Advanced Reinforcement Learning-based Optimization Techniques for Wireless Access Networks

by

Pedro Iturria Rivera

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the
Doctorate of Philosophy
in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

To
my beloved parents Mercy and Pedro,
my "abuelo" Delio and "abuela" Norma,
my sister Daylin,
my lovely Sonia and my family.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Melike Erol-Kantarci, for the opportunity to work under her guidance during these past four years. She has been an exceptional role model, teaching me not only how to be a better professional, but also how to be a better person. Without her constant guidance, this compiled work would not have been possible.

I would also like to extend my sincere thanks to all of my collaborators and friends who have supported me throughout this journey. In particular, I would like to thank my lab colleagues Ycaro Dantas, Arafat Habib, Han Zhang, Hao Zhou, Shavbo Salehi, Medhat Elsayed, Shahram Mollahasani, Turgay Pamuklu, Roghayeh Joda, and Marco Skocaj for their invaluable contributions and for sharing many memorable moments with me.

Furthermore, I would like to express my gratitude to all of my industry collaborators, especially Professor Burak Kantarci, Marcel Chenier, Majid Bavand, and the NetExperience/Ericsson team, for their support and collaboration.

Thank you all for your guidance, encouragement, and friendship. I am truly grateful for the opportunity to have worked with such an amazing group of people.

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University's regulations concerning plagiarism, including those concerning consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

Next-generation wireless networks have grown increasingly complex over the years due to the continuous demand generated by newer services like Extended Reality (XR), encompassing Augmented Reality (AR), Mixed Reality (MR), and Virtual Reality (VR).

5^{th} generation networks (5G) introduced three main network services to provide support to new traffic types with diverse quality of service demands: Enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low Latency Communications (URLLC), and massive Machine Type Communications (mMTC). Despite the advancements achieved by 5G, there is still much room for improvement when meeting the needs of more data-intensive, low-latency, and ultra-high-reliability applications. In new generation Wi-Fi networks, Wi-Fi 6, 7, and the futuristic Wi-Fi 8, the race has been dedicated to providing extremely high throughput required by throughput-hungry services such as XR. Despite the success of the changes introduced in such amendments, Wi-Fi still falls short of providing ultra-high throughput with ultra-low latency and high reliability for applications such as cooperative mobile robots and others [1].

To suffice such complexity and myriad of requirements, Machine Learning (ML)-based solutions have stepped up and provided elegant and efficient solutions for many challenging wireless communications problems [2]. Several advances in ML in the field of computer vision, natural language processing, game playing, and robotics have allowed the migration of many of these applications to the field of wireless communications. More specifically, we foresee the immense application potential that Reinforcement Learning (RL), a subfield of ML, can provide due to its capacity to learn, as humans do, complex systems by interacting with an unknown environment.

In this work, we aim to apply advanced state-of-the-art RL techniques on wireless access networks to optimize network performance. As case studies, we selected load balancing, handover in multi-RAT networks, XR codec selection, resource management in 4^{th}, 5^{th}, and beyond networks. We propose centralized and hierarchical RL-based solutions and explore the advantages of team learning and multi-agent reinforcement learning (MARL) in the proposed use cases. Our novel MARL algorithms demonstrate their capabilities to overperform centralized ones when the problem can be modeled as cooperative or competitive. Similarly, we study advanced state-of-the-art RL techniques in Wi-Fi networks. In this case, we explored spatial reuse, traffic allocation, and channel selection for IEEE 802.11ax and IEEE 802.11be networks.

The results of this investigation underscore the effectiveness of various reinforcement learning (RL) techniques, especially in the multi-agent setting, across diverse application

domains and wireless access networks. Firstly, competitive multi-agent reinforcement learning (MARL) schemes yield better results than centralized ones, enabling agents to compete more efficiently for resources in a load-balancing scenario. Secondly, hierarchical algorithms offer more optimal solutions in dual connectivity handovers compared to centralized ones, thanks to their unique capacity to handle sparse rewards. Thirdly, in Wi-Fi networks, cooperative spatial reuse appears to enhance collaboration among agents, and transfer reinforcement learning (TRL) facilitates quick adaptation in dynamic environments. Fourthly, additional methods need to be incorporated to address partially observable Markov decision process (POMDP) problems, a common characteristic in wireless networks, to enable successful utilization of RL. Finally, parallel transfer reinforcement learning (PTRL) can significantly improve convergence speed without the need for the classical teacher-student paradigm in sequential transfer reinforcement learning (TRL).

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Acronym	Meaning
AI	Artificial Intelligence
AP	Access Point
AR	Augmented Reality
BS	Base Station
CDQL	Clipped Double Q-Learning
DDQN	double Deep Q-Network
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
DTRL	deep transfer reinforcement learning
eMBB	Enhanced Mobile Broadband
HiDQL	Hierarchical Deep Q-Learning
HO	Handover
hRL	hierarchical Reinforcement Learning
KPI	Key Performance Indicator
LB	Load Balancing
MAB	Multi-Armed Bandit
MAC	Media Access Control
MADDPG-AP	Multi-Agent Deep Deterministic Policy Gradient with Adaptive Policies
MA-MAB	Multi-Agent Multi-Armed Bandit
MA-CMAB	Contextual Multi-Agent Multi-Armed Bandit
MADDPG	Multi-agent Deep Deterministic Policy Gradient
MARL	multi-agent reinforcement learning
MAS	Multi-Agent Systems
MDP	Markov Decision Process
MH-RSAC	Multi-Headed Recurrent Soft-Actor Critic

ML	Machine Learning
MLO	Multi-Link Operation
mMTC	massive Machine Type Communications
MR	Mixed Reality
oQMIX	Optimistic QMIX
O-RAN	Open RAN
PDB	Packet Delay Budget
PLR	Packet Loss Ratio
PTRL	Parallel Transfer Reinforcement Learning
QMIX	Mixture of Q-Functions
QoE	Quality of Experience
QoS	Quality of Service
RAN	radio access network
RAT	Radio Access Technology
RBG	Resource Block Group
ReBUHA	Resource Block Utilization based Handover Algorithm
RL	Reinforcement Learning
RRM	Radio Resource Management
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
SL	Supervised Learning
SLO	Single-Link Operation
SR	Spatial Reuse
TD	Temporal Difference
TL	Transfer Learning
TRL	Transfer Reinforcement Learning
UE	User Equipment
UEs	User Equipments
UL	Unsupervised Learning
URLLC	Ultra-Reliable and Low Latency Communications
VDN	Value Decomposition Networks
VR	Virtual Reality
XR	Extended Reality

List of Symbols

α	learning rate
ϵ	exploration
γ	discount factor
$\mathbb{E}$	expected value
$\mathcal{D}$	experience buffer
$\mathcal{G}$	set of goals
$\mathcal{M}$	set of access points
ϕ	cell individual offset
π	policy
τ	action-observation history
θ	target network parameters
θ'	evaluation network parameters
A	The set of actions
a	action
D	delay
f_c	carrier frequency
G	discount future reward

M	set of base stations
N	set of user devices
N^{RB}	set of resource blocks
N_0	noise power density
O	complexity
P_cs	CCA threshold
P_tx	transmission power
Pr	transition probability
Q	state-action value function
r	reward
R_T	composite reward
$rand$	a random number between 0 and 1
$randint$	a random integer number
S	The set of states
s	state
T	episodes
u	the agents' joint action
V	state value
W	bandwidth

Chapter 1

Introduction

1.1 Motivations

In the era of Artificial Intelligence (AI)-based solutions, ML and its subfield RL have shown enormous potential across a myriad of services in our daily lives. Their applications span from industrial and robotics to healthcare and wireless communications. The complexity of today's systems has led to the usage of AI to provide solutions that were not possible to achieve with the existing methods. ML utilizes existing data and learns from it, which is the primary distinction from traditional algorithms based on rule lists [5]. ML can be classified into three subfields: Supervised Learning (SL), Unsupervised Learning (UL), and RL. SL necessitates that each training example has its corresponding label for proper functioning. The training example corresponds to the input, while the label represents the output. This method is primarily suitable for classification or prediction tasks. In contrast, UL does not require labels and aims to infer inherent features of the data. This enables inference of information such as grouping similar data points, detection of anomalous instances, or dimensionality reduction, among others, in tasks where labels are not available [6]. Lastly, RL is a learning method used in uncertain environments where the dynamics of the process can be modeled as a Markov Decision Process (MDP). In this setting, an agent or learning entity interacts with the environment using a defined action, receives a reward or penalty for such action, and observes the current state of the environment. This corresponds to a cyclic process, which aims to select, in the long term, such actions that provide the highest reward [7].

Wireless access networks are characterized by their inherent uncertainty. Moreover, adjusting the multiple parameters involved in any network decision can be challenging, and

in some cases, heuristic methods are utilized due to the absence of a closed-form solution [8]. In multi-agent environments, such as in multi-base station and multi-user settings in wireless networks, these challenges become even more pronounced [5]. Furthermore, many network functions in wireless access networks require optimization of certain objectives. In next-generation networks, such optimization often involves solving non-linear or polynomial problems that cannot be addressed by classical optimization techniques. Consequently, less expensive solutions, such as those provided by Machine Learning (ML), are necessitated [9].

For this reason, we intend to explore the application of several advanced RL techniques, mostly in the multi-agent context, in wireless access networks: 5^{th} and 6^{th} generation cellular networks and Wi-Fi. In this thesis introduction, we present in more detail the motivations behind this research.

1.1.1 Reinforcement Learning for Next Generation Cellular Networks

The fifth and sixth generation (5^{th} and 6^{th}) cellular networks extend their capabilities to offer services with more demanding requirements. These include increased throughput, reduced latency, enhanced reliability, higher connection density, improved energy efficiency, and integrated intelligence with machine learning capabilities. Several studies have shown the success of RL, a subfield of ML, in next-generation cellular networks. Several surveys can be found throughout the literature where ML and 5G and 6G have been the main focus. Such surveys span from general state of the art [10–15], positioning [16], vehicular networks [17], large scale 6G networks [18], quantum ML, [19], QoS and Quality of Experience (QoE) [20], non-terrestrial networks [21], distributed ML [22] and handover for 5G and 6G networks [23]. It can be seen, that most of the ML algorithms applied in radio access network (RAN) need to be adapted, tweaked, or enhanced to perform properly in wireless networks. This makes sense, since typically the ML community proposed their methods in environments more related to robotics or game playing. Differently from the classical RL research, we delve in this thesis, in specific areas of Radio Access Networks where RL can provide improved solutions when compared with traditional mechanisms. Among the plethora of applications of RL-enabled Wireless RAN, we study load balancing, handover, and XR codec adaptation. In each of these studies, we intend to optimize Key Performance Indicator (KPI)s of interest and comply with the newest stringent requirements imposed by next-generation networks.

In recent years, RL-based load balancing solutions have been proposed throughout the literature. The term load balancing in wireless networks is a broad concept and if

used, must be specified the context in which is employed. In this thesis, we refer to load balancing as the balance and management of resource utilization across different network components, such as base stations, to ensure efficient and optimal network performance. The goal of load balancing is to prevent network congestion, enhance resource utilization, and improve overall user experience. The term "traffic steering" can also be employed as a form of load balancing since it deals with the distribution of data traffic; however, we do not study load balancing in such a sense.

Load balancing within the RAN involves redistributing User Equipments (UEs) to less congested base stations. The load on a base station is determined by the number of UEs associated with it and the traffic demand generated by these UEs. To redistribute UEs, handover mechanisms are used [24]. All the existing handover mechanisms use some metrics such as Reference Signal Received Quality (RSRQ) and Reference Signal Received Power (RSRP) to trigger the handover and TTT, time to trigger, which sets the time in which the handover mechanisms should wait to be triggered to avoid ping-pong scenarios. Despite, the effectiveness of the existent algorithms, the increase User Equipment (UE)'s density and diversity of traffic types' quality of service requirements sets some challenges in finding the optimal load distribution in RAN. Recent developments with Cloud RAN (C-RAN) have enabled the proposal of centralized RL solutions where an agent can control the configuration parameters of several Base Stations (BSs). Existing literature about this topic does not consider QoS parameters at the time of triggering handover. To this end, in Chapter 3, we introduce in section 3.1 our proposed CDQL algorithm for load balancing [25] that considers QoS metrics to improve KPI of interest. However, centralized solutions are not pragmatic and decentralization is needed. In a decentralized scenario, each Base Station (BS) tries to improve its own KPIs and trigger the handover based on the optimized Handover (HO). Such scenario can be modeled as a multi-agent system, that could present two characteristics according to the nature of the agents' interactions. Agents can act collaboratively or competitively nature. In the cooperative setting, agents collaborate to optimize a common long-term goal, meanwhile, in the competitive setting, the return of agents usually sums up to zero. Collaboration seems to contribute to better performance in these settings but typically the cost of communication and ideal communication among these *networked agents* is often assumed. Thus, competition is preferred in specific scenarios where the performance is reasonably maintained comparably to cooperative agents. Thus, in section 3.2 we propose a novel state-of-the-art MARL competitive algorithm named Multi-Agent Deep Deterministic Policy Gradient with Adaptive Policies (MADDPG- AP) that is capable of overperforming the centralized our previous CDQL algorithm [26].

Due to the evolution of the RAN from 4^{th} to 5^{th} generation, handover mechanisms

adapted to support different Radio Access Technology (RAT) users. Service to older terminals was continuously offered and new terminals with multi-RAT capability needed to seamlessly use both networks that have different radio characteristics. Some proposals have been made for handover in multi-RAT networks [27]. In section 3.3, we address the handover problem in an LTE-NR network with dual connectivity using a hierarchical algorithm named Hierarchical Deep Reinforcement Learning (HiDQL) [28] that is capable of offering better performance than CDQL and the existent algorithmic solutions.

In the context of the new generation RAN, O-RAN emerges as a novel and promising architectural paradigm. Among its many characteristics, it features community collaboration, interoperability, a multi-vendor ecosystem, and the decomposition of network elements, essentially embodying the concept of openness. To enable these exciting features, O-RAN incorporates applications referred to as xApps and rApps. These applications are specifically designed to enhance the functionality and capabilities of the open architecture, with their main differentiation lying in their location and requirements within this new framework. In Chapter 4, we present a pioneering study supporting the implementation of team learning in O-RAN. In this context, two distinct xApps—power allocation and radio resource allocation—can significantly improve network performance when team learning is employed instead of individual learning agents [29].

1.1.2 Reinforcement Learning for Wi-Fi Networks

The upcoming generations of Wi-Fi networks, including IEEE 802.11be (commercially known as Wi-Fi 7) and the future IEEE 802.11bn (potentially named Wi-Fi 8), address extremely high and ultra-high throughput services with stringent latency requirements, respectively. Such services demand high reliability and coordination, necessitating intelligent management capable of adapting to dense and crowded environments. The success of machine learning (ML) in cellular networks can be analogously claimed in Wi-Fi. Several studies have demonstrated the success of reinforcement learning (RL), a subfield of ML, in next-generation Wi-Fi. Despite Wi-Fi and ML works are not abundant as its competitor 5G and 6G some studies can be found with the topic of general state of the art [30–32], Indoor Localization [33–36], QoS [37], sensing [38], human recognition [39,40] and intrusion detection [41]. As for 5G and 6G, the same conclusion can be drawn for Wi-Fi: the majority of ML algorithms used in Wi-Fi require adaptation, adjustments, or enhancements to achieve optimal performance.

Amidst the numerous applications of Wi-Fi empowered by ML, we explore coordinated spatial reuse, traffic allocation, and channel selection. Spatial reuse in Wi-Fi involves

efficiently utilizing available frequency channels, enabling multiple devices to transmit and receive data simultaneously without interference. Optimizing this task becomes challenging in densely populated areas like airports and shopping malls. Motivated by these challenges, we present several contributions to coordinated spatial reuse in Chapter 5.1, including the Contextual Multi-Agent Multi-Armed Bandit (MA-CMAB) algorithm and the proposal of Transfer Reinforcement Learning (TRL) to expedite its adaptation [42]. In Section 5.2, we focus on adaptability and propose a meta-learning algorithm based on Multi-Agent Multi-Armed Bandit (MA-MAB). Meta-learning demonstrates its immense potential when access to several tasks is possible, surpassing TRL techniques with the advantage of not experiencing negative transfer learning [43].

The arrival of the new IEEE 802.11be (Wi-Fi 7) amendment brought radical changes such as Multi-Link Operation (MLO). MLO in Wi-Fi refers to a device's capability to establish and manage connections with multiple channels simultaneously with one Access Point (AP), enhancing network performance and reliability. Consequently, this new feature affects traffic allocation and channel selection for those MLO-capable devices. Despite state-of-the-art algorithmic methods showing good performance, in Section 5.4, we demonstrate that RL is capable of taking better actions. To do so, in our first work, we propose the usage of Multi-Headed Recurrent Soft-Actor Critic (MH-RSAC) algorithm to address several challenges imposed by this new feature on traffic allocation [44]. Later on, we present our Parallel Transfer Reinforcement Learning (PTRL) Optimistic-Weighted Value Decomposition Networks algorithm to improve intelligent channel selection in IEEE 802.11be MLO-capable networks [45]. We observe that PTRL is a feasible alternative to classical sequential TRL.

1.2 Thesis Contributions and Organization

This thesis applies advanced single and multi-agent state-of-the-art RL algorithms to wireless access networks such as Wi-Fi and 4^{th} and 5^{th} Generation Radio Access Networks. We explore diverse algorithms from CDQL, MADDPG, Mixture of Q-Functions (QMIX) and address common challenges in RL such as convergence time using diverse techniques such as TRL, meta-learning, and PTRL. Our playground corresponds to the latest Wi-Fi standards as IEEE 802.11be and IEEE 802.11ax and 5G RAN. The organization is centered around the idea of addressing current challenges in wireless networks using reinforcement learning (RL) optimization techniques. Starting with load balancing and handover in 4G/5G, we proceed with an analysis of team learning in Open RAN architecture. Then, we continue with a chapter dedicated to Wi-Fi capacity and conclude with a study on application codec

optimization in 6G. All works share a common focus: the integration of RL and wireless networks.

As a closing remark, this thesis proves the promising applications of RL methods for optimization into modern wireless access networks and what could be done to make them a reality. More details are summarized as follows.

1.2.1 Reinforcement Learning based Load Balancing and Handover in 4G and 5G Networks

In chapter 3, we study load balancing in 4G and 5G networks using two state-of-the-art algorithms CDQL and MADDPG [25, 46]. We address the load balancing problem where we seek to balance resource block utilization and QoS metrics such as latency and throughput. To do so, in subsection 3.1 we use Cllipped Double Q-learning which tackles the overestimation issues presented by previous deterministic action space RL algorithms. We consider a centralized approach where our agent can choose the best cell individual offset (CIO) per BSs that maximizes the key performance indicators of the network. Such centralized learning can be utilized in Cloud RAN (C-RAN). We compare our results with the classical A3 Handover algorithm and a simplistic resource block utilization-based baseline handover algorithm. Although single-agent RL has shown successful performance in many problems, they might not be sufficient to fulfill the expectation in terms of reliability, latency, and efficiency, especially in wireless networks where either competition for a resource or coordination among a set of agents is essential [5]. Therefore a more realistic approach requires considering the interaction between multiple agents with the environment such as in MARL [47]. Consequently, we proposed a MARL algorithm named Multi-Agent Deep Deterministic Policy Gradient with Adaptive Policies (MADDPG-AP) in subsection 3.2 that represented a more realistic scenario and offered better performance than the centralized approach.

Furthermore, we address the optimization of handover latency in dual-connectivity architectures using a novel hierarchical Reinforcement Learning (hRL) method in subsection 3.3. We propose two reinforcement learning algorithms: a single agent RL algorithm named Clipped Double Q-Learning (CDQL) and a hierarchical Deep Q-Learning (HiDQL) to improve Multiple Radio Access Technology (multi-RAT) dual-connectivity handover [28]. We compare our proposal with two algorithmic existent baselines: a fixed parameter and a dynamic parameter solution. We obtained that both RL solutions improved algorithmic baselines and realized that HiDQL presented a slower convergence time concerning CDQL, but offered a more optimal solution than CDQL. Additionally, we foresee the advantages

of utilizing context-information as geo-location of the UEs to reduce the beam exploration sector, thus improving further multi-RAT handover latency results.

1.2.2 Multi-Agent Team Learning in Virtualized Open Radio Access Networks (O-RAN)

In chapter 4, we presented a novel work about the possible uses, open issues, and future directions of team learning in O-RAN. Starting from the concept of the Cloud Radio Access Network (C-RAN), continuing with the virtual Radio Access Network (vRAN), and most recently with the Open RAN (O-RAN) initiative, Radio Access Network (RAN) architectures have significantly evolved in the past decade. In the last few years, the wireless industry has witnessed a strong trend towards disaggregated, virtualized, and open RANs, with numerous tests and deployments worldwide. One unique aspect that motivates this paper is the availability of new opportunities that arise from using machine learning, more specifically multi-agent team learning (MATL), to optimize the RAN in a closed-loop where the complexity of disaggregation and virtualization makes well-known Self-Organized Networking (SON) solutions inadequate. In our view, Multi-Agent Systems (MASs) with MATL can play an essential role in the orchestration of O-RAN controllers, i.e., near-real-time and non-real-time RAN Intelligent Controllers (RIC). In this article, we first provide an overview of the landscape in RAN disaggregation, virtualization, and O-RAN, and then we present the state-of-the-art research in multi-agent systems and team learning as well as their application to O-RAN. We present a case study for team learning where agents are two distinct xApps: power allocation and radio resource allocation. We demonstrate how team learning can enhance network performance when team learning is used instead of individual learning agents. Finally, we identify challenges and open issues to provide a roadmap for researchers in the area of MATL-based O-RAN optimization.

1.2.3 Multi-agent Deep Reinforcement Learning for Next Generation Wi-Fi Networks

The exponential increase in the demand for high-performance services such as streaming video and gaming by wireless devices has posed several challenges for Wireless Local Area Networks (WLANs). In chapter 5, we study several topics that affect the newest Wi-Fi standards IEEE 802.11ax and IEEE 802.11be. Among them, we find spatial reuse, traffic allocation, and channel selection for Multi-Link Operation-capable Wi-Fi devices.

The newest standards, IEEE 802.11ax, and 802.11be, bring high data rates in dense user deployments. Additionally, they introduce new flexible features in the physical layer, such as dynamic Clear Channel-Assessment (CCA) thresholds, intending to improve spatial reuse (SR) in response to radio spectrum scarcity in dense scenarios. We formulate the Transmission Power (TP) and CCA configuration problem to maximize fairness and minimize station starvation. In the first section of this chapter, we present five main contributions to distributed SR optimization using Multi-Agent Multi-Armed Bandits (MA-MABs). First, we provide regret analysis for the distributed Multi-Agent Contextual MABs (MA-CMABs) proposed in this work. Second, we propose reducing the action space given the large cardinality of action combinations of TP and CCA threshold values per Access Point (AP). Third, we present two deep MA-CMAB algorithms, named Sample Average Uncertainty (SAU)-Coop and SAU-NonCoop, as cooperative and non-cooperative versions to improve SR. Additionally, we analyze the viability of using MA-MABs solutions based on the -greedy, Upper Bound Confidence (UCB), and Thompson (TS) techniques. Finally, we propose a deep reinforcement transfer learning technique to improve adaptability in dynamic environments. However, transfer learning techniques can sometimes suffer from negative transfer, where knowledge from the source task hinders performance on the target task. Thus, to mitigate Quality of Service (QoS) degradation, we introduce a solution that builds on meta-learning and multi-arm bandits in the second section of this chapter. Simulation results show that the proposed solution can adapt with an average Meta-learning techniques often incorporate mechanisms to minimize negative transfer effects.

In the second section of this chapter, we delve into Multi-Link Operation and its implications for traffic allocation in IEEE 802.11be networks. IEEE 802.11be -Extremely High Throughput- is the newest IEEE 802.11 amendment that comes to address the increasingly throughput-hungry services such as Ultra High Definition (4K/8K) Video and Virtual/Augmented Reality (VR/AR). To achieve this, IEEE 802.11be presents a set of novel features that will push Wi-Fi technology to its limits. Among them, MLO devices are anticipated to become a reality, surpassing Single-Link Operation (SLO) Wi-Fi of the past. To achieve superior throughput and very low latency, a careful design approach must be taken regarding how incoming traffic is distributed in MLO-capable devices. We present a Reinforcement Learning (RL) algorithm named MH-RSAC to distribute incoming traffic in 802.11be MLO capable networks. Moreover, we compare our results with two non-RL baselines previously proposed in the literature named: Single Link Less Congested Interface (SLCI) and Multi-Link Congestion-aware Load balancing at flow arrivals (MCAA).

At the end of this chapter, we address channel selection in IEEE 802.11be MLO. Among others, this new feature will increase the complexity of channel selection due to the novel multiple interfaces proposal. In this section, we present a Parallel Transfer Reinforce-

ment Learning (PTRL)-based cooperative Multi-Agent Reinforcement Learning (MARL) algorithm named Parallel Transfer Reinforcement Learning Optimistic-Weighted Value Decomposition Networks (oVDN) to improve intelligent channel selection in IEEE 802.11be MLO-capable networks. Additionally, we compare the impact of different parallel transfer learning alternatives and a centralized non-transfer MARL baseline. Two PTRL methods are presented: Multi-Agent System (MAS) Joint Q-function Transfer, where the joint Q-function is transferred, and MAS Best/Worst Experience Transfer where the best and worst experiences are transferred among MASs.

1.2.4 Multi-Agent Reinforcement Learning-based XR Codec Adaptation

Extended Reality (XR) services are set to bring revolutionizing applications over 5^{th} and 6^{th} generation wireless networks by delivering seamless and immersive virtual and augmented reality interactions to users. However, the stringent requirements of the new applications in several areas of industry and healthcare impose great challenges to the network infrastructure. Consequently, machine learning algorithms are considered promising techniques due to their adaptability and responsiveness in dynamic and complex environments. In chapter 6, we present a Multi-Agent Reinforcement Learning (MARL) solution capable of realizing codec parameter cross-optimization of XR traffic. Additionally, we compare our proposal with an existent state-of-the-art baseline named algorithm. Our proposal consists of a cooperative multi-agent system based on an Optimistic Mixture of Q-Values (oQMIX), where two agents handle one type of XR traffic: Augmented Reality (AR) and Virtual Reality (VR), and a third handles Cloud Gaming (CG) traffic. Furthermore, we leverage an attention mechanism and slate-Markov Decision Process (MDP) to improve the oQMIX algorithm's action selection. We observed that presented a more aggressive behavior with the tendency of higher throughput in all XR flows, increasing packet collisions and packet losses when the distance between the UE and the gNB increases. Conversely, oQMIX presented a more conservative behavior reducing PLR and maintaining a similar behavior in terms of goodput to while achieving lower PLR, delay, and jitter.

1.3 Publications

In this thesis, the research considered exclusively pertains to first authorship. When accounting for research with second and third authorship, the total number of compiled works

amounts to 6 patents, 4 journal articles, 13 conference proceedings, and 3 studies currently under submission.

1.3.1 Patents

[P01] **P. E. Iturria Rivera**, M. Erol-Kantarci, M. Bavand, R. Gaigalas, M. Elsayed, S. Furr, "Dual Connectivity Handover Optimization using Reinforcement Learning", US provisional patent filed on 13 April 2023.

[P02] **P. E. Iturria Rivera**, M. Erol-Kantarci, M. Bavand, R. Gaigalas, M. Elsayed, Y.Ozcan, "XR Codec Adaptation using Multi-Agent Reinforcement Learning with Attention Action Selection in 5G Networks", US provisional patent application on 9 November, 2023.

1.3.2 Journals

[J01] **P. E. Iturria-Rivera**, H. Zhang, H. Zhou, S. Mollahasani, and M. Erol-Kantarci, "Multi-Agent Team Learning in Virtualized Open Radio Access Networks (O-RAN)," *Sensors*, vol.22, no.14, pp.1-13, Jul. 2022.

[J02] **P. E. Iturria-Rivera**, M. Chenier, B. Herscovici, B. Kantarci and M. Erol-Kantarci, "Meta-Bandit: Spatial Reuse Adaptation via Meta-Learning in Distributed Wi-Fi 802.11ax", in *IEEE Networking Letters*, vol. 5, no. 4, pp. 179–183, 2023, doi: 10.1109/LNET.2023.3268648.

[J03] **P. E. Iturria-Rivera**, M. Chenier, B. Herscovici, B. Kantarci and M. Erol-Kantarci, "Cooperate or Not Cooperate: Transfer Learning With Multi-Armed Bandit for Spatial Reuse in Wi-Fi", in *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 2, pp. 351-369, 2024, doi: 10.1109/TMLCN.2024.3371929.

1.3.3 Conference Papers

[C01] **P. E. Iturria-Rivera** and M. Erol-Kantarci, "QoS-Aware Load Balancing in Wireless Networks using Clipped Double Q-Learning", IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS), Denver, CO, USA, 2021, pp. 10-16, doi: 10.1109/MASS52906.2021.00011.

[C02] **P. E. Iturria-Rivera** and M. Erol-Kantarci, "Competitive Multi-Agent Load Balancing with Adaptive Policies in Wireless Networks", IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2022, pp. 796-801, doi: 10.1109/CCNC49033.2022.9700667.

[C03] **P. E. Iturria-Rivera**, M. Elsayed, M. Bavand, R. Gaigalas, S. Furr and M. Erol-Kantarci, "Hierarchical Deep Q-Learning Based Handover in Wireless Networks with Dual Connectivity," IEEE Global Communications Conference, Rio de Janeiro, Brazil, 2022, pp. 6553-6558, doi:10.1109/GLOBECOM48099.2022.10000894.

[C04] **P. E. Iturria-Rivera**, M. Chenier, B. Herscovici, B. Kantarci and M. Erol-Kantarci, "RL meets Multi-Link Operation in IEEE 802.11 be: Multi-Headed Recurrent Soft-Actor Critic-based Traffic Allocation," IEEE International Conference on Communications, Rome, Italy, 2023, pp. 4001-4006, doi: 10.1109/ICC45041.2023.10279008 **(Best Paper Award)**.

[C05] **P. E. Iturria-Rivera**, M. Chenier, B. Herscovici, B. Kantarci and M. Erol-Kantarci, "Channel Selection for Wi-Fi 7 Multi-Link Operation via Optimistic-Weighted VDN and Parallel Transfer Reinforcement Learning," IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Toronto, ON, Canada, 2023, pp. 1-6, doi: 10.1109/PIMRC56721.2023.10293832.

[C06] **P. E. Iturria-Rivera**, R. Gaigalas, M. Elsayed, M. Bavand, Y. Ozcan and M. Erol-Kantarci, "Extended Reality (XR) Codec Adaptation in 5G using Multi-Agent Reinforcement Learning with Attention Action Selection", IEEE 35th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Valencia, Spain, 2024, pp. 1-6.

Chapter 2

Background and Literature Review

This chapter first introduces the research background, especially reinforcement learning techniques such as RL, Deep Reinforcement Learning (DRL), MARL, TRL, PTRL, and so on. Then, we investigate existing studies on load balancing and handover, XR codec optimization in 5G networks, spatial reuse, and channel selection in Wi-Fi and summarize related works in the literature review.

2.1 Background

Reinforcement Learning

Reinforcement learning (RL) is a framework for sequential decision-making where agents learn to take actions in an environment to maximize received rewards. To illustrate, consider an RL algorithm guiding the movements (actions) of a robot (the agent) in a factory (the environment), to succeed in moving boxes from one place to another (the reward). The RL algorithm could govern the motions (actions) of the robot within the world (the environment) to accomplish a specific task (earning a reward). In other words, RL maps observations of an environment to actions, aiming to maximize a numerical quantity that is connected to the rewards received. After this procedure, a policy that maximizes the expected return in a Markov Decision Process MDP is learned by the agent. But, what is an MDP? Let us define it.

