.5, 156.1 and 153.2, respectively, which means the *Exponential* rule outperforms the *Linear* rule by 11.18% and the *Quartic* rule by 24.32%, on average. This is because

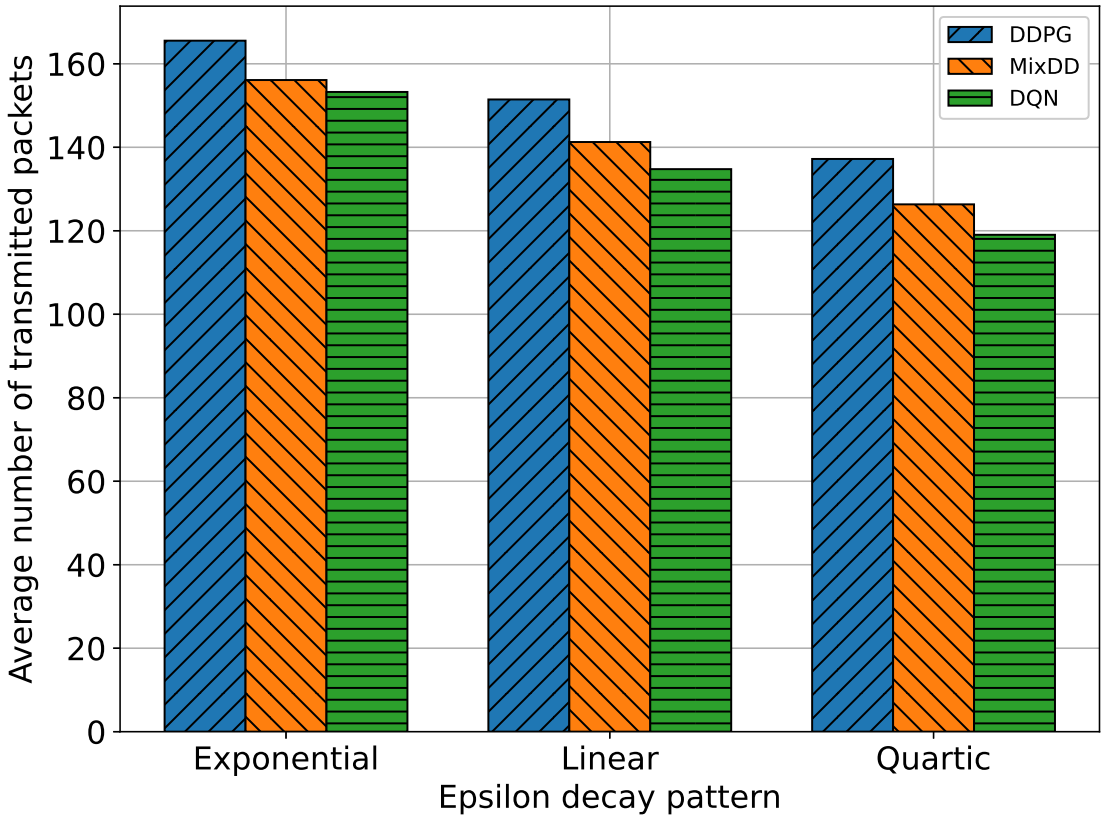

Figure 5.5: Average throughput versus Epsilon decay pattern.

when ϵ reduces to a low value, an agent will exploit its learned policy with a high probability. The *Exponential* rule reduces ϵ to the minimum value ϵ_{min} with the fastest rate. In addition, in Fig. 5.4, all three ϵ decay rules converge to 190 packets per time slot. Consequently, the *Exponential* rule has the highest average throughput among all ϵ decay rules. Hence, in all subsequent simulations, the *Exponential* rule is used as the ϵ decay rule when training agents.

5.4.1.2 Convergence

Referring to Fig. 5.6, the average number of transmitted packets for DDPG, MixDD and DQN increases over time. The average number of transmitted packets for DDPG, MixDD and DQN increases from 102.9, 97.8 and 88.8, and converges to 190.2, 189.6 and 186.0 packets per time slot, respectively. This is because these three learning algorithms are able to learn the optimal policies for channel selection,

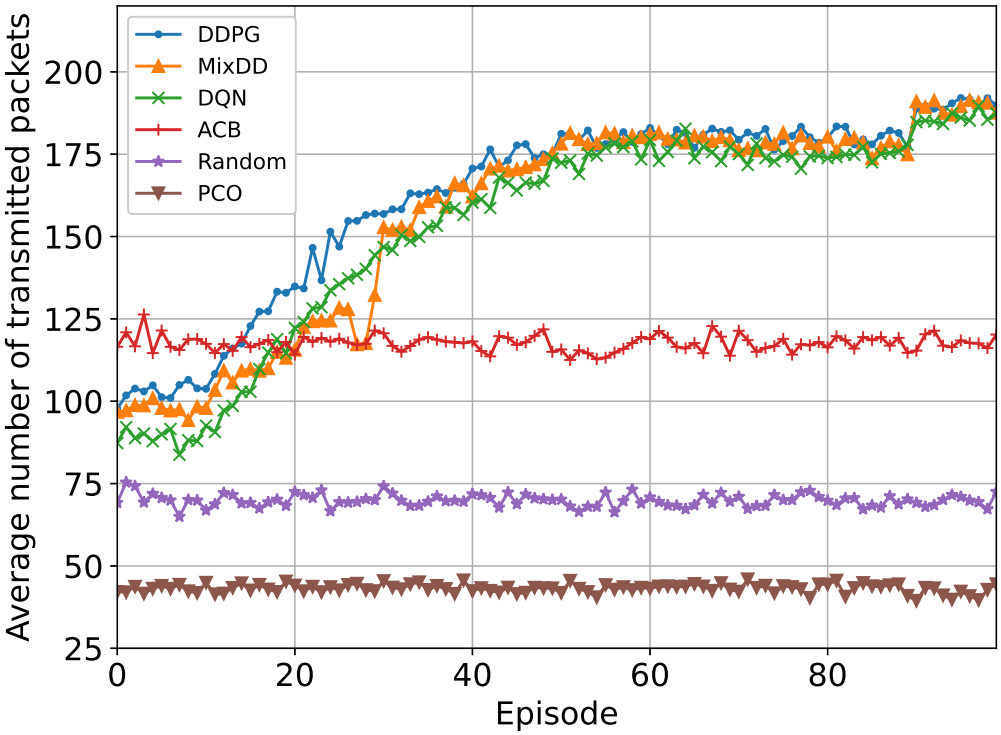

Figure 5.6: Elapsed time versus the number of transmitted packets.

power distribution and CCA threshold adjustment over time. Further, the performance for DDPG is better than the other two learning algorithms. For example, the number of transmitted packets per time slot for DDPG is 4.43% and 6.84% higher than that of MixDD and DQN on average. This is because DDPG is able to learn the power distribution and CCA threshold selection in a continuous manner. This allows DDPG to learn the optimal action from the action space. By contrast, MixDD learns discrete CCA thresholds on each channel; DQN learns both discrete power distribution and CCA threshold on each channel. Thus, MixDD and DQN are not able to learn the optimal action if the action is not discretized into their action spaces. Therefore, DDPG outperforms the other two learning algorithms. In addition, the average number of transmitted packets for ACB, Random and PCO is the same over time, where they are able to transmit 118.6, 70.6 and 42.8 packets per time slot, respectively. This is because these three algorithms are deterministic and

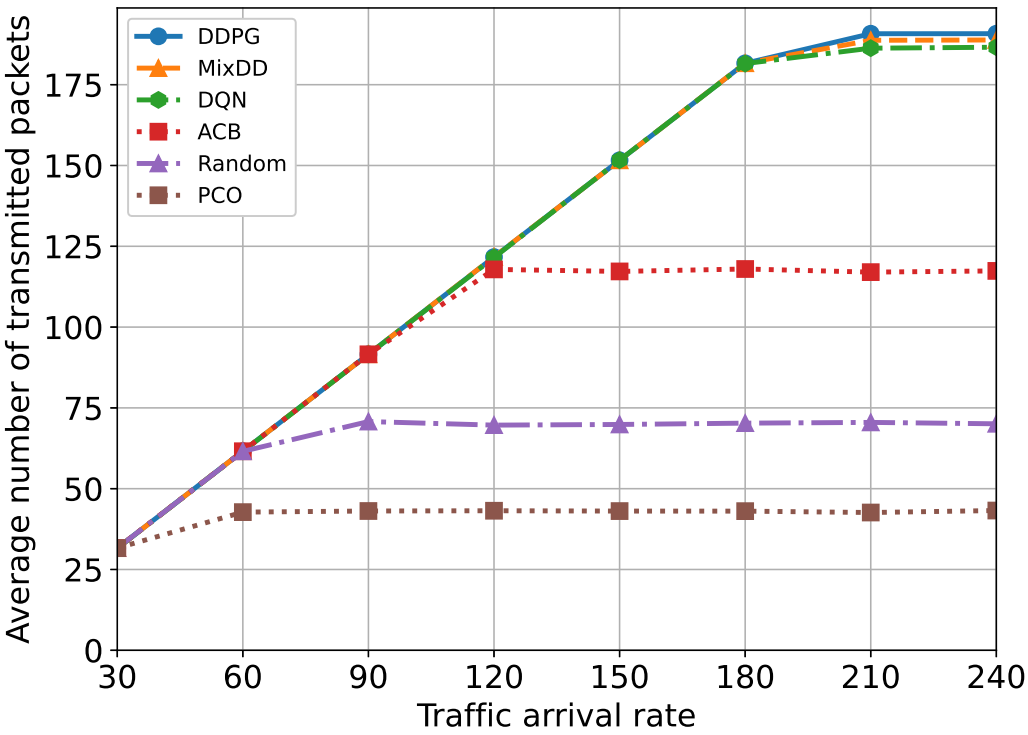

Figure 5.7: Impact of traffic arrival rate on throughput.

they do not learn to select channels, and assign transmit power and CCA threshold on each channel to improve their throughput.

5.4.2 Test Stage

5.4.2.1 Poisson traffic

Three channels and Poisson traffic models are considered. Specifically, the traffic arrival rate increases from 30 to 240 packets per time slot; the average result is collected by running 5000 time slots for each traffic arrival rate. Fig. 5.7 and 5.8 show the impact of arrival rates. From Fig. 5.7, DDPG, MixDD and DQN show an increasing throughput trend. They are able to transmit 31.56 packets per time slot when the traffic arrival rate is 30 packets per time slot. This number then increases to 190.8 for DDPG, 188.8 for MixDD and 186.6 for DQN when the traffic arrival rate is 240 packet per time slot. This is because these three learning algorithms

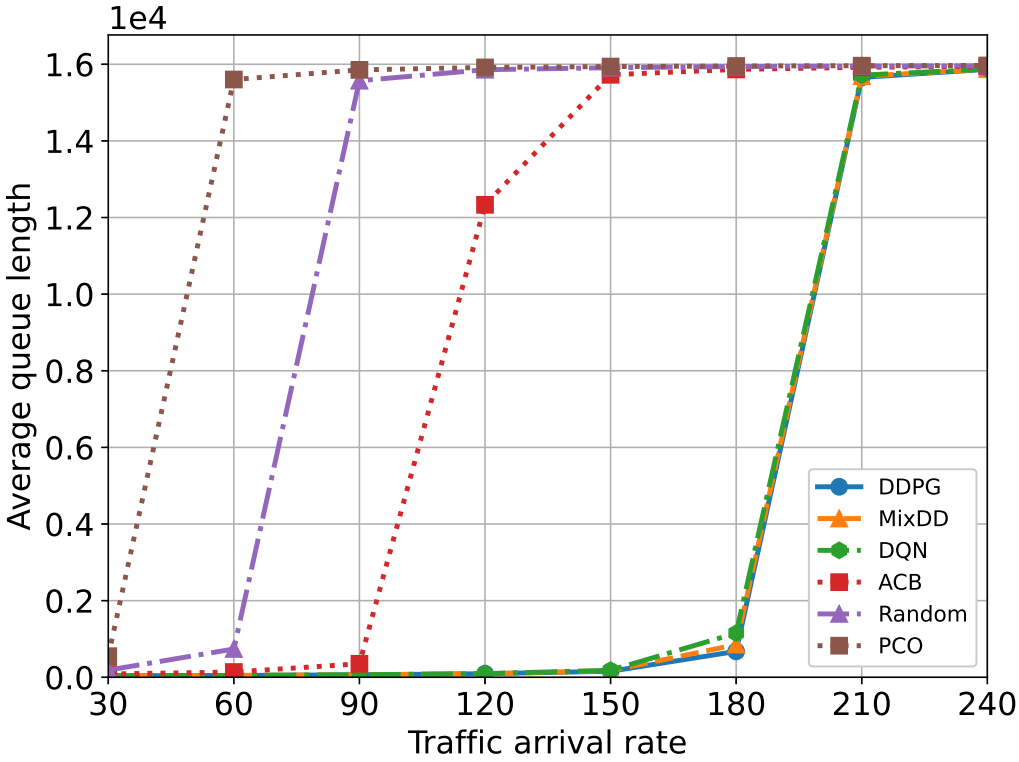

Figure 5.8: Impact of traffic arrival rate on average queue length.

are able to transmit more than 186 packets per time slot after training as shown in Fig. 5.6. Therefore, when the traffic arrival rate is lower than 180 packets per time slot, DDPG, MixDD and DQN are able to empty the queue of AP i, and thereby deliver all arriving packets in each time slot. As a result, the throughput is limited by the low arriving traffic and it is approximately the same as the traffic arrival rate. This is also shown in Fig. 5.8 where the average queue length of DDPG, MixDD and DQN is lower than 1100 packets when the traffic arrival rate is no larger than 180 packets per time slot. However, as the traffic arrival rate exceeds 180 packets per time slots, DDPG, MixDD and DQN do not have sufficient throughput to deliver all arriving packets, which increases the queue length of AP i. From Fig. 5.8, the average queue length of these three learning algorithms increases significantly from 1500 to 15968 packets when the traffic arrival rate increases from 180 to 210 packets per time slot. Similar trends are observed for ACB, Random and PCO, where they

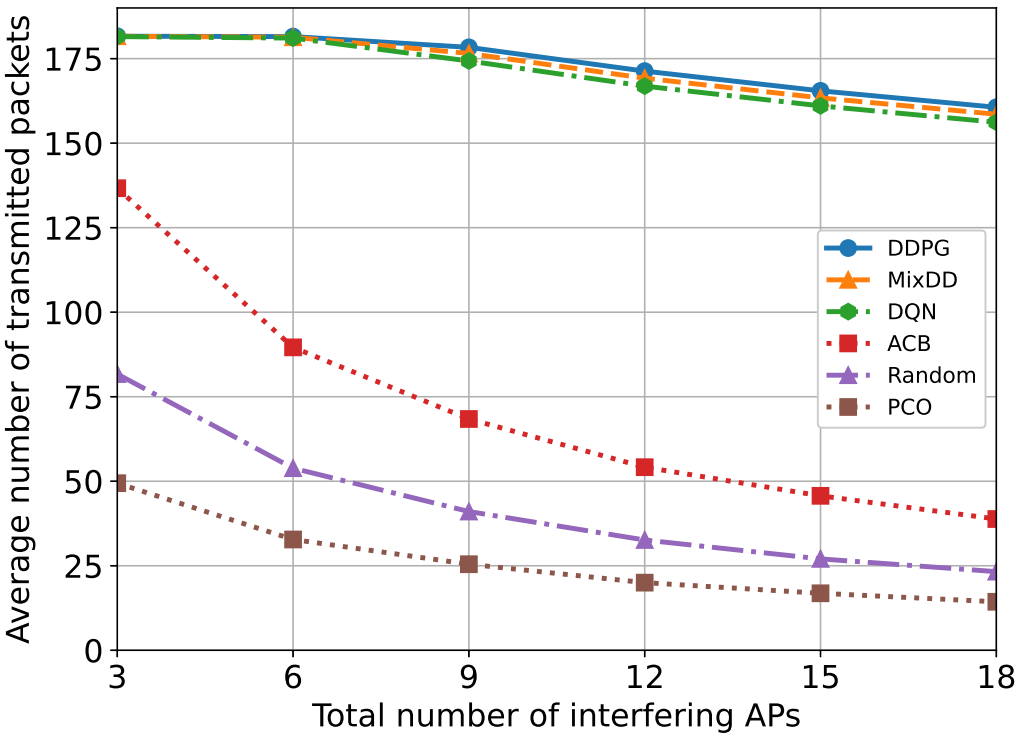

Figure 5.9: Number of interfering APs versus throughput.

have an average throughput of 117.2, 71.8 and 42.4 packets per time slot as shown in Fig. 5.6. Therefore, they are only able to reduce the queue length of AP i when the traffic arrival rate is lower than their average throughput.

5.4.2.2 Interfering APs

This simulation concerns Poisson traffic with an arrival rate of 180 packets per time slot. The network has three channels, and the number of interfering APs on each channel increases from one to six, which means the total number of interfering APs increases from three to 18. Interfering APs are uniformly placed within the range of 40 m around AP i. For each number of interfering APs, the simulation runs ten times, with 5000 time slots in each run, to collect the average result.

Fig. 5.9 and 5.10 show the average throughput and queue length of AP i. From Fig. 5.9, the throughput decreases as the number of interfering APs increases.

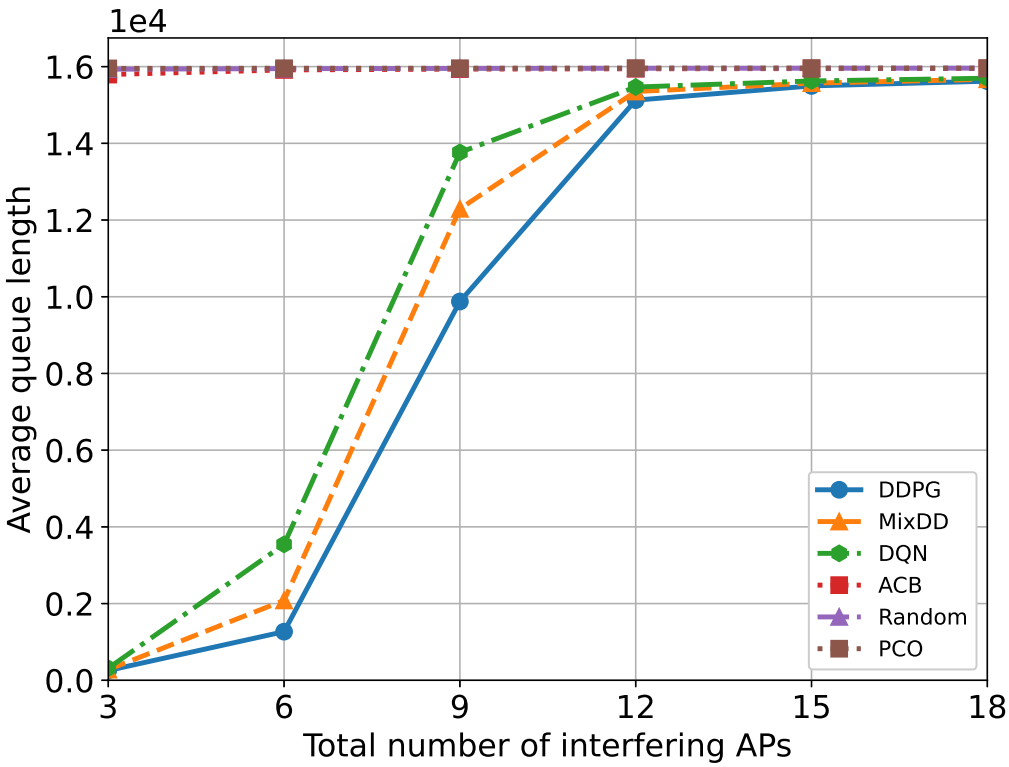

Figure 5.10: Number of interfering APs versus the average queue length.