In a Markov process, it is assumed that the world consistently occupies one of several possible states. The term "Markov" signifies that the likelihood of being in a particular

state relies solely on the preceding state and is independent of the states preceding it. The transitions between these states are encapsulated by transition probabilities, denoted as $Pr(s_{t+1}|s_t)$, representing the probability of moving from the current state, s_t, to the subsequent state, s_{t+1}, at a given time step, indexed by t. Additionally, we define a Markov reward process. A Markov reward process also includes a distribution $Pr(r_{t+1}|s_t)$ over the possible rewards r_{t+1} received at the next time step, given that we are in state s_t. The reward return is calculated as a discounted future reward G_t at time t as follows:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2.1}$$

where $\gamma \in (0, 1]$ is the discount factor. When the discount factor is less than one, rewards that are closer in time carry a higher value than more distant rewards.

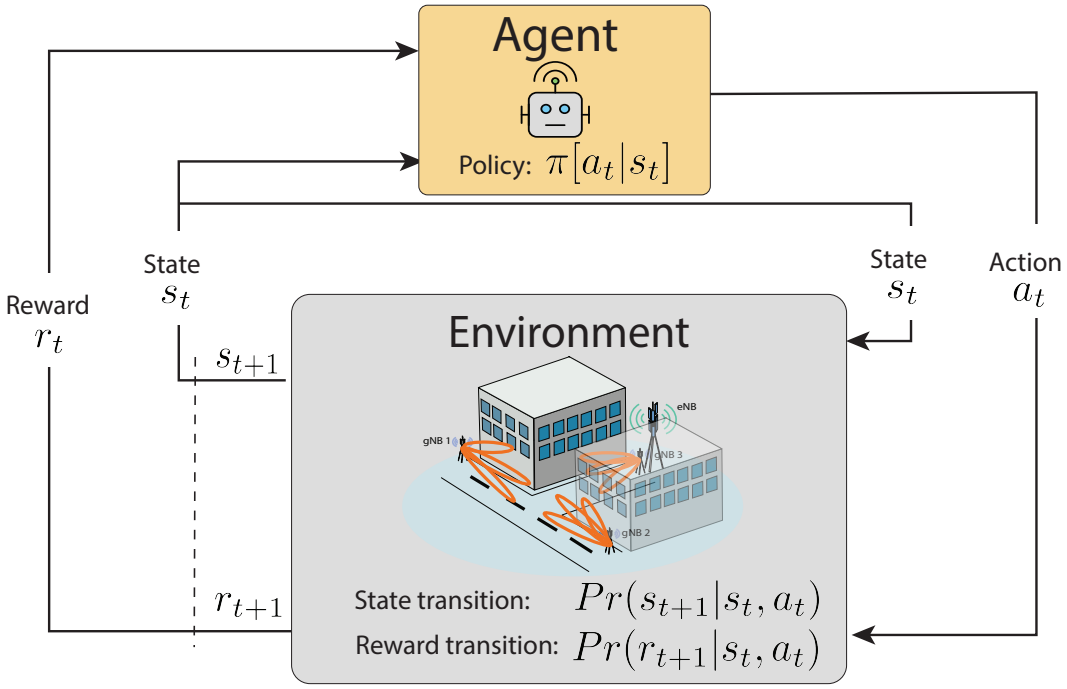

Figure 2.1: Reinforcement learning cycle.

In Figure 2.1 we show the Reinforcement learning cycle representing a Markov Decision Process (MDP). In this figure, a set of possible actions is introduced at each time step.

The action taken at a given time influences the transition probabilities, now expressed as $Pr(s_{t+1}|s_t, a_t)$. Additionally, rewards may be contingent on the action, and this relationship is represented as $Pr(r_{t+1}|s_t, a_t)$. Consequently, an MDP generates a sequential series of states, actions, and rewards defined by the tuple $< S, A, Pr, R >$ where S is the state set, A is the action set, Pr is the transition probability, and R is the reward function, respectively.

Finally, three main challenges are faced by any RL algorithm. The sparse nature of the reward, the balance between exploration and exploitation or exploration-exploitation trade-off, and the offset rewards due to the temporal nature of an MDP [7].

Multi-Armed Bandits

Multi-Armed Bandits (MABs) are a widely used RL approach that addresses the exploration-exploitation trade-off problem. Their implementation is usually simpler when compared with full RL off-policy or on-policy algorithms. It is important to note that there is some overlap between bandit algorithms and RL algorithms. Multi-armed bandit problems can be seen as a simpler case of RL problems. One of the main differences that can be found is related to the exploration-exploitation dilemma. Indeed, both deal with this problem, but bandits typically focus solely on this tradeoff, whereas RL algorithms often deal with more complex environments where the agent can learn from the consequences of its actions. Another important characteristic is the sequential decision-making nature of both algorithms. Bandit algorithms typically rely on independent, one-shot decisions, without taking into account long-term consequences. This is a quite convenient feature in scenarios where learning must be performed rapidly. In RL, the agent's decisions can have long-term consequences and influence the subsequent states and rewards it encounters. Finally, bandit algorithms are simpler complexity-wise, which makes them appealing in scenarios where computational resources are scarce, as considered in our work. However, simplicity often comes with the cost of obtaining suboptimal solutions [48].

The basic model of MABs corresponds to the stochastic bandit, where the agent has K possible actions to choose, called arms, and receives a certain reward R as a consequence of pulling the k^{th} arm over T environment steps [49]. The rewards can be modeled as independent and identically distributed (i.i.d), adversarial, constrained adversary, or random-process rewards [50]. Of the four models previously mentioned, two are more commonly found in the literature: the i.i.d and the adversarial models. In the i.i.d model, each pulled arm's reward is drawn independently from a fixed but unknown distribution D_k with an unknown mean μ_k^*. On the other hand, in the adversarial model, each pulled

arm's reward is randomly sampled from an adversary or alien to the agent (such as the environment) and not necessarily sampled from any distribution [51].

The performance of MABs is measured in terms of cumulative regret $R(T)$ or total expected regret over the T steps. Regret quantifies the missed opportunity in a multi-armed bandit problem, computed as the difference between the expected reward attainable by an oracle that selects the optimal arm at each time step and the actual reward obtained by a given policy. Thus, the regret for an Multi-Armed Bandit (MAB) can be formally defined as:

$$R(T) = \mu^* T - \sum_{k>1}^{K} \mu_k \mathbb{E}[n_k(T)], \tag{2.2}$$

where μ_k is the mean of the D_k random reward distribution, $\mu^* = \max\{\mu_1, \ldots, \mu_K\}$, and $n_k(t)$ is the number of times the k^{th} arm has been chosen at time t. The utmost goal of the agent is to minimize $R(T)$ over the T steps, such that $\lim_{T\to\infty} R(T)/T = 0$, which means the agent will identify the action with the highest reward in such a limit.

ϵ-greedy, Upper-Confidence-Bound, and Thompson Sampling MAB

The ϵ-greedy MAB is one of the simplest MABs, and as the name suggests, it is based on the ϵ-greedy policy. In this method, the agent selects greedily the best arm most of the time, and once in a while, with a predefined small probability (ϵ), it selects a random arm [7].

The UCB MAB tackles some of the disadvantages of the ϵ-greedy policy at the moment of selecting non-greedy arms. Instead of drawing randomly, the UCB policy measures how promising non-greedy arms are close to optimal. In addition, it takes into consideration the rewards' uncertainty in the selection process. The selected arm is obtained by drawing the action from $\texttt{argmax}_a \left[Q_t(a) + c\sqrt{\ln t / N_t(a)} \right]$, where $N_t(a)$ corresponds to the number of times that action a via the k^{th} arm has been chosen, and $Q_t(a)$ is the Q-value of action a [7, 52].

Finally, Thompson Sampling MAB action selection is based on the Thompson Sampling algorithm. Thompson sampling or posterior sampling is a Bayesian algorithm that constantly constructs and updates the distribution of the observed rewards given a previously selected action. This allows the MAB to select arms based on the probability of how

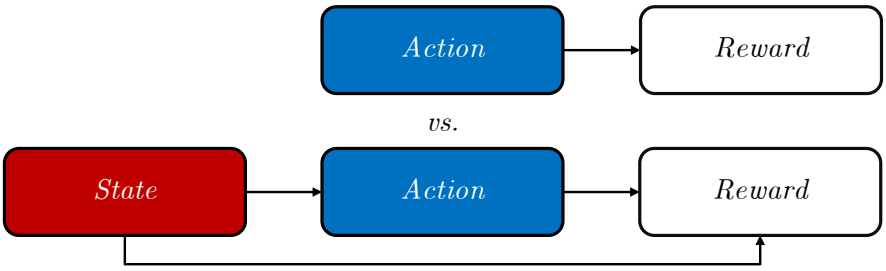

Figure 2.2: MAB vs. contextual MAB

optimal the chosen arm is. The parameters of the distribution are updated depending on the selection of the distribution class [53].

Deep Contextual Multi-Armed Bandits

Contextual Multi-Armed Bandit (CMAB) is a variant of MABs that, before selecting an arm, observes a series of features commonly named context [48]. Fig. 2.2 depicts the difference between the stateless MAB and CMAB. Different from the stateless MAB, a CMAB is expected to relate the observed context with the feedback or reward gathered from the environment in T episodes and consequently predict the best arm given the received features [51]. In other words, the context plays a crucial role in contextual bandits because it provides valuable information that can help the algorithm make more informed decisions and ultimately lead to better rewards.

Deep Reinforcement Learning

In this section, we introduce Tabular Q-learning which is the basic method of the Q-learning family in RL. It belongs to the family of Tabular RL methods which do not require any function approximators such as neural networks. Two main classifications can be found for Tabular RL algorithms: model-based and model-free. In model-based algorithms, the MDP structure is known and it can be exploited using classical dynamic programming. On the other hand, model-free algorithms are classified into two groups: value estimation and policy estimation. The first estimates the optimal state-action function and then selects the action based on the highest value of the set of available actions. The second estimates the optimal policy by using gradient descent which can be used to select the action given the current state. The RL taxonomy does not stop there and within each group, three methods subgroups are found: Monte Carlo, Temporal difference (TD), and Dynamic

Programming (DP). Monte Carlo methods simulate several trajectories over the MDP for a given policy and improve it. However, in most practical scenarios this is unfeasible and the policy needs to be updated while the agent transverses the MDP. DP methods consider complete knowledge of the transition and reward structure, which is the most noticeable difference between other methods that need to receive indirect values of the transition probability and reward. Finally, TD combines elements of both dynamic programming and Monte Carlo methods. TD methods update value estimates based on observed transitions between states, without requiring a model of the environment dynamics, as in dynamic programming. The key idea behind TD methods is to estimate the value function by bootstrapping from the current estimate, incorporating both the observed reward and the estimated value of the next state. This allows for online learning, meaning the agent can update its value estimates after each time step, using the new experience gained.

Additionally, RL methods can be classified into two broad groups according to the way they learn: on-policy and off-policy. For instance, on-policy methods focus on improving the current strategy, learning from the agent's own experiences while off-policy methods allow for learning from a broader range of data, potentially making better use of past experiences generated by different policies. Among on-policy methods, SARSA (State-Action-Reward-State-Action) is one of the most famous algorithms and it is characterized by the following update rule of its Q-function:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r(s_t, a_t) + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)), \tag{2.3}$$

where α is the learning rate. In Eq. 2.3 can be seen the Temporal Difference (TD) that tries to capture the error between the estimated q-value and the future estimate in $r(s_t, a_t) + \gamma \cdot Q(s_{t+1}, a_{t+1})$. On the other hand, Q-Learning is a model-free reinforcement learning algorithm used to find the optimal action-selection policy for a given finite Markov decision process. It involves estimating the quality of different actions in a given state and updating these estimates iteratively through exploration and exploitation. Tabular Q-learning proposes an update rule as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r(s_t, a_t) + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t)). \tag{2.4}$$

Despite the success of Q-learning, it faced some challenges when learning optimal policies in environments with large and complex state spaces [54]. Thus, the family of deep RL was proposed to tackle some of the previously mentioned challenges and we intend to introduce them in the next subsection.

Note that, a large number of algorithms exist in the literature that fit in each of the classifications mentioned above. However, the description of each of them and the techniques they use are considered out of the scope of this thesis.

2.1.1 Clipped Deep Double Q-learning

Clipped Deep Double Q-learning (CDQL) builds upon the successful algorithm Deep Q-learning [54]. To better understand CDQL, we introduce some preliminaries of Deep Q-Network (DQN). Deep Q-Learning (DQN) is a powerful reinforcement learning algorithm that combines Q-Learning, a traditional reinforcement learning technique, with deep neural networks.

DQN incorporates deep neural networks, typically convolutional neural networks (CNNs), to approximate the Q-function. The neural network takes the environment state as input and outputs Q-values for each possible action.

$$Q(s_t, a_t; \theta) \approx Q^*(s_t, a_t), \tag{2.5}$$

where θ corresponds to the main neural network that parameterizes the Q-function. To improve stability and sample efficiency, DQN uses experience replay. It stores and randomly samples experiences from a replay buffer, breaking the temporal correlation of consecutive experiences. The experience replay buffer stores experiences (s, a, r, s') and samples a random batch during training. The update rule becomes:

$$Q(s_t, a_t; \theta) \leftarrow (1 - \alpha) \cdot Q(s_t, a_t; \theta) + \alpha \cdot \Big(r_t + \gamma \cdot \max_a Q(s_{t+1}, a_t; \theta^-) \Big) \tag{2.6}$$

This helps in mitigating the risk of overfitting to recent experiences. DQN employs a target Q-network to stabilize the learning process. The target network is a delayed copy of the online Q-network, and its parameters are periodically updated via Polyak averaging. This helps in reducing the correlation between the target and current Q-values.

$$\theta^- \leftarrow \tau \cdot \theta + (1 - \tau) \cdot \theta^-, \tag{2.7}$$

where θ^- corresponds to the target neural network.

Finally, to control the scale of rewards and improve learning stability, DQN often incorporates reward clipping. This involves constraining the magnitude of the rewards and preventing excessively large gradients during training.

$$r_t^{'} = \text{clip}(r_t, r_{\min}, r_{\max}) \tag{2.8}$$

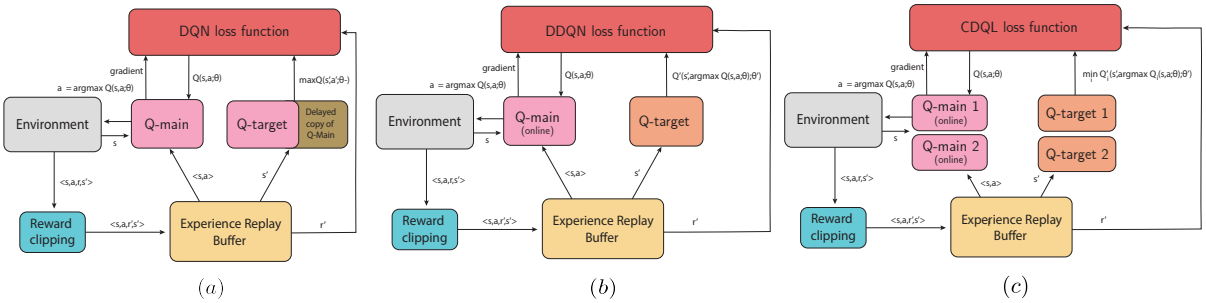

Figure 2.3: (a) Deep Q-learning, (b) Double Deep Q-learning and (c) Clipped Double Q-Learning schemes

Figure 2.3(a) illustrates a summary of the Deep Q-Network (DQN). In the current state s, the primary network initially chooses the action a, which is subsequently executed in the environment. The agent then obtains a new state s and receives a reward r from the environment. The tuple $< s, a, r, s' >$ is saved in the experience replay buffer, which will be used for the network training as shown by equation (2.6). Finally, a delayed copy of the main Q-function will be used as the target Q-function.

Double Deep Q-Learning (DDQN) [55] in figure 2.3(b) is an extension of Deep Q-Learning (DQN) that addresses certain issues related to overestimation of Q-values, which can occur in standard DQN due the use of a delayed copy of its Q-function in the evaluation process. One of the changes corresponds to having an independent target Q-function, different from the delayed target Q-function of DQN. Then, the update rule changes as:

$$Q(s_t, a_t; \theta) \leftarrow (1-\alpha) \cdot Q(s_t, a_t; \theta) + \alpha \cdot \left(r_t + \gamma \cdot Q\left(s_{t+1}, \arg\max_{a'} Q(s_{t+1}, a'; \theta_{\text{online}}); \theta_{\text{target}} \right) \right), \tag{2.9}$$

where θ_{online} are the parameters of the online Q-network, θ_{target} are the parameters of the target Q-network, $\arg\max_{a'} Q(s_{t+1}, a'; \theta_{\text{online}})$ represents the action that maximizes the Q-value for the next state according to the online Q-network.

Finally, we can present the algorithm Clipped Double Q-Learning [56]. As observed in Figure 2.3(c), this algorithm may be seen as a continuation of the work of [55] with DDQN. It was introduced as part of TD3 (Twin Delayed Deep Deterministic Policy Gradient Algorithm), which builds on the Deep Deterministic Policy Gradient algorithm (DDPG) [57]. The aforementioned algorithms suffered from Q-value overestimation due to the usage of the $argmax$ operator for selecting the maximum Q-value. CDQL tackles overestimation issues by following the strategy of having two neural networks that learn at the same time

meanwhile the reward is calculated based on the minimum Q-value of such networks hence, reducing overestimation.

2.1.2 Hierarchical Reinforcement Learning

Typically, an RL agent optimizes its action based on a customized reward or objective function. The design of a reward function will aim to maximize or minimize certain metrics of interest. In addition, a reward function could also be used in scenarios where our agent's goal is known. However, in goal-directed problems as in the majority of RL problems, sparse rewards are a significant challenge that affects the learning of robust value functions and thus, optimal action selection. Hierarchical reinforcement learning (hRL) [58] helps to solve the aforementioned problem by splitting the value function into two levels: a meta-controller and a controller. Given the simpler scenario where the value function has been divided into two levels, the agent will be able to act over two timescales which in some applications is needed given the sparcity of rewards.

In hierarchical reinforcement learning, the MDP is redefined as $< S, A, Pr, R, \mathcal{G} >$, where $\mathcal{G}$ represents the set of goals. Given the current state $s \in S$, the meta-controller generates high-level goals $g \in c$ for the controllers. As a result, the controller selects low-level actions $a \in A$ based on high-level policies and receives an intrinsic reward r_{in}. Finally, the meta-controller obtains an extrinsic reward r_{ex} from the environment and chooses new goals g' for the controller. The extrinsic reward is typically computed as the sum of all inner steps of the controller's rewards.

The concept behind involves incorporating a hierarchical architecture into RL. Specifically, the meta-controller formulates high-level policies to guide the controller's selection of low-level actions. Compared to traditional RL, hRL is acknowledged as a more efficient learning method, leveraging its hierarchical structure for improved task management through the division of sub-goals.

2.1.3 Multi-Agent Systems and Team Learning in Reinforcement Learning

Multi-Agent Systems (MAS) is composed of a group of autonomous and intelligent agents that act in an environment to accomplish a common goal or individual goals. In the following, we introduce different types of MAS, team learning in MAS, and finally how MAS with team learning can be employed in O-RAN architecture.

A recent survey on open virtualized networks [59], gives a comprehensive overview of the state-of-the-art in modern RANs. Following, we introduce the existent types of MAS.

Types of MAS

There can be several types of MAS, such as homogeneous/heterogeneous, communicating/non-communicating, and cooperative(collaborative)/competitive.

Homogeneous/Heterogeneous: The homogeneous multi-agent system consists of agents with a similar internal structure, which means that all agents have the same local goals, capabilities, actions, and inference models. In homogeneous architecture, the main difference among agents is based on the place where their actions are applied over the environment. By contrast, in the heterogeneous MAS, agents may have different goals, capabilities, actions, and inference models. From the O-RAN perspective, homogeneous agents would be instances of the same xApp instantiated for different slices, while different xApps would be heterogeneous agents.

Communicating/non-communicating: A group of agents can be designed to communicate with each other or not. When there is no communication, agents act independently without receiving any feedback from other agents. However, since they are working in the same environment, indirect feedback on the actions of the other agents will be observed by the individual agent. In the case of communicating agents, there is explicit feedback among agents. This is more suitable for many O-RAN ML algorithms. Note that, the way agents communicate will impact the required bandwidth on the O-RAN interfaces. It is also important to note that in a multi-vendor environment, some agents controlling different vendor functions might not communicate with each other.

Cooperative/Competitive: In a MAS, agents can be cooperative, competitive, or mixed. In a cooperative setting, the agents need to take collaborative actions to achieve a shared goal, or in other words, agents must jointly optimize a single reward signal. On the other hand, in a competitive setting, each agent tries to maximize its received reward under the worst-case assumption; meanwhile, the other agents always try to minimize the reward of others. It is also possible to have groups with mixed behavior agents where some are cooperative while others are competitive. In the O-RAN architecture, this kind of behavioral differences may be needed for various use cases, where certain xApps might be cooperating while others are competing due to the nature of shared resources. Although many machine learning techniques have been considered for MAS, team learning could have particular significance in the future since it can have applications in O-RAN by organizing

xApps as teams of teams in hierarchies. In the chapter 4, we focus on MATL and its potential use in O-RAN.

Background on Disaggregated, Virtualized RAN and O-RAN

Traditional RAN solutions offer an architecture where BBUs and RUs are co-located. This brought limitations in terms of not being able to pool BBU resources. Therefore the following generation of RAN architectures considered BBU resources that are pooled close to the radios but not co-located, where geographical proximity is necessary due to latency limitations. The pool of BBUs is called a Distributed Unit (DU), and the radios constitute the Radio Unit (RU). Within O-RAN specifications, another level of processors is also defined, which is called as O-RAN Central Unit (O-CU) [1].

The most appealing reason behind RAN disaggregation was to reduce costs and bring more versatility to the technological market. An earlier version of RAN disaggregation was seen in C-RAN where some hardware functions are implemented as software functions, and BBU functionality is collected at the centralized cloud. C-RAN offers improvements in network capacity, handling cooperative processing, mobility, coverage, energy-efficient network operation, and reduced deployment costs [60].

On the evolution path of RAN architectures, the most recent development comes with O-RAN, in which interfaces between O-RU and O-DU and between O-DU and O-CU are based on open specifications [61]. This paves the way for interoperability between vendor products and the possibility of selecting the best set of products by Mobile Network Operators (MNOs). In addition, O-RAN embraces intelligence in every layer of its architecture and aims to leverage new machine learning-based technologies [62]. As seen in Fig. 2.4, the RAN Intelligent Controller (RIC) is composed of the non-Real Time RIC (non RT-RIC) and the near-Real Time RIC (near RT-RIC), which have been considered as fundamental platforms to enable intelligence [63]. The primary objective of the non RT-RIC is to support applications that operate in non-real-time or above the one-second time frame. In the non-RT RIC, we employ the rApps that aim at non-real-time parameter optimization, RAN analytics, and management, as well as model training for the lower layer of the RIC. On the other hand, the near RT-RIC supports applications that are executed in near-real-time, such as between 10ms and 1ms. The near-RT RIC has two interfaces with a centralized unit user plane (O-CU-UP) and control plane (O-CU-CP) used for data transmission and signaling, and configuration, respectively. In the case of near RT-RIC,

[1] In O-RAN specs, the radio unit is called O-RU, and the distributed computational unit is called (O-DU).

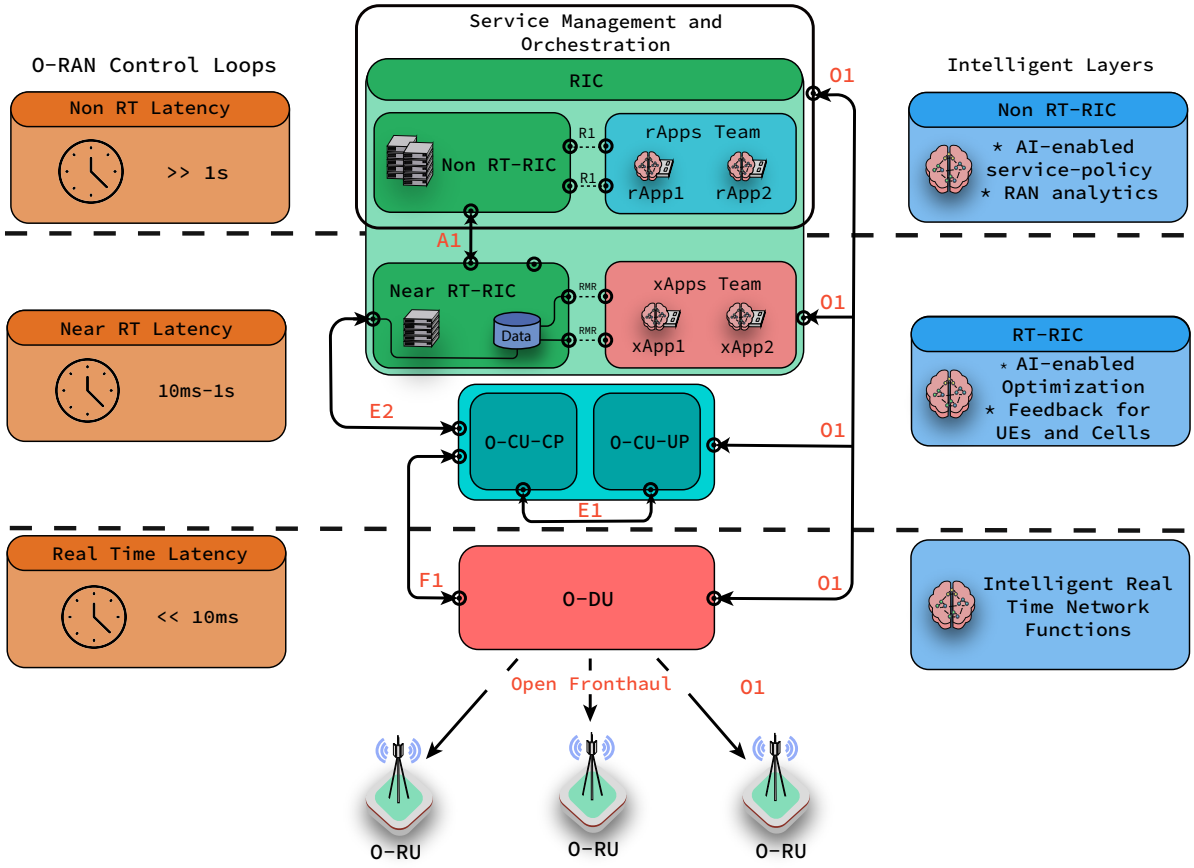

Figure 2.4: O-RAN interfaces, closed-loop latency requirements, and intelligence at different layers of O-RAN. O1 corresponds to the interface between the SMO and O-RAN managed elements. A1 interfaces the SMO and RAN. E2 interfaces the control and optimization of the O-CU and O-DU nodes (E2 nodes) resources by the Near RT-RIC. F1 interfaces the O-DU and O-CU. E1 interface interconnects O-CU-CP and O-CU-UP. The Open Fronthaul refers to the interface that connects the O-RU with the O-DU. R1 interfaces rApps and the Non-RT RIC. Finally, xApps are capable of accessing to E2 nodes' real-time information and general-purpose data in a central database using a service bus implemented by a peer-to-peer low latency ($\sim$0.02 ms) library named RIC Message Router (RMR) [4].

xApps enable intelligence by enforcing policies from the non RT-RIC and are capable but not restricted to executing Radio Resource Management (RRM) tasks such as cell load prediction, anomaly detection, traffic steering and data collection from E2 Nodes (O-DU, O-CU and O-RU). Access from the near RT-RIC and non RT-RIC to xApps and rApps is provided through Application Programming Interfaces (APIs) with software updates and data gathering purposes. The introduction of xApps and rApps increases the interoperability and the openness of the O-RAN architecture by allowing developers from third-party software providers to share, publish, and provide access to their applications. The figure above illustrates that the O-DU is connected to the O-CU and uses O-RUs to provide services to UEs.

The Role of Intelligence in O-RAN

O-RAN defines certain guidelines to employ AI in its architecture. Offline and online learning are expected to coexist with a modular design as best practice to follow. This will enable service providers to decide the location of intelligence in the network functions according to their best interests. As a recommendation, xApps and rApps are expected to fall in certain control loops [64, 65] according to the time budget needed for such applications (see Figure. 2.4). Furthermore, open-source solutions, such as Acumos AI, emerge as a potential development platform for ML models. Several AI/ML use cases are already identified by the O-RAN community, such as QoE (Quality of Experience) optimization, traffic steering [66], user access control [67] and V2X handover management [62]. It is important to highlight that although some ML applications are described in O-RAN reports, the huge potential of applying MATL in O-RAN is not covered in any of the use cases. On the other hand, MAS has been studied in the context of wireless networks in a few studies. Most of these prior works consider multiple uncoordinated agents working within the same environment, driven by the same reward or goal. However, when multiple agents interact with the environment independently, they change the environment of each other. Additionally, when agents have different goals, the problem cannot be simplified to deploying independent agents. Therefore, using teams of agents in O-RAN emerges as a promising approach.

In section IV, we present a case study where a team of two xApps exchange information to enhance performance and avoid conflict among team members. Note that, the organization of xApps in teams will incur communication overhead. Before we elaborate more on such opportunities and challenges, in the following section, we provide background on MAS and team learning.

2.1.4 Multi-agent Reinforcement Learning: MADDPG, VDN and QMIX

In this subsection, we introduce the MARL technique, which is an important variant of RL. Compared with RL, the main difference of MARL is that multiple agents are involved simultaneously.

Several MARL algorithms can be found throughout the literature. According to [68] MARL can be classified either as Independent Learning (IL), Centralized-Training Decentralized Execution (CTDE), or Centralized Learning (CL) depending on how action selection and training are performed among agents. In some challenges, IL has shown good performance such as in [69], however, instability is typically observed due to the concurrent exploration/training of the agents [47]. On the other side, CL assumes complete access to the agents' information and hence, it becomes a single-agent problem. However, CL approaches are not realistic and can present several issues in terms of increasing action and state space. Finally, CTDE approaches allow for have middle ground between the two previously described techniques with an independent action selection and a centralized training step.

Multi-agent Deep Deterministic Policy Gradient (MADDPG)

In this work, we use a state-of-the-art policy gradient algorithm. This algorithm is presented in [47] and builds on the generalization for the multi-agent domain of Deep Deterministic Policy Gradient algorithm (DDPG) [57]. MADDPG is a multi-agent off-policy and continuous action space algorithm constrained to three main assumptions: (1) the learned policy per agent uses local observations, (2) the environment is non-deterministic, thus a differentiable model is not assumed and (3) a communication structure is not assumed among agents. Furthermore, this algorithm belongs to the centralized training with a decentralized execution family.

Value Decomposition Networks (VDN)

Value Decomposition Networks (VDN) is proposed in [70] and can be classified as a fully cooperative CTDE algorithm that lies between Independent Q-Learning and Centralized Q-Learning. VDN builds upon the assumption that the multiagent system joint action-

value function can be decomposed into individual agent's value functions as:

$$Q_{tot}(\{\boldsymbol{h}^1, ..., \boldsymbol{h}^n\}, \{\boldsymbol{a}^1, ..., \boldsymbol{a}^n\}) \approx \sum_{n=1}^{|\mathcal{N}|} \bar{Q}^n(h^n, a^n), \tag{2.10}$$

where $\bar{Q}^n$ corresponds to the Q-function of the n^{th} agent, $n \in \mathcal{N}$ where each agent's policy only depends on its individual observation history. In addition, $\{\boldsymbol{h}^1, ..., \boldsymbol{h}^n\}$ and $\{\boldsymbol{a}^1, ..., \boldsymbol{a}^n\}$ corresponds to the tuple of histories and actions of the n^{th} agent, respectively. As mentioned previously, VDN assumes full cooperation among agents which allows the definition of a team reward. Thus, the team reward can be decomposed as:

$$r_{tot} = \sum_{n=1}^{|\mathcal{N}|} r^n(h^n, a^n) \tag{2.11}$$

Then, from the perspective of one agent (say $n = 1$), the additive joint Q-function can be defined as:

$$\begin{aligned} Q_{tot}(\boldsymbol{s}, \boldsymbol{a}) &= \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r_{tot}(\boldsymbol{s}_t, \boldsymbol{a}_t) | \boldsymbol{s}_1 = \boldsymbol{s}, \boldsymbol{a}_1 = \boldsymbol{a}] \\ &= \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r^1(h_t^1, a_t^1) | \boldsymbol{s}_1 = \boldsymbol{s}, \boldsymbol{a}_1 = \boldsymbol{a}] + \\ &\sum_{n=2}^{|\mathcal{N}|} \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r^n(h_t^n, a_t^n) | \boldsymbol{s}_1 = \boldsymbol{s}, \boldsymbol{a}_1 = \boldsymbol{a}] \\ &=: \tilde{Q}^1(\boldsymbol{s}, \boldsymbol{a}) + \sum_{n=2}^{|\mathcal{N}|} \tilde{Q}^n(\boldsymbol{s}, \boldsymbol{a}) \\ &\approx \bar{Q}^1(h^1, a^1) + \sum_{n=2}^{|\mathcal{N}|} \bar{Q}^n(h^n, a^n), \end{aligned} \tag{2.12}$$

where γ corresponds to the discount factor, $\tilde{Q}^n(\boldsymbol{s}, \boldsymbol{a})$ the Q-function dependable on the state $\boldsymbol{s}$ and action $\boldsymbol{a}$. Notice that the term $\bar{Q}^n(h^n, a^n)$ is used to differentiate a Q-function that depends on h from the previous term that depends on s. Finally, it can be seen that the last equation on (2.12) corresponds to (2.10).

QMIX: Monotonic Value Function Factorisation

Despite VDN having proven to be quite successful in many RL tasks and some recently in wireless access networks [45], QMIX has demonstrated its capacity to propose richer action-value functions without full factorization of decentralized policies.

Optimistic Weighted QMIX

In [71], the authors introduce two distinct weighting schemes for the QMIX algorithm. QMIX, much like VDN, operates as a fully cooperative algorithm, involving a factorization of the value function among the participating agents. Instead of employing an additive approach for mixing strategies, QMIX leverages hypernetworks to enforce a monotonic behavior in its Q-function. However, a limitation of QMIX lies in its presumption of equal importance across actions, which can result in suboptimal policy outcomes. To address this shortcoming, the authors present the Optimistic Weighted QMIX, which allows for the assignment of individual weights to each Q-function, thereby facilitating better collective action decisions. The weighting parameter $w : S \times \boldsymbol{U} \rightarrow (0, 1]$ is defined as follows:

$$w(s, \boldsymbol{u}) = \begin{cases} 1 & Q_{tot}(\boldsymbol{\tau}, s, \boldsymbol{u}; \theta') < y_i, \\ \alpha & otherwise, \end{cases} \tag{2.13}$$

where $y_i = r_i + \gamma Q_{tot}(s, o; \theta)$. Additionally, $\boldsymbol{\tau}$, $\boldsymbol{u}$, s, o, θ, θ' and $\alpha \in (0, 1]$ corresponds to the action-observation history, the agents' joint action, the observation state, the single agent's action, the target policy, the evaluation policy, and a predefined weight, respectively. As observed in equation 2.13 the contribution of actions considered suboptimal is reduced.

2.1.5 Meta-Learning, Sequential and Parallel Transfer Reinforcement Learning

In this section, we present an overview of Meta-learning, Sequential, and Parallel Transfer Reinforcement Learning. Most RL methods applied in Wireless Networks do not consider the necessity of accelerating learning. Among the diverse techniques that allow improving the learning rate of common RL algorithms, we can find Meta-Learning and Transfer Learning (TL).

Meta-Learning

Spatial Reuse adaptability in dynamic scenarios is an open challenge. One of the possible solutions to this problem is to employ Meta-Reinforcement Learning (meta-RL) methods. However, convergence of the learning process remains a challenge. Ideally, the exploration performed by RL agents should be completed promptly to further exploit the best action. However, a premature exploration could lead to a policy that ends up at a local minima or to the agent's total failure [72]. Meta-learning has been conceptualized as "learning to learn" to tackle the learning convergence challenge with just a few learning instances [73]. Meta-learning has been previously applied in the wireless communications domain. In [74], the authors propose a meta-learning and recurrent neural network (RNN) approach to predict mmWave/THz link blockages. In the Wi-Fi context, the authors in [75] propose using meta-learning to adapt quickly in the presence of new types of impersonation attacks.

To the best of our knowledge, we proposed in this thesis the first study that addresses adaptability via meta-learning and contextual multi-armed bandits in the 802.11ax SR context. More specifically, a Deep Contextual MAB based proposed in [76], namely Sample Average Uncertainty (SAU)-Sampling is leveraged to apply a Model Agnostic Meta-Learning (MAML) algorithm [77]. Figure 5.7 illustrates a high-level overview of the meta-learning strategy in Open-WiFi settings [78]. In the proposed MA-CMAB scheme, each Contextual Multi-Armed Bandit (CMAB) agent experiences a set of $\mathcal{T}_N$ scenarios that experience mobility-induced fluctuation of the user demand. The acquired knowledge is stored in the Open-WiFi controller for posterior meta-training in an offline fashion. Finally, the meta-agent named "meta-bandit" is used to further few-shot adaptation in the unseen $\mathcal{T}_{N+1}$ scenario. The main goal of this work is to propose a method capable of accelerating the learning convergence of SR-based RL in highly dense Wi-Fi networks while maintaining comparable Key Performance Indicators (KPIs) with state-of-the-art methods such as transfer learning. Moreover, our results show that the Meta-Bandit exhibits at convergence time a considerable improvement in terms of network fairness when compared with the presented baseline.

Sequential Transfer Learning

Transfer learning or knowledge transfer techniques improve learning time efficiency by utilizing prior knowledge. Typically, this is done by extracting the knowledge from one or more source tasks and then applying such knowledge in a target task [79]. If the tasks are related in nature and the target task benefits positively from the acquired knowledge from the source, then it is called inductive transfer learning [80]. This type of learning is

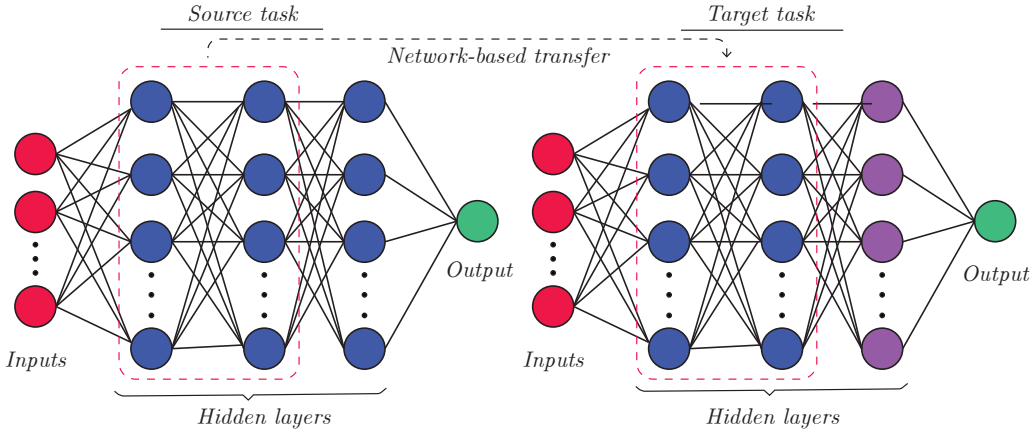

Figure 2.5: Network-based transfer learning: the neural network source task's hidden layers are reutilized in the target network

not uncommon and it is used by the human brain daily. However, a phenomenon called negative transfer can occur if, after knowledge transfer, the target task performance is negatively affected [81].

In the realm of transfer learning, we can find Deep Transfer Learning (DTL). DTL is a subset of transfer learning that studies how to utilize knowledge in deep neural networks. In the context of classification/prediction tasks, a large amount of data is required to properly train the model of interest [82]. In many practical applications where training time is essential to respond to new domains [83], retraining using a large amount of data is not always feasible and possibly catastrophic in terms of performance. "What to transfer" corresponds to one of the main research topics in transfer learning. Specifically, in the case of deep transfer learning, four categories have been broadly identified: instance-based transfer, where data instances from a source task are utilized; mapping-based transfer, where a mapping of two tasks is used on a new target task; network-based transfer, where the network pre-trained model is transferred to the target task; and adversarial-based transfer, where an adversarial model is employed to find which features from diverse source tasks can be transferred to the target task [84].

In this work, we utilize the DTL form called network-based transfer learning to adapt efficiently TP and CCA parameters in dynamic scenarios. An example of network-based transfer learning technique is presented in Fig. 2.5. Such a technique is utilized in deep transfer reinforcement learning as part of a transfer learning type called policy transfer [85]. In particular, policy transfer takes a set of source policies $\pi_{S_1}, ..., \pi_{S_K}$ that is trained on a set of source tasks and uses them in a target policy π_T in a way that can leverage the

former knowledge from the source policies to learn its own. More specifically, the weights and biases that comprise each of the hidden layers of the source policies are the elements transferred to the target policies. Note that in practice policies are modeled as neural networks. Furthermore, we take advantage of the design of a contextual multi-armed bandit presented in [76] and apply policy transfer to improve the agent's SR adaptability in dynamic environments. The results and observations of applying DTRL are discussed in section 5.1.5.

Parallel Transfer Learning

As presented in the previous subsection, Transfer Reinforcement Learning is a technique employed to accelerate convergence and improve learning in RL. The idea comes from human psychology where learned actions or concepts can be utilized to speed up the acquisition of new knowledge [86]. In the context of multi-agent RL, transfer learning typically happens in a sequential and unidirectional flow, from agents defined as source or teacher agents towards agents with some or minimal knowledge named target or student agents. Evidently, in some scenarios, the assumption of the existence of a source-target-agent relationship is not known; in other words, there is no hint of which agent is the source or the target. To address the previously mentioned issue, Parallel Transfer Reinforcement Learning (PTRL) in intra-multi-agent systems is introduced in [87] where source and target agents run concurrently. Thus, knowledge transfer occurs in a parallel and bidirectional way. Consequently, it removes the need to have a set of previously trained source agents, as required by sequential RL.

2.2 Literature Review

In this section, we provide an overview of this thesis-related work. Subsection 2.2.1 presents background information on Load Balancing (LB) and Handover ML-based on 4G and 5G Networks. Subsection 2.2.2 introduces team learning in the O-RAN architecture. Additionally, subsections 2.2.3 to 2.2.5 introduce some works on Spatial Reuse and Channel Selection ML-methods, and finally, subsection 2.2.6 depicts existent research on XR codec optimization in 5G networks.

2.2.1 ML-based Load Balancing and Handover

Several works in the literature relate to load balancing and handover in RANs due to their significant impact on saving resources and maximizing network performance. In most cases, the latter problem is tackled by modifying the existing handover (HO) strategies either by changing some HO parameters or by constantly tracking the network Key Performance Indicators (KPIs) [88, 89].

Recently, machine learning has been used for load balancing as well. In [90], the authors proposed a 5G handover algorithm based on Q-learning for optimizing the handover mechanism by the selection of data link beams and access beams in 5G cellular networks. In [91] the authors presented a supervised learning solution based on deep learning by considering the variation of the signal-to-interference-plus-noise ratio (SINR) to calculate the probability of Radio Link Failure (RLF) and based on this metric, they performed handover to the cell that is less prone to experience RLF. In [92] the authors presented an RL-based mobility load balancing (MLB) algorithm addressed to deal with ultra-dense networks. The proposal consisted of a two-layer architecture with the first layer in charge of building small clusters and the second layer where in each intra-cluster, an MLB algorithm is executed to obtain the optimal HO parameters by minimizing the resource block utilization. In [93] the authors presented an RL-based load balancing algorithm for LTE where maximizing the instantaneous throughput of the overall network is the main objective.

In summary, previous works do not consider end-to-end delay and channel quality in the optimization process of HO parameters. Different than other research, we consider additional QoS metrics alongside the resource block utilization seeking to balance both while maximizing the network performance.

In the realm of multi-agent systems, to the best of our knowledge, there is no previous work that has dealt with load balancing in RAN from the competitive multi-agent off-policy learning perspective. However, multi-agent settings have been studied in other load-balancing approaches. Additionally, several single and centralized RL load balancing studies can be found in the literature where HO parameters are modified based on network KPIs. These studies are summarized below.

In [94], the authors describe a message-passing-based cooperative multi-agent Q-learning algorithm to obtain the optimal bias offset in dense HetNets (Heterogeneous Networks). In [95], the authors study load balancing in a wired network topology to evenly distribute traffic among the nodes without incurring congestion. To do so, the authors propose a multi-agent actor-critic to address traffic optimization. Additionally, in [96], the authors propose an RL-load balancing algorithm based on the exploration of the BS's transmission

power and the CIO parameter. In this work, the authors use Double Deep Q-Learning to maximize the reward function based on throughput. Finally, in [25], we have performed load balancing by considering resource block utilization, delay, and metrics in its reward objective function. This work is different than our previous work in terms of the consideration of a multi-agent domain with an adaptive and continuous reinforcement learning scheme and a new formulation of the Markov Decision Process (MDP).

Concerning the AI-based dual connectivity handover proposal we list some of the related works. We must note, that to the best of our knowledge, there was no previous work where hRL has been utilized to optimize handover key parameters such as TTT and SINR cell outage in a 5G dual connectivity scenario. The aforementioned parameters are defined in detail in section V. However, dual connectivity (DC) has been studied thoroughly in the literature. These studies are summarized below. In [97], the authors give an overview of the 4G LTE-NR DC based on the new specifications given by 3GPP. A study case showed that DC is capable of providing coverage and capacity improvements when compared to standalone LTE-NR deployed individually. More recently, in [98] the authors perform an analysis of the dual connectivity proposal in the 3GPP release 15 [99]. In addition to the existent specification, the authors remarked on the possibility of including a Secondary Cell controller to manage the multi-radio DC signaling. Furthermore, in [100], the authors present a survey on recent developments of 4G/5G DC, as well as 4G/5G internetworking performance and future research challenges and open issues.

Besides, the body of works in DC, the concept of hRL has not been explored much in the the wireless community. In [101], the authors study outage avoidance in a two-hop cooperative relay network by utilizing hRL in optimizing relay selection and both source and relay transmission power. In [102], the authors propose a hierarchical deep Q-network to perform dynamic multi-channel spectrum sensing in cognitive networks. In this work, the actions correspond to the selection of a channel and the reward to the channel status (busy/idle). Finally, in [103] the authors leverage a hierarchical architecture using Deep Deterministic Policy Gradient for the optimization of the spectrum slice and computing/storing resource allocation in a MEC-based vehicular network. In the following sections, we will introduce the dual connectivity architecture utilized in this work and explain how RL emerges as an effective solution in the optimization of the handover in DC scenarios. Additionally, we foresee the advantage of embedding context-awareness in the DC architecture.

2.2.2 ML-based Multi-agent Team Learning

As shown in Fig. 2.6, in team learning, teams of distributed agents cooperate to achieve a common goal. They usually share complete information on observations. More specifically, team learning tries to maximize a team's objective function comprised of its agents' contributions. Consequently, this learning approach could be useful in O-RAN architecture where a team of xApps (scheduler, power adaptation, beamforming, etc.) needs to find the best arrangement for their members to improve the overall network performance. Such tasks could be comprised of several individual subtasks corresponding to each team member. Hence, team learning shows itself as a promising approach for the decentralized and granular architecture of O-RAN. Although the application of team learning to O-RAN is new, team learning has been studied and applied for a wide range of applications. For instance, team learning has been used to address complex problems; from multi-robot teams (e.g., robot soccer), multi-player games (e.g., Dota, StarCraft), predator-prey pursuit and capture problems, to search and rescue missions, as well as mitigating traffic congestion.

For instance, [104] presents an interesting work related to team learning in urban network traffic congestion. The authors proposed a Dynamic Traffic Assignment (DTA) algorithm based on a collaborative, decentralized heterogeneous reinforcement learning approach to mitigate the effects of the randomness of urban traffic scenarios. To this end, two-agent teams are defined as advisers and deciders. The decider team task consists of assigning traffic flows to a specific traffic network. Meanwhile, the adviser team is responsible for supporting the deciders with additional information on the environment.

In particular, the adviser-decider type of team learning can be used in O-RAN considering the different time granularities of non-real-time RIC (non-RT RIC) and near-Real-Time RIC (near-RT RIC). For instance, a team of agents at non-RT RIC can advise another team at near-RT RIC on long-term policies. In section, we explore more on how MATL can be useful for O-RAN.

Finally, it is also important to highlight the differences between team learning and distributed or federated learning. Federated learning indicates how the data is used to train/update local models using the information from heterogeneous/homogeneous agents [105] and does not cover the specific dynamics of agent interactions such as in a team. Team learning is not constrained to a specific data model; agents learn by interacting with the environment and they converge on a policy.

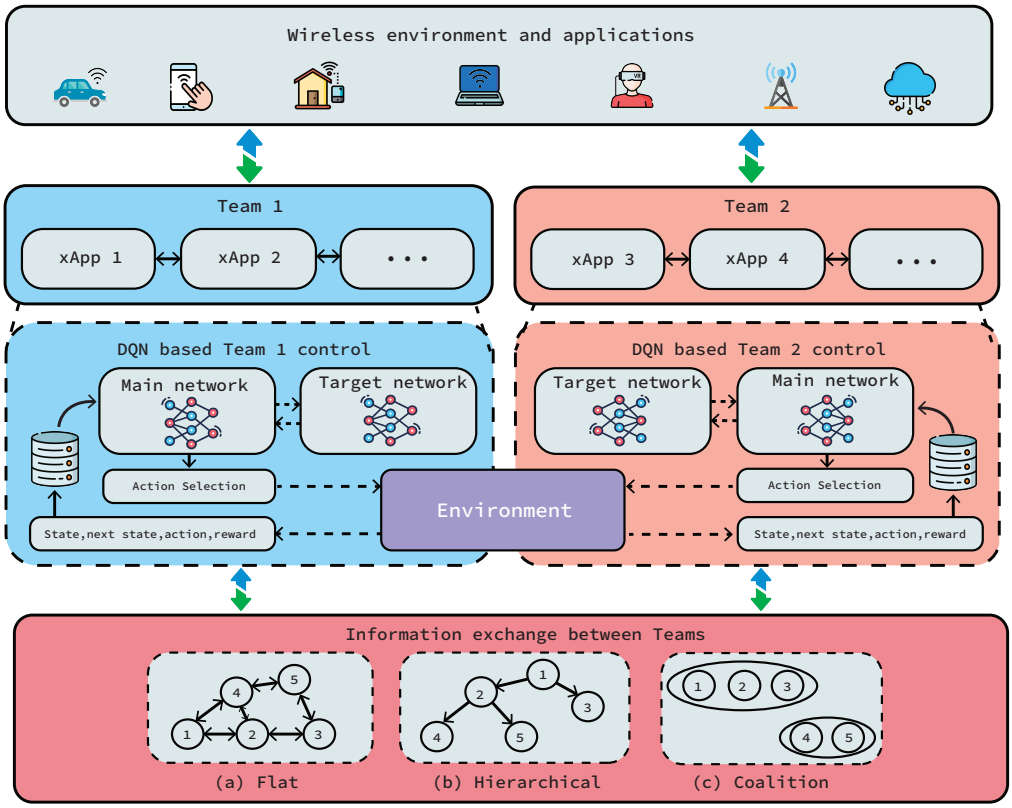

Figure 2.6: Multi-agent team Learning: The figure shows two teams comprised of more than one xApp where each xApp is a DQN agent that interacts with the environment. Teams can form different communication structures to share information: (a) flat, (b) hierarchical, and (c) coalition.

2.2.3 ML-based Spatial Reuse in Wi-Fi

Machine Learning (ML) has been considered a viable instrument for the optimization of SR in the context of Wi-Fi 802.11ax. More specifically, Reinforcement learning-based spatial reuse has been of interest in recent literature. The studies have focused on distributed solutions with no cooperation or centralized schemes of multi-armed bandits. These studies are summarized below.

In [106] presents a centralized MAB consisting of an optimizer based on a modified Thompson Sampling (TS) algorithm and a sampler based on the Gaussian Mixture (GM) algorithm to improve spatial reuse in 802.11ax Wi-Fi. More specifically, the authors pro-

pose dealing with the large action space comprised of TP and Overlapping BSS/Preamble-Detection (OBSS/PD) thresholds by utilizing a MAB variant called the Infinitely Many-Armed Bandit (IMAB). Furthermore, a distributed solution based on Bayesian optimizations of Gaussian processes to improve spatial reuse is proposed in [107]. In [108], the authors present a comparison among well-known MABs such as ϵ-greedy, UCB, Exp3, and Thompson sampling in the context of decentralized spatial reuse via Dynamic Channel Allocation (DCA) and Transmission Power Control (TPC) in WLANs. The results showed that "selfish learning" in a sequential manner presents better performance than "concurrent learning" among the agents. The authors in [109] propose a federated learning-based solution in a multi-Basic Service Set (multi-BSS) scenario to improve SR. Furthermore, in [42], the authors present a comparison among cooperative and non-cooperative algorithms to improve SR in decentralized settings. Moreover, they propose to model the problem as a Multi-Agent Contextual Multi-Armed Bandit (MA-CMAB), and for the first time, set the grounds for the application of transfer learning to deal with dynamic environments in the SR context. Despite the advantages of transfer learning techniques in RL, they are susceptible to negative transfer in which the target task underperforms after receiving knowledge from the source task [110]. To address the aforementioned issue, newer techniques such as meta-learning allow us to quickly adapt to unseen tasks given sufficient source tasks expertise [77].

Other solutions not related to reinforcement learning can be found in the literature to improve spatial reuse in WLANs. For instance, in [111], the authors propose a distributed algorithm where the APs decide their Transmission Power based on their Received Signal Strength Indicator (RSSI). Moreover, in [112], the authors present an algorithm to improve spatial reuse by utilizing diverse metrics such as SINR, proximity information, RSSI, and Basic Service Set (BSS) color and compare it with the legacy existing algorithms. The ultimate goal of the previous algorithm is the selection of the channel state (IDLE or BUSY) at the moment of an incoming frame given the previous metrics. In all the above works, the authors employ either centralized or decentralized schemes with no cooperation to address SR optimization in Wi-Fi.

2.2.4 ML-based Traffic Allocation in Multi-Link Operation Wi-Fi 7

To the best of our knowledge, Reinforcement Learning-based traffic allocation policy in 802.11be Multi-Link Operation was non-existent throughout the literature. However, 802.11be MLO has been recently of great interest among researchers. Some of these works are referred to below.

In [113], the authors present a study of MLO in Wi-Fi 802.11be in the context of a sub-standard of 802.1, Time Sensitive Networking (TSN). Results show that MLO can improve latency over a single link, however, the usage of uncorrelated channels is recommended to increase the average SNR perceived by the stations. In addition, the authors in [114] perform an experimental study using a real dataset measurement; a dataset containing 5 GHz spectrum occupancy measurements. In the previously mentioned study, it is proved that when MLO is available in symmetrical occupied channels, latency can be reduced by one order of magnitude over a single link operation. Interestingly, it is observed that when the channels do not fulfill the symmetry condition, the latency performance worsens. The authors solve the previous issue by adding a backoff counter in each interface to avoid packet collision. Related to our current work, the authors in [115] propose three non-dynamic flow traffic allocation policies in Wi-Fi 802.11be: SLCI, MCAA, and *Multi Link Same Load to All interfaces* (MLSA), respectively. They prove that MLO performance is tightly related to the efficiency of the flow distribution over the available interfaces in a Multi-Link Device. The authors present in [116], a dynamic flow traffic allocation policy that can adapt its traffic policy not only upon the arrival of the traffic flow but every 1 second. Finally, some surveys can be found throughout the literature that study the challenges and perspectives of 802.11be and MLO such as in [117, 118].

2.2.5 ML-based Channel Selection in Multi-Link Operation Wi-Fi 7

Channel selection algorithms in Wi-Fi have been of great interest in the recent literature. However, channel selection in IEEE 802.11be MLO has not been studied due to its novelty. In addition, to the best of our knowledge, this is the first work that tackles Parallel Transfer Reinforcement Learning among MASs and applies such a technique in the context of channel selection in IEEE 802.11be. Below, we list some of the works related to parallel transfer learning and channel selection.

In [87] the authors utilized parallel transfer learning among intelligent agents to coordinate electrical vehicle charging to prevent transformer overload in smart grid scenarios. This work introduces, for the first time, parallel transfer learning and discusses the advantages over sequential transfer learning techniques. Moreover, the same authors present in a later work [119] a survey study where possible alternatives to employ parallel transfer learning are considered. On the other hand, channel selection has been studied in the previous Wi-Fi standards where only Single Link Operation is considered. For instance, in [120], the authors study channel selection for a single user utilizing a Deep Q-Network

agent. In this work, at the beginning of each time slot, the user selects a channel and receives a reward based on how successful the transmission was. Additionally, in [121], a Q-Learning-based algorithm is utilized in the selection of the channel/subframe in LTE-Wi-Fi scenarios. In the previous work, the authors measured the effectiveness of the algorithm based on channel utilization and channel utilization fairness among the APs.

2.2.6 ML-based XR Codec Adaptation in 5G Networks

In recent years, XR traffic has attracted the interest of academia and industry due to its stringent requirements that have posed great challenges to 5G and beyond networks. XR, together with Cloud Gaming (CG) is currently one of the most important 5G media applications under consideration in the industry [122]. XR is focused on creating immersive experiences that blend the digital and physical worlds. Cloud gaming, on the other hand, is specifically about delivering gaming content via the cloud, reducing the need for powerful hardware on the user's end. XR involves VR, AR, and Mixed Reality (MR) technologies, requiring specific hardware like VR headsets or AR glasses. Cloud gaming leverages cloud computing and high-speed internet to deliver gaming experiences. XR can be used in various fields such as entertainment, education, healthcare, and more, while cloud gaming is primarily focused on gaming. Recently, some studies have proposed adaptive mechanisms to improve KPIs for XR. For instance in the works [123, 124], the authors study novel Quality of Service (QoS) control procedures to handle the peculiarities of XR traffic. In addition, it presents three traffic adaptation mechanisms at the application layer (named XR loopback) to improve performance between XR applications and the 5G Radio Access Network (RAN). Each mechanism consists either of adjusting in real-time XR codec parameters such as data rate or frame per second (FPS) or adjusting both of them concurrently. In conjunction with the previous mechanism, the authors propose a QoS-based scheduler and conclude that even though the scheduler improves the classical resource allocation schedulers such as Proportional Fairness (PF) the loopback mechanisms contribute the most to improving QoS metrics. However, some unwanted behavior can be seen in the proposed adaptive mechanisms such as unfairness among flows and increasing packet loss rate when the channel conditions worsen.

On the other hand, several works in the literature focus their interest on codec adaptation in the context of video streaming and RL. For instance, in [125] the authors optimize video streaming bitrate using an Actor-Critic RL architecture while considering users' preferences. In addition, in [126] the authors study video streaming chunk representation by using a two-layer deep-RL framework: at the physical layer adjusting beamforming parameters and at the application layer the video chunk representation. Finally [127] proposes

GreenABR, an energy-aware adaptive bitrate (ABR) algorithm based on deep RL. The authors in the former work utilize real cellphone terminal energy consumption data to decide on the best bitrate without jeopardizing user QoE.

In the realm of 5G, 6G, and Wi-Fi, previous works have consistently asserted and demonstrated that ML, particularly its subfield RL, offers effective solutions to intricate optimization problems within a reasonable convergence time, surpassing more conventional solvers like convex optimization. Through the application of RL and the innovative techniques proposed in this thesis, we present various applications of RL in both single and multi-agent contexts within wireless networks. Our focus extends beyond achieving superior results compared to traditional methods and existing RL baselines; we are also keen on enhancing adaptation and convergence speed. To this end, we employ techniques that concentrate on reducing the action state space, including TRL, PTRL, meta-learning, and attention mechanisms.

Chapter 3

RL based Load balancing and Handover in 4G and 5G Networks

Load balancing in next-generation wireless networks remains a challenging problem. Resource orchestration in wireless networks with multiple BSs requires an intelligent solution. In the first work of this chapter, we propose a centralized RL algorithm named CDQL that considers QoS metrics for load balancing. Our results show that CDQL outperforms existing baselines considerably. Concerned about convergence time, we propose MADDPG-AP, a competitive distributed algorithm that improves convergence time by 70% when compared to the CDQL algorithm. Finally, we study handover in multi-RAT networks. We compare a hierarchical RL algorithm named Hierarchical Deep Q-Learning (HiDQL) with a centralized deep Q-Learning algorithm and existing algorithmic baselines. The results show that both RL solutions improve existing algorithms' KPIs, with HiDQL demonstrating the best performance among the intelligent solutions.

3.1 QoS-Aware Load Balancing in Wireless Networks using Clipped Double Q-Learning

The exponential increase in mobile network usage alongside the growing data consumption demand by several use cases such as AR/VR (Augmented Reality/Virtual Reality) and video streaming, have led wireless networks to evolve to satisfy the pressing requirements. More specifically, video-dominated applications related to video streaming, video conferencing, and high-quality buffered media have faced a great spike due to the recent impact

of the COVID-19 pandemic. Some studies showed that after the lockdown, an increase of 215-285% in VoIP and videoconferencing traffic and a 20-40% increment in streaming and web video consumption have been observed [128]. Besides, the bandwidth requirement of the aforementioned use cases, they also have tight delay requirements. The trend for multimedia usage is expected to continue after the pandemic and mobile network operators will shoulder a majority of the load. This calls for optimized use of resources at the RAN, the transport network, and the core.

LTE (Long Term Evolution) and its successor 5G (5^{th} Generation) NR (New Radio) can support self-optimization functionalities that 3GPP has identified in the context of SON. These also include load balancing. Load balancing for the RAN involves handing over UEs to less-occupied base stations. Naturally, a base station's load would be for the number of UEs that are associated with it and the traffic demand of these UEs.

As part of the handover mechanism, each UE in the network will send periodical measurement reports to its respective serving cell. In practical terms, the serving cell will send via a Radio Resource Control (RRC) message the indication of what type of measurements each connected UE must gather and consequently report of the vicinity cells and of itself. Then, BSs will look for handover opportunities by verifying some possible events to trigger or not the handover procedure. Some of the events described by 3GPP are [24]:

- A2: Serving cell Reference Signal Received Quality (RSRQ) becomes worse than threshold
- A3: Neighbour Reference Signal Received Power (RSRP) becomes better than serving cell
- A4: Neighbour cell RSRQ becomes better than threshold

In the context of handover, there are two well-known mechanisms. These are A2-A4 handover where two conditions must be satisfied corresponding to the events A2 and A4: *i*) to trigger handover the serving cell of the UE must fall below a certain RSRQ serving cell threshold and *ii*) the handover is only performed if the difference between the best neighbor and the serving cell RSRQ is greater than certain predefined neighbor cell offset. The A3 handover algorithm or "strongest cell handover algorithm" is a simpler approach where handover is triggered for the UE to the best cell in the measurement report. When the best cell in terms of RSRP is selected, this value must be greater than the current serving cell by a hysteresis value and must be maintained for a time (TTT, time to trigger) to avoid ping-pong effect. Handover using the previous mechanism is illustrated in Fig. 3.1.