DDPG, MixDD and DQN are able to transmit 181 packets per time slot when the number of interfering APs is three. This number decreases to 160.6 for DDPG, 158.6 for MixDD and 156.2 for DQN when the number of interfering APs increases to 18. As a comparison, the average number of transmitted packets per time slot for ACB, Random and PCO decreases from 136.7, 81.7 and 49.4 to 38.8, 23.3 and 14.3, respectively. This is because the level of interference on each channel increases as the number of interfering APs increases, which causes throughput degradation on all algorithms/rules. However, DDPG, MixDD and DQN continue to have better performance against increasing level of interference as compared to ACB, Random and PCO. The throughput of DDPG, MixDD and DQN reduces by 12.74% on average as the number of interfering APs increases from three to 18. In contrast, the performance of ACB, Random and PCO drops by 71.37% on average under the same circumstance. From Fig. 5.9, DDPG, MixDD and DQN always have the

highest throughput. In particular, these three learning algorithms achieve 137.72%, 296.67% and 548.5% higher throughput than ACB, Random and PCO on average, respectively. This is because these three learning algorithms learn the optimal channel selection, transmit power distribution and CCA threshold adjustment when interference level from interfering APs varies. From Fig. 5.9, the throughput of DDPG outperforms MixDD by 1.21% and DQN by 2.65% when the number of interfering APs is larger than six. The difference in throughput leads to different performance on average queue length. Referring to Fig. 5.10, the average queue length for DDPG is 5.88% and 10.51% shorter than MixDD and DQN. This reveals that continuous control of transmit power and CCA threshold has better performance than discrete control. In contrast, the average queue length for ACB, Random and PCO is always around 16000 packets. These three rules are not able to transmit more than 180 packets per time slot. Therefore, they are not able to reduce the queue length of AP i and thus, experience queue overflow.

5.4.2.3 Channel gain variance

In this simulation, the Poisson traffic with an arrival rate of 180 packets per time slot is considered. The network has three channels, and the channel gain variance σ^2 increases from zero to 80 dB^2. Referring to Fig. 5.11 and 5.12, the performance of DDPG, MixDD and DQN is not affected by the changing channel gain variance. In Fig. 5.11, DDPG, MixDD and DQN are able to transmit 181 packets per time slots for all values of channel gain variance. The average queue length of these three algorithms is lower than 2000 packets. The reason is that DDPG, MixDD and DQN learn to optimally assign transmit power and CCA threshold on each channel to gain high throughput against various levels of interference. In contrast, the throughput of ACB, Random and PCO increases with increasing channel gain variance. The average number of transmitted packets per time slot for ACB, Random and PCO increases from 117.8 to 127.0, 70.2 to 75.7 and 42.8 to 46.8, respectively. Their throughput improves by 8.25% on average as the channel gain variance increases

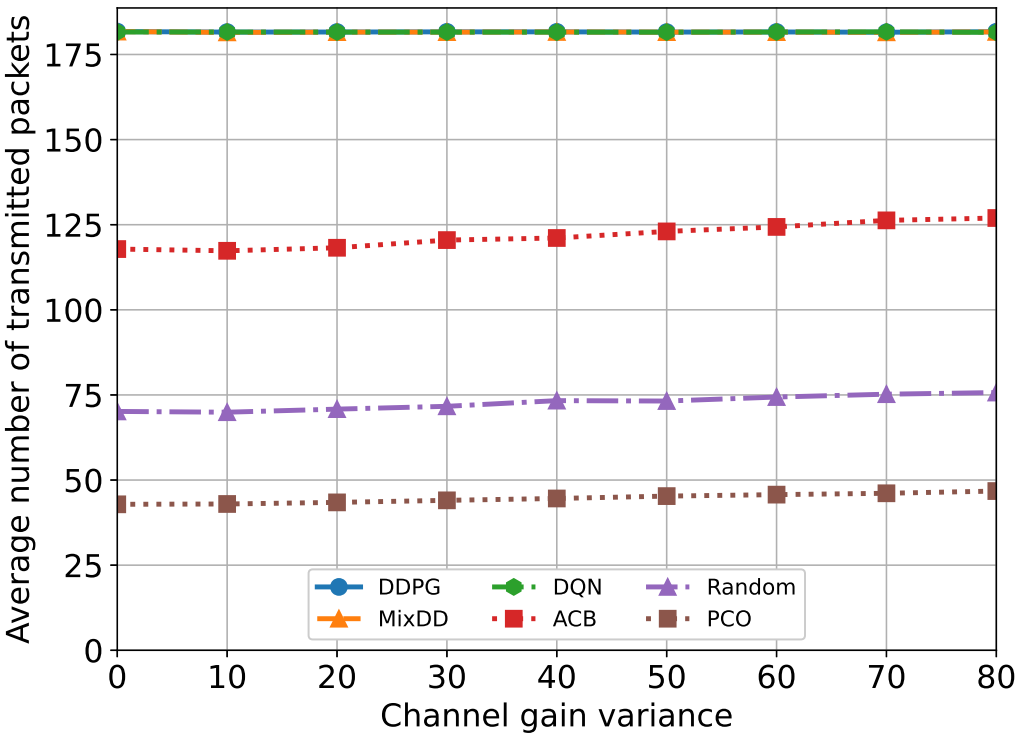

Figure 5.11: Impact of channel gain variance on throughput.

from zero to 80. This is because as the variance increases, the corresponding Cumulative Distribution Function (CDF) changes, which increases the probability that the interference on each channel is lower than CCA threshold, i.e., -82 dBm. Therefore, when channel gain variance is high, ACB, Random and PCO have more opportunities to transmit than when channel gain variance is low. Consequently, the average number of transmitted packets increases. However, the increased in throughput is lower than the traffic arrival rate. Hence, ACB, Random and PCO are not able to reduce the queue length of AP i, and suffer from queue overflow. Referring to Fig. 5.12, the average queue length for ACB, Random and PCO is always at the maximum queue length of 16000 packets for all channel gain variances.

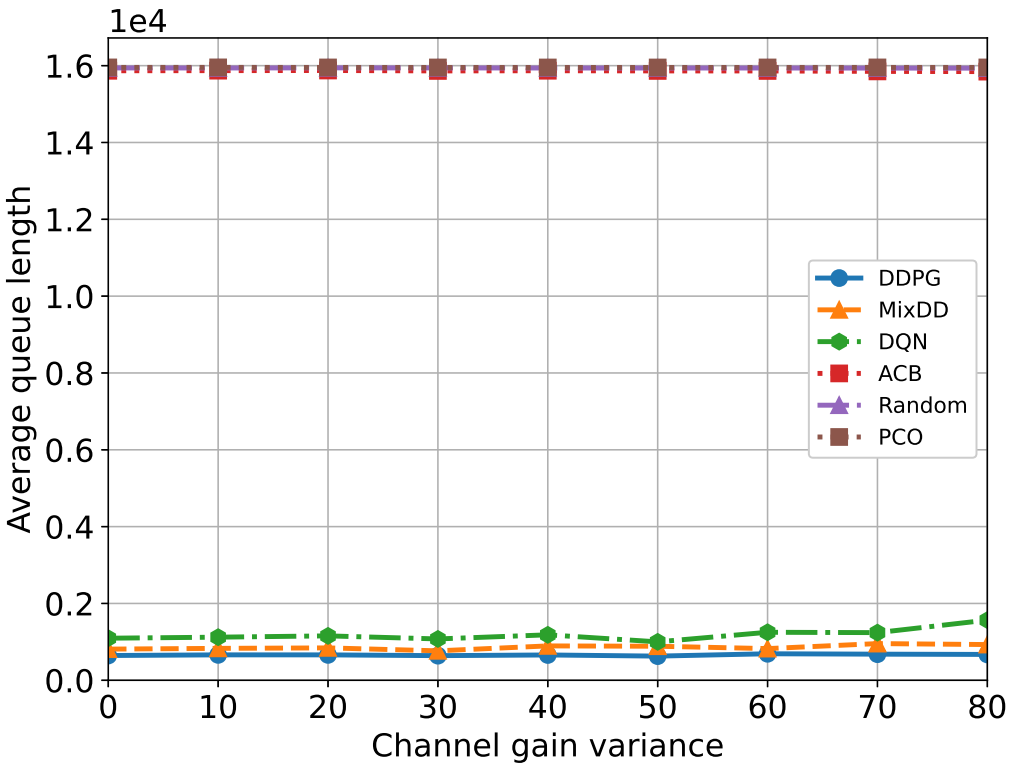

Figure 5.12: Impact of channel gain variance on the average queue length.

5.4.2.4 Trace-based study

Next, the performance of all compared algorithms/rules are evaluated using the traffic trace file provided in [145]. Eight days of traffic are extracted, from 19 October 2014 to 26 October 2014, see Fig. 5.13, and they are used as the traffic arriving at AP i. Fig 5.14 and 5.15 show the average number of transmitted packets and queue length with different date of traffic trace data.

From Fig 5.14 and 5.15, DDPG, MixDD and DQN always have the highest throughput and lowest queue length. In Fig 5.14, the average number of transmitted packets per time slot for DDPG, MixDD and DQN is around 39.4. As a comparison, ACB, Random and PCO are able to transmit 38.8, 37.5 and 32.8 packets per time slot. The average throughput of DDPG, MixDD and DQN is only 1.4%, 4.87% and 20.13% higher than ACB, Random and PCO, respectively. The reason is that the arriving traffic is not high throughout a day. From Fig. 5.13, the average number of