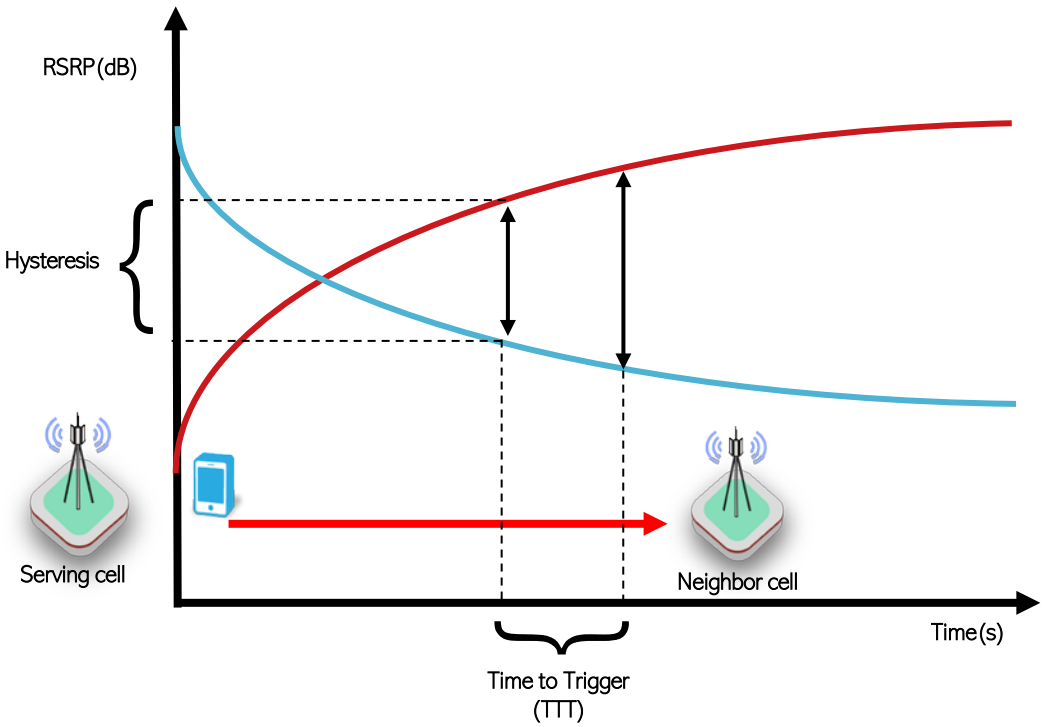

Figure 3.1: Illustration of A3 handover algorithm.

3.1.1 System model

We consider a network consisting of a set Γ of size M_T of base stations (BS). The network serves a set of Ψ of size N_T stationary mobile users deployed randomly around the BSs.

According to the Downlink (DL) bandwidth configuration chosen, B MHz, let us define N_{RB}^{DL} as the number of resource blocks available to be assigned by the Media Access Control (MAC) scheduler. The Channel and QoS Aware (CQA) scheduler is chosen as MAC scheduler. This scheduler assigns resource blocks by prioritizing users with greater Head of Line (HOL) delay and maximizing MAC layer throughput [129]. In this study, we consider one Resource Block Group (RBG) ($RBG = N_{RB}^{DL}/K$) as the smallest resource unit where $K \in [1, 2, 3, 4]$ according to the DL bandwidth used. In addition, a centralized approach is assumed where a central agent can monitor the BSs and UEs' Key Performance Indicators (KPIs). This can be conveniently applicable to C-RANs, as mentioned before. The agent can alter the CIO of each i cell based on the proposed machine-learning algorithm. The CIO of each cell is denoted as $\phi_i \in [\phi_{min}, \phi_{max}]$ dB and is modified to trigger the handover algorithm among cells. For A3 handover algorithm, if a user is served by some cell i, it will start a handover to cell j request through the $X2$ interface if the following condition holds:

$$RSRP_j + \phi_{j \to i} > Hys + RSRP_i + \phi_{i \to j}, \tag{3.1}$$

where $RSRP_j$ and $RSRP_i$ are the measured RSRP values in dB of the serving cell and the neighbor cell. Hys is the hysteresis value used to avoid ping-pong scenarios. $\phi_{j \to i}$ and $\phi_{i \to j}$ are the cell individual offsets. The values are independent for each cell.

3.1.2 Clipped Double Q-Learning Based Load Balancing

In the proposed approach described in Algorithm 3.1, an agent observes the environment parameters, such as CQI, packet delay, and resource block utilization per BS. The agent's actions consist of modifying the BS's individual CIO values to maximize the agent's reward. Note that in this case, we employ a decaying epsilon (ϵ) approach for exploration purposes. This exploration strategy involves encouraging the agent to select random actions at the beginning of the training phase and to act more greedily as it progresses through later steps in the environment.

Action space selection

The actions of our agent are defined as a vector with the CIO values assigned to each BS. For CDQL the set of actions will be deterministicly predefined by the possible permutations of the set of predefined values.

For the CDQL algorithm, the size of the action space will be ${}_lP_k = \frac{l!}{(l-k)!}$, where l is the size of the set of the possible CIO values that can take each BS and k will be equivalent to M_T or the amount of BSs in the network. Thus,

$$A(t) = \begin{bmatrix} \phi_1(t) & \phi_2(t) & \dots & \phi_{M_T}(t) \end{bmatrix} \tag{3.2}$$

The individual CIO value defined as ϕ_i will be lower and upper bounded by two predefined values ϕ_{min} and ϕ_{max} as described as follows: $\phi_i(t) \in [\phi_{min}, \phi_{max}]$.

Algorithm 3.1 Clipped Double Q-Learning (CDQL)

Initialize Q_{θ_1},Q_{θ_2} and target networks with random weights, replay buffer J, ϵ, ϵ_{decay} and ϵ_{min}
for environment step **do**
  Observe state s_t
  Select $a_t \sim \pi(a_t, s_t)$ **if** $\epsilon \geq x \sim U(0,1)$ **otherwise** select random action a_t
  Execute a_t and observe next state s_{t+1} and reward $r_t = R(s_t, a_t)$
  Store (s_t, a_t, r_t, s_{t+1}) in replay buffer J
  for each update **do**
    Sample $(s_t, a_t, r_t, s_{t+1}) \sim J$
    Compute $Y_t^{CDQL} = r_t + \gamma min_{i=1,2} Q_{\theta_i}(s_{t+1}, argmax_{a'}\ Q_{\theta_i}(s_{s+1}, a'))$
    Perform gradient descent step on $(Y_t^{CDQL} - Q_{\theta_i}(s_t, a_t))^2$
    Update target networks parameters: $\theta_i' \leftarrow \tau * \theta_i + (1-\tau) * \theta_i'$
    Update ϵ **if** $\epsilon > \epsilon_{min}$
    $\epsilon = \epsilon * \epsilon_{decay}$
  end for
end for

State space selection

The state space is composed of two terms. The first one corresponds to the attached UEs' ratio in each BS and the second term is the Resource Block Utilization (RBU) vector comprised by the RBU of each BS. Thus, the state $S(t)$ will be represented by the concatenation of both metrics as:

$$S(t) = \big[\mathbf{U}(\mathbf{t}), \mathbf{P}(\mathbf{t})\big] \tag{3.3}$$

where $\mathbf{U}(t)$ corresponds to a M_T-length vector comprised by the ratio of UEs attached to each cell i.

$$\mathbf{U}(t) = \big[u_1(t) \quad , \ldots , \quad u_{M_T}(t)\big] \tag{3.4}$$

where Ψ_{Γ_i} is the total of UEs attached to cell i and $u_i = \dfrac{\Psi_{\Gamma_i}}{N_T}$

Additionally, $\mathbf{P}(t)$ corresponds to a M_T-length vector conformed by the RBU for each BS at time t as:

$$\mathbf{P}(t) = \big[p_1(t) \quad , \ldots , \quad p_M(t)\big] \tag{3.5}$$

Given the fact that resource allocation in each cell is reported every TTI, which is a smaller time frame compared with the observable time interval of our agent (1s), we consider the resource block utilization in each TTI as a discrete random variable. Thus, we can model the resource block utilization during the observation time p_i as the expected value of the resource block utilization for each TTI p_{tti}.

$$p_i = \mathbb{E}[p_{tti}] \tag{3.6}$$

Reward

The total reward function is calculated based on Quality of Service (QoS) parameters and the load of each BS in terms of resource block utilization as follows:

$$R_T(t) = w_1 * R_D(t) + w_2 * R_{RB}(t) + w_3 * R_{CQI}(t) \tag{3.7}$$

Here $R_D \in [-1, 1]$ is the reward related to delay constraints, $R_{RB} \in [-1, 1]$ is the reward related to resource allocation usage, and $R_{CQI} \in [-1, 1]$ corresponds to a reward measuring the quality of the modulation used by the users in the network. Finally, w_1, w_2, w_3 are the weights for each reward.

The first component of the proposed reward can be defined as:

$$R_D(t) \triangleq \frac{1}{N_T} \sum_{i=1}^{N_T} \mathbb{1} \cdot \delta_i(t) \tag{3.8}$$

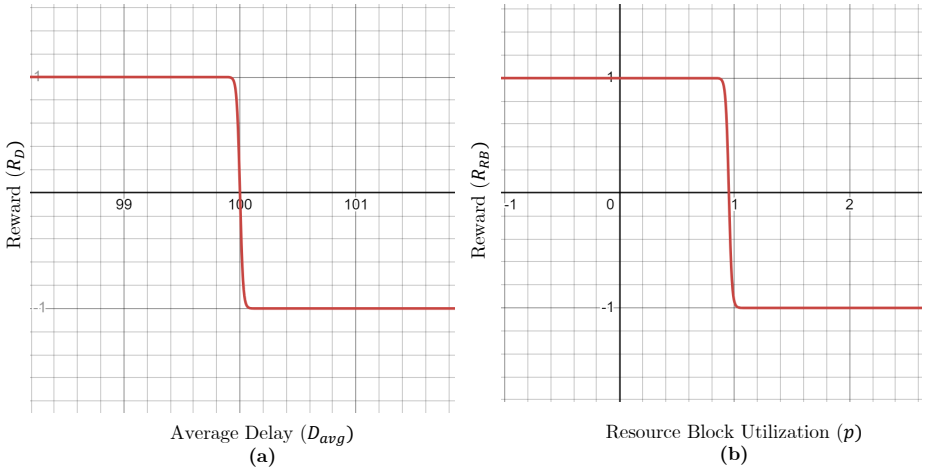

Figure 3.2: Reward design: ***(a)*** Sigmoid function for the delay reward; an average delay above $2/3PDB$ is penalized ***(b)*** Sigmoid function for the resource block utilization reward; a resource block utilization above 95% is penalized.

and,

$$\delta_i(t) = \begin{cases} -1 & \text{if } \exists \Psi_i \notin \Psi_C \ , \\ \urcorner(D_{avg}) & otherwise \end{cases} \tag{3.9}$$

where Ψ_i represents the ith UE and Ψ_C, $\Psi_C \subset \Psi$ is the set of connected UEs to any BS $\in \Gamma$. On the other hand, $\urcorner(D_{avg})$ is a sigmoid function defined as follows:

$$\urcorner(D_{avg}) = 1 + \frac{c}{1 + e^{-o(D_{avg} - \mathcal{F})}} \tag{3.10}$$

Here c establishes the upper bound of the slope, o adjusts the slope of the sigmoid, and $\mathcal{F} = 2/3 * PDB$ controls the target packet delay. PDB is the Packet Delay Budget which is according to the type of traffic used in the network.

The objective of the first term is to reward each UE's average latency based on the PDB of a defined packet type. As can be seen, we penalize the cases if a UE gets disconnected from the network as a result of the load balancing decision.

The second component of the proposed reward can be defined as:

$$R_{RB}(t) \triangleq 1 + \frac{c}{1 + e^{-a(max(\boldsymbol{P}(t)) - \mathcal{D})}} \tag{3.11}$$

where $\boldsymbol{P}(t)$ corresponds to a M_T-length vector composed of the resource block utilization of each BS at time t. The term $max(P(t))$ allows selecting the most loaded BS to either

penalize or reward based on a predefined threshold defined as γ_{RB}. As the maximum value of $P(t)$ decreases, a higher reward is obtained. c establishes the upper bound of the slope, a adjusts the slope of the sigmoid as in (3.11) and $\mathcal{D}$ controls the target data utilization. For both R_D and R_{RB} objectives, the sigmoid function allows to penalize with -1 the average delay and RBU if the thresholds $\mathcal{F}$ and $\mathcal{D}$ are not satisfied. If satisfied the agent receives a positive feedback of 1. The behavior for both rewards can be observed in Fig. 3.2.

Finally, the third component is defined by:

$$R_{CQI}(t) \triangleq \frac{1}{N_T} \sum_{i=1}^{N_T} \mathbb{1} \cdot \Omega_i(t), \tag{3.12}$$

where,

$$\Omega_i(t) = \begin{cases} -1 & \text{if } \Psi_{i_{CQI}}(t) < 6, \\ 0 & \text{if } 7 \leq \Psi_{i_{CQI}}(t) \leq 9 \ , \\ 1 & \textit{otherwise}, \end{cases} \tag{3.13}$$

where $\Psi_{i_{CQI}}$ corresponds to the CQI measurement of the ith UE. The objective of this term is to reward or penalize each UE's CQI based on the fact that higher CQI will be translated in a higher modulation scheme and thus a more efficient usage of the resource block allocation.

3.1.3 Baseline: Resource Block Utilization based Handover Algorithm (ReBUHA)

In this work, we consider a baseline algorithm named ReBUHA in addition to the classical A3 handover algorithm to perform a comparison of our results. ReBUHA algorithm replaces the A3 handover which mainly relies on received power (see equation 3.1) with a resource allocation awareness method to trigger the handover procedure. As seen in Algorithm 13, $\mathbf{P}(t)$ is M_T-size vector where each element corresponds to the ratio of RBs used in the m^{th} BS at the time t. Every $t = 1s$, the algorithm will use $\mathbf{P}(t)$ information and look for handover opportunities in a centralized way. γ_{RB} corresponds to a resource block utilization threshold defined as the ratio of usage in which the handover algorithm will look for opportunities for handover.

Algorithm 3.2 ReBUHA algorithm

Data: RBU information of all BSs
Result: Resource allocation based handover
if all $p_i(t) \in \mathbf{P}(t) : p_i(t) > \gamma_{RB}$ **then**
    break
end if
for $p_{j \neq i}(t) \in \mathbf{P}(t)$ **do**
    Check if UE Ψ_i is attached to an overloaded BS
    if $p_j(t) < \gamma_{RB}$ **then**
        Trigger handover to cell $j \in \Gamma$
    else
        break
    end if
end for

3.1.4 Performance Evaluation

Simulation Setting

The simulations are performed by using the discrete network simulator ns-3 [130]. In Table 3.1 and Table 3.2 we provide the settings utilized in our simulations and the RL parameters, respectively. Each BS is positioned with an inter-site distance of 720 meters. Five different scenarios are tested under the proposed algorithms with 30, 35, 40, 45, and 50 UEs. Each scenario is designed by distributing a percentage of the total users on the edge of each cell in a random disc and the rest is uniformly allocated throughout the coverage of the middle BS. We initialize the simulations by attaching all UEs to the BS that sits in between 2 BSs and then according to the policy of each strategy, whether handover is triggered or not. The traffic is a mixture of 20 UEs using CBR, and the rest following Poisson arrivals. For the case of Poisson traffic, we use a small payload of 32 bytes with a traffic load of 0.1 Mbps, meanwhile, for the CBR which emulates video traffic we use a larger payload of 250 bytes with an interval of 10 ms. Simulation results are collected by averaging 15 simulations per scenario. Each simulation consists of 150 episodes with 50 iterations per episode. OpenAI Gym is used as the interface between ns-3 and our agent [131]. Lastly, we consider each parameter of our reward of equal importance, thus the weights of equation 3.7 are equal to 1.

Table 3.1: Network settings

Parameter	**Value**
Inter-site Distance	720 m
M_T	3
N_T	30,35,40,45,50
Center Frequency	2 GHz
System Bandwidth	5 MHz (25 resource blocks)
Pathloss Model	Log Distance Propagation Loss Model
	$95 + 27\ log_{10}(distance[km])$
BS antenna height	30 m
UE antenna height	1.5 m
Max Tx power	20 dBm
MAC scheduler	CQA scheduler
User distribution	Stationary and uniformly distributed
Traffic Model	Conversational video (live streaming) and Poisson
	Packet payload size = 250 Bytes
	Interval = 10 ms
	Packet delay budget = 150 ms
Handover algorithm	A3-event based
	Time to trigger = 8 ms
	Hysteresis = 2 dBm

3.1.5 Simulation Results

To assess the performance of our proposed scheme, we present throughput, delay, jitter, and Packet Loss Ratio (PLR) with a 90% confidence interval as well as the convergence of the machine learning algorithm.

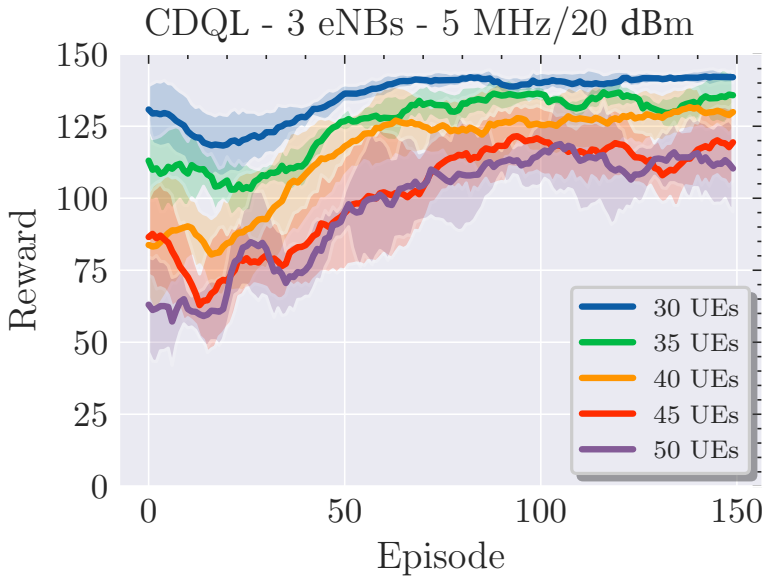

Figure 3.3: Learning performance for a varying number of UEs and the CDQL algorithm.

Figure 3.3 shows the learning performance achieved by CDQL for the simulated scenarios. The trend shows how the reward value per episode converges in all cases and the converged reward value becomes lower as the number of UE increases. This is due that some of the KPIs that are tracked by our agent are affected by the increment of the number of users. Figure 3.4 presents throughput, end-to-end delay, jitter, and PLR. In Fig. 3.4 (a) it can be seen that our algorithm achieves an average improvement of 6.1% and 9.5% in throughput in comparison with the A3 and ReBUHA algorithms, respectively. Similarly, the other figures (b, c, d, and e) show a noticeable improvement to the baselines with a gain of 49.8% and 52.9% in terms of delay, 55%, and 51% in terms of jitter and 34% and 55.2% in terms of PLR for the cases of the A3 and ReBUHA algorithms, respectively. Note that, the delay results of ReBUHA algorithm decreases at 45 UEs in comparison with A3. This is because, after 45 UEs the algorithm triggers its handover procedure as the middle BS surpasses the resource block utilization threshold predefined by the value of γ_{RB}. However, metrics such as PLR continue to worsen reassuring our thesis that in high-traffic scenarios, a closed track of QoS metrics is needed. In addition to the previous results, an unexpected behavior is noticeable in the delay and jitter metrics for 45 and 50 UEs, where these metrics decrease when they are expected to increase. This behavior corresponds to a mere statistical calculation by the network simulator and would not be possible to explain without the PLR graph. As the number of UEs increases, the PLR, BS

Table 3.2: RL environment settings

Parameter	**Value**
Number of iterations/episode	50
Number of episodes	150
Gym environment step time	1s
Batch size	32
CDQL	ϕ set $\{-9, -6, -3, 0, 3, 6, 9\}$ dBm $w_1, w_2, w_3 = 1$ $\mathcal{F} = 2/3 * PDB$ where $PDB = 150$ ms, $c = -2$, $o = 75$ $\gamma_{RB} = 0.6$, $a = 20$ Optimizer : Adam (1e-3) Number of hidden layers (N_h) : 2 Loss function : Huber Loss (clip delta = 1.0) Update target model type: Polyak averaging $\gamma = 0.95, \epsilon = 1.0, \epsilon_{min} = 0.001, \epsilon_{decay} = 0.995$

load, and packet collisions also increase, resulting in more UEs being left without service. Since the simulator only calculates its statistics based on the connected UEs, it may appear that jitter and delay have improved. However, this improvement is only partial because, although these metrics have improved, many users are left without service.

Figure 3.5 presents the components of the vectors **P** and **U** at the end of the simulated scenarios. It can be seen how the A3 handover algorithm does not trigger in any of the scenarios by keeping all the UEs attached in the middle BS. The latter occurs because none of the UEs comply with the event trigger condition of the A3 algorithm. Furthermore, it is noticeable that in Fig. 3.5 (d) the ReBUHA algorithm starts distributing the load in agreement with the behavior described in the previous figure. For our proposed algorithm it is

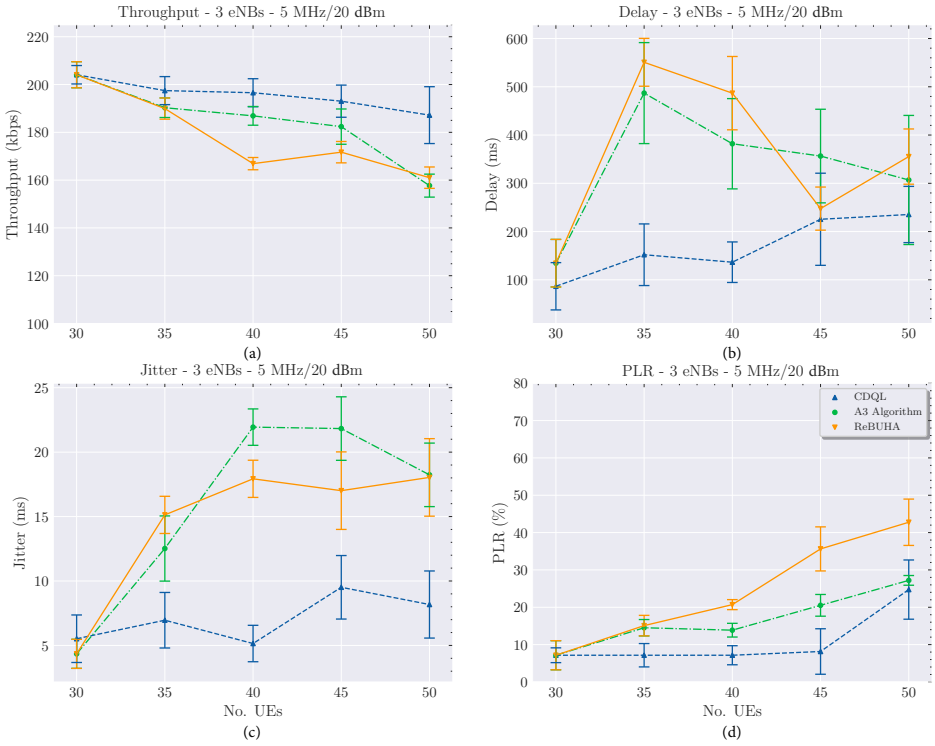

Figure 3.4: Performance metrics of CDQL, A3 and ReBUHA algorithms. (a) Throughput, (b) Delay, (c) Jitter and (d) PLR

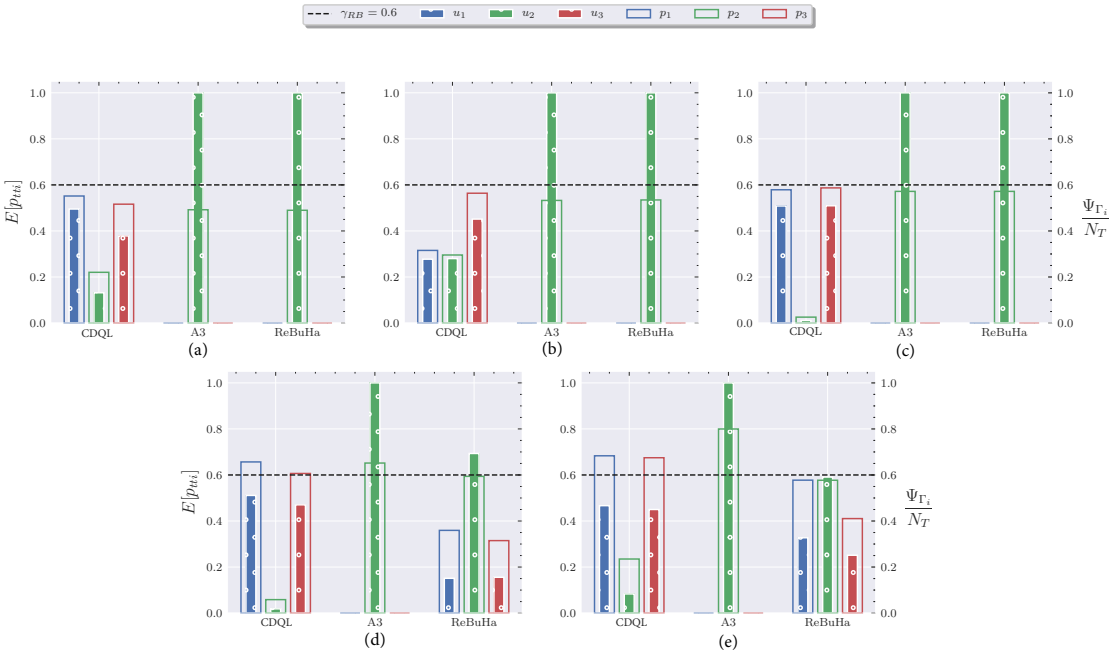

Figure 3.5: Average BS resource block utilization (RBU) and UE per BS ratio after convergence of each strategy. (a) 30 UEs, (b) 35 UEs, (c) 40 UEs, (d) 45 UEs and (e) 50 UEs

.

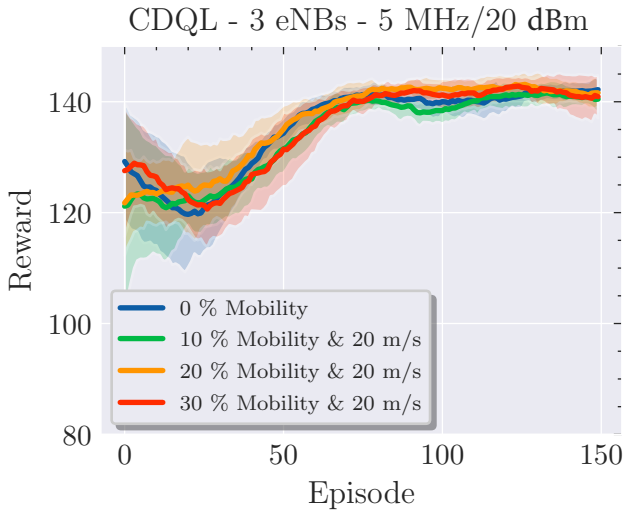

Figure 3.6: Learning performance of CDQL for 30 UEs and 0%, 10%, 20% and 30% of mobile users with UE's speed of 20 m/s.

observable that it distributes the UEs over the BSs based on choosing the "best" CIO value per BS that will maximize our agent's reward objective function. Note that the proposed algorithm for (d) and (e) does not meet the target resource block utilization goal which is established by γ_{RB}. This behavior can be explained based on the needed minimization of not only the resource block utilization in the network but also the improvement of QoS metrics as well.

Additionally, we present the performance of our scheme under mobility. We consider 30 UEs where 10%, 20%, and 30% of the total UEs in the network are mobile and they use random walk with a speed of 20 m/s. Figure 3.6 shows that similar to the non-mobility scenario (represented as 0%), our scheme similarly converges for mobile scenarios. Furthermore, we show in Fig. 3.7 throughput, end-to-end delay, jitter, and PLR for non-mobile and mobile scenarios. Note that we only presented the A3 Algorithm and CDQL since for such a number of users the ReBUHA algorithm behaves identically as the A3. In terms of throughput and PLR, both CDQL and A3 algorithms perform similarly. For delay, the proposed CDQL offers slightly lower latency than the A3 algorithm. More specifically, our scheme can achieve an improvement of 64% respecting delay with no considerable difference in the other KPIs.

Finally, it is worth mentioning that our scheme shows its real potential when the number of users increases, in other words, when the scarcity of resources increases. We chose the

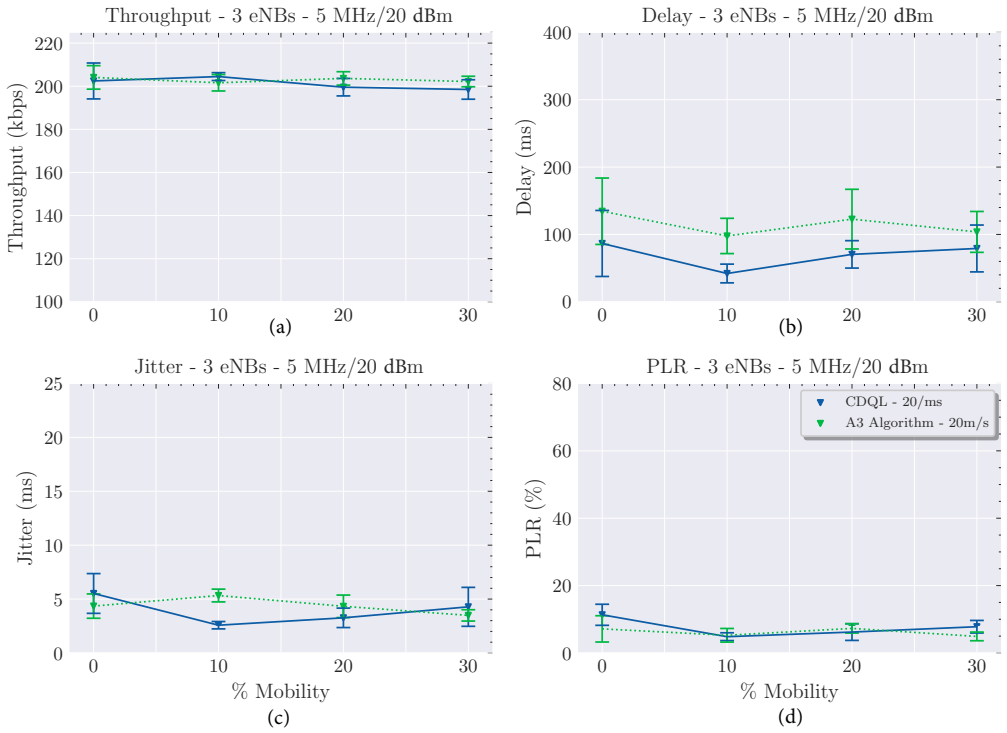

Figure 3.7: Different performance metrics of CDQL and A3 algorithms for different mobility percentages and UE's speed of 20 m/s. (a) Throughput, (b) Delay, (c) Jitter and (d) PLR

A3 best performance scenario (30 UEs) to show the steady behavior of our scheme.

3.2 Competitive Multi-Agent Load Balancing with Adaptive Policies in Wireless Networks

An undeniable growth of connected devices is experienced in the era of 5^{th} Generation (5G), which is expected to continue in the future 5G and beyond networks. Besides, tremendous volume and vast diversity in high data traffic applications with stringent QoS (Quality of Service) requirements are posing a challenge for these next-generation wireless networks.

Machine learning techniques, such as Reinforcement Learning (RL), have raised a strong interest among the research community and the industry for their use in performance enhancement of wireless networks. The similarity of RL techniques with the "trial and

error" human-like behavior conveys perfectly their usage in dynamic environments that seek "near" optimal control. Two main RL taxonomies are present in the literature: single-agent RL and multi-agent RL. In single-agent domains, an agent is the decision instance that oversees and controls the state of the environment [68].

AI-driven intelligent self-optimization functionalities are supported by 5G in which load balancing falls under the umbrella of applications considered in Self-Organizing Networks (SON). Load balancing in the context of radio access refers to the handover of UEs of high-loaded BSs (Base Stations) to less-loaded BSs. To perform such action, the resource block utilization (RBU) per BS is observed. The amount of RBU will be related not solely to the amount of UEs attached to a specific BS but also to the traffic characteristics. One of the most known and used handover algorithms named A3 is defined in [24]. This release specifies the required events in the BS that must occur to perform a handover procedure. More specifically, it requires that neighbor BS Reference Signal Received Power (RSRP) becomes better than serving BS and this condition must hold during a predefined time named Time to Trigger (TTT) to avoid ping-pong scenarios.

3.2.1 System model

We consider a network consisting of M_T base stations (BS) where for each i_{th} BS, $i \in \Lambda$. The network serves a set of Ψ of size N_T stationary mobile users randomly deployed around the BSs. The Channel and QoS Aware (CQA) scheduler is chosen as MAC scheduler. In this work, we consider one Resource Block Group (RBG) ($RBG = N_{RB}^{DL}/K$) as the smallest resource unit where $K \in [1, 2, 3, 4]$ according to the DL bandwidth used. Additionally, each BS is considered an agent of our environment. Each agent is capable of modifying their own CIO value based on the exploration performed by our RL scheme. The CIO value is utilized as part of the A3 handover algorithm that will indicate handover between BS i and BS j if the following condition holds: $RSPR_j + \phi_{j \to i} > Hys + RSPR_i + \phi_{i \to j}$. $RSPR_j$ and $RSPR_i$ are the measured RSRP (Reference Signal Received Power) values in dB of the serving BS and the neighbor BS, respectively. Hys is the hysteresis value used to avoid ping-pong scenarios.

3.2.2 Multi-Agent Deep Deterministic Policy Gradient with Adaptive Policies

In the proposed approach, each agent observes the attached UEs' ratio, the resource block utilization, and the currently chosen CIO value and receives its reward based on feedback

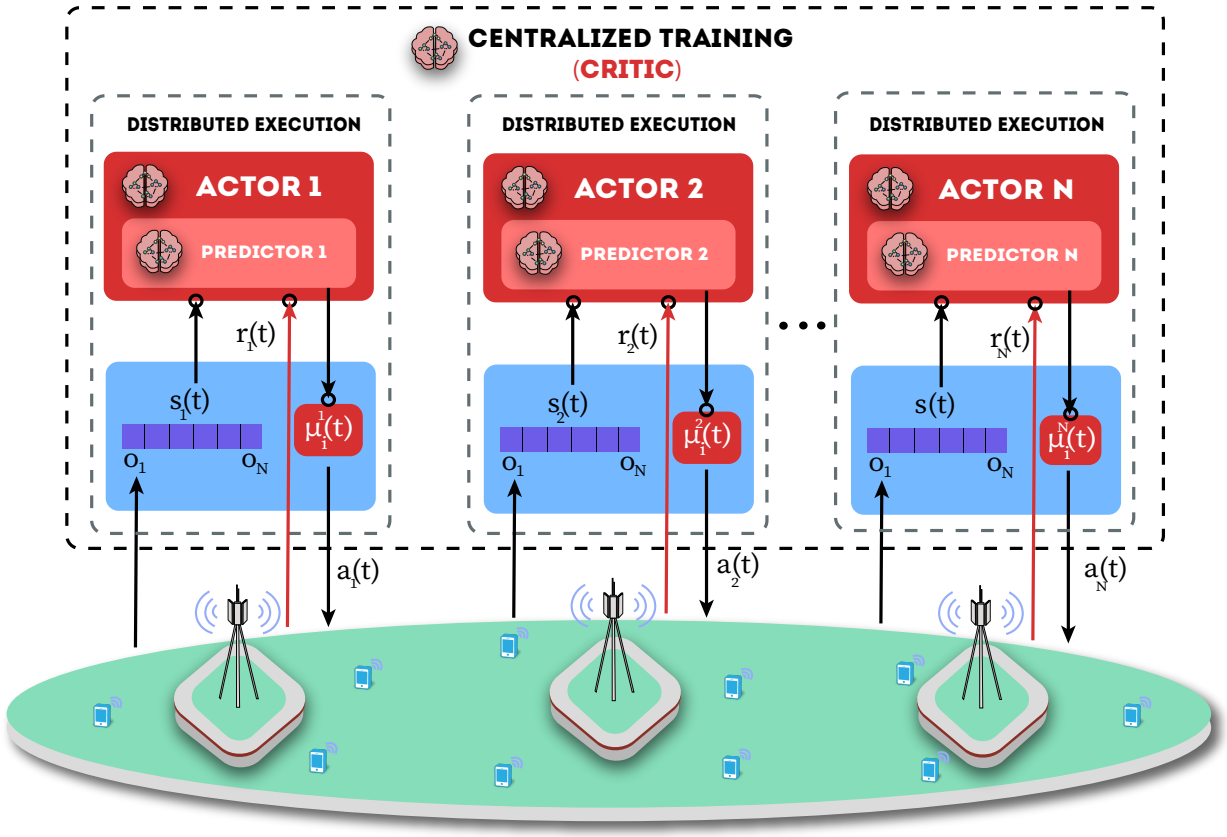

Figure 3.8: Multi-Agent Deep Deterministic Policy Gradient with Adaptive Policies scheme applied for load balancing in RAN. Each BS uses a sequence of past observations from the environment (state space) and utilizes it as input of the policy predictor that outputs a policy and consequently, the action to take. In this context, observations correspond to the attached UE's ratio, the resource block utilization and the CIO currently applied whereas the action yields the CIO value to apply.

from the environment in terms of throughput, packet delay, and resource block utilization. The agent's actions consist of modifying the BS's individual CIO values to maximize the agent's reward. In the following subsection, we present an overview of MADDPG-AP and then formally define our solution.

Addressing non-stationarity among competitive agents in MADDPG

Non-stationarity, mainly caused by the joint interaction of the agents with the environment, is one of the primary issues presented by the MARL setting. When the environment experiments a non-stationary behavior it becomes a moving target problem where each agent's best policy changes as individual policies change [132]. The authors in [47] called attention to the previously mentioned issue in the particular case of competitive agents due to the natural overfitting of a single agent's behavior over its competitors. To address this problem, it was proposed to train an ensemble of K sub-policies per agent where each agent had to choose a random k sub-policy per episode. Another proposed solution was M3DDPG (MiniMax Multi-agent Deep Deterministic Policy Gradient) [133], inspired by MADDPG, where the authors consider a minimax approach to train robust agents even when the opponent agent's performance is considered not good. Finally, in [134] the authors consider that by having K learned subpolicies per agent under the same constraints as MADDPG, the agent would be able to choose a corresponding policy according to the state observed. Our algorithm is mainly based on [134] saving some particular differences that will be addressed in the next subsection.

Adaptive policies and ranked buffer

The curse of dimensionality and collecting data during training are the main reasons why a slow exploration is experienced in off-policy MARL algorithms [135]. For instance, in the context of the RIC (RAN Intelligent Controller) in the O-RAN architecture (Open RAN) there are some close loop requirements in terms of latency that algorithms must comply [25]. Thus, a reasonable time must be considered for executing any models. To accelerate such convergence time and inspired by [136], we introduce a self-supervised technique using a ranked buffer in the policy predictors' training stage. As a main difference, we rank state sequences, instead of local observations, by the sum of the reward observed during such sequence and discard those sequences with low reward (see Algorithm 3.3) to form a new buffer that is used to train the policy predictor as seen in Fig. 3.9. By doing the latter, we select the best subpolicy given an observed sequence.

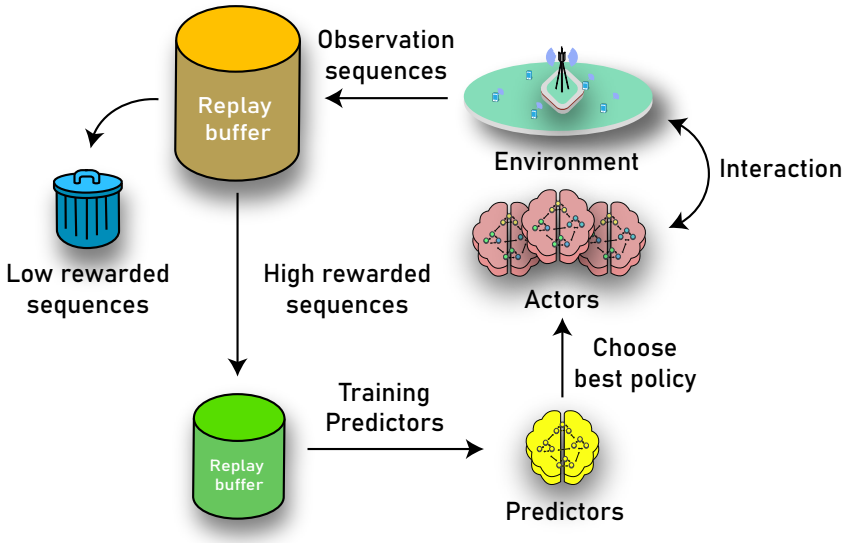

Figure 3.9: Ranked buffer in MADDPG-AP

Algorithm 3.3 Ranked buffer

1: **for** agent i to N **do**
2: Create sequences from $\mathcal{B}$ $\boldsymbol{w}_i = \langle (o_1, \ldots, o_W)_1, \ldots, (o_1, \ldots, o_W)_{N_s} \rangle$ using a sliding window of size W
3: Rank sequences in $\boldsymbol{w}_h$ based on $\hat{r}_{N_s} = \sum_{i=0}^{W} r_i$ and obtain a new buffer $\hat{\mathcal{B}}$ and $B_s = N_s/K$.
4: **if** $|\hat{\mathcal{B}}| > B_s$ **then**
5: Discard sequences with low reward in $\hat{r}_{N_s}$.
6: **end if**
7: **end for**

The pseudo-code for MADDPG-AP with ranked buffer is presented in Algorithm 3.4 whereas the training stage of MADDPG-AP is depicted in Algorithm 3.5.

Action space selection

The action value of each agent corresponds to the CIO value selected in each timestep and is defined as a continuous variable limited by a predefined exploration bound. Thus,

$$A_j(t) = \phi_j(t) \tag{3.14}$$

Algorithm 3.4 Training and execution stages for MADDPG-AP algorithm

# Training stage:
for agent i to N **do**
  Init set of policies $\Pi_i^K = \langle \mu_i^1, \ldots, \mu_i^K \rangle$ and predictors ρ_i
  for $k \leftarrow 0$ subpolicy **to** K **do**
    Learn Π_i^k and ρ_i
  end for
end for
# Execution stage:
Load predictors ρ_i and set of policies Π_i^K per agent, respectively.
for environment step $t \leftarrow 1$ **to** T **do**
  for agent i to N **do**
    $o_i^t \leftarrow$ receive observation and append it to $\boldsymbol{w}_i^t$
    $\mu_i^k \leftarrow$ predict policy by $\rho_i(\boldsymbol{w}_i^t)$;
    $|\boldsymbol{w}_i^t| = W$ $a_i^t \leftarrow$ select action by μ_i^k
  end for
  Execute actions $\boldsymbol{a} = (a_1^t, ..., a_N^t)$
  Collect rewards $\boldsymbol{r} = (r_1^t, ..., r_N^t)$
end for

The individual CIO value is lower and upper bounded by two predefined values ϕ_{min} and ϕ_{max}, respectively, as described as follows: $\phi_i(t) \in [\phi_{min}, \phi_{max}]$. Furthermore, Ornstein–Uhlenbeck noise is used to encourage exploration among the agents.

State space selection

Each BS state space is composed of three terms. The first term corresponds to the attached UEs' ratio, the second term is the Resource Block Utilization (RBU), and lastly, the current CIO value. Thus, the state $S(t)$ will be defined by the concatenation of such metrics as:

$$S_j(t) = \begin{bmatrix} u_j(t) & p_j(t) & \phi_j(t) \end{bmatrix}, \tag{3.15}$$

where $u_j = \dfrac{\Psi_{\Gamma_i}}{N_T}$ and Ψ_{Γ_j} is the total of UEs attached to BS j.

Additionally, $p(t)$ corresponds to the RBU for BS j at time t. Resource block allocation for each BS is gathered from every agent's observation time (0.2s) which is greater than

Algorithm 3.5 Multi-Agent Deep Deterministic Policy Gradient with Adaptive Policies and Ranked Buffer

Learning stage: Init N predictors networks ρ_i
for all K subpolicy **do**
  Initialize N actors and target networks μ_i and μ_i' respectively, the centralized critic Q_i^{μ} and target critic network with bounded random weights and replay buffer $\mathcal{B}$
  for all episode step **do**
    Initialize N Ornstein–Uhlenbeck random processes for action exploration
    for all environment step **do**
      for each agent i, select action $a_i = \mu_{\theta_i} + \mathcal{N}_t$ w.r.t the current policy and exploration
      Execute actions $\mathbf{a} = (a_1, ..., a_N)$
      Store $(\mathbf{x}, \mathbf{a}, \mathbf{r}, \mathbf{x}')$ in replay buffer $\mathcal{B}$
      $\mathbf{x} \leftarrow \mathbf{x}'$
      for agent i to N **do**
        Sample a random minibatch of S samples $(\boldsymbol{x}^j, \boldsymbol{a}^j, \mathbf{r}^j, \boldsymbol{x}^{j'})$ from $\mathcal{B}$
        Set $y^j = r_i^j + \gamma Q_i^{\boldsymbol{\mu}'}(\boldsymbol{x}^{j'}, a_1^{'}, \ldots, a_N^{'})|_{a_k^{'}=\boldsymbol{\mu}_k'(o_k^j)}$
        Update actor i by using the sampled policy gradient:
        $\nabla_{\theta_i} J \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_{\theta_i} \boldsymbol{\mu}_i(o_i^j)$
        $\nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_i^j, ..., a_i, ..., a_N^j)|_{a_i=\boldsymbol{\mu}_i(o_i^j)}$
        Update critic by minimizing the loss: $\mathcal{L}(\theta_i) = \frac{1}{S} \sum_{s=1}^{S} (y^j - Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_i^j, ..., a_N^j))^2$
      end for
      Update target networks parameters: $\theta_i' \leftarrow \tau * \theta_i + (1 - \tau) * \theta_i'$
    end for
  end for
  Sample training data K from $\hat{\mathcal{B}}$ and train LSTM predictor ρ_i networks with sequences as input and the corresponding subpolicy as output by minimizing the following negative log-likelihood loss:
  for agent i to N **do**
    $\nabla_{\rho_i} J(\rho_i) \approx \frac{1}{K} \sum_{k=1}^{K} \sum_{\mu_k'} -y^{\mu_i'} log(p_i(\mu_i'))$
  end for
end for

the TTI. Thus, we model the resource block utilization during the observation time p_i as the expected value of the resource block utilization for each TTI as p_{tti}.

$$p_j = \mathbb{E}[p_{tti}] \tag{3.16}$$

Reward

The total reward function is calculated based on Quality of Service (QoS) parameters and the load of each BS in terms of resource block utilization. Let us start with a well-known reward based on throughput [96]:

$$R_j = \sum_{i=0}^{U_c} r_i, \tag{3.17}$$

where U_c is the number of UEs connected to BS j and r_i corresponds to the throughput measured at time t by UE i. This reward does not consider RBU, latency, and any information related to the UE's connection status (For example, given a certain value of CIO, UE i can be attached to BS j but not achieve connectivity, in other words, the delay is infinite or throughput is equal to 0). Our approach consists of adding an extra parameter that will either punish or reward the utility function defined in equation 3.17. The mean throughput is chosen as defined in equation 3.18. The idea behind it is to use an approximation of the average throughput and to penalize if there are disconnected UEs. The second term in the sum of equation 3.18 will penalize or reward the equation 3.17 depending on ω_1 where ω_1, is a latency-based parameter as defined later in equation 3.21 that will oversee penalizing or rewarding according to the latency requirements for a specific application.

$$R_j = \sum_{i=0}^{U_c} r_i + \left[\frac{\omega_{1i}}{U_c} \sum_{i=0}^{U_c} r_i \right] \tag{3.18}$$

After some algebraic modification equation 3.19. is obtained.

$$R_j = \left[\frac{U_c + \sum_{i=0}^{U_c} \omega_{1i}}{U_c} \right] \sum_{i=0}^{U_c} r_i \tag{3.19}$$

The current reward formulation in equation 3.19 is composed of two KPIs that are UE-dependent. However, the RBU-based reward tries to minimize the number of resources that an individual BS is managing. Since our $\mathcal{R}_j$ is calculated based on BS, the reward based on RBU ω_2 depicted in equation 3.24 is multiplied by the term in equation 3.19 obtaining:

$$R_j = \omega_{2j} \left[\frac{U_c + \sum_{i=0}^{U_c} \omega_{1i}}{U_c} \right] \sum_{i=0}^{U_c} r_i \tag{3.20}$$

$\omega_1 \in [-1, 1]$ can be defined as:

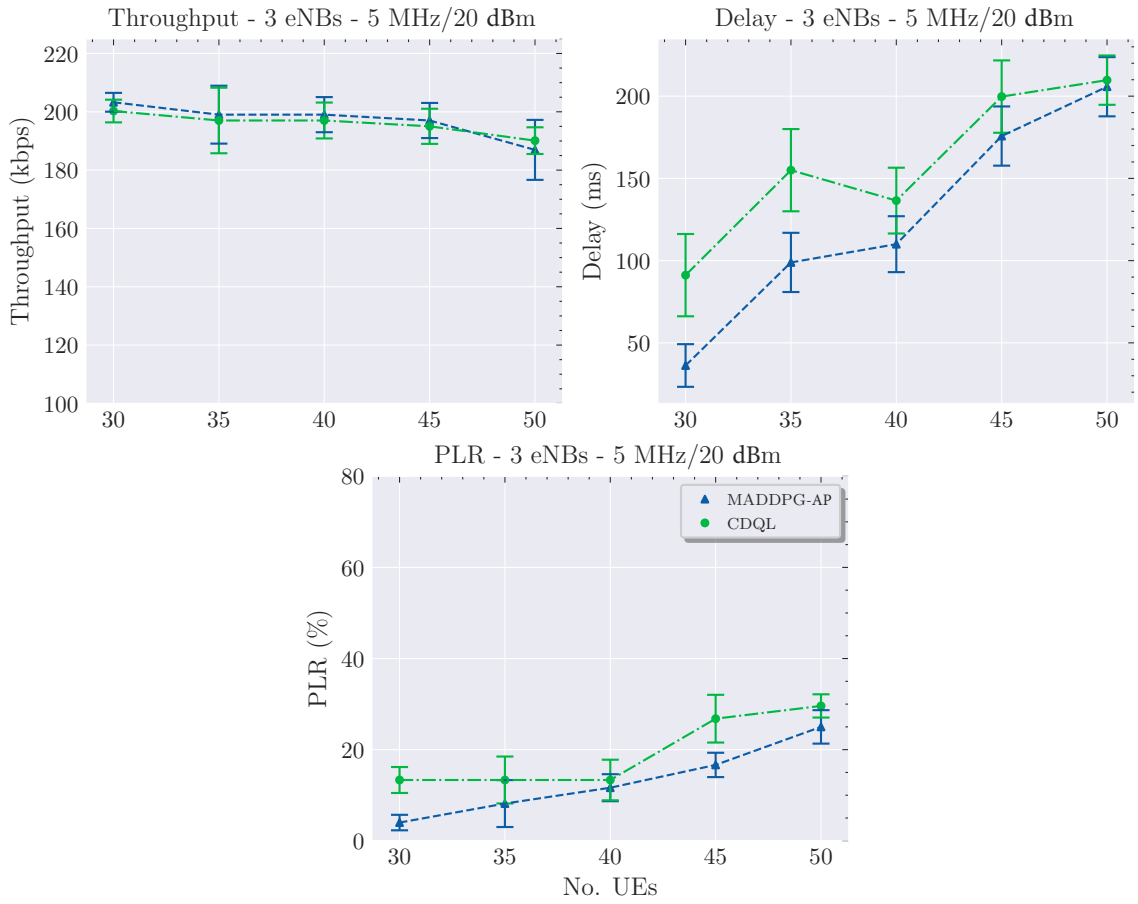

Figure 3.10: Performance metrics of MADDPG-AP and CDQL algorithms. (a) Throughput, (b) Delay and (c) PLR

$$\omega_1(t) \triangleq \frac{1}{N_T} \sum_{i=1}^{N_T} \mathbb{1} \cdot \delta_i(t) \tag{3.21}$$

and,

$$\delta_i(t) = \begin{cases} -1 & \text{if } \exists \Psi_i \notin \Psi_C, \\ \urcorner(D_{avg}) & otherwise \end{cases} \tag{3.22}$$

where Ψ_i represents the i^{th} UE and Ψ_C, $\Psi_C \subset \Psi$ is the set of connected UEs to any BS $\in \Gamma$. On the other hand, $\urcorner(D_{avg})$ is a sigmoid function defined as follows:

$$\urcorner(D_{avg}) = 1 + \frac{c}{1 + e^{-o(D_{avg} - \mathcal{F})}} \tag{3.23}$$

Here c establishes the upper bound of the slope, o adjusts the slope of the sigmoid, and $\mathcal{F} = 2/3 * PDB$ controls the target packet delay. PDB is the Packet Delay Budget according to the type of traffic used in the network.

The objective of ω_1 itself is to reward each UE's average latency based on the Packet Delay Budget (PDB) of a defined packet type. A penalization is applied if a UE is found disconnected from the network due to a load-balancing decision.

Finally, $\omega_2 \in [0, 1]$ can be defined as:

$$\omega_2(t) \triangleq 1 + \frac{c}{2 + 2e^{-a(p(t) - \mathcal{D})}} \tag{3.24}$$

where $p(t)$ corresponds to the resource block utilization measured value by the BS at time t, as $p(t)$ decreases, a higher reward is obtained. c establishes the upper bound of the slope, a adjusts the slope of the sigmoid as in equation 3.24 and $\mathcal{D}$ controls the target resource block utilization.

3.2.3 Baseline: Clipped Double Q-Learning (CDQL)

For the present work, as a baseline, we consider a single and centralized reinforcement learning algorithm named CDQL that was proposed in [25]. This technique showed an improvement in throughput, latency, jitter, and PLR when compared to traditional handover algorithms. The CDQL presents two main differences with our current work: *i*) It considers a single and centralized agent with a deterministic action space and *ii*) It does not consider maximizing throughput in its reward design.

Table 3.3: Network settings

Parameter	**Value**
BS Inter-site Distance	720 m
M_T	3
N_T	30,35,40,45,50
Center Frequency	2 GHz
System Bandwidth	5 MHz (25 resource blocks)
Pathloss Model	Log Distance Propagation Loss Model
	95 + 27 $log_{10}(distance[km])$
BS antenna height	30 m
UE antenna height	1.5 m
Max Tx power	20 dBm
MAC scheduler	CQA scheduler
User distribution	Stationary and uniformly distributed
Traffic Model	Conversational video (live streaming) and Poisson
	Packet payload size = 250 Bytes
	Interval = 10 ms
	Packet delay budget = 150 ms
Handover algorithm	A3-event based
	Time to trigger = 8 ms
	Hysteresis = 2 dBm

3.2.4 Performance Evaluation

Simulations are implemented using the discrete network simulator ns-3. Additionally, OpenAI Gym is utilized to interface the ns-3 wireless environment and our proposed algorithm. In Table 3.3 and Table 3.4 the settings used in our simulations and the RL parameters are

Table 3.4: Learning parameters

Parameter	**Value**
Training stage (Number of iterations/episode)	50
Number of episodes	300
Execution stage (Number of iterations/episode)	50
Number of episodes	150
Gym environment step time	0.2s
Batch size	100
MADDPG-AP	$\phi_{\in}[-9, 9]$ dBm $K = 3$ $\mathcal{F} = 2/3 * PDB$ where $PDB = 150$ ms, $c = -2$, $o = 75$ $\gamma_{RB} = 0.8$, $a = 20$ Optimizer : Actor: Adam (1e-4), Critic: Adam (1e-3) Number of hidden layers (N_h) : 2 Number of neurons/layer (n_l) : 128 Update target model type: Polyak averaging $\gamma = 0.95, \epsilon = 1.0, \epsilon_{min} = 0.001, \epsilon_{decay} = 0.995$

given, respectively. The inter-site distance between BS is set to 720 meters. Five different scenarios are tested under the proposed algorithms with 30, 35, 40, 45, and 50 UEs. The UEs are distributed on the edge of each BS in a random disc and the rest is uniformly allocated throughout the coverage of the middle BS.

3.2.5 Simulation Results

We present the performance results of our proposed scheme in terms of throughput, delay, and packet loss ratio (PLR) with a 90% confidence interval. Figures 3.10 (b and c) show a positive change for the baseline with a gain of 20.89% in terms of delay and 32.1% in

Table 3.5: Convergence comparison

RL Scheme	**Avg. Convergence in Episodes**	**Avg. Improvement**
CDQL	61 (12.2s)	-
MADDPG-AP	**18** (**3.6**s)	70.49%

terms of PLR. No evident improvement was observed in terms of throughput, as seen in Fig. 3.10 (a).

Figure 6.3 presents a convergence comparison for the 30 UEs scenario for the MADDPG-AP and CDQL algorithms where an improvement of up to 60 episodes is achieved. Finally, Table 3.5 shows an average convergence time over all scenarios with an improvement of 70.49% for MADDPG-AP concerning the CDQL scheme.

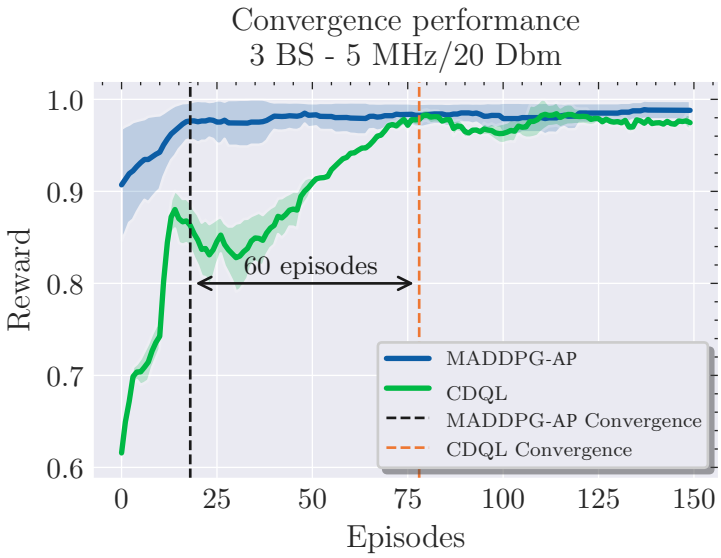

Figure 3.11: Convergence performance for the 30 UEs scenario for the CDQL and MADDPG-AP schemes.

3.3 Hierarchical Deep Q-Learning Based Handover in Wireless Networks with Dual Connectivity

With the deployment of 5^{th} Generation New Radio (5G) and the existing 4^{th} Generation LTE (4G) technologies, mobile users with LTE or 5G capabilities must be able to seamlessly adapt to the dual existent infrastructure. Back in 2013, the 3rd Generation Partnership Project (3GPP) proposed dual connectivity (DC) architectures in [137] that allowed master eNodeBs (eNBs) and secondary eNodeBs to share partially their IP layer in a dual connection manner to maximize network performance. A more recent technical report [99] proposed a generalization of multi-radio dual connectivity architectures to support LTE and 5G users. Dual connection architectures take advantage of the technologies involved to improve KPI of interest. Concurrently, the stringent requirements of diverse services in terms of latency and throughput, such as URLLC and eMBB have demanded a more optimized parameter tuning in any wireless mechanism utilized.

Reinforcement Learning (RL) techniques have been widely recognized for their effectiveness in the autonomous learning context. More specifically, RL has sparked the wireless network community's attention [9] since optimized parameter learning has become a challenge in such dynamic environments.

In this work, we address the handover problem in an LTE-NR network with dual connectivity using a hierarchical and a non-hierarchical architecture. Additionally, we consider the context information to further improve the DC algorithm performance. To do so, we use a novel hierarchical Deep Q-learning algorithm named hierarchical Deep Q-Learning (HiDQL) and a non-hierarchical RL approach named CDQL to improve handover latency in a DC architecture. HiDQL presents an architecture comprised of a meta-controller and a controller. Two parameters are adjusted in this problem: Threshold to Time (TTT) and signal-to-interference-plus noise ratio (SINR) outage. TTT is defined as the time to hold the condition of triggering the handover algorithm named Secondary Cell Handover (SCH), discussed in subsection 3.3.1. On the other hand, an SINR outage is defined as the SINR threshold in which we consider a cell being in an outage from the User Equipment (UE). The controller will propose actions in terms of TTT and SINR outage aided by the proposed goals by the meta-controller to minimize handover latency. We compare our results with an algorithm presented in [25] named CDQL that proved its efficiency in load balancing problems. Our results show an improvement in terms of latency with a gain of 47.6% and 26.1% for Digital-Analog beamforming (BF), 17.1% and 21.6% for Hybrid-Analog BF, and 24.7% and 39% for Analog-Analog BF when comparing the RL-schemes with the baseline's best results. Additionally, we observed faster convergence when uti-

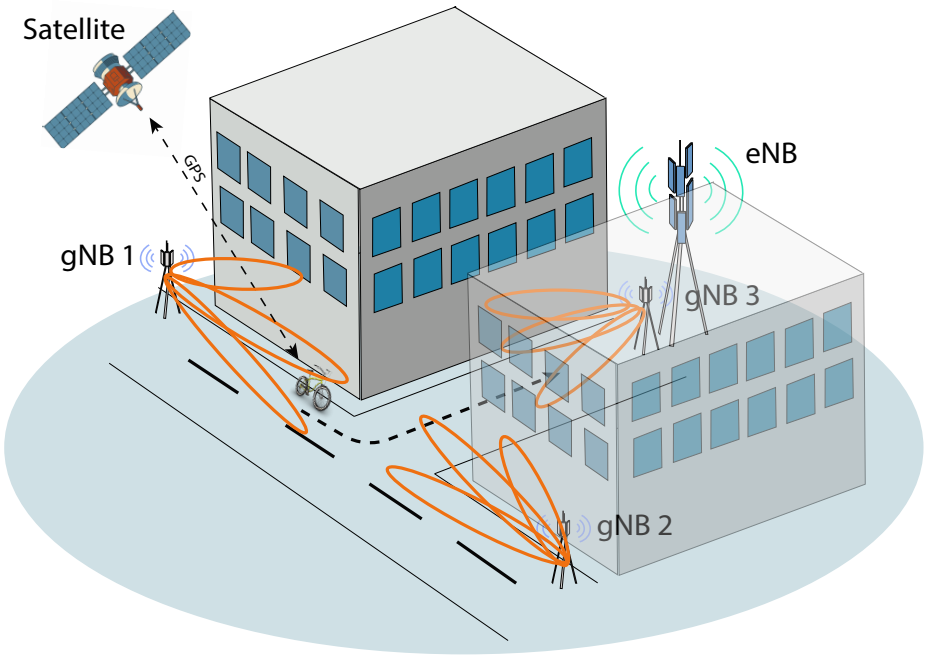

Figure 3.12: Overview of the scenario: a UE uses GPS context information and dual connectivity to improve handover latency in an urban scenario.

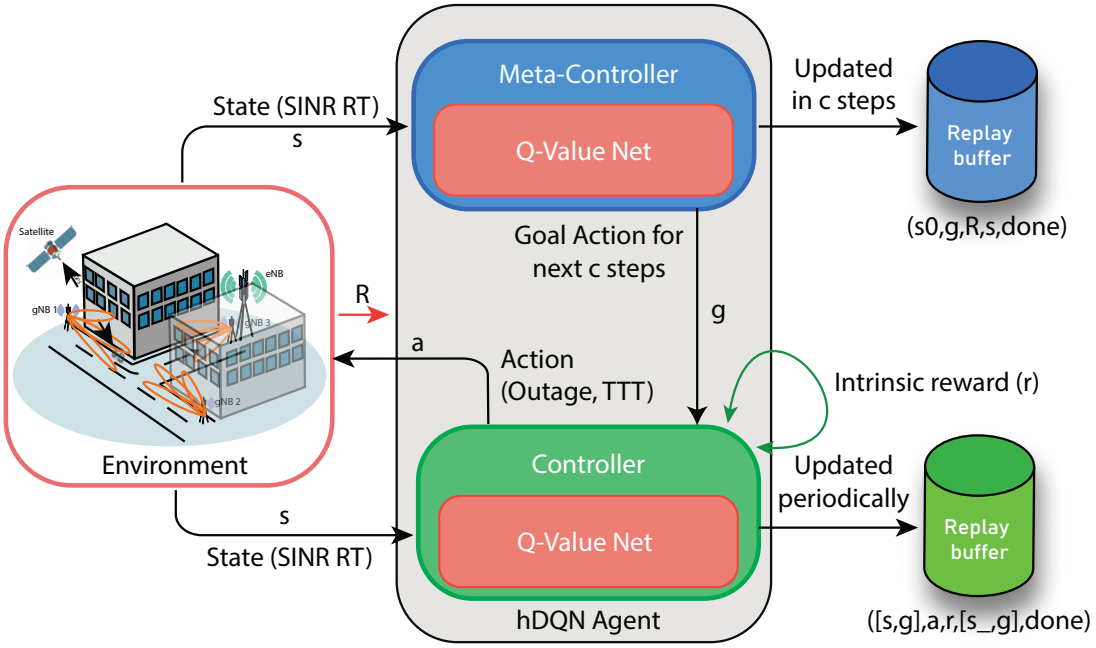

Figure 3.13: Hierarchical Deep Q-Learning (HiDQL) applied in a Multiple Radio Access Technology (multi-RAT) LTE-NR scenario. The HiDQL agent located in the coordinator will optimize TTT and SINR outage metrics to reduce multi-RAT handover latency.

lizing the CDQL algorithm, meanwhile, the HiDQL showed improved results in terms of handover latency. Finally, we obtained an improvement regarding handover latency under line-of-sight assumptions when considering context information in comparison with no context available.