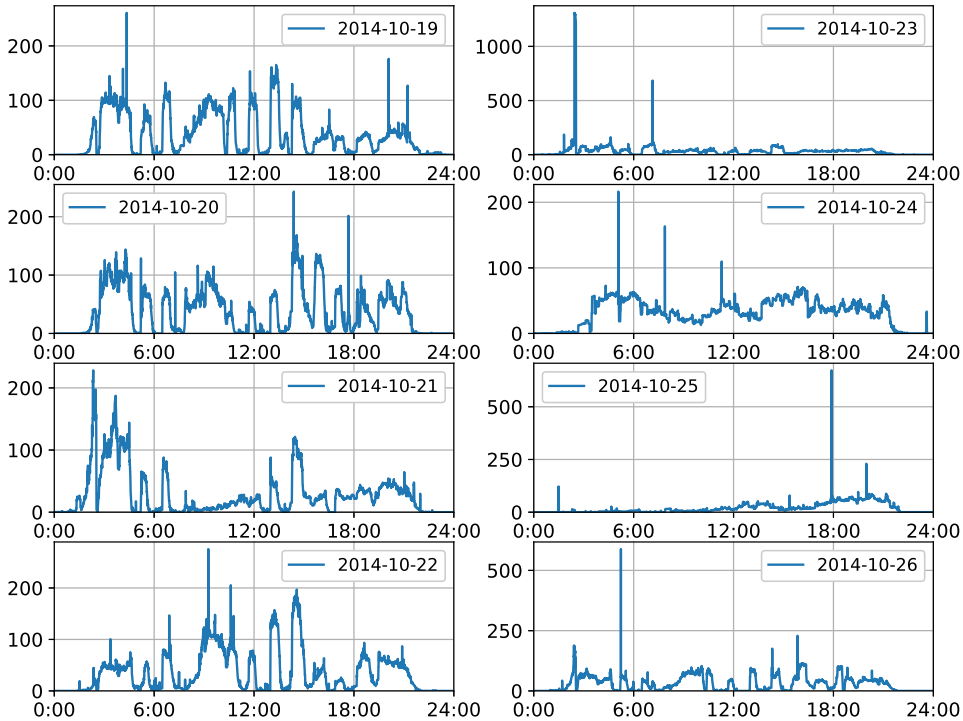

Figure 5.13: Arriving traffic for AP i throughout eight days. The X-axis represents the time across a day, and Y-axis represents the number of packets arriving in each time slot, respectively.

arriving packets in each time slot is around 30.6 across the eight days. Therefore, DDPG, MixDD and DQN are able to empty the queue of APs quickly. This can also be seen from Fig. 5.15, where DDPG, MixDD and DQN have the smallest average queue length of 3.05 packets. As a result, DDPG, MixDD and DQN do not have a large number of packets to transmit in each time slot, which results in a low average number of transmitted packets per time slot. In contrast, the average queue length for ACB, Random and PCO is 4.64, 6.52 and 8.24. The average queue length of DDPG, MixDD and DQN is 32.94%, 50.99% and 62.52% shorter than ACB, Random and PCO.

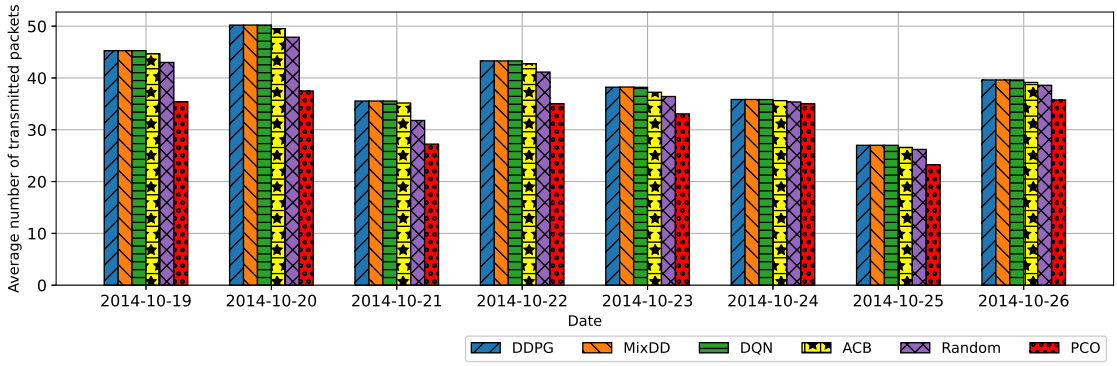

Figure 5.14: Average throughput with difference trace data.

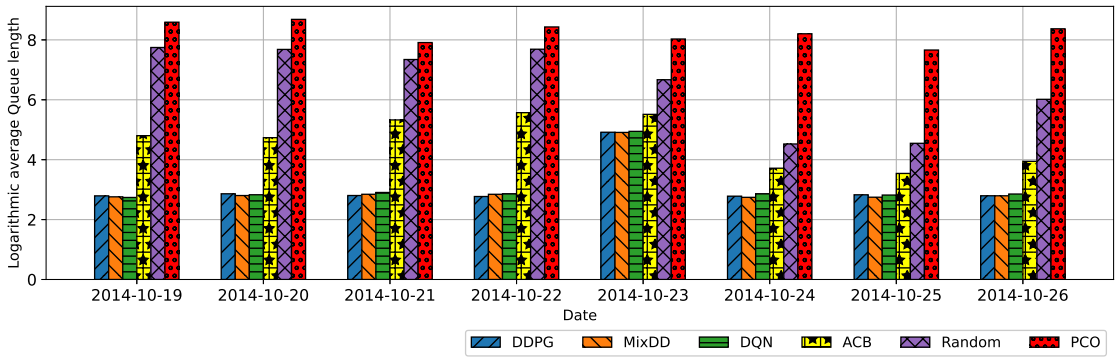

Figure 5.15: Average queue length with difference trace data. Note that the *logarithmic* value of the original queue length (in packets) is taken to have a better view.

5.4.3 Supplementary Stage

5.4.3.1 Number of channels

Here, different numbers of channels are studied. For each channel, there is one interfering AP placed 20 m away from AP i. As the number of channels differs, the action space for the learning agents is different. Therefore, for each channel number, the learning agents will be re-built and trained until convergence. Fig. 5.16 shows the converged throughput of each algorithm with different channel numbers. The performance of DDPG, MixDD and DQN have no difference when the number of channels is no larger than four. The average number of transmitted packets per time slot for DDPG, MixDD and DQN increases from 130 to 245 when the channel number increases from two to four. This means all three learning agents are able to learn the optimal policy when the channel number is low. However, as the number

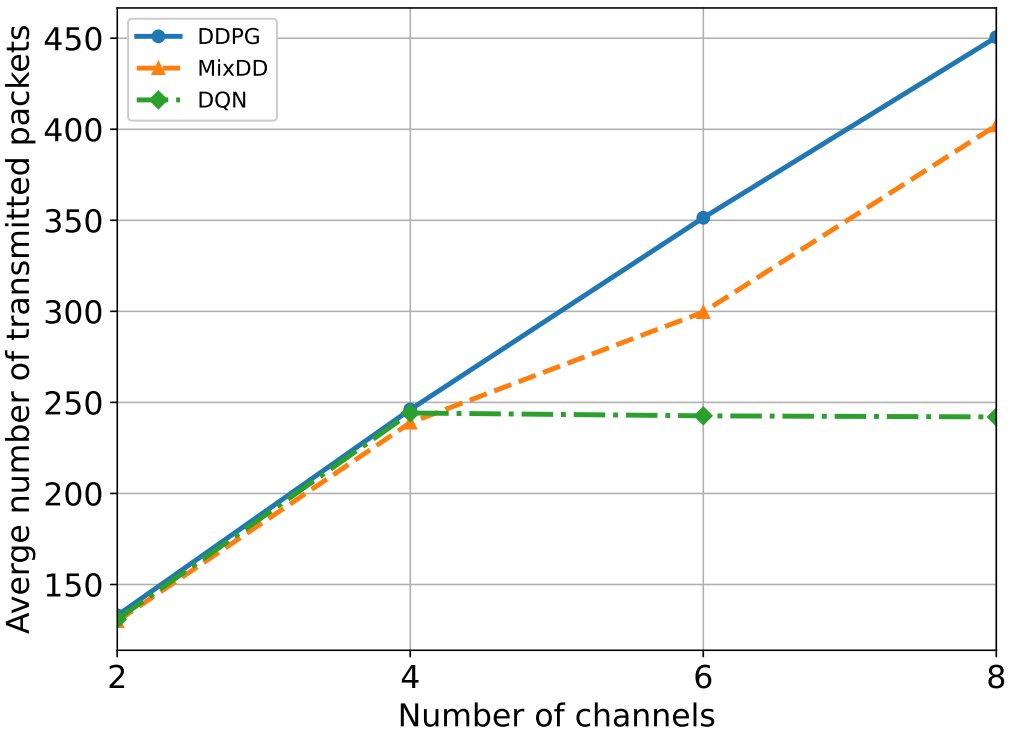

Figure 5.16: Converged throughput verses number of channels.

of channels increases, DDPG achieves the highest throughput. The average number of transmitted packets per time slot for DDPG increases from 351 to 451 when the number of channels increases from six to eight, which is 14.67% and 65.51% higher than MixDD and DQN on average, respectively. This means DDPG is able to learn the optimal policy with different number of channels. In contrast, the throughput of MixDD and DQN shows less improvement than DDPG when the number of channels is larger than four. The throughput of MixDD increases from 245 to 400 as the number of channels increases from four to eight. This is because MixDD uses *DQN* for CCA threshold adjustment. Considering *DQN* works based on discretized action space, MixDD is not able to learn the optimal CCA threshold if it is not included in the action space, which results in lower performance than DDPG. The converged throughput for DQN remains at around 243 packets per time slot as the number of channels increases from four to eight. This is because the action space

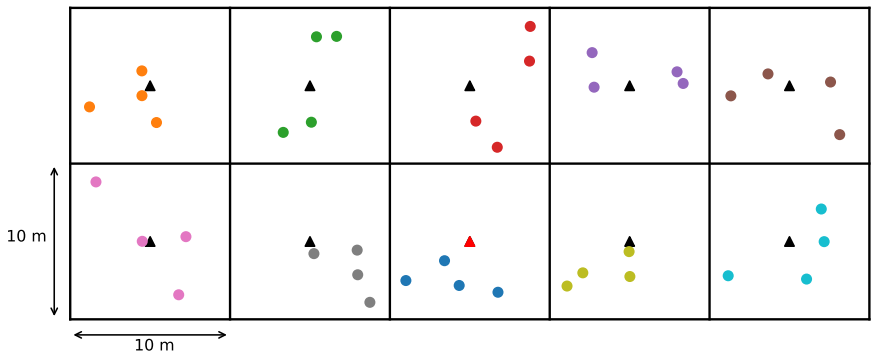

Figure 5.17: The topology of IEEE simulation scenario for apartment [2], where each apartment has a dimension of 10 m × 10 m. Each AP is placed in the center of an apartment. The target AP i (red triangle) is located at the center bottom apartment and all other APs (black triangles) act as the interference sources. The penetration loss of each wall is set to 5 dB [3].

for DQN increases significantly with increasing number of channels. For example, the number of actions for distributing transmit power over six and eight channels is 1287 and 6435. As a result, agents are not able to explore and learn each action efficiently during training. Therefore, agents are not able to learn the optimal policy, which results in poor performance.