3.3.1 System model

In this work, we use a DC architecture presented in [27] inspired by 3GPP's previous DC proposal [137]. In [27], the authors leverage the usage of a DC framework to improve, among others, handover latency by using an algorithm named Secondary Cell Handover (SCH). SCH enables fast switching between the LTE and 5G RATs and is managed by a defined entity named the coordinator. In this architecture, the eNB takes the role of coordinator and controls the multi-RAT switch. Two main parameters are controlled by the coordinator: TTT and SINR cell outage. As part of the SCH algorithm, the coordinator will trigger the SCH when any of the RATs involved report a better SINR and neither of

the RATs is in an outage. Additionally, it will check the condition to trigger the SCH algorithm for TTT seconds to avoid ping-pong scenarios. This algorithm resembles the classical LTE and 5G handover with the exception that no initial access is necessary and no interaction with the Mobility Management Entity (MME) is required thanks to the nature of the DC architecture. As mentioned, the handover will be triggered based on SINR and specifically using a table named the Complete Report Table (CRT) comprised of the UE's SINR report tables from each gNodeB (gNB). Each gNB will perform a sweep over a number of predefined directions and will sense the Sounding Reference Signals from the UE's to obtain SINR measurements. The collected data will be sent via X2 interface to the coordinator. The delay incurred to perform the measurement sweeps is directly related to the beamforming (BF) technology used by the UEs and gNBs. The aforementioned delay, D, is calculated as:

$$D = \frac{N_{gNB} N_{UE} T_{per}}{L}, \tag{3.25}$$

where N_{gNB} and N_{UE} correspond to the number of required sweep directions needed for measurement collection by the gNBs and UEs, respectively. T_{per} is the Sounding Reference Signal (SRS) periodicity and L corresponds to the BF capabilities that will present different values if the transceiver is fully digital or analog. The relationship between D and L can be calculated using typical values as depicted in Table 3.6 (Modified from [27]).

Table 3.6: Relationship between L and D

gNB-UE BF	L **(gNB/UE)**	D^***(ms)**
Analog-Analog	1/1	25.6
Hybrid-Analog	2/1	16.8
Digital-Analog	N_{gNB}/1	1.6

* D in (3.25) is calculated with $T_{per} = 200\mu s$, $N_{gNB} = 16$ and $N_{UE} = 8$

We consider an LTE-NR network with dual connectivity and consisting of M_T gNBs. Additionally, an eNB will act as a coordinator thus, serving the M_T gNBs. A mobile user is dual-connected to one gNB and one eNB at the same time. We consider an urban scenario with two buildings as shown in Fig. 3.12. The channel considered for the eNB is the 3GPP Channel Model, meanwhile, the 3GPP building channel type and losses based on the 3GPP UMi Street Canyon propagation model are used for the gNBs.

3.3.2 Hierarchical Deep Q-Learning (HiDQL)

In this work, we use a state-of-the-art hierarchical Deep Q-learning (HiDQL) algorithm. This algorithm is presented in [58], where the goal-directed behavior is studied under some specific sparse reward problems such as the Montezuma Revenge and a complex discrete stochastic decision process.

As shown in Fig. 3.13, a hierarchical agent located in the coordinator observes the user's SINR report table and receives the extrinsic reward based on the feedback from the environment in terms of handover latency. This agent will adjust key parameters such as TTT and SINR outage to maximize the agent's extrinsic reward. In the following subsection, we will present an overview of HiDQL and then formally define our solution.

Meta-controller and controller action space selection

In the proposed hierarchical approach described in Algorithm 3.6, the meta-controller proposes goals or high-level actions when the maximum c steps, c_{max} is achieved or when the similarity between the goal and low-level action, $\mathcal{S}_\kappa$ falls under a predefined threshold index κ as defined in the function `Done`. Meanwhile, the controller's low-level actions are executed c inner steps. For both levels, the actions are defined in the same fashion as:

$$A(t) = \left[a_{out}(t), a_{ttt}(t)\right], \tag{3.26}$$

where a_{out} and a_{ttt} are the SINR outage and TTT values, respectively. Such values are discretized into K_o and K_{TTT} levels according to the maximum and minimum defined values. The possible values for a_{out} and a_{ttt} are defined as:

$$A_{out} = \{O_{min}, O_{min} + \frac{O_{max} - O_{min}}{K_o - 1}, ..., O_{max}\}, \tag{3.27}$$

$$A_{ttt} = \{TTT_{min}, TTT_{min} + \frac{TTT_{max} - TTT_{min}}{K_{TTT} - 1}, \cdots, TTT_{max}\}, \tag{3.28}$$

where O_{min}, O_{max}, and TTT_{min}, TTT_{max} are predefined maximum and minimum values of SINR outage and TTT values, respectively. Finally, the size of the meta-controller and controller action spaces become $S = |A_{out}| * |A_{ttt}|$.

Algorithm 3.6 Hierarchical Deep Q-Learning (HiDQL)

Initialize c_{max}, κ, experience replay buffers $\{B_{mc}, B_c\}$, policy networks $\{\mu_{mc}, \mu_c\}$, for the meta-controller and controller, respectively.
Function Done (c, c_{max}, $\mathcal{S}_\kappa$, κ):
if ($c = c_{max}$ **or** $\mathcal{S}_\kappa < \kappa$) **then**
  return True
else
  return False
end if
End Function
for environment step $t \leftarrow 1$ **to** T **do**
  $c \leftarrow 0$
  Init environment and initial state
  Execute goal from meta-controller: $g = \mu_{mc}(s)$
  while c_{max} **not** reached **do**
    $R \leftarrow 0$; $s_0 \leftarrow s$; $\mathcal{S}_\kappa =$ **false**
    while not `Done` **do**
      Execute action from controller: $a = \mu_c(s, g)$
      Get next state s' and similarity condition $\mathcal{S}_\kappa$
      Calculate intrinsic reward r and receive extrinsic reward r_g
      Store $([s, g], a, r, [s', g],$ `Done`$)$ in replay buffer B_c
      Update μ_c
      $c := c + 1$; $R := R + r_g$; $s := s'$
    end while
    Store $(s_0, g, R, s,$ `Done`$)$ in replay buffer B_c;
    Update μ_{mc};
    if not `Done` **then**
      Execute goal from meta-controller: $g = \mu_{mc}(s)$
    end if
  end while
end for

State space selection

The state space is comprised of the Complete Report Table (CRT), which contains the relationship in terms of the SINR of each user and the perceived gNB SINR. The SINR

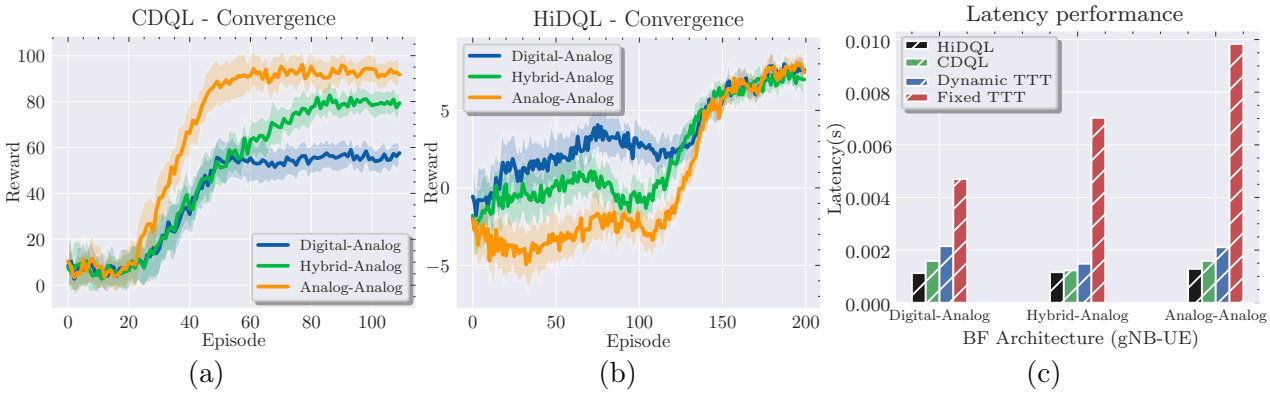

Figure 3.14: (a) CDQL convergence performance. (b) HiDQL convergence performance. contain. (c) Latency performance of the baselines Dynamic TTT and Fixed TTT and the proposed RL schemes CDQL and HiDQL.

between an i^{th} gNB and n^{th} UE is calculated as follows:

$$SINR_{i,n} = \frac{\frac{P_{TX}}{PL_{i,n}} G_{i,n}}{\sum_{k \neq i} \frac{P_{TX}}{PL_{k,n}} G_{k,n} + W_{tot} \times N_0}, \tag{3.29}$$

where $G_{i,n}$ and $PL_{i,n}$ are the beamforming gain and the pathlosss obtained between i^{th} gNB and n^{th} UE. P_{TX} is the transmit power and $W_{tot} \times N_0$ is the thermal noise power. Thus, the state space is defined as follows:

$$S(t) = \left[\boldsymbol{C}(t) \right], \tag{3.30}$$

where $\boldsymbol{C}(t)$ corresponds to the CRT with a size of $M_T = |\boldsymbol{C}(t)|$.

Intrinsic and extrinsic reward

The intrinsic reward function is calculated by the controller by taking into account the goal selected by the meta-controller in c inner steps. Such a reward is defined as:

$$r(t) = r_i(t) + r_{l^2}(t), \tag{3.31}$$

where r_i corresponds to the immediate reward obtained from the environment and r_{l^2} corresponds to the reward based on the l^2-norm between the controller's action and the meta-controller's goal. Such rewards are defined as follows:

$$r_i(t) = \begin{cases} -f_{-1}(-D^u_{avg}) & \text{if } D^u_{avg} < D^{BF}_{tol}, \\ 0 & otherwise \end{cases} \tag{3.32}$$

where $f_{-1}(\cdot) : [-e^{-1}, 0) \rightarrow [-1, -\infty)$ is the lower branch of the Lambert function $y = f_{-1}(ye^y)$ [138]. D^{BF}_{tol} is the maximum tolerable delay corresponding to the value of the best latency results per BF technology in the non-RL approaches. D^u_{avg} is the average latency of the UE. Additionally, we scale linearly $D^u_{avg} \rightarrow [e^{-1}, 0]$.

$$r_{l^2}(t) = \begin{cases} 1 & \text{if } \mathcal{S}_\kappa < \kappa, \\ -1 & otherwise, \end{cases} \tag{3.33}$$

where $\mathcal{S}_\kappa = |a - g|_2$ and κ is the similarity threshold between the action and goal. Finally, we can express the extrinsic reward as:

$$R_T = \sum_{j=t-c}^{c} r(j), \tag{3.34}$$

where the extrinsic reward is defined as the sum of the intrinsic reward during the c inner steps.

3.3.3 Clipped Double Q-Learning: RL scheme

In addition to HiDQL, we utilized an RL solution named CDQL [29]. For the CDQL algorithm, the state and action space correspond to the ones used in the lower level by the HiDQL algorithm in (3.26) and (3.30) respectively, meanwhile, the reward corresponds to the intrinsic reward defined in (3.32).

3.3.4 Baselines: Fixed TTT and Dynamic TTT

For the present work, as baselines, we take a fixed and a dynamic solution into consideration. The fixed and dynamic solutions were proposed in [27], where the fixed solution has a fixed value of TTT and in the dynamic solution TTT is calculated as follows:

$$f_{TTT}(\boldsymbol{\Delta}) = TTT_{max} - \frac{\boldsymbol{\Delta} - \boldsymbol{\Delta}_{min}}{\boldsymbol{\Delta}_{max} - \boldsymbol{\Delta}_{min}}(TTT_{max} - TTT_{min}), \tag{3.35}$$

where $\boldsymbol{\Delta}$ corresponds to the difference between the serving cell and the best gNB in terms of SINR.

3.3.5 Performance Evaluation

Simulation Settings

We implement our proposed solution using the discrete network simulator ns-3 and its module mmWave [139]. Additionally, ns3-ai module [140] is used to interface the simulation environment to our RL algorithm written in Python 3.9. Simulation settings and RL parameters are depicted in Table 3.7 and Table 3.8, respectively.

3.3.6 Simulation Results

We present the performance results of our proposed scheme in terms of convergence and handover latency. Figure 3.14 (a), (b) depicts the convergence behavior for the CDQL and HiDQL algorithms for different BF mechanisms, respectively. It can be observed that CDQL presents a faster convergence when compared with HiDQL. However, converged reward values are not achieved equally among the tested BF architectures. HiDQL presents a slower convergence time than CDQL however with a similar maximum convergence reward value among BF methods. The intuition behind the previous results corresponds to the inner nature of hierarchical learning. HiDQL performs an iterative search through the meta-controller's proposal of goals to the controller and thus, obtaining optimum action selection in challenging scenarios as described in [58]. The difference between reward scales between HiDQL and CDQL corresponds to how the reward is defined in both algorithms, thus a comparison in terms of the value of the reward is not relevant.

In Fig. 3.14 (c), we present the latency performance of our techniques and the baselines: fixed TTT (F-TTT), and dynamic TTT (D-TTT). From the two baselines, the best performance in terms of latency is achieved by the dynamic TTT parameter selection. On the other hand, we obtained an average 9% improvement when comparing HiDQL and CDQL in terms of latency results. However, based on more detailed look into results, we further focus in Table 3.9 on the latency results improvements over the previous solutions F-TTT and D-TTT.

As shown in Table 3.9, when compared with the dynamic TTT results, a gain of 47.6% and 26.1%, for Digital-Analog BF, 17.1% and 21.6%, for Hybrid-Analog BF, and 24.7% and 39% for Analog-Analog BF were obtained for HiDQL and CDQL, respectively.

Table 3.7: Network settings

Parameter	**Value**
M_T	3
Number of UEs	1
gNB center frequency	28 GHz
gNB system bandwidth	200 MHz
gNB numerology	2
gNB Pathloss Model	3GPP Umi-Street Canyon
gNB Channel condition model	Buildings
gNB antenna height	10 m
gNB number of antennas	64
eNB Center Frequency	2 GHz
eNB System Bandwidth	100 MHz
eNB Pathloss Model	3GPP
Max Tx power	30 dBm
UE number of antennas	16
UE antenna height	1.6 m
UE speed	13 m/s
UE Tx power	10 dBm
Traffic Model	On-Off UDP application Packet payload size = 512 Bytes Packet window size = 256 Interval = 20 μs

Table 3.8: Learning parameters

Parameter	Value
Handover algorithm	Threshold to time: $TTT_{max} = 150$ ms; $TTT_{min} = 25$ ms
	SINR Outage: $O_{max} = -3$ dBm; $O_{min} = -8$ dBm
	$D_{tol}^{BF} = min(D_f^{BF}, D_d^{BF})$ *
Gym environment step time	Event-based
Batch size	32
	Number of hidden layers (N_h) : 2
	Number of neurons/layer (n_l) : 64
HiDQL	$\kappa = 0.25$
	Optimizer : Meta-Controller: Adam (1e-4),
	Controller: Adam (1e-4)
c_max	50 steps
CDQL	Update target model type: Polyak averaging
	$\gamma = 0.95, \epsilon = 1.0, \epsilon_{min} = 0.001, \epsilon_{decay} = 0.995$
	Optimizer : Adam (1e-4)

*D_f^{BF} and D_d^{BF} correspond to the latency performance per BF technology of the fixed TTT and dynamic TTT solutions, respectively.

Context information aware DC handover

In addition to the SCH algorithm used as part of the proposed DC architecture, we foresee the possible advantages of using context information to improve handover time. In [141], the authors give an overview of the typical initial access (IA) cell search algorithms where contrary to LTE, 5G NR provides mechanisms to determine suitable initial directions of transmission to avoid high isotropic losses. Among the IA algorithms, context information (CI)-based algorithms allow users to be aware of the location of the surrounded 5G NR antennas through an LTE link and GPS. In this work, we leverage GPS information as

Table 3.9: RL vs non-RL latency comparison

	Latency improvement (%)					
	Digital-Analog BF		Hybrid-Analog BF		Analog-Analog BF	
RL Scheme	F-TTT	D-TTT	F-TTT	D-TTT	F-TTT	D-TTT
HiDQL	76.1	47.6	83.5	21.6	87.0	39.0
CDQL	66.2	26.1	82.5	17.1	83.8	24.7

Table 3.10: CI vs. non-CI latency comparison

	Latency improvement (%)		
	Digital-Analog BF	Hybrid-Analog BF	Analog-Analog BF
CI	Non-CI		
F-TTT	74.9	2.57	28.5
D-TTT	40.1	8.7	29.7

context and reduce the sector in which a gNB should sense the space towards a UE. Figure 3.15 depicts the beforementioned mechanism, where given the user position $P1$ the gNB can calculate a reduced sector to obtain the periodic SRS signals from the surrounded UEs. We also assume that each UE has a line of sight both with the coordinator and the gNBs. The proposed mechanism is described in Algorithm 3.7. As shown in Table 3.10, context awareness directly improves the DC handover mechanism with a considerable latency improvement.

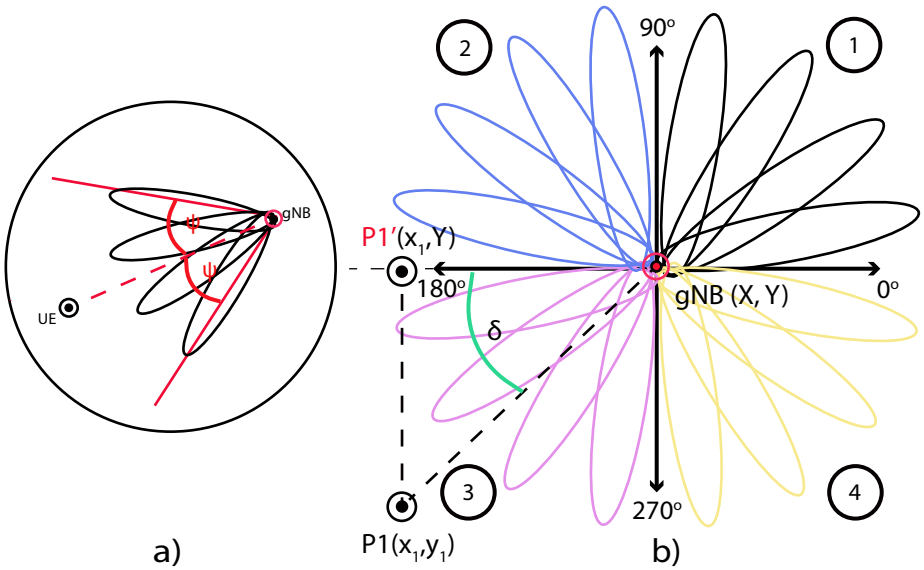

Figure 3.15: a) $\pm\psi$ indicates the sweep angle to obtain SINR measurements. b) δ corresponds to the relative angle used to derive the sweep sector angle.

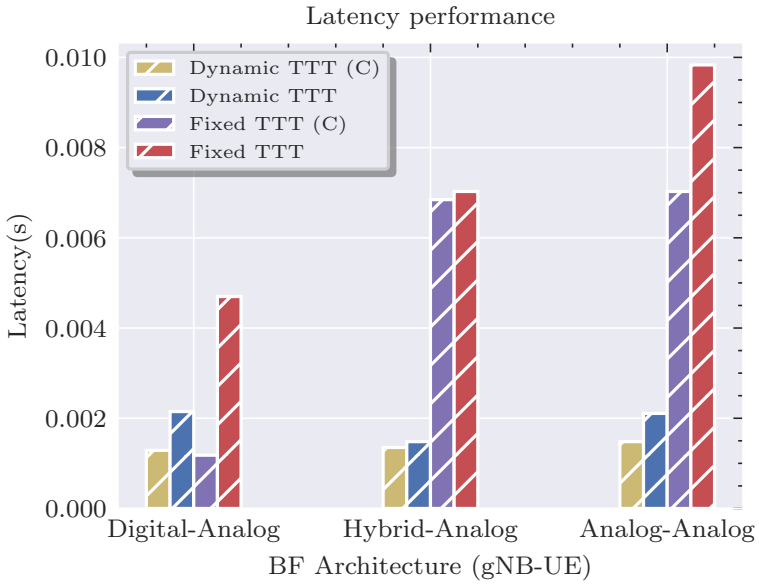

Figure 3.16: Context information (C) latency performance results when utilizing Dynamic TTT and Fixed TTT baselines.

Algorithm 3.7 CI-aware sector sweep selection algorithm

Result: Sector's angle to perform measurement sweeps.
Given P1(x_1, y_1) and (X, Y), the UE's and gNB geolocations, respectively.
Additionally, $\delta = \arcsin \frac{\vec{d_2}}{\vec{d_1}}$ and $\Theta^{gNB} = \frac{2\pi}{N}$; where
$\vec{d_2} = \overrightarrow{P_1 P_1'}$ and $\vec{d_1} = \overrightarrow{P_1 gNB}$
# Step 1: Detect UE's relative quadrant $q \in \{1, 2, 3, 4\}$. Calculate angle ϕ respect gNB quadrant one.
if $y_1 > Y$ and $x_1 > X$ **then**
    #UE in 1^{st} quadrant
    $\phi = \delta$
else if $y_1 > Y$ and $x_1 < X$ **then**
    #UE in 2^{nd} quadrant
    $\phi = \frac{\pi}{2} + \delta$
else if $y_1 < Y$ and $x_1 < X$ **then**
    #UE in 3^{rd} quadrant
    $\phi = \pi + \delta$
else
    #UE in 4^{th} quadrant
    $\phi = \frac{3\pi}{2} + \delta$
end if
# Step 2: Calculate sector angle (S_A)
$S_A \in [\phi - 2\Theta, \phi + 2\Theta]$

Chapter 4

Multi-Agent Team Learning in Virtualized Open Radio Access Networks (O-RAN)

The demand for mobile connectivity has been growing over the past decades, including a parallel increase in the demand for better Quality of Service (QoS). On top of that, 5G and the next generations of mobile networks will not only serve smartphone users but also verticals like self-driving cars, healthcare, manufacturing, gaming, marketing, Internet of Things (IoT) and many more [142]. Almost in all generations of mobile networks, resource optimization has been a challenge, yet with 5G, despite new spectrum allocations in the mmWave band, the spectrum is still scarce due to increasing demand for wireless connectivity due to emerging new applications such as virtual reality, remote surgery, and so on [143]. Moreover, starting from Long-Term Evolution (LTE), the densification trend continues with 5G, at the cost of increased investments. It is well-known that densification is essential due to the limited coverage of mmWave bands. Hence, the increasing complexity of mobile networks is reaching the limits of model-based optimization approaches and yielding to data-driven approaches, also known as AI-enabled wireless networks [9]. In this context, the interest in self-optimizing networks that use machine learning is growing [144]. In the meanwhile, an interesting nexus is emerging between AI-enabled networks and RAN disaggregation, virtualization, and open interfaces.

In the 2010s, Cloud RAN (C-RAN) introduced a disaggregated architecture that allowed the grouping of Baseband Units (BBUs) into a centralized BBU pool, potentially reducing deployment costs in the long term and improving network capacity. Other architectures have continued from the path of C-RAN, such as virtualized RAN (vRAN)

which adds the concept of hardware functions being softwarized. RAN disaggregation and virtualization offer many advantages over traditional RAN solutions, such as reduced network deployment time, improved energy efficiency, and enhanced mobility support. Most recently, the Open Radio Access Network (O-RAN) Alliance has brought together those prior efforts around a new RAN architecture with open interfaces, aiming to replace some of the interfaces that have become proprietary over time due to vendor-specific implementations of standards. Open interfaces are considered to be important for equipment from multiple vendors to be stacked together and for the selection of best-of-breed solutions.

The timing of O-RAN coincides with new advances in Software Defined Networking, Network Function Virtualization, dynamic function splitting, high-capacity data centers, and cloud and edge computing, all of which have set the stage for O-RAN [145]. Meanwhile, the increased complexity and the flexible architecture of the O-RAN specifications call for the use of machine learning techniques more than ever.

As of 2021, there are many implementations of open RAN around the globe. For instance, in Japan, Rakuten has deployed distributed data centers hosting both Distributed Unit (DU) and Central Unit (CU) functions. The Spanish operator Telefónica has invested in the integration of its open RAN solution with its multi-access edge computing (MEC) solution. In addition, some operators such as Dish, Sprint, and T-Mobile in the US and Etisalat in the middle-east are in the testing phase of open RAN. There are many other emerging vendors, system integrators, and operators who are experimenting with open RAN, in addition to the above-mentioned players in the market. It is worth noting that, some of these implementations are not necessarily following O-RAN specifications defined by O-RAN Alliance.

The multitude of layers, functions, splits, and the consequent complexity in O-RAN, position machine learning techniques that involve multiple agents and team learning as potential solutions. In recent years, MASs have been used to address complex problems in robotics and other self-driving systems therefore they can be promising alternatives for the self-driving (or self-optimizing) capability of O-RAN. In team learning, distributed agents form teams that cooperate to reach a common goal. The utmost goal of this aggregation is to maximize the team's objective function. The network function softwarization capabilities of O-RAN present an excellent opportunity to utilize teams of distributed agents as cooperating xApps to improve main key performance indicators while sharing partial or complete information on observations. To the best of our knowledge, this work is the first to elaborate on multi-agent team learning (MATL) and its use in O-RAN. Note that, providing a survey on MAS or a description of O-RAN specifications is beyond the scope of this study. In the literature, there are several comprehensive surveys on MAS [146] and O-RAN specifications [147–149]. This work focuses on the application of MATL in

O-RAN. In subsection 4.0.1 we continue with an introduction to MAS and team learning. In subsection 4.0.2, we discuss a case study where MATL is employed in O-RAN. More specifically, we study three schemes for team learning where each team is comprised of two xApps for optimizing power allocation and resource allocation. Finally, in subsection 4.0.5 we provide a detailed discussion of the challenges of MATL in O-RAN and identify open issues in the field.

4.0.1 Embedding Intelligence in O-RAN

Multi-agent team learning presents itself as an exciting technique where agents can join forces to achieve a common goal. Modular disaggregation and feedback availability throughout the O-RAN architecture intuitively leads us to the applicability and integration of team learning in O-RAN. To integrate MATL into O-RAN and develop an AI-enabled RIC, some steps need to be taken.

In the first step, model capabilities and its required resources, including the accessibility of data (observation level), availability of processing power, interaction time, and maximum tolerable latency (processing time and communication time), need to be examined. This evaluation can be executed by the service management and orchestration (SMO) platform. The corresponding RAN counters from O-CUs and O-DUs are gathered in the data collector, which is placed in the SMO. Based on the SMO platform type, various entities can perform the data collection. For instance, if an open network automation platform (ONAP) is employed to play the role of SMO, the virtual event streaming (VES), which is a subsystem in the ONAP will take care of collecting data collection, analytics, and events (DCAE). After collecting analytics, the SMO shares the collected data through a data bus with non-RT RIC. Selecting a proper model and training the model is the next step, where AI agents run and train a model at non-RT RIC, near-RIC, or even O-DU based on the NF resource and latency requirements.

Based on the model's outcome, agents apply the corresponding actions. Depending on the AI model, type of actions, and location in O-RAN architecture, various interfaces, including A1, O1, and E2, can be used. A1 interface can be used for policy-based data format, ML/AI model management, and delivering information to the near-RT RIC to apply RAN optimization. Similarly, the E2 interface is located between O-CU, O-DU, and near-RT RIC. Additionally, the O1 interface is responsible for delivering management and operational information on O-DU, O-CU, near-RT RIC, and radio unit (O-RU), including performance, configuration, and fault management, to the SMO. It can also be used for the configuration of O-DU, O-CU, near-RT RIC, and O-RU to the application.

When intelligence is implemented in O-RAN, using a team learning approach, it is important to select a common goal and the sub-goals properly. In addition, the overhead of sharing feedback needs to be considered. In the next section, we present a study case showing the advantages of MATL in the O-RAN architecture.

4.0.2 A Case Study on Multi-agent Team Learning in O-RAN

As a case study, we investigate the advantages of using MATL for optimizing resource block and power allocation in O-RAN. Prior studies have shown reinforcement learning-based resource block and power allocation outperform traditional approaches [150]. Therefore, we focus on only machine learning approaches and extend them with the notion of team learning. In our case study, for simplicity, we consider a team of two heterogeneous xApps: one optimizing power allocation and another xApp optimizing resource block allocation.

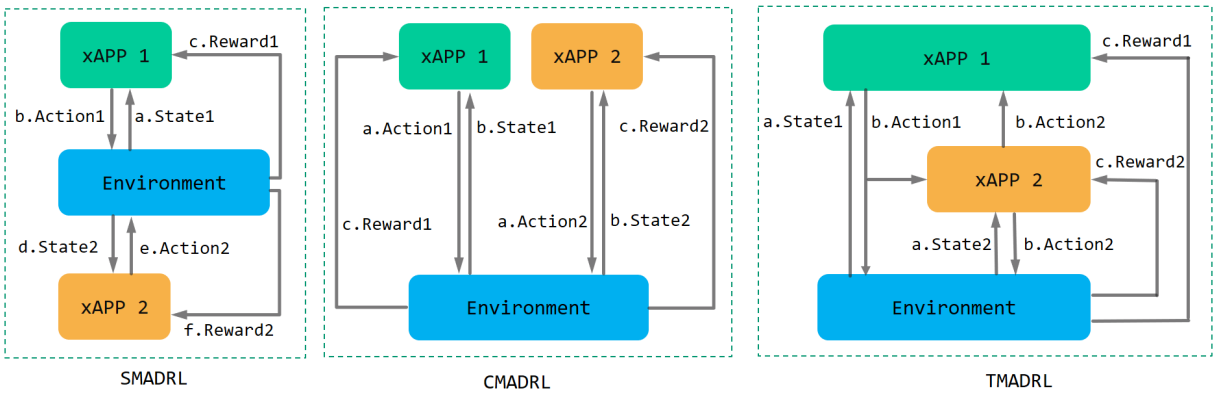

Figure 4.1: Sequential multi-agent deep reinforcement learning (SMADRL), Concurrent multi-agent deep reinforcement learning (CMADRL), and Team multi-agent deep reinforcement learning (TMADRL) schemes.

The core idea is to let xApps take into account the actions of other team members. This is achieved by changing the state of the xApps to a combination of both environmental information and action information from other team members. In addition, xApps also inform other team members about the actions they decide to take. By exchanging information, team members consider the actions of others and hence improve the team member's coordination as well as the overall performance of the RAN. We define our performance metrics as system throughput and energy efficiency. As shown in Fig. 4.1, we propose three schemes to study the interactions among agents and the repercussions of exchanging information. We define our schemes as follows:

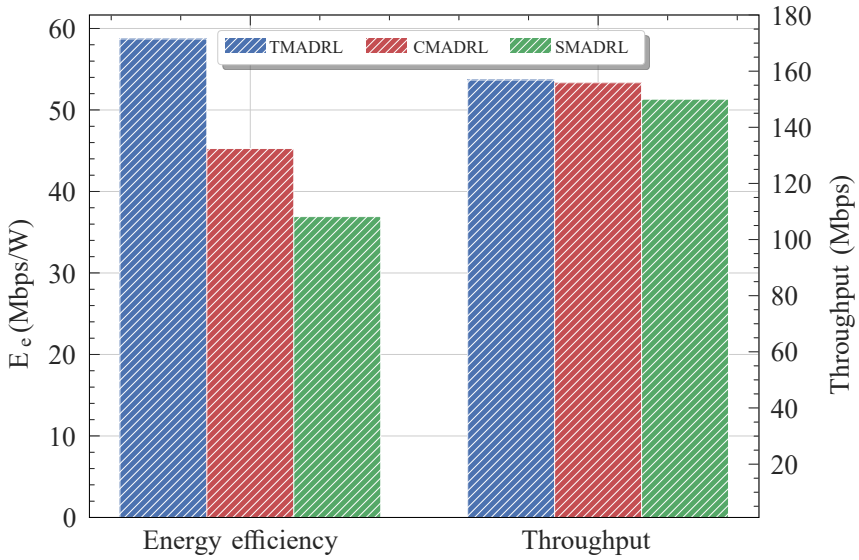

Figure 4.2: Average energy efficiency (E_e) and throughput performance metrics for the TMADRL, CMADRL, and SMADRL schemes.

- **SMADRL**: Team members act sequentially in sequential multi-agent deep reinforcement learning (SMADRL). xApp 1 first observes the environment and takes action, then xApp 2 acts.
- **CMADRL**: In the concurrent multi-agent deep reinforcement learning (CMADRL), team members act concurrently, allowing in such arrangement, the simultaneous observation and the reward gathering for both xApps.
- **TMADRL**: In the team multi-agent deep reinforcement learning (TMADRL), team members exchange the action information with each other, and make their own decisions accordingly. Specifically, xApp 1 and xApp 2 will take their actions in each time slot based on the team member's previous action and the environmental states. After sharing such feedback in terms of actions, the agents apply their actions to the system. Finally, they update their models with a single-team reward.

4.0.3 Team Learning xApps Markov Decision Process (MDP)

In our case study, both xApps are implemented using deep Q-learning (DQL) [151]. The MDP of two of the xApps in the proposed TMADRL scheme are defined as follows:

Table 4.1: Simulation settings

Parameter		Value
Network settings	Networking Environment	4 Base Stations with 1 kilometre inter-site distance, 30 user equipments.
	Propagation	120.9+37.6 log10(distance) dB, Log-Normal shadowing: 8 dB.
	Carrier configuration	20MHz bandwidth, 100 resource blocks, 12 subcarriers per resource block, 12 resource block groups (RBGs).
	PHY configuration	Maximum transmission power of 38 dBm, minimum transmission power of 1 dBm, Additive white Gaussian noise = -114 dBm.
	Traffic model	Poisson distribution with varying load between 4-7 Mbps.
	Simulation time	20,000 time slots. 1 time slot is 100 ms.
Machine learning settings	Network structure	Number of hidden layers: 2
	Number of neurons	xApp 1: $N_h^1 = 256$ and $N_h^2 = 128$, xApp 2: $N_h^1 = 512$ and $N_h^2 = 256$.
	Discount factor	$\gamma = 0.2$
	Learning rate	$\alpha_l = 0.001$
	Reward importance ratio	$\beta = 0.35$

1. Power allocation xApp:

- State space is defined as:

$$S_t^l = \{\Gamma_t^{l,m}, R_t^{l,m}, p_t^{l,m}, \Gamma_t^{l',m'}, R_t^{l',m'}, p_t^{l',m'}, B_t^{l,m} | m \in M\} \tag{4.1}$$

where $\Gamma_t^{l,m}$ denotes the signal-to-interference-plus-noise ratio (SINR) of the transmission link of the l^{th} RBG of the m^{th} base station (BS), $R_t^{l,m}$ denotes the current transmission rate and $p_t^{l,m}$ denotes the current transmission power assigned to that resource block. The subscript t indicates the time step and $B_t^{l,m}$ is the length of queued data in the buffer. Additionally, l', and m' are respectively the index of the resource block that will be allocated to the n^{th} user and the index of BS that the n^{th} user will be associated with in the next time slot. This is given as:

$$l', m' \in \{l' \in L, m' \in M, \alpha_{t+1}^{l',m',n} = 1 | n \in \{\alpha_t^{l,m,n} = 1\}\} \tag{4.2}$$

where $\alpha_t^{l,m,n}$ is a binary indicator to indicate whether the l^{th} RBG of the m^{th} BS is allocated to the n^{th} user equipment. L corresponds to the number of RBGs of a BS and M corresponds to the number of BSs in the environment. The binary RBG allocation indicator of the next time slot is indicated by the action of resource block allocation xApp. In this way, the chosen action of resource block allocation xApp will be included in the state of power allocation xApp.

- Action space is defined as:

$$A_t^{l,m} = \{P_{min}, P_{min} + \frac{P_{max} - P_{min}}{\mathcal{B} - 1}, ..., P_{max}\} \tag{4.3}$$

The transmission power values are discretized into $\mathcal{B}$ levels according to the maximum and the minimum defined values to be assigned to each RBG.

- Reward: The reward is defined as the total throughput minus the sum transmission power, which is given as:

$$r_t = \Sigma_{m \in M} \Sigma_{l \in L} (R_t^{l,m} - \beta P_t^{l,m}) \tag{4.4}$$

where β is a factor used to balance the reward of throughput and transmission efficiency.

2. Radio Resource block allocation xApp:

- State space is given as:

$$S_t^m = \{\alpha_t^{l,m,n} \Gamma_t^{l,m}, \alpha_t^{l,m,n} R_t^{l,m}, \alpha_t^{l,m,n} p_t^{l,m}, \alpha_t^{l,m,n} L_t^{l,m}, p_t^{l',m'}, | l \in L, n \in N\}, \tag{4.5}$$

where $p_t^{l',m'}$ is obtained from the power allocation xApp.

- Action space is defined as:

The action of BS n is to choose a user by

$$A_t^{l,m} = \{n_0, n_1, ..., n_{N-1} | n_d \in H_t^m\} \tag{4.6}$$

where H_t^m denotes the set of all the user equipment attached to the m^{th} BS.

- Reward: The reward of resource block allocation xApp is defined likewise the power allocation xApp.

Both xApps learn and act in the same system with four BSs and 30 user equipment. We use a self-made Python network environment that includes the communication models needed to simulate a mobile network. Finally, the deep reinforcement learning algorithm is implemented with the TensorFlow library. We set up and train the xApps models independently as it is done typically in the Near RT-RIC of the O-RAN architecture. The simulation settings are shown in Table 4.1.

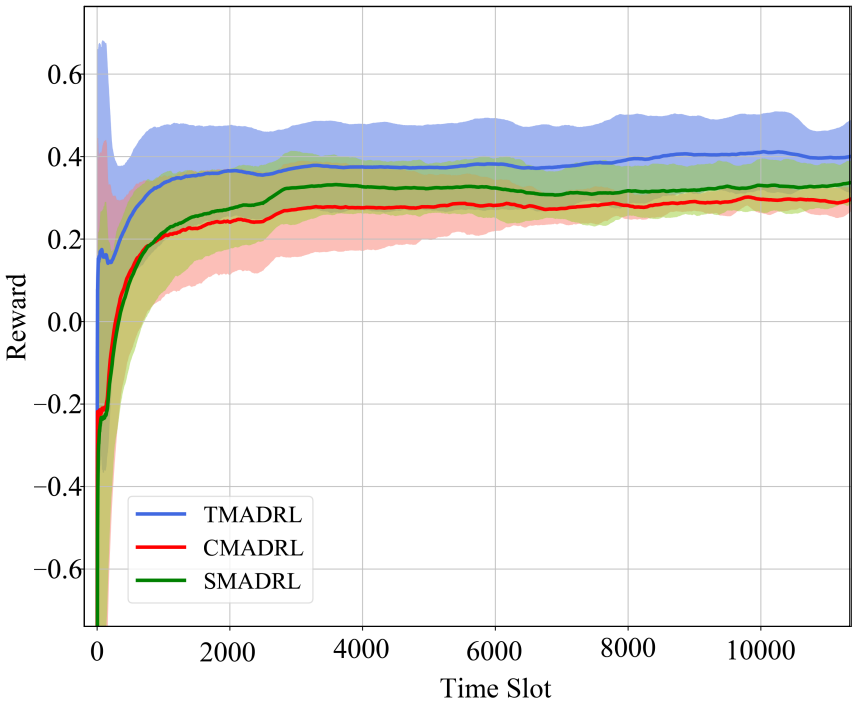

Figure 4.3: Learning convergence for the TMADRL, CMADRL, and SMADRL schemes.

4.0.4 Performance Evaluation

Figure 4.3 shows the convergence curves of the three different algorithms for five trials. It can be observed that the TMADRL algorithm receives a higher reward from the environment compared with the two baselines. Furthermore, in Fig. 4.2, we present the energy efficiency and throughput results of the three schemes under study. The figure indicates

that TMADRL achieves a more efficient energy utilization with a gain of 29.7% and 59.04% when compared with CMADRL and SMADRL. Additionally, it can be seen that TMADRL can achieve an improvement in throughput with a gain of 1% and 4.56%. It is important to mention that despite the throughput improvement is not relevant, TMADRL achieves a significant improvement in terms of energy efficiency while maintaining throughput rates. The throughput is measured by the average amount of useful data rate per unit of time and the energy efficiency is defined as the ratio of the throughput and the power consumed by the BSs. The reason behind this performance enhancement is related to the capabilities of our team members to exchange information in terms of actions which avoids non-Markovian environment behavior and thus, allows team convergence. Lastly, in Fig. 4.4, we present four simulation results, each one representing the MATL and baseline behavior under four different traffic loads per user. We can observe a clear positive trend in terms of throughput for the presented algorithms, TMADRL being the best performer among all. Furthermore, an interesting uptrend behavior is observed in the energy efficiency metric that has its roots in the user traffic and the BS resource capacity. When low user traffic is employed, the transmission capacity is underutilized, meanwhile when the traffic increases, the BS is capable of more efficient utilization of the resources. Under all the traffic loads, the TMADRL scheme outperforms the baselines presented in this work with a more evident gain in higher loads such as 6 and 7 Mbps. To sum up, MATL shows promising results in the O-RAN architecture with additional improvements when feedback exchange is considered. Note that, despite our use cases' focusing on xApps, the employed MATL schemes and techniques are extensible to reps as well. It is also worth mentioning that under this cooperative setting, the overhead and required processing memory will grow as a trade-off of the increased throughput and efficiency. Specifically, the processing memory will rise by an order of $2N_{xApps}$ and the communication cost between the two xApps will increase by an order of 2^{N}_{xApps} where N_{xApps} corresponds to the number of xApps.

In the next subsection, we elaborate on open issues, as well as the opportunities of MATL in O-RAN.

4.0.5 Open Issues and Future Directions for Multi-Agent Team Learning in O-RAN

Despite the many potential benefits of MATL in O-RAN, several challenges need to be addressed before it can be dominantly used in O-RAN control and optimization. In this section, we identify the challenges, open issues, and opportunities in MATL for O-RAN. It should be noted that some challenges are already fundamental in learning in the MAS environment, while others are only specific to O-RAN.

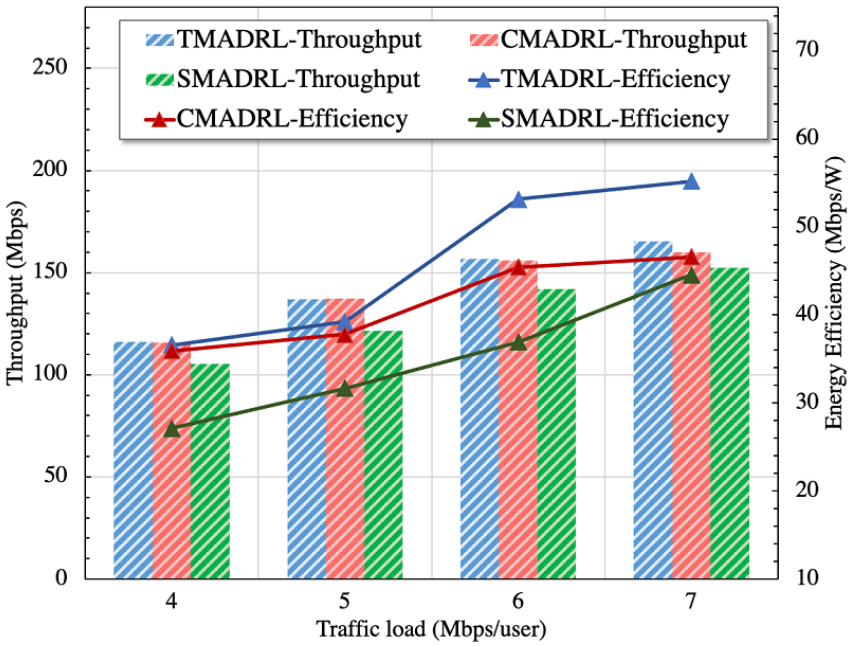

Figure 4.4: Average energy efficiency (E_e) and throughput performance metrics for the TMADRL, CMADRL, and CMADRL schemes for 4,5,6 and 7 Mbps traffic load per user.

- **Convergence:** Similar to all other machine learning-based techniques in wireless, MATL for the O-RAN should converge fast. Divergent behavior will cause instability and slow convergence might hinder decision capability in real-time. However, convergence guarantee in decentralized learning for stochastic teams is known to be a challenging problem [152]. For fast convergence, bootstrapping techniques or offline training can be considered.

- **Scalability:** Dimensionality issues have been a recurrent issue when the number of agents (so as the states and actions) in a MAS tends to increase. As more xApps become intelligent agents in O-RAN, the scalability of learning, inter-agent communication, and environment feedback need to be addressed.

- **Lack of full observability:** As xApps act in an environment, they will be simultaneously modifying the environment of other agents. Therefore, to take optimal action, each agent will need to predict other agents' actions unless all agents can fully observe the environment, which is unlikely in O-RAN. Hence, decisions need to be made based on the agents' partial observations. This can result in a sub-optimal

solution. To this end, learning models that are robust under partial observability need to be explored.

- **Information sharing:** It is essential to decide how much of the available local information should be shared among agents for enhancing the modeling of an environment. This should be jointly considered with fronthaul and midhaul capacity and the functional split decisions in O-RAN.

- **Selecting the optimal team size:** In MATL, choosing the optimal team size can affect learning and system performance. Although a larger team can provide more comprehensive visibility over the environment and access more relevant information, each agent's incorporation and learning experience can be affected. Meanwhile, one can obtain faster learning within a smaller team size, but a sub-optimal performance may be achieved due to the limited system view. Therefore, optimal team organization will be fundamental to MATL in O-RAN.

- **Goal selection:** In the O-RAN environment, conflicting actions of agents may degrade the network performance, even though each agent intends to maximize the performance. The goal of MATL should be minimizing conflicts and focusing on overall performance enhancement.

- **Impact of delayed feedback and jitter:** Most MAS studies consider that the feedback from the environment is immediate, and if agents are communicating, their feedback is instant. However, in disaggregated RANs, feedback can be delayed. Delayed feedback may cause agents to interact with different observations of the environment and lead to degraded RAN performance.

- **Security and trust:** Most MAS rely on the truthfulness of information shared among agents. Although there are studies on uncertainty or partial observability, intentional wrong reporting and adversarial behavior should also be addressed. This is particularly important when embedding intelligence in O-RAN.

Chapter 5

Multi-agent Deep Reinforcement Learning for Next-Generation Wi-Fi Networks

Next-generation Wi-Fi networks bring several novel features to provide high reliability in dense environments. Among them, coordinated spatial reuse and MLO-capable devices correspond to some of the new additions to the newest amendments IEEE 802.11ax and 802.11be. Due to the high complexity of Wi-Fi scenarios, in the first work of this chapter, we study RL-based solutions to provide spatial reuse through coordination and present a comparison with the absence of it. Additionally, we leverage TRL to provide reliability in such dynamic scenarios. In the second work of this chapter, we focus on the adaptability of RL and propose a meta-learning algorithm capable of diminishing some negative factors such as negative transfer learning presented in the previous work about spatial reuse. The third study of this chapter delves into traffic allocation in MLO-capable devices. For the first time, we propose a RL-based algorithm that distributes incoming XR traffic in devices with concurrent and multiple operational interfaces. Finally, the fourth study focuses on channel allocation in MLO-capable networks. We propose a cooperative and parallel transfer learning algorithm that can outperform existing baselines by improving convergence speed and reducing the complexity of multi-agent reinforcement learning problem formulation.

5.1 Cooperate or not Cooperate: Transfer Learning with Multi-Armed Bandit for Spatial Reuse in Wi-Fi

Wireless connectivity has become an irreplaceable commodity in our modern society. The exponential trend expected in wireless technology usage has led analysts to predict that by the end of 2023, 71% of the global population will enjoy some kind of wireless service. In the group of Wireless Local Area Networks (WLANs), Wireless Fidelity (Wi-Fi) technology presents a growth of up to 4-fold over 5 years from 2018 to 2023 [153]. The current Wi-Fi standard, IEEE-802.11ax, also known as Wi-Fi 6 and its "Extended" version Wi-Fi 6E, are expected to constitute 75% of all Wi-Fi chipset shipments by the beginning of 2024. Moreover, Wi-Fi 6E is forecasted to represent the 32% of all Wi-Fi chipset shipments by 2025 [154]. Additionally, the IEEE standardization body has recently been working on a new standard named IEEE 802.11be or Wi-Fi 7, scheduled to be released by May 2024 [155]. This standard will replace IEEE 802.11ax in the coming years.

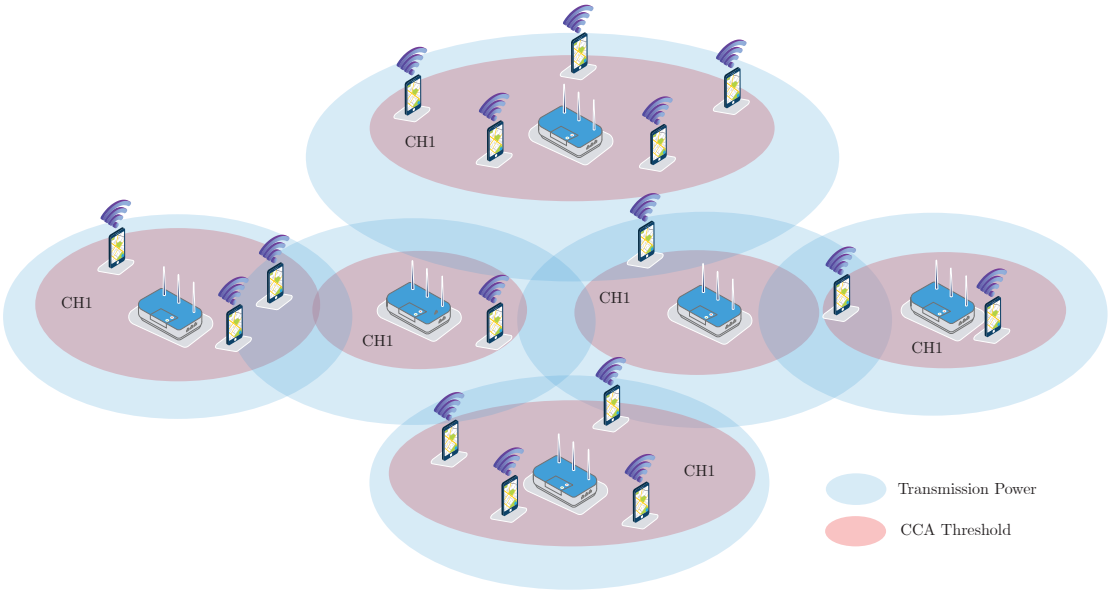

Figure 5.1: Typical operational scenario: Access Points (APs) adjust their Transmission Power and CCA threshold towards an efficient spatial reuse.

Spatial reuse (SR) has been of interest for more than 20 years in the wireless community, as it contributes to the reduction of collisions among stations and the determination of channel access rights [156]. As the number of dense WLAN deployments increases,

SR becomes more challenging in the context of Carrier Sense Multiple Access (CSMA) technology, as used in Wi-Fi [157]. Firstly, Wi-Fi 6 was introduced to address diverse challenges, such as the increasing number of Wi-Fi users, dense hotspot deployments, and the high demand for services like Augmented, Mixed, and Virtual Reality. It included additional features, such as dynamic adjustment of the Clear Channel Assessment (CCA) threshold and Transmission Power (TP). Before Wi-Fi 6, the CCA threshold configuration was a static value per Access Point (AP), causing inefficient channel utilization in dense Wi-Fi deployments [158]. Additionally, adjusting TP allows the reduction of interference among the APs and consequently maximizes network performance [159]. Thus, SR and network performance can be positively improved by adjusting CCA and TP. However, the complex interactions between CCA and TP call for intelligent configuration of both. As part of the forthcoming IEEE 802.11be standard, Coordinated Spatial Reuse (CSR) becomes one of several proposals to improve the current 802.11ax. Unlike the uncoordinated version introduced in 802.11ax, CSR requires inter-AP feedback to combat interference with neighboring APs [160]. Additionally, Wi-Fi 7 will include a widening of the channel bandwidth to a substantial 320 MHz, elevating the modulation rate to an impressive 4096 QAM, Multi-Link Operation (MLO), and Multiple Resource Units (MRU) capabilities.

To this end, data scarcity and data access are key for any Machine Learning (ML) method [161]. Recently, AI-based wireless networks have garnered remarkable interest among researchers in both the Wi-Fi domain [30, 44, 162] and the 5G domain [9]. However, the proposed solutions usually require complete availability of the data. In reality, data access is not always feasible due to privacy restrictions. Recent wireless network architectures have started to shift towards a more open and flexible design. In 5G networks, as well as in the O-RAN Alliance architecture, support is provided for the utilization of artificial intelligence to orchestrate main network functions [29]. In the context of Wi-Fi, a novel project named OpenWiFi [78], released by the Telecom Infra Project, intends to disaggregate the Wi-Fi technology stack by utilizing open-source software for the cloud controller and AP firmware operating system. These paradigm changes allow for the development of many applications in the area of ML, and more specifically, in RL applications, to become a reality.

In this research, we intend to optimize TP and CCA thresholds to improve SR and overall network Key Performance Indicators (KPIs). More importantly, we aim to study whether cooperation significantly impacts SR, demonstrating if the newly proposed features, like CSR, are indeed a necessity in Wi-Fi networks. To do so, we formulate the TP and CCA configuration problem to maximize product network fairness and minimize station starvation. We model our solution as a distributed multi-agent decision-making problem and use a Multi-Agent Multi-Armed Bandit (MA-MAB) approach to solve it.

The contributions of this work, different from those found in the literature, can be summarized in the following points:

1. We present the regret analysis for the distributed non-cooperative contextual MA-MAB (MA-CMAB) version of Sample Average Uncertainty-Sampling (SAU). SAU is based on a deep Contextual MAB.

2. We propose a solution for reducing the inherently huge action space given the possible combinations of TP and CCA threshold values per AP. We derive our solution via worst-case interference analysis.

3. We analyze the performance of the network KPIs of well-known distributed MA-MAB implementations such as ϵ-greedy, UCB, and Thompson on the selection of the TP and CCA values in cooperative and non-cooperative settings.

4. We introduce an MA-CMAB in its cooperative and non-cooperative settings and analyze its performance.

5. We propose, for the first time to the best of our knowledge, a deep transfer learning solution to adapt TP and CCA parameters efficiently in dynamic scenarios.

In this work, when referring to dynamic scenarios, we mean scenarios where there are changes in the user load per AP. This way, we aim to mimic real-life situations. With these contributions, our simulation results show that the ϵ-greedy MAB solution improves throughput by at least 44.4%, provides a 12.2% improvement in terms of fairness, and achieves a 94.5% reduction in Packet Loss Ratio (PLR) over typical configurations when a reduced set of actions is known. Additionally, we show that the SAU-Coop algorithm improves throughput by 14.7% and PLR by 32.5% when compared with non-cooperative approaches with a full set of actions. Moreover, our proposed transfer learning-based approach reduces service drops by at least 60%.

The rest of the subsections are organized as follows. Subsection 5.1.1 covers the basics of Multi-Armed Bandits and subsection 5.1.2 an analysis of the regret of the proposed algorithm. In subsection 5.1.3, we present our system model along with an analysis to reduce the action space via worst-case interference. Subsection 5.1.4 presents the proposed schemes, and the results are discussed in subsection 5.1.5.

5.1.1 Multi-Agent Multi-Armed Bandits (MA-MABs)

In this subsection, we intend to formally introduce the Multi-Agent Multi-Armed Bandit (MA-MAB) problem. An MA-MAB is the multi-agent variant of an MAB where several agents pull their arms and obtain feedback from the environment [163]. MA-MABs can be modeled as centralized or distributed. In centralized settings, the agents' actions are taken by a centralized controller, and in distributed settings, each agent will independently choose their actions. Distributed decision-making settings scale more effectively [164] and naturally deal easily with a large set of arms (K) when compared with centralized settings that suffer from K arms' cardinality explosion.

In this work, we consider a distributed scenario defined as follows. Let us consider a contextual K-armed bandit with N players or agents that form a team. Notice that in this work we use M to denote the number of APs, and consequently $N = M$. However, in this subsection, we reserve N to describe the number of agents in any multi-agent multi-armed bandit problem. Denote $X_{n,k}(t)$ the reward obtained by the n^{th} agent by pulling its k^{th} arm. Additionally, let $\mu_{n,k}$ denote the mean of $X_{n,k}(t)$. If only one agent is considered, we can assume that the yielded reward is $X_{n,k}(t) = Y_{n,k}(t)$, where $Y_{n,k}(t)$ is a random variable that models the reward of the environment. We assume a non-collision scheme [165–167] where if more than two agents play the same arm k at time t the yielded reward is not affected by that specific condition. Differently from [168], where channel access is studied, our problem becomes non-trivial since agents can select the same spatial reuse parameters and still have good performance. Thus, we assume agents' decisions can generate conflict or not and the perceived reward can change according to the team's action decision. Thus, $X_{n,k}(t) = \hat{Y}_{n,k}(t)$, where $\hat{Y}_{n,k}(t)$ is a random variable that models the reward obtained from the environment when more than one agent interacts with it. All agents choose their arms concurrently in this setting. Let us define the action taken by agent n at time t by $k_n(t) \in \mathcal{A} := \{1, .., K\}$. In addition, the context of each agent observed by agent n at time t is defined by $c_n(t) \in \mathcal{C} := \{C_1, .., C_Q\}$, where Q is the number of contextual information used in the CMAB. Consequently, the history seen by agent n at time t corresponds to $\mathcal{H}_n(t) = \{k_n(1), c_n(1), X_{n,k_{n(1)}}(1), ..., k_n(t), c_n(t), X_{n,k_{n(t)}}(t)\}$. The final policy for agent n corresponds to $\pi_n : \mathcal{H}_n(t) \rightarrow \mathcal{A}$ where $\pi_n = (k_n(t))_{t=1}^{\infty}$. Finally, the action system vector can be defined as bipartite matching $\mathbf{k}^* = \{(k_1, ..., k_N) : k_n \in \mathcal{A}\}$. The end goal is to maximize the system reward in two distributed approaches: non-cooperative and cooperative. The expected sum of rewards corresponds to $\mathbb{E}_\pi[\sum_{t=1}^{T} X_{\pi(t)}(t)]$. If the means $\mu_{n,k}$ are known, the set of actions can be selected via picking a bipartite matching as:

$$\mathbf{k}^{**} \in \arg\underset{k \in k^*}{max} \sum_{n=1}^{N} \mu_{n,k_n}, \tag{5.1}$$

where $\mathbf{k}^{**}$ is the optimal and not unique bipartite matching. The solution to the proposed problem would be finding a maximum weight matching on a labeled bipartite graph between agents' actions and the number of agents. A bipartite matching in this context refers to a set of edges in the bipartite graph such that no two edges share a common vertex. In this case, actions are chosen by selecting a specific arm in each of the N agents comprising the Multi-Agent MAB. Since each action is taken concurrently by all agents, the condition of bipartite matching is always fulfilled. The selection of the bipartite matching can be represented as a combinatorics problem where the permutation of each arm in a multi-agent scenario is mapped as:

$$\mathbf{k}^*(t) \in [1, P(K, N)], \tag{5.2}$$

where $\mathbf{k}^*(t)$ is the bipartite matching selected at time t and $P(K, N)$ corresponds to the number of permutations given K arms and N agents. Finally, the expected regret of a multi-agent multi-armed bandit can be defined as:

$$R(T) = T\mu_{k^{**}} - \mathbb{E}_{\pi}\left[\sum_{t=1}^{T} X_{\pi(t)}(t)\right], \tag{5.3}$$

where $\mu_{k^{**}}$ corresponds to the optimal team policy to select $\mathbf{k}^{**}$ bipartite matching.

5.1.2 Analysis of Regret

The expected regret of a single agent obtained with the SAU-Sampling algorithm has a logarithmic nature (*Proof:* See [76, Appendix A.8]) as its counterparts TS and UCB, and shows an improvement in comparison to the linear regret nature of the ϵ-greedy MAB [50]. Building on and extending the work of [76,168] along with the considerations in [169,170], we can obtain the expected regret for our distributed SAU-Sampling MA-CMAB algorithm as follows:

Theorem 5.1. *If SAU-Sampling MA-CMAB is run on a distributed K-armed N-agents bandit problem ($K \geq 2$, $N \geq 1$) having an arbitrary reward distribution $X_{1,1}, \cdots, X_{N,K}$ with support in $[0, 1]$, then its expected regret after n number of plays is at most*[1]*:*

$$R(T) \leq N^2 K \Delta_{k^{**}} \left(\frac{96 \log n}{\Delta^2_{k^{**}}} + \frac{1}{1+N}\right), \tag{5.4}$$

[1] A proof of Theorem 5.1 is provided in Appendix A.1.1

Table 5.1: Notations

Notation	Definition
s and $\mathcal{S}$	Index and set of stations
m and $\mathcal{M}$	Index and set and the number of APs
$x^{\lvert\mathcal{S}\rvert}$ and $y^{\lvert\mathcal{M}\rvert}$	Stations' positions and AP's positions
N_S	Total number of nodes
P_{cs}^{m}	CCA threshold of m^{th} AP
P_{tx}^{m}	Transmission Power of m^{th} AP
$R_T^{(s,m)}$	Throughput of s^{th} STA of m^{th} AP
$R_A^{(s,a)}$	Achievable throughput of s^{th} STA of m^{th} AP
$D_l^{(m,s)}$	Adaptive data link rate of s^{th} STA of m^{th} AP
$u_{IDLE}^{(s,m)}$	Binary function, $u_{IDLE}^{(s,m)} = 1$ if STA s is idle with respect to its m^{th} BSS and $u_{IDLE}^{(s,m)} = 0$, otherwise
$u_{SUCC}^{(s,m)}$	Binary function, $u_{SUCC}^{(s,m)} = 1$ if STA s successfully transmits a packet to its m^{th} AP and $u_{SUCC}^{(s,m)} = 0$, otherwise
$\phi_p^{(s,m)}$	Corresponds to the set of binary variables that define each s^{th} STA transmission properties to their m^{th} AP, where $\phi_p^{s,m} := \{u_{IDLE}^{(s,m)}, u_{SUCC}^{(s,m)}, \xi_{CCA}^{(s,m)}, \xi_{ED}^{(s,m)}\}$
$\xi_{CCA}^{(s,m)}$	Binary function, $\xi_{CCA}^{(s,m)} = 1$ if the sensed energy signal of a packet sent by the s^{th} STA to the m^{th} AP is below the CCA threshold (P_{cs}^{m}), and $\xi_{CCA}^{(s,m)} = 0$, otherwise
$\xi_{ED}^{(s,m)}$	Binary function, $\xi_{ED}^{(s,m)} = 1$ if the sensed energy signal of a packet sent by the s^{th} STA to the m^{th} AP is below the Energy Detection (ED) threshold (P_{ed}^{m}), and $\xi_{ED}^{(s,m)} = 0$, otherwise
$\xi_{STA}^{(s,m)}$	Binary function, $\xi_{STA}^{(s,m)} = 1$ if s^{th} station's throughput is greater than $\omega R_A^{(s,a)}$ and $\xi_{STA}^{(s,m)} = 0$, otherwise
$\mathbb{E}(T_g^{(s,m)})$ and $\mathbb{E}(I_g^{(s,m)})$	Expected length of general time-slot and expected information transmitted by the s^{th} STA of m^{th} AP
T_{TXOP}, T_{EDCA}	Packet transmission duration and time required for a successful Enhanced Distributed Channel Access (EDCA) transmission
$\delta^{(s,m)}$	Packet transmission duration and time required for a successful Enhanced Distributed Channel Access (EDCA) transmission
$\bar{P}^{fair}$ and $\bar{U}$	Average linear product-based network fairness and average station starvation
ω, $g^{(s,m)}$ and σ^2	Fraction of $R_A^{(s,a)}$ in which STAs are considered in starvation, the channel power gain and the power noise
P_{tx}^{m} and P_{tx}^{r}	The transmission power at the m^{th} transmitter (AP) and the received signal strength at the r^{th} receiver
$d_{r,m}$ and θ	Distance between the m^{th} transmitter and r^{th} receiver and path loss exponent
$\mathcal{F}_{+}^{m}$ and $\mathcal{F}_{-}^{m}$	Subset of interferers and non-AP interferers
$\gamma^{(r,m)}$, $C^{(r,m)}$ and C_T	Worst-case SINR and Shannon's maximum capacity of m^{th} transmitter and r^{th} receiver and cumulative maximum network capacity

where n is the number of times each agent in SAU-Sampling MA-CMAB chooses an action. Finally, $\Delta_{k^{**}}$ is defined as $\Delta_{k^{**}} = \mu_{n,k_n^{**}} - \mu_{n,k_n}$.