5.4.3.2 IEEE scenarios

In this simulation, it considers IEEE scenario and channel model proposed in [2] and [3]. Referring to Fig. 5.17, it considers an apartment with ten cells; each cell has a dimension of 10 m × 10 m. In each cell, an AP is placed at the center; the AP in the center bottom cell is selected as AP i that runs the learning algorithm. Each AP has four associated users that are uniformly placed within its cell. The indoor channel model provided in [3] is considered, and the wall penetration loss is set to 5 dB.

Fig. 5.18 shows the evolution of the average number of transmitted packets for different algorithms/rules. DDPG, MixDD and DQN are able to achieve the highest throughput over time. The average number of transmitted packets per time slot for

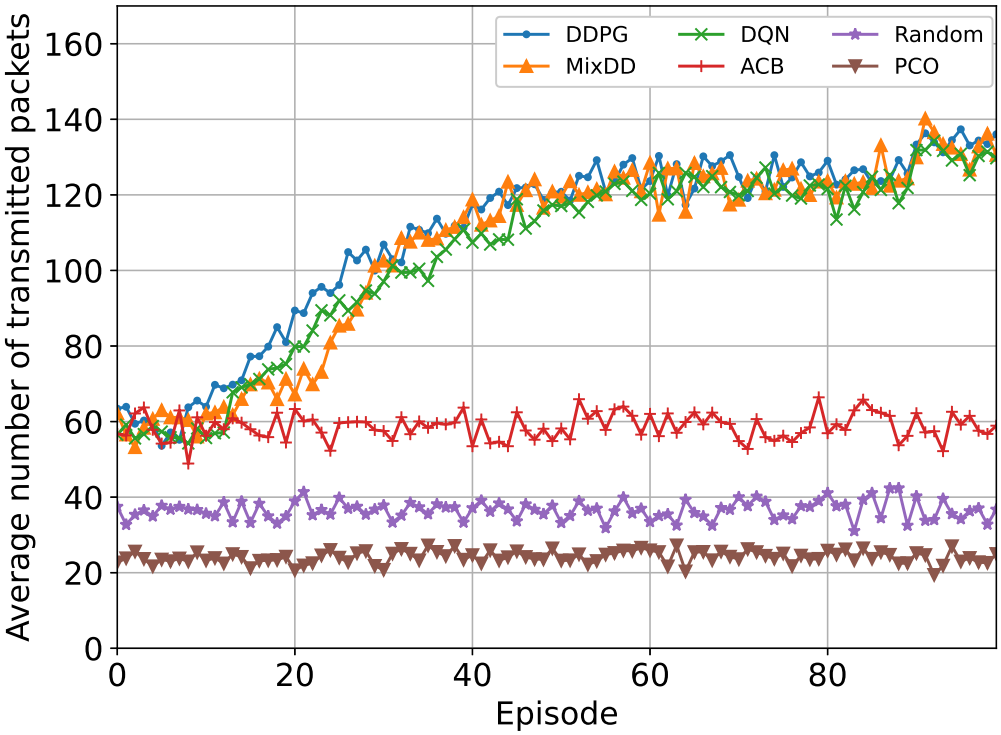

Figure 5.18: Elapsed time versus average number of transmitted packets with IEEE simulation scenario.

DDPG, MixDD and DQN increases from 61.2, 59.3 and 56.8 to 135.0, 133.0 and 130.6, respectively. These three learning algorithms are still able to learn the optimal policy with IEEE scenario and channel model. DDPG achieves the highest throughput among all three learning algorithms, where its average number of transmitted packets per time slot is 4.01% and 5.83% higher than that of MixDD and DQN. In contrast, ACB, Random and PCO have a stable throughput. The average number of transmitted packets for ACB, Random and PCO is 58.5, 36.6 and 23.5 per time slot.

5.5 Conclusion

This chapter aims to improve the multi channel utilization and minimize the queue length of an AP in a WiFi network. The proposed three-tier learning approach runs

on an AP and learns to select a set of channels and allocate transmit power and CCA threshold for each selected channel. Advantageously, an AP only require local measured information, such as queue length and interference to learn the optimal policy. Simulation results show that the proposed learning approach is able to learn the optimal policy, and achieve the best performance under multiple scenarios. Numerical results show that the proposed three-tier learning approach is able to reduce the average queue length of an AP by up to 62.52% when compared to an AP using a fixed strategy over realistic traffic trace data.

Chapter 6

Conclusion

Future WiFi networks are required to provide transmissions with high capacity and low latencies to satisfy users demands. This results in increasing level of interference due to the presence of densely deployed APs. To this end, improving spatial reuse and transmission capacity are two promising solutions. This thesis conducts research into improving spatial reuse and transmission capacity. Specifically, it studies how to optimize the CCA threshold, transmit power or channel bonding policy in order to improve network capacity. A challenging issue is that the traffic arrival, amount of interference and channel conditions are random over time. This means any methods used to determine an optimal policy must be able to adapt to changing wireless network conditions.

Henceforth, this thesis outlines a number of reinforcement learning based solutions. These solutions run on each AP and allow each AP to independently learn the optimal CCA threshold, transmit power or channel bonding policy. Critically, these solutions are model-free and only rely on locally observed information, such as interference and current queue length. Briefly, this thesis makes the following contributions:

- In Chapter 3, this thesis considers a dense WiFi network where a number of APs are closely placed together. Each AP runs a DQN agent to learn to the

optimal CCA threshold for each device in its cell. It proposes two learning models, namely EpL and InstL, that rely on locally measured historical interference data. The simulation results indicate that with sufficient learning, both EpL and InstL converge to the same optimal value. InstL converges faster than EpL in a fixed network topology, i.e., 20 episodes versus 250 episodes, as it has more times of learning within one episode. In addition, both EpL and InstL outperform Le-DSC by 62.4%.

- In Chapter 4, this thesis aims to improve spatial reuse and minimize the queue length of APs with random traffic arrivals. Each AP runs a hierarchical reinforcement learning (HRL) agent that learns the optimal transmit power and CCA threshold. Advantageously, each AP only requires local information such as interference and queue length. The simulation results indicate that the proposed HRL based approach is able to learn the optimal transmit power and CCA threshold. Experiments over traffic trace data indicate that the average queue length of an HRL-equipped AP is shorter than DSC and Legacy by 24.4% and 52.4%, respectively.

- Lastly in Chapter 5, this thesis jointly optimizes the CCA threshold, transmit power and bonded channels of an AP with random traffic arrivals. The aim is to minimize the queue length and maximize the throughput of an AP. Unlike prior works that only consider to bond adjacent channels in WiFi networks, this thesis considers both adjacent and non-adjacent channel bonding. The joint optimization problem is first formulated as a three-layer MDP. Then, this thesis outlines a three-tier RL approach that runs on an AP to determine a set of channels, transmit power and CCA threshold. Specifically, for a precise control of the CCA threshold and transmit power, it uses DDPG for continuous control of the said parameters. Numerical results show that the proposed three-tier RL approach is able to reduce the average queue length of an AP by up to 62.52% when compared to an AP using a fixed strategy.

There are a number of potential future research directions. First of all, in this thesis, all APs treat other APs/Cells as part of environment and compete with each other for channel resources, which will inevitably generate a significant amount of interference in networks. Therefore, an interesting future work is to consider cooperative APs. Specifically, APs use a *cooperative* multi-agent reinforcement learning algorithm [149] to cooperatively optimize their CCA threshold, transmit power or channel bonding policies based on their current queue length or interference. The aim is to further improve network capacity. In addition, another future work is to consider traffic with different priorities when a cooperative scheme is considered. For example, services such as video meetings or online streaming require both high capacity and low latency and hence need high priority to ensure Quality of Service (QoS). Therefore, an AP needs to adaptively adjust its CCA threshold, transmit power and channel bonding policies to satisfy the demands of different types of traffic. Moreover, as discussed in Chapter 2, directional antenna also helps to improve spatial reuse. In particular, by using directional antenna, an AP transmitting packets to its associated user may not generate interference to neighboring cells. However, the transmitting AP may suffer from hidden terminal/channel problem as devices in a neighboring cell may not hear its transmission. Therefore, a future work is to apply intelligent and cooperative control of directional antenna on each AP to further improve spatial reuse. Another research direction is to consider power consumption. As noted in [12], WiFi users have finite energy to transmit or receive data. Therefore, effective power management for transmission is critical as it determines the lifetime of WiFi users. To this end, a future study can be conducted to focus on energy efficiency optimization with spatial reuse in WiFi networks. Lastly, one possible research direction is to study the performance of the proposed RL approach in settings with mobile users. This is because the channel condition changes significantly when users move within a cell. Therefore, an AP needs to predict the movement pattern of users and learn the optimal control policy on CCA threshold, transmit power and channel bonding to address the varying location of users.

Bibliography

[1] N. A. Smith and R. W. Tromble, "Sampling uniformly from the unit simplex," *Johns Hopkins University, Tech. Rep*, vol. 29, 2004.

[2] S. Merlin *et al.*, "TGax simulation scenarios," *doc. IEEE 802.11-14/0980r16*, July 2015. Accessed on 2022-04-12.

[3] J. Liu *et al.*, "IEEE 802.11ax channel model document," *doc. IEEE 802.11-14/0882r4*, Sept. 2014. Accessed on 2022-04-12.

[4] A. Vitosinschi, "802.11ac wireless throughput testing and validation guide." accessed 2019-09-24.

[5] A. von Nagy, "MCS value achieved by clients at various signal to noise ratio levels (SNR)." accessed 2021-12-21.

[6] M. S. Gast, *802.11 ac: a survival guide: Wi-Fi at gigabit and beyond.* Sebastopol, CA, USA: O'Reilly Media, 2013.

[7] G. R. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X. P. Costa, and B. Walke, "The IEEE 802.11 universe," *IEEE Commun. Mag.*, vol. 48, pp. 62–70, Jan. 2010.

[8] Cisco, "Cisco annual internet report (2018–2023) white paper," June 2020. Accessed July 2022.

[9] Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022," June 2019. Accessed July 2022.

[10] H. A. Omar, K. Abboud, N. Cheng, K. R. Malekshan, A. T. Gamage, and W. Zhuang, "A survey on high efficiency wireless local area networks: Next generation WiFi," *IEEE Commun. Surveys Tuts.*, vol. 18, pp. 2315–2344, Apr. 2016.

[11] K. K. Sreedhar, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications," in *IEEE ISM*, (San Jose, CA, USA), pp. 583–586, Dec. 2016.

[12] E. Khorov, A. Kiryanov, A. Lyakhov, and G. Bianchi, "A tutorial on IEEE 802.11ax high efficiency WLANs," *IEEE Commun. Surveys Tuts.*, vol. 21, pp. 197–216, Sept. 2018.

[13] Q. Qu, B. Li, M. Yang, Z. Yan, A. Yang, J. Yu, M. Gan, Y. Li, X. Yang, O. Aboul-Magd, E. Au, D.-J. Deng, and K.-C. Chen, "Survey and performance evaluation of the upcoming next generation wlan standard - IEEE 802.11ax," *arXiv preprint*, June 2018.

[14] T. Nitsche, C. Cordeiro, A. B. Flores, E. W. Knightly, E. Perahia, and J. C. Widmer, "IEEE 802.11ad: directional 60 ghz communication for multi-gigabit-per-second wi-fi [invited paper]," *IEEE Communications Magazine*, vol. 52, pp. 132–141, Dec. 2014.

[15] A. B. Flores, R. E. Guerra, E. W. Knightly, P. Ecclesine, and S. Pandey, "IEEE 802.11af: a standard for TV white space spectrum sharing," *IEEE Communications Magazine*, vol. 51, pp. 92–100, Oct. 2013.

[16] S. Aust, R. V. Prasad, and I. G. M. M. Niemegeers, "Ieee 802.11ah: Advantages in standards and further challenges for sub 1 ghz wi-fi," in *IEEE ICC*, (Ottawa, ON, Canada), pp. 6885–6889, June 2012.

[17] S. H. R. Bukhari, M. H. Rehmani, and S. Siraj, "A survey of channel bonding for wireless networks and guidelines of channel bonding for futuristic cognitive radio sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 18, pp. 924–948, Apr.-Jun. 2016.

[18] M. Han, Z. Chen, L. X. Cai, T. H. Luan, and F. Hou, "A deep reinforcement learning based approach for channel aggregation in IEEE 802.11 ax," in *IEEE GLOBECOM*, (Taipei, Taiwan), pp. 1–6, Dec. 2020.

[19] D.-J. Deng, K.-C. Chen, and R.-S. Cheng, "IEEE 802.11ax: Next generation wireless local area networks," in *IEEE Qshine*, (Rhodes, Greece), pp. 77–82, Aug. 2014.

[20] E. Khorov, A. Kiryanov, and A. Lyakhov, "IEEE 802.11ax: How to build high efficiency WLANs," in *IEEE EnT*, (Moscow, Russia), pp. 14–19, Nov. 2015.

[21] B. Bellalta, "IEEE 802.11ax: High-efficiency WLANS," *IEEE Wireless Commun.*, vol. 23, pp. 38–46, Feb. 2016.

[22] B. Bellalta, L. Bononi, R. Bruno, and A. Kassler, "Next generation IEEE 802.11 wireless local area networks: Current status, future directions and open challenges," *Computer Communications*, vol. 75, pp. 1–25, Feb. 2016.

[23] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, pp. 388–404, Mar. 2000.

[24] M. Kamel, W. Hamouda, and A. Youssef, "Ultra-dense networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, pp. 2522–2545, May 2016.

[25] L. Kleinrock and J. Silvester, "Spatial reuse in multihop packet radio networks," *Proc. IEEE*, vol. 75, pp. 156–167, Jan. 1987.

[26] B. Alawieh, Y. Zhang, C. Assi, and H. Mouftah, "Improving spatial reuse in multihop wireless networks - a survey," *IEEE Commun. Surveys Tuts.*, vol. 11, pp. 71–91, Aug. 2009.

[27] IEEE, "IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pp. 1–2793, Mar. 2012.

[28] V. Bharghavan, "Performance evaluation of algorithms for wireless medium access," in *IEEE IPDS*, (Durham, NC, USA), pp. 86–95, Sept. 1998.

[29] H. Dai, K.-W. Ng, and M.-Y. Wu, "An overview of MAC protocols with directional antennas in wireless ad hoc networks," in *IEEE ICWMC*, (Bucharest, Romania), pp. 84–91, July 2006.

[30] I. Jamil, L. Cariou, and J. Helard, "Improving the capacity of future IEEE 802.11 high efficiency WLANs," in *IEEE ICT*, (Lisbon, Portugal), pp. 303–307, May 2014.

[31] A. Valkanis, A. Iossifides, and P. Chatzimisios, "An interference based dynamic channel access algorithm for dense WLAN deployments," in *IEEE PACET*, (Xanthi, Greece), pp. 1–4, Nov. 2017.

[32] V. Kawadia and P. Kumar, "Principles and protocols for power control in wireless ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, pp. 76–88, Jan. 2005.

[33] L. Lanante and S. Roy, "Analysis and optimization of channel bonding in dense IEEE 802.11 WLANs," *IEEE Trans. Wireless Commun.*, vol. 20, pp. 2150–2160, Mar. 2021.

[34] L. Deek, E. Garcia-Villegas, E. Belding, S.-J. Lee, and K. Almeroth, "The impact of channel bonding on 802.11n network management," in *7th Conf. Emerg. Netw. Exp. Technol.*, (Tokyo, Japan), pp. 1–12, Dec. 2011.

[35] S. Barrachina-Muñoz, F. Wilhelmi, and B. Bellalta, "To overlap or not to overlap: Enabling channel bonding in high-density WLANs," *Comput. Netw.*, vol. 152, pp. 40–53, Apr. 2019.

[36] S. Jang, K. G. Shin, and S. Bahk, "Post-CCA and reinforcement learning based bandwidth adaptation in 802.11ac networks," *IEEE Trans. Mobile Comput.*, vol. 17, pp. 419–432, May 2018.

[37] L. Deek, E. Garcia-Villegas, E. Belding, S.-J. Lee, and K. Almeroth, "Intelligent channel bonding in 802.11n WLANs," *IEEE Trans. Mobile Comput.*, vol. 13, pp. 1242–1255, June 2014.

[38] R. S. Sutton, A. G. Barto, F. Bach, *et al.*, *Reinforcement learning: An introduction.* MIT press, 1998.

[39] S. Szott, K. Kosek-Szott, P. Gawłowicz, J. T. Gómez, B. Bellalta, A. Zubow, and F. Dressler, "Wi-Fi meets ML: A survey on improving IEEE 802.11 performance with machine learning," *IEEE Commun. Surveys Tuts.*, pp. 1–54, June 2022.

[40] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, pp. 436–444, May 2015.

[41] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, pp. 3133–3174, 4th Quart. 2019.

[42] M. S. Afaqui, E. Garcia-Villegas, E. Lopez-Aguilera, G. Smith, and D. Camps, "Evaluation of dynamic sensitivity control algorithm for IEEE 802.11ax," in *IEEE WCNC*, (New Orleans, LA, USA), pp. 1060–1065, Mar. 2015.

[43] A. Aijaz and P. Kulkarni, "On performance evaluation of dynamic sensitivity

control techniques in next-generation WLANs," *IEEE Syst. J.*, pp. 1–4, May 2018.

[44] M. S. Afaqui, E. Garcia-Villegas, E. Lopez-Aguilera, and D. Camps-Mur, "Dynamic sensitivity control of access points for IEEE 802.11ax," in *IEEE ICC*, (Kuala Lumpur, Malaysia), pp. 1–7, May 2016.

[45] P. Kulkarni and F. Cao, "Dynamic sensitivity control to improve spatial reuse in dense wireless LANs," in *ACM MSWiM*, (Malta), pp. 323–329, Nov. 2016.

[46] Y. Hua, Q. Zhang, and Z. Niu, "Distributed physical carrier sensing adaptation scheme in cooperative MAP WLAN," in *IEEE GLOBECOM*, (Honolulu, HI, USA), pp. 1–6, Nov. 2009.

[47] Z. Lv, H. Hu, D. Yuan, and J. Ran, "An adaptive rate and carrier sense threshold algorithm to enhance throughput and fairness for dense WLANs," in *IEEE ICCC*, (Chengdu, China), pp. 453–458, Dec. 2017.

[48] X. Zhang, H. Zhu, and G. Qiu, "Optimal physical carrier sensing to defend against exposed terminal problem in wireless ad hoc networks," in *IEEE ICCCN*, (Shanghai, China), pp. 1–6, Aug. 2014.

[49] Y. Zhu, Q. Zhang, Z. Niu, and J. Zhu, "On optimal physical carrier sensing: Theoretical analysis and protocol design," in *IEEE INFOCOM*, (Barcelona, Spain), pp. 2351–2355, May 2007.

[50] Y. Zhu, Q. Zhang, Z. Niu, and J. Zhu, "QoS-aware adaptive physical carrier sensing for wireless networks," in *IEEE WCNC*, (Kowloon, China), pp. 2311–2316, Mar. 2007.

[51] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* New York, NY, USA: John Wiley & Sons, Inc., 1994.

[52] IEEE, "IEEE standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless LAN medium access

control (MAC) and physical layer (PHY) specifications amendment 1: Radio resource measurement of wireless LANs," *IEEE Std 802.11k-2008 (Amendment to IEEE Std 802.11-2007)*, pp. 1–244, June 2008.

[53] J. Deng, B. Liang, and P. K. Varshney, "Tuning the carrier sensing range of IEEE 802.11 mac," in *IEEE GLOBECOM*, (Dallas, TX, USA), pp. 2987–2991, Nov. 2004.

[54] J. Zhu, X. Guo, L. L. Yang, and W. S. Conner, "Leveraging spatial reuse in 802.11 mesh networks with enhanced physical carrier sensing," in *IEEE ICC*, (Paris, France), pp. 4004–4011, June 2004.

[55] H. Ma, H. M. K. Alazemi, and S. Roy, "A stochastic model for optimizing physical carrier sensing and spatial reuse in wireless ad hoc networks," in *IEEE MASS*, (Washington DC, USA), pp. 8 pp.–622, Nov. 2005.

[56] H.-A. Safavi-Naeini, E. Tuomaala, O. Alanen, S. Choudhury, E. Rantala, and J. Kneckt, "Adapting CCA and receiver sensitivity," *IEEE*, Nov. 2014. doc. IEEE 802.11-14/1443r0.

[57] E. Ak and B. Canberk, "FSC: Two-scale AI-driven fair sensitivity control for 802.11ax networks," in *IEEE GLOBECOM*, (Taipei, Taiwan), pp. 1–6, Dec. 2020.

[58] Y. Wen, H. Fujita, and D. Kimura, "Throughput-aware dynamic sensitivity control algorithm for next generation WLAN system," in *IEEE PIMRC*, (Montreal, QC, Canada), pp. 1–7, Oct. 2017.

[59] T. Nakahira, K. Ishihara, Y. Asai, Y. Takatori, R. Kudo, and M. Mizoguchi, "Centralized control of carrier sense threshold and channel bandwidth in high-density WLANs," in *IEEE APMC*, (Sendai, Japan), pp. 570–572, Nov. 2014.

[60] P. Kulkarni and F. Cao, "Taming the densification challenge in next generation

wireless LANs: An investigation into the use of dynamic sensitivity control," in *IEEE WiMob*, (Abu Dhabi, United Arab Emirates), pp. 860–867, Oct. 2015.

[61] H. Ma, S. Y. Shin, and S. Roy, "Optimizing throughput with carrier sensing adaptation for IEEE 802.11 mesh networks based on loss differentiation," in *IEEE ICC*, (Glasgow, UK), pp. 4967–4972, June 2007.

[62] J. Zhu, B. Metzler, X. Guo, and Y. Liu, "Adaptive CSMA for scalable network capacity in high-density WLAN: A hardware prototyping approach," in *IEEE INFOCOM*, (Barcelona, Spain), pp. 1–10, Apr. 2006.

[63] K. Park, L. Kim, and J. C. Hou, "Adaptive physical carrier sense in topology-controlled wireless networks," *IEEE Transactions on Mobile Computing*, vol. 9, pp. 87–97, Jan. 2010.

[64] Y. Zhu, Q. Zhang, Z. Niu, and J. Zhu, "On optimal QoS-aware physical carrier sensing for IEEE 802.11 based WLANs: Theoretical analysis and protocol design," *IEEE Trans. Wireless Commun.*, vol. 7, pp. 1369–1378, Apr. 2008.

[65] C. Thorpe, S. Murphy, and L. Murphy, "IEEE802.11k enabled adaptive carrier sense management mechanism (KAPCS2)," in *IFIP/IEEE IM*, (Dublin, Ireland), pp. 509–515, May 2011.

[66] E. Haghani, M. N. Krishnan, and A. Zakhor, "Adaptive carrier-sensing for throughput improvement in IEEE 802.11 networks," in *IEEE GLOBECOM*, (Miami, FL, USA), pp. 1–6, Dec. 2010.

[67] J. A. Fuemmeler, N. H. Vaidya, and V. V. Veeravalli, "Selecting transmit powers and carrier sense thresholds in CSMA protocols for wireless ad hoc networks," in *ACM 2nd Annu. Int. Workshop on Wireless Internet*, (New York, NY, USA), p. 15, Aug. 2006.

[68] M. Iwata, K. Yamamoto, B. Yin, T. Nishio, M. Morikura, and H. Abeysekera, "Analysis of inversely proportional carrier sense threshold and transmission

power setting based on received power for IEEE 802.11ax," in *IEEE CCNC*, (Las Vegas, NV, USA), pp. 1–6, Feb. 2019.

[69] T. Ropitault and N. Golmie, "ETP algorithm: Increasing spatial reuse in wireless lans dense environment using ETX," in *IEEE PIMRC*, (Montreal, QC, Canada), pp. 1–7, Oct. 2017.

[70] T. Ropitault, "Evaluation of RTOT algorithm: A first implementation of OBSS_PD-based SR method for IEEE 802.11ax," in *IEEE CCNC*, (Las Vegas, NV, USA), pp. 1–7, Jan. 2018.

[71] T.-S. Kim, H. Lim, and J. C. Hou, "Improving spatial reuse through tuning transmit power, carrier sense threshold, and data rate in multihop wireless networks," in *ACM 12th Annu. Int. Conf. Mobile Comput. Netw*, (New York, NY, USA), pp. 366–377, Sept. 2006.

[72] S. Kim, S. Yoo, J. Yi, Y. Son, and S. Choi, "FACT: Fine-grained adaptation of carrier sense threshold in IEEE 802.11 wlans," *IEEE Trans. Veh. Technol.*, vol. 66, pp. 1886–1891, May 2017.

[73] I. Jamil, L. Cariou, and J.-F. Hélard, "Novel learning-based spatial reuse optimization in dense WLAN deployments," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, p. 184, Aug. 2016.

[74] C. Chen, H. Zhao, H. Xiang, C. Sun, J. Sui, L. Zhu, S. Wang, L. Cong, and Y. Zhou, "A QoS enhancement scheme through joint control of clear channel assessment threshold and contending window for IEEE 802.11e broadcasting," *Mob. Inf. Syst.*, vol. 2017, pp. 1–9, Jan. 2017.

[75] M. Y. Arslan, K. Pelechrinis, I. Broustis, S. Singh, S. V. Krishnamurthy, S. Addepalli, and K. Papagiannaki, "ACORN: An auto-configuration framework for 802.11n WLANs," *IEEE/ACM Trans. Netw.*, vol. 21, pp. 896–909, Oct. 2013.

[76] T. Song, T.-Y. Kim, W. Kim, and S. Pack, "Channel bonding algorithm for densely deployed wireless LAN," in *ICOIN*, (Kota Kinabalu, Malaysia), pp. 395–397, Jan. 2016.

[77] T. Song, T.-Y. Kim, W. Kim, and S. Pack, "Adaptive and distributed radio resource allocation in densely deployed wireless LANs: A game-theoretic approach," *IEEE Trans. Veh. Technol.*, vol. 67, pp. 4466–4475, May 2018.

[78] Y. Li, W. Zhang, C.-X. Wang, J. Sun, and Y. Liu, "Deep reinforcement learning for dynamic spectrum sensing and aggregation in multi-channel wireless networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, pp. 464–475, Mar. 2020.

[79] W. Wang, F. Zhang, and Q. Zhang, "Managing channel bonding with clear channel assessment in 802.11 networks," in *IEEE ICC*, (Kuala Lumpur, Malaysia), pp. 1–6, May 2016.

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

[81] R. Karmakar and G. Kaddoum, "IBAC: An intelligent dynamic bandwidth channel access avoiding outside warning range problem," *IEEE Trans. Mobile Comput.*, pp. 1–15, Jan. 2022.

[82] H. Zhou, B. Li, Z. Yan, and M. Yang, "A channel bonding based QoS-aware OFDMA MAC protocol for the next generation WLAN," *Mobile Networks and Applications*, vol. 22, pp. 19–29, Feb. 2017.

[83] Q. Wu, W. Chen, M. Tao, J. Li, H. Tang, and J. Wu, "Resource allocation for joint transmitter and receiver energy efficiency maximization in downlink OFDMA systems," *IEEE Trans. Commun.*, vol. 63, pp. 416–430, Dec. 2015.

[84] Y.-D. Chen, D.-R. Wu, T.-C. Sung, and K.-P. Shih, "DBS: A dynamic bandwidth selection MAC protocol for channel bonding in IEEE 802.11ac WLANs," in *IEEE WCNC*, (Barcelona, Spain), pp. 1–6, Apr. 2018.

[85] J. Herzen, R. Merz, and P. Thiran, "Distributed spectrum assignment for home WLANs," in *IEEE INFOCOM*, (Turin, Italy), pp. 1573–1581, July 2013.

[86] S. Rayanchu, V. Shrivastava, S. Banerjee, and R. Chandra, "FLUID: Improving throughputs in enterprise wireless LANs through flexible channelization," *IEEE Trans. Mobile Comput.*, vol. 11, pp. 1455–1469, Apr. 2012.

[87] M. Han, S. Khairy, L. X. Cai, Y. Cheng, and F. Hou, "Capacity analysis of opportunistic channel bonding over multi-channel WLANs under unsaturated traffic," *IEEE Trans. Commun.*, vol. 68, pp. 1552–1566, Dec. 2020.

[88] H. Qi, H. Huang, Z. Hu, X. Wen, and Z. Lu, "On-demand channel bonding in heterogeneous wlans: A multi-agent deep reinforcement learning approach," *Sensors*, vol. 20, pp. 1–16, May 2020.

[89] S.-s. Lee, T. Kim, S. Lee, K. Kim, Y. H. Kim, and N. Golmie, "Dynamic channel bonding algorithm for densely deployed 802.11ac networks," *IEEE Trans. Commun.*, vol. 67, pp. 8517–8531, Sept. 2019.

[90] X. Wang, P. Huang, J. Xie, and M. Li, "OFDMA-based channel-width adaptation in wireless mesh networks," *IEEE Trans. Veh. Technol.*, vol. 63, pp. 4039–4052, Feb. 2014.

[91] A. Nabil, M. J. Abdel-Rahman, and A. B. MacKenzie, "Adaptive channel bonding in wireless LANs under demand uncertainty," in *IEEE PIMRC*, (Montreal, QC, Canada), pp. 1–7, oct 2017.

[92] O. Iacoboaiea, J. Krolikowski, Z. Ben Houidi, and D. Rossi, "Real-time channel management in WLANs: Deep reinforcement learning versus heuristics," in *IEEE IFIP Networking*, pp. 1–9, July 2021.

[93] T. Moscibroda, R. Chandra, Y. Wu, S. Sengupta, P. Bahl, and Y. Yuan, "Load-aware spectrum distribution in wireless LANs," in *IEEE ICNP*, (Orlando, FL, USA), pp. 137–146, Oct. 2008.

[94] R. Karmakar, S. Chattopadhyay, and S. Chakraborty, "SmartBond: A deep probabilistic machinery for smart channel bonding in IEEE 802.11ac," in *IEEE INFOCOM*, (Toronto, ON, Canada), pp. 2599–2608, July 2020.

[95] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *NeurIPS*, (Los Angeles, CA, USA), p. 6379–6390, Dec. 2017.

[96] A. Galindo-Serrano, L. Giupponi, and M. Dohler, "Cognition and docition in OFDMA-based femtocell networks," in *IEEE GLOBECOM*, (Miami, FL, USA), pp. 1–6, Dec. 2010.

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

[98] H. Saad, A. Mohamed, and T. ElBatt, "A cooperative Q-learning approach for distributed resource allocation in multi-user femtocell networks," in *IEEE WCNC*, (Istanbul, Turkey), pp. 1490–1495, Apr. 2014.

[99] A. Galindo-Serrano, L. Giupponi, and G. Auer, "Distributed learning in multiuser OFDMA femtocell networks," in *IEEE VTC*, (Yokohama, Japan), pp. 1–6, May 2011.

[100] A. Galindo-Serrano and L. Giupponi, "Distributed Q-learning for aggregated interference control in cognitive radio networks," *IEEE Transactions on Vehicular Technology*, vol. 59, pp. 1823–1834, May 2010.

[101] A. Galindo-Serrano and L. Giupponi, "Aggregated interference control for cog-

nitive radio networks based on multi-agent learning," in *IEEE CROWNCOM*, (Hannover, Germany), pp. 1–6, June 2009.

[102] O. van den Biggelaar, J. Dricot, P. D. Doncker, and F. Horlin, "Power allocation in cognitive radio networks using distributed machine learning," in *IEEE PIMRC)*, (Sydney, NSW, Australia), pp. 826–831, Sept. 2012.

[103] R. Karmakar, S. Chattopadhyay, and S. Chakraborty, "Dynamic link adaptation in IEEE 802.11ac: A distributed learning based approach," in *IEEE LCN*, (Dubai, United Arab Emirates), pp. 87–94, Nov. 2016.

[104] R. Karmakar, S. Chattopadhyay, and S. Chakraborty, "SmartLA: Reinforcement learning-based link adaptation for high throughput wireless access networks," *Computer Communications*, vol. 110, pp. 1–25, Sept. 2017.

[105] S. Cho, "Reinforcement learning for rate adaptation in CSMA/CA wireless networks," in *Advances in Computer Science and Ubiquitous Computing*, (Singapore), pp. 175–181, Feb. 2021.

[106] R. Karmakar, S. Chattopadhyay, and S. Chakraborty, "Intelligent MU-MIMO user selection with dynamic link adaptation in IEEE 802.11ax," *IEEE Trans. Wireless Commun.*, vol. 18, pp. 1155–1165, Feb. 2019.

[107] P. V. R. Ferreira, R. Paffenroth, A. M. Wyglinski, T. M. Hackett, S. G. Bilén, R. C. Reinhart, and D. J. Mortensen, "Multiobjective reinforcement learning for cognitive satellite communications using deep neural network ensembles," *IEEE J. Sel. Areas Commun.*, vol. 36, pp. 1030–1041, May 2018.

[108] G. Naddafzadeh-Shirazi, P.-Y. Kong, and C.-K. Tham, "Distributed reinforcement learning frameworks for cooperative retransmission in wireless networks," *IEEE Trans. Veh. Technol*, vol. 59, pp. 4157–4162, July 2010.

[109] N. Mastronarde, J. Modares, C. Wu, and J. Chakareski, "Reinforcement

learning for energy-efficient delay-sensitive CSMA/CA scheduling," in *IEEE GLOBECOM*, (Washington, DC, USA), pp. 1–7, Feb. 2016.

[110] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Trans. Veh. Technol.*, vol. 68, pp. 3163–3173, Feb. 2019.

[111] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, pp. 257–265, June 2018.

[112] Q. Zhao, B. Krishnamachari, and K. Liu, "On myopic sensing for multi-channel opportunistic access: structure, optimality, and performance," *IEEE Trans. Wireless Commun.*, vol. 7, pp. 5431–5440, Dec. 2008.

[113] M. Chu, H. Li, X. Liao, and S. Cui, "Reinforcement learning-based multiaccess control and battery prediction with energy harvesting in IoT systems," *IEEE Internet Things J.*, vol. 6, pp. 2009–2020, Apr. 2019.

[114] U. Challita, L. Dong, and W. Saad, "Proactive resource management for LTE in unlicensed spectrum: A deep learning perspective," *IEEE Trans. Wireless Commun.*, vol. 17, pp. 4674–4689, July 2018.

[115] B. Yin, K. Yamamoto, T. Nishio, M. Morikura, and H. Abeysekera, "Learning-based spatial reuse for WLANs with early identification of interfering transmitters," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, pp. 151–164, Nov. 2020.

[116] M. Tonnemacher, C. Tarver, V. Chandrasekhar, H. Chen, P. Huang, B. L. Ng, J. Charlie Zhang, J. R. Cavallaro, and J. Camp, "Opportunistic channel access using reinforcement learning in tiered CBRS networks," in *IEEE Int. Symp. Dyn. Spectr. Access Netw.*, (Seoul, South Korea), pp. 1–10, Jan. 2018.

[117] F. Wilhelmi, B. Bellalta, C. Cano, and A. Jonsson, "Implications of decen-

tralized Q-learning resource allocation in wireless networks," in *IEEE PIMRC*, (Montreal, QC, Canada), pp. 1–5, Feb. 2017.

[118] O. Avner and S. Mannor, "Multi-user lax communications: A multi-armed bandit approach," in *IEEE INFOCOM*, (San Francisco, CA, USA), pp. 1–9, Apr. 2016.

[119] Y. Luo and K.-W. Chin, "Learning to bond in dense WLANs with random traffic demands," *IEEE Trans. Veh. Technol.*, vol. 69, pp. 11868–11879, July 2020.

[120] Y. Luo and K.-W. Chin, "An energy efficient channel bonding and transmit power control approach for WiFi networks," *IEEE Trans. Veh. Technol.*, vol. 70, pp. 8251–8263, July 2021.

[121] Q. Li, Z. Hu, X. Wen, Z. Lu, and H. Qi, "On-line learning algorithm for dynamic sensitivity control in IEEE 802. 11ax network," *The Journal of China Universities of Posts and Telecommunications*, vol. 25, pp. 67–74, Oct. 2018.

[122] P. Zhu, J. Pinto, T. Nguyen, and A. Fern, "Achieving quality of service with adaptation-based programming for medium access protocols," in *IEEE GLOBECOM*, (Anaheim, CA, USA), pp. 1932–1937, Apr. 2012.

[123] R. Ali, A. Nauman, Y. B. Zikria, B.-S. Kim, and S. W. Kim, "Performance optimization of QoS-supported dense WLANs using machine-learning-enabled enhanced distributed channel access (MEDCA) mechanism," *Neural Computing and Applications*, vol. 32, pp. 13107–13115, Aug. 2020.

[124] W. Wydmański and S. Szott, "Contention window optimization in IEEE 802.11ax networks with deep reinforcement learning," in *IEEE WCNC*, (Nanjing, China), pp. 1–6, May 2021.

[125] F. Wilhelmi, S. Barrachina-Muñoz, B. Bellalta, C. Cano, A. Jonsson, and G. Neu, "Potential and pitfalls of multi-armed bandits for decentralized spatial

reuse in WLANs," *Journal of Network and Computer Applications*, vol. 127, pp. 26–42, Feb. 2019.

[126] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997.

[127] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, pp. 3133–3174, May 2019.

[128] R. A. Howard, "Dynamic programming and markov processes.," *Cambridge:Massachusetts: MIT Press*, 1960.

[129] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[130] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, May 1992.

[131] L.-J. Lin, *Reinforcement Learning for Robots Using Neural Networks.* PhD thesis, Carnegie Mellon University, USA, 1992.

[132] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning (ICML)*, (Amherst MA USA), pp. 1–5, July 1993.

[133] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *13th AAAI Conf. Artif. Intell.*, (Phoenix, AZ, USA), pp. 2094–2100, Mar. 2016.

[134] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. 33rd Int.*

Conf. Mach. Learn. (ICML), (New York, New York, USA), pp. 1995–2003, June 2016.

[135] I. Selinis, M. Filo, S. Vahid, J. Rodriguez, and R. Tafazolli, "Evaluation of the DSC algorithm and the BSS color scheme in dense cellular-like IEEE 802.11ax deployments," in *IEEE PIMRC*, (Valencia, Spain), pp. 1–7, Sept. 2016.

[136] M. Viswanathan, "Log distance path loss or log normal shadowing model," Sept. 2013. Accessed 2022-04-12.

[137] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Int. Conf. Learn. Represent. (ICLR)*, (San Diego, CA, USA), May 2015.

[138] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," *arXiv preprint arXiv:1603.04467*, Mar. 2016. Software available from tensorflow.org, accessed on 2019-02-02.

[139] F. Chollet *et al.*, "Keras," 2015. accessed on 2019-02-02.

[140] J. Heaton, *Introduction to neural networks with Java*, ch. 5, pp. 157–158. Heaton Research, Inc., 2008.

[141] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, pp. 181–211, Aug. 1999.

[142] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall, Sept. 2002.

[143] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete event dynamic systems*, vol. 13, pp. 41–77, Jan. 2003.

[144] T. Zahavy, M. Haroush, N. Merlis, D. J. Mankowitz, and S. Mannor, "Learn

what not to learn: Action elimination with deep reinforcement learning," in *NeurIPS*, (Montréal, CANADA), p. 3562–3573, Dec. 2018.

[145] J. Liu, B. Krishnamachari, S. Zhou, and Z. Niu, "DeepNap: Data-driven base station sleeping operations through deep reinforcement learning," *IEEE Internet Things J.*, vol. 5, pp. 4273–4282, Dec. 2018.

[146] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, *et al.*, "Continuous control with deep reinforcement learning.," in *ICLR*, (San Juan, Puerto Rico), May 2016.

[147] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *ICML*, (Beijing, China), pp. 387–395, June 2014.

[148] IEEE, "IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pp. 1–3534, Dec. 2016.

[149] L. Busoniu, R. Babuska, and B. D. Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C*, vol. 38, pp. 156–172, Mar. 2008.