Corollary 5.1. *The upper bound of the regret in the distributed and non-cooperative solution is polynomial in K and N, respectively, and exhibits logarithmic behavior in the time domain.*

According to Corollary 5.1, the regret will suffer a polynomial and logarithmic increase, which is highly problematic, especially if K or N grows. This motivates subsection 5.1.3, where we reduce the $P(K, N)$ by analytically finding a reduced set of arms to alleviate the poor performance of such regret. On the other hand, we can proceed to define the expected regret for the cooperative scenario:

$$R(T) = TX_k - \mathbb{E}_\pi \left[\sum_{t=1}^{T} X_{\pi(t)}(t) \right] + \mathbb{E}[Cost], \tag{5.5}$$

where $\mathbb{E}[Cost]$ is the expected cost of communication among agents. To further obtain the upper bounds in the cooperative case, a similar procedure should be followed as in [164, 171], where agents can communicate an estimate of their reward. However, we leave the derivation of this regret outside the scope of this work due to its lengthy characteristic and empirically prove cooperation superiority when compared with the non-cooperative case. The goal in both scenarios is to find the set of policies for each agent in the distributed algorithm to minimize the regrets presented in (5.3) and (5.5). In this work, we consider two main approaches: distributed non-cooperative and cooperative MA-MABs with adversarial rewards. Next, we will discuss the details of the system model and present an analysis of reducing the cardinality of the action space in the proposed SR formulation.

5.1.3 System model and Problem Formulation

In this work, we consider an infrastructure mode Wi-Fi 802.11ax network $\mathcal{N}$ with $N_S = |\mathcal{S}|+|\mathcal{M}|$ nodes where $\mathcal{S}$ is the set of stations with $\{\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^{|\mathcal{S}|}\} \in \mathbb{R}^2$ positions and $\mathcal{M}$ is the set of APs with $\{\boldsymbol{y}^1, \boldsymbol{y}^2, ..., \boldsymbol{y}^{|\mathcal{M}|}\} \in \mathbb{R}^2$ positions. We can assume that $|\mathcal{M}|$ APs positions correspond to cluster centers and the stations will attach to their closest AP. In addition, the list of notations utilized in this work can be found in Table 5.1.

In this study, we improve SR via maximization of the linear product-based fairness and minimization of the number of stations under starvation by configuring TP and CCA

parameters.

$$\textbf{Max} \quad \begin{pmatrix} \text{fairness} \\ \text{avg. station starvation complement} \end{pmatrix} \tag{5.6a}$$

$$\textbf{s.t.} \quad \text{Throughput,} \tag{5.6b}$$

$$\textbf{var.} \quad \text{Transmission power and CCA threshold,}$$
$$\text{Idle and success transmission indicators,}$$
$$\text{EDCA Idle and success transmission duration.} \tag{5.6c}$$

Each s^{th} STA is defined by a set of binary variables of transmission properties to their m^{th} AP, where $\phi_p^{s,m} := \{u_{IDLE}^{(s,m)}, u_{SUCC}^{(s,m)}, \xi_{CCA}^{(s,m)}, \xi_{ED}^{(s,m)}\}$. Let us define the binary variable $u_{IDLE}^{(s,m)}$ of an STA being idle in a BSS as:

$$\sum_{m \in \mathcal{M}} u_{IDLE}^{(s,m)} = 1 \qquad \forall s \in \mathcal{S}, \tag{5.7}$$

where $u_{IDLE}^{(s,m)} = 1$ if s^{th} STA is idle with respect to its AP and $u_{IDLE}^{(s,m)} = 0$, otherwise. In addition, we proceed to define the binary variable $u_{SUCC}^{(s,m)}$ in which an STA will successfully transmit a packet as,

$$\sum_{s \in \mathcal{S}} u_{SUCC}^{(s,m)} u_{IDLE}^{(s,m)} \xi_{CCA}^{(s,m)} \xi_{ED}^{(s,m)} \leq 1 \qquad \forall m \in \mathcal{M}, \tag{5.8}$$

$$\sum_{m \in \mathcal{M}} \xi_{CCA}^{(s,m)} = 1 \qquad \forall s \in \mathcal{S}, \tag{5.9}$$

$$\sum_{m \in \mathcal{M}} \xi_{ED}^{(s,m)} = 1 \qquad \forall s \in \mathcal{S}, \tag{5.10}$$

where $\xi_{CCA}^{(s,m)} = 1$ if the sensed energy signal of a packet sent by the s^{th} STA to the m^{th} AP is below the CCA threshold (P_{cs}^m), otherwise becomes zero. Here, $\xi_{ED}^{(s,m)} = 1$ if the sensed energy signal of a packet sent by the s^{th} STA to the m^{th} AP is below the Energy Detection (ED) threshold (P_{ed}^m), otherwise becomes zero. Additionally, we consider $P_{cs}^m = P_{ed}^m$.

The CCA threshold is a predefined signal strength level that a channel must fall below for a device to consider the channel clear and decide to transmit. Pragmatically speaking,

if the signal strength on a channel is above the CCA threshold, a device assumes that the channel is busy, and it refrains from transmitting to avoid interference with ongoing transmissions. On the other hand, if the signal strength falls below the CCA threshold, the device assumes that the channel is clear and proceeds with transmission. The rationale for this threshold is to help avoid collisions and interference by ensuring that a device does not transmit when it detects ongoing transmissions on the channel.

While CCA is primarily used to assess whether the channel is clear for the initiation of Wi-Fi transmissions, the ED threshold is used to detect the overall energy level on a channel, including both Wi-Fi and non Wi-Fi signals. Devices use energy detection to determine if a channel is busy or idle. If the energy level is above the ED threshold, the channel is assumed to be busy, and devices may defer their transmissions. If the energy level is below the ED threshold, the channel is assumed to be clear, and devices may proceed with transmissions. In dynamic channel selection scenarios, where Wi-Fi networks may change channels dynamically to avoid interference, the ED threshold is used to make decisions about channel switching. Even though ED and CS thresholds are different in terms of what type of energy they track, both minimize interference, avoid collisions, and improve the overall reliability and performance of wireless communication by managing channel access and transmission.

P_{ed}^{m} by definition has a higher value than P_{cs}^{m} since it is firstly checked by the AP to set an STA on IDLE status when compared to the received power. Afterwards, the AP checks P_{cs}^{m}. Thus, we could assume that for all APs at least the condition $P_{cs}^{m} \leq P_{ed}^{m}$ holds and consequently we can simplify our analysis with equality.

As indicated in [172], two procedures can be defined when stations are using ECDA. The first one indicates the duration of a successful EDCA transmission whereas the second one indicates the duration of a collision. The first procedure can be generally described as:

$$T_{EDCA}^{s} = T_{TXOP} + SIFS + T_P + ACK + AIFS, \tag{5.11}$$

where T_{TXOP} defines the transmission duration, SIFS corresponds to the short inter-frame space, ACK denotes the time to send an acknowledgment signal, AIFS represents the arbitration inter-frame space, and T_P stands for the propagation delay. On the other hand, we can define the second procedure as the duration of a collision. The previous event can be calculated as:

$$T_{EDCA}^{c} = T_{TXOP} + T_P + AIFS. \tag{5.12}$$

All of the variables in Eqs. (5.11) and (5.12) are in the μs order with exception of T_{TXOP} which falls in the order of ms. Thus, with the following approximation, $T_{EDCA}^{c} \approx T_{EDCA}^{s}$, we can denote both values as T_{EDCA}.

As indicated by [173], the expected conditional length of the general time-slot $\mathbb{E}(T_g)$ and the expected conditional transmitted information $\mathbb{E}(I_g)$ by the s^{th} STA to m^{th} AP can be expressed as:

$$\begin{aligned}\mathbb{E}(T_g^{(s,m)}|u_{IDLE}^{(s,m)}) = \delta^{(s,m)}u_{IDLE}^{(s,m)}+ \\ u_{IDLE}^{(s,m)}T_{ECDA} \quad \forall m \in \mathcal{M}, s \in \mathcal{S},\end{aligned} \tag{5.13}$$

$$\mathbb{E}(I_g^{(s,m)}|u_{SUCC}^{(s,m)}) = u_{SUCC}^{(s,m)}D_l^{(s,m)}T_{TXOP} \qquad \forall m \in \mathcal{M}, s \in \mathcal{S}, \tag{5.14}$$

where $D_l^{(s,m)}$ denotes the link data rate, T_{EDCA} is a variable that corresponds to the time required for a successful Enhanced Distributed Channel Access (EDCA) transmission and $\delta^{(s,m)}$ represents the duration of an idle time slot. The link data rate adaptively depends on SNR [174] and is mapped based on the SNR/BER curves [175]. The received SNR can be defined as $P_{tx}^m g^{(s,m)}/\sigma^2$ where P_{tx}^m is the transmission power, $g^{(s,m)}$ is the channel power gain and σ^2 is the power noise.

According to Bianchi's work that is commonly used as a reference study [176], the value of the throughput R can be calculated as follows:

$$R = \frac{\mathbb{E}[\text{payload information transmitted in a slot time}]}{\mathbb{E}[\text{length of a slot time}]}. \tag{5.15}$$

Analogously, we can define the throughput of the s^{th} station attached to the m^{th} AP as:

$$\begin{aligned}R_T^{(s,m)} &= \frac{\mathbb{E}(I_g^{(s,m)}|u_{IDLE}^{(s,m)})}{\mathbb{E}(T_g^{(s,m)}|u_{SUCC}^{(s,m)})} \\ &= \frac{u_{SUCC}^{(s,m)}D_l^{(s,m)}T_{TXOP}}{\delta^{(s,m)}u_{IDLE}^{(s,m)} + u_{IDLE}^{(s,m)}T_{EDCA}}.\end{aligned} \tag{5.16}$$

Additionally, let us define the average linear product-based network fairness and average station starvation in a distributed setting by,

$$\bar{P}^{fair}(t) \leq 1, \tag{5.17}$$

$$\sum_{s\in\mathcal{S}} \xi_{STA}^{(s,m)}(t) = 1 \qquad \forall m \in \mathcal{M}, \tag{5.18}$$

$$\bar{P}^{fair}(t) = \frac{1}{|\mathcal{M}|} \sum_{m\in\mathcal{M}} \prod_{s\in\mathcal{S}} \frac{R_T^{(s,m)}(t)}{R_A^{(s,a)}}, \tag{5.19}$$

$$\bar{U}(t) = \frac{1}{|\mathcal{M}|} \sum_{m\in\mathcal{M}} \frac{1}{|\mathcal{S}|} \sum_{s\in\mathcal{S}} \xi_{STA}^{(s,m)}(t), \tag{5.20}$$

where $R_A^{(s,a)}$ is the achievable throughput of the s^{th} station attached to the m^{th} AP. In practice, the value of $R_A^{(s,a)}$ is set by running simulations with one STA and gathering the corresponding throughput upper bound. Additionally, $\xi_{STA}^{(s,m)} = 1$ if s^{th} station's throughput is greater than $\omega R_A^{(s,a)}$ where the station starvation ratio, $\omega \in (0, 1]$, otherwise becomes zero, in which case the station is considered in starvation. ξ_{STA} measures the number of starving stations at t time. It uses throughput that behaves as a random variable due to many factors such as congestion, packet loss, and varying levels of network activity. Due to the property of *induced randomness*, ξ_{STA} takes the form of a random variable as well.

The considered problem is a multi-objective problem and can be addressed with the weighted sum approach. Thus, in each time step, the problem can be formulated as,

$$\max_{P_{tx},P_{cs}} \; B_1\bar{P}^{fair}(t) + B_2(1 - \bar{U}(t)) \tag{5.21}$$

s.t.

$$(5.7)\text{-}(5.18), \tag{5.22}$$

$$P_{tx}^m \in [P_{tx}^{min}, P_{tx}^{max}], P_{cs}^m \in [P_{cs}^{min}, P_{cs}^{max}] \qquad \forall m \in \mathcal{M}, \tag{5.23}$$

where B_1 and B_2 are the importance coefficients associated to the variables $\bar{P}^{fair}$ and the complement of $\bar{U}(t)$, respectively.

As specified in equation (22), Problem 1 requires resolution in each time slot t. The parameters chosen in a given time slot and their outcomes are contingent not only on the selected transmission power and CCA threshold at time t but also on decisions made in the past and uncertain future events. Consequently, we can conceptualize this problem as a sequential decision problem, wherein actions optimizing performance are taken across a

sequence of steps. This behavior is dictated by our setup, where each AP independently selects its parameters, sharing varying degrees of information or no information at all. Due to the dynamic nature of the scenario, the binary variables of each STA $\phi_p^{(s,m)}$ have a probabilistic nature, requiring an additional step to map them to EDCA parameters [173]. Instead, we simplify our analysis by utilizing a network simulator to model such dynamics. We propose to solve the previous mixed fractional and stochastic polynomial problem by using an MA-CMAB solution, as described in subsection 5.1.4.

Optimal action set via worst-case interference

Motivated by subsection 5.1.2, we aim to find a reduced action set via worst-case interference analysis. Wi-Fi typical scenarios consist of APs and stations distributed non-uniformly. Contrary to the analysis presented in [177], we aim to obtain an optimal subset of TP and CCA threshold values to further reduce the action space size that is utilized in the optimization of SR. In this analysis, we only consider the Carrier Sense (CS) threshold term as a form of the CCA threshold.

First, let us consider the worst-case interference scenario in an arrangement with $N_S \geq 2$. For the sake of simplicity, we use the path-loss radio propagation model:

$$P_{rx}^{r} = \frac{P_{tx}^{m}}{{d_{r,m}}^{\theta}}, \tag{5.24}$$

where P_{tx}^{m} and P_{rx}^{r} are the TP at the m^{th} transmitter (AP) and the received signal strength at the r^{th} receiver, respectively. In addition, $d_{r,m}$ is the distance between the transmitter and receiver. Finally, $\theta \in [2, 4]$ corresponds to the path loss exponent. Thus, from the perspective of the m^{th} AP, the worst-case interference I^m is defined as:

$$I^m = \sum_{v \in \mathcal{F}_+^m} \frac{P_{tx}^{v}}{X^{(m,v)^{\theta}}} + P_{tx}^{sta} \sum_{w \in \mathcal{F}_-^m} \frac{1}{X^{(m,w)^{\theta}}}, \tag{5.25}$$

where $\mathcal{F}_+^m$ is the subset of interference sources $|\mathcal{F}_+^m| = |\mathcal{M}| - 1$, corresponding to APs interfering with the m^{th} AP, and $\mathcal{F}_-^m$ is the subset of non-AP interference sources $|\mathcal{F}_-^m| = |\mathcal{S}|$, corresponding to the stations interfering with the m^{th} AP. Furthermore, P_{tx}^{v} is the TP of the v^{th} interference source, and P_{tx}^{sta} is a constant corresponding to the fixed power assigned to all the stations, based on the fact that typically stations are not capable of modifying their TP. Additionally, $X^{(m,v)}$ and $X^{(m,w)}$ correspond to the distance from the m^{th} AP to the v^{th} AP interference source and m^{th} AP to the w^{th} station interference source, respectively.

$X^{(m,.)}$ is calculated as,

$$X^{(m,.)} = \sqrt{(D^m_{CCA} + x^{(m,.)})^2 + d^2_{r,m} - \frac{2(D^m_{CCA} + x^{(m,.)})}{(d_{r,m}\cos\varsigma_{r,.})^{-1}}}, \tag{5.26}$$

where (.) refers either to the AP or non-AP interference source, D^m_{CCA} is the CCA threshold range of the m^{th} AP, $\varsigma_{r,.}$ is the distance between the receiver to the interference source (.) and $x^{(m,.)}$ corresponds to the distance between any (.) interference source and D^m_{CCA}.

The corresponding worst-case SINR $\gamma^{(r,m)}$ at the receiver is defined as:

$$\gamma^{(r,m)} = \frac{P^m_{tx}}{{d_{r,m}}^{\theta}(I^m + N_0)}. \tag{5.27}$$

Let us assume that $N_0 << I^m$, thus the equation is reduced to:

$$\gamma^{(r,m)} = \frac{P^m_{tx}}{{d_{r,m}}^{\theta} I^m}. \tag{5.28}$$

By substituting equations (5.25) and (5.26) in (5.28), we obtain equation (5.29). The previous mentioned equation describes $\gamma^{(r,m)}$ in function of D^m_{CCA} and $d_{r,m}$. Additionally, we substitute $D^m_{CCA} = (P^m_{tx}/T^m_{cs})^{1/\theta}$ in equation (5.29) obtaining,

$$\gamma^{(r,m)} = \frac{\frac{P^m_{tx}}{{d_{r,m}}^{\theta}}}{\sum\limits_{v\in\mathcal{F}^m_+} \frac{P^m_{tx}}{\left(\sqrt{(D^m_{CCA}+x_{m,v})^2+d^2_{m,r}-\frac{2(D^m_{CCA}+x_{m,v})}{(d_{r,m}\cos\varsigma_{r,.})^{-1}}}\right)^{\theta}} + P^{sta}_{tx}\sum\limits_{w\in\mathcal{F}^m_-} \frac{1}{\left(\sqrt{(D^m_{CCA}+x_{m,w})^2+d^2_{r,m}-\frac{2(D^m_{CCA}+x_{m,w})}{(d_{r,m}\cos\varsigma_{r,w})^{-1}}}\right)^{\theta}}} \tag{5.29}$$

$$\gamma^{(r,m)} = \frac{\frac{P^m_{tx}}{{d_{r,m}}^{\theta}}}{\sum\limits_{v\in\mathcal{F}^+_m} \frac{P^m_{tx}}{\Gamma^m + P^{sta}_{tx}\sum_{w=1}^{K^-_m}\iota^{(m,w)}}}, \tag{5.30}$$

where,

$$\Gamma^m = \left(\sqrt{\left[\left(\frac{P^m_{tx}}{T^m_{cs}}\right)^{\frac{1}{\theta}} + x_{m,v}\right]^2 + d^2_{r,m} - H_1}\right)^{\theta},$$

$$H_1 = 2\left[\left(\frac{P_{tx}^{m}}{T_{cs}^{m}}\right)^{\frac{1}{\theta}} + x_{m,v}\right] d_{r,m}\cos\varsigma_{r,v},$$

$$\iota^{(m,w)} = \left(\sqrt{(\Omega_{sta}+x_{m,w})^2 + d_{r,m}^2 - \frac{2(\Omega_{sta}+x_{m,w})}{(d_{r,m}\cos\varsigma_{r,w})^{-}1}}\right)^{-\theta},$$

$$\Omega_{sta} = \left(\frac{P_{tx}^{sta}}{T_{cs}^{sta}}\right)^{\frac{1}{\theta}}.$$

Hereafter, we proceed to define the maximum channel capacity in terms of TP and Carrier Sense (CS) threshold (T_{cs}). Given a certain value of SINR, the Shannon maximum capacity is expressed as:

$$C^{(r,m)} = W\log_2(1+\gamma^{(r,m)}), \tag{5.31}$$

where W is the channel bandwidth in Hz. Then, the cumulative maximum network capacity can be calculated as:

$$C_T = \sum_{m=1}^{|\mathcal{M}|-1}\sum_{r=1}^{N} C^{(r,m)}. \tag{5.32}$$

In Fig. 5.2, a graph of the network maximum capacity is shown as a function of TP and CS threshold. As observed, the network capacity achieves its higher values when a combination of high TP and low CS threshold is utilized. Note that, prior knowledge of the locations is required.

5.1.4 Proposed Multi-Agent Multi-Armed Bandit Algorithms

In this subsection, we present the action space, context definition, and reward function for the MA-MAB algorithms utilized in this work.

Action space

In this work, we consider a discrete action space that corresponds to the number of combinations of P_{cs} and P_{tx}. In the context of MABs, this translates into the number of arms for each MAB agent. The action space is defined as,

$$A_{cs} = \{P_{cs}^{\min}, P_{cs}^{\min} + \frac{P_{cs}^{\max} - P_{cs}^{\min}}{L_{cs} - 1}, ..., P_{cs}^{\max}\}, \tag{5.33}$$

$$A_{tx} = \{P_{tx}^{\min}, P_{tx}^{\min} + \frac{P_{tx}^{\max} - P_{tx}^{\min}}{L_{tx} - 1}, ..., P_{cs}^{\max}\}, \tag{5.34}$$

where $P_{cs}^{\min}$, $P_{cs}^{\max}$, $P_{tx}^{\min}$, and $P_{tx}^{\max}$ are the minimum and maximum values of the CCA threshold and TP values, respectively. L_{cs} and L_{tx} denote the number of levels used to discretize the CCA threshold and TP values, respectively. Finally, the number of arms corresponding to the action space for the m^{th} agent, K_m^{AP}, is given by $|A_{cs}^m| \cdot |A_{tx}^m|$.

Reward function in distributed non-cooperative settings

The reward is defined following the optimization problem 5.1.3. Such problem statement presents a general framework of the presented reward function whose objectives are maximizing fairness and minimizing starvation. Defined in a distributed manner, it resembles the reward presented in [106], which includes a linear product-based fairness and the starvation term of a station [106, 112]. A station is considered to be in starvation when its performance is below a predefined percentage of its achievable throughput. The reward is defined as,

$$r_{AP}^m = \underbrace{\frac{|\Psi_{AP}^m| \prod_{s \in \Psi_{AP}^m} \frac{R_T^{(s,m)}}{\omega R_A^{(s,m)}}}{N_{AP}^m (N_{AP}^m + 1)}}_{\text{maximize starving stations}} + \underbrace{\frac{|N_{AP}^m \setminus \Psi_{AP}^m|}{N_{AP}^m + 1}}_{\text{maximize non-starving stations}} + \underbrace{\frac{|N_{AP}^m \setminus \Psi_{AP}^m| \prod_{s \in \Psi_{AP}^m} \frac{R_T^{(s,m)}}{R_A^{(s,m)}}}{N_{AP}^m (N_{AP}^m + 1)}}_{\text{maximize fairness}} \tag{5.35}$$

Equation (5.35) can be reduced to,

$$r_{AP}^m = \frac{|\Psi_{AP}^m| \prod_{s \in \Psi_{AP}^m} \frac{R_T^{(s,m)}}{\omega R_A^{(s,m)}} + H_2}{N_{AP}^m (N_{AP}^m + 1)}, \tag{5.36}$$

$$H_2 = |N_{AP}^m \setminus \Psi_{AP}^m| \left(N_{AP}^m + \prod_{s \in N_{AP}^m \setminus \Psi_{AP}^m} \frac{R_T^{(s,m)}}{R_A^{(s,m)}} \right),$$

where Ψ_{AP}^m is the set of starving stations attached to the m^{th} AP, N_{AP}^m the set of stations attached to the m^{th} AP. We can also observe that $r_{AP}^m \propto C^{(r,m)}$ as defined in (5.31). The term $N_{AP}^m(N_{AP}^m + 1)$ is a normalizing factor that ensures that as more users are added, the reward is adjusted more granularly than just linearly. In practice, this is equivalent to having an interval-based reward that adapts to the number of users. The term related to user starvation, as referred to in (5.1.3), is derived using a form of fairness that employs the ω parameter to control what is considered a starving station. This method practically derives a binary variable, such as ξ_{STA}, based on throughput. Note that the term related to station starvation is divided into two parts: one that seeks to maximize the performance of starving stations through fairness, and the other that addresses the performance of non-starving stations via a ratio of the total stations. The third term tries to maximize the fairness of the non-starving stations.

In the next subsection, we present the definition of the context considered in our MA-CMAB solution.

Distributed Sample Average Uncertainty-Sampling MA-CMAB

In [76], the authors present an efficient contextual multi-arm bandit based on a "frequentist approach" to compute uncertainty instead of using Bayesian solutions such as Thompson Sampling. The frequentist approach consists of measuring the uncertainty of the action-values based on the sample average rewards just computed, rather than relying on the posterior distribution given the past rewards. In this work, we introduce multi-agent cooperative and non-cooperative variants of the previously mentioned RL algorithm.

Our problem definition describes the context with only the local observations of the APs:

1. Number of starving stations, $|\Psi_{AP}^m|$, where m corresponds to the m^{th} AP under ω fraction of their attainable throughput during episode t.
2. Average RSSI, $\overline{S}_{AP}^m$, where m is the m^{th} AP during episode t.
3. Average Noise, $\overline{\Upsilon}_{AP}^m$, where m denotes the m^{th} AP during episode t.

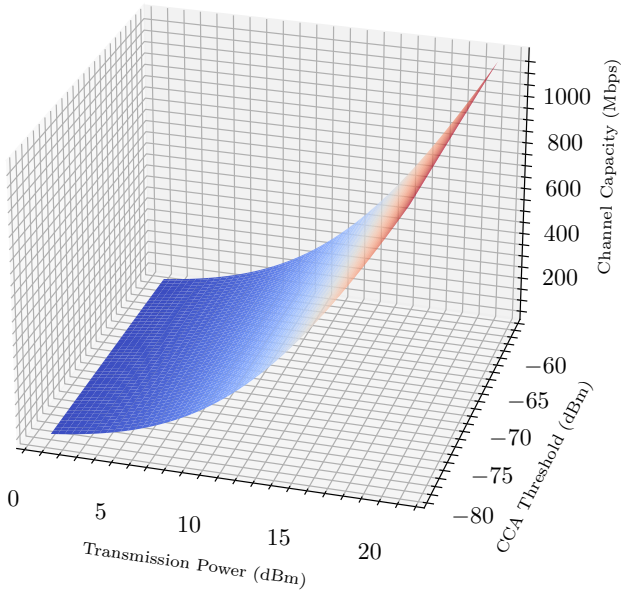

Figure 5.2: Network capacity as a function of TP and CS threshold.

Additionally, the context is normalized as,

$$\psi_{AP}^{m} = |\Psi_{AP}^{m}|/N_{AP}^{m}, \tag{5.37}$$

$$s_{AP}^{m} = \begin{cases} 0, & -50 \text{ dBm} \leq \overline{S}_{AP}^{m} \leq -60 \text{ dBm}, \\ 0.25, & -60 \text{ dBm} \leq \overline{S}_{AP}^{m} \leq -70 \text{ dBm}, \\ 0.5, & -70 \text{ dBm} \leq \overline{S}_{AP}^{m} \leq -80 \text{ dBm}, \\ 0.75, & -80 \text{ dBm} \leq \overline{S}_{AP}^{m} \leq -90 \text{ dBm}, \\ 1, & -90 \text{ dBm} \geq \overline{S}_{AP}^{m}, \end{cases} \tag{5.38}$$

$$\hat{\Upsilon}_{AP}^{m} = \overline{\Upsilon}_{AP}^{m}/100. \tag{5.39}$$

The SAU-Sampling MA-CMAB algorithm, in its non-cooperative version (SAU-NonCoop), is described in Algorithm 14. The algorithm begins with the initialization of action-value functions $\mu(\boldsymbol{x}m|\hat{\boldsymbol{\theta}}m)$ as deep neural networks and the exploration parameters $J_{m,k}^{2}$ and

$n_{m,k}$ for each m^{th} AP. Here, $n_{m,k}$ represents the number of times action a was selected in the m^{th} AP, and $J^2_{m,k}$ is defined as an exploration bonus. In each environmental step (Algorithm 14, `line 2`), each agent observes its local context and computes the selected arm based on the reward prediction. In (Algorithm 5.1, `line 11`), each CMAB agent updates $\hat{\boldsymbol{\theta}}_{m,k}$ using stochastic gradient descent on the loss between the predicted reward and the real observed reward. Finally, the exploration parameters are updated accordingly, given the prediction error, as depicted in (Algorithm 5.1, `line 12`).

Algorithm 5.1 SAU-Sampling MA-CMAB

1: **Initialize** network $\hat{\boldsymbol{\theta}}_m$, exploration parameters $J^2_m(t=0)=1$ and $n_m(t=0)=0$ for all actions $k \in K_m$.
2: **for** environment step $t \leftarrow 1$ **to** T **do**
3: **for** agent m **do**
4: Observe context $\boldsymbol{x}_m(t) = [\psi^m_{AP}(t), s^m_{AP}(t), \hat{\Upsilon}^m_{AP}(t)]$
5: **for** $k = 1, ..., K_m$ **do**
6: Calculate reward prediction $\hat{\mu}_{i,t}(t) = \mu(\boldsymbol{x}_m|\hat{\boldsymbol{\theta}}_m)$ and $\tau^2_{m,k}(t) = J^2_{m,k}/n_{m,k}$
7: $\tilde{\mu}_{m,k} \sim \mathcal{N}(\hat{\mu}_{m,k}, n^{-1}_{m,k}\tau^2_{m,k})$
8: **end for**
9: Compute $a_m(t) = \texttt{argmax}_a(\{\tilde{\mu}_{m,k}(t)\}_{a\in K_m}\})$ if $t > K_m$, otherwise $a_m(t) \sim \mathcal{U}(0, K)$;
10: Select action $a_m(t)$, observe reward r^m_{AP};
11: Update $\hat{\boldsymbol{\theta}}_{m,k}$ using SGD with gradients $\partial l_m/\partial\theta$ where $l_m = 0.5(r^m_{AP} - \hat{\mu}_{m,k}(t))$;
12: Update $J^2_{m,k} \leftarrow J^2_{m,k} + e^2_m$ using prediction error $e_m = r^m_{AP}(t) - \hat{\mu}_{m,k}(t)$ and $n_{m,k} \leftarrow n_{m,k} + 1$;
13: **end for**
14: **end for**

Cooperative Sample Average Uncertainty-Sampling MA-CMAB

In this subsection, we present a cooperative version of SAU-Sampling MA-CMAB named SAU-Coop, which differs from the non-cooperative version by having the total reward r^m_C consider the network Jain's fairness index in addition to their local reward r^m_{AP} as:

$$r^m_C = r^m_{AP} + r_{\mathcal{J}}, \tag{5.40}$$

where $r_{\mathcal{J}}$ as the overall network Jain's fairness index is defined as,

$$r_{\mathcal{J}} = \mathcal{J}(R^1, ..., R^M) = \frac{(\sum_{m=1}^{|\mathcal{M}|} R^m)^2}{|\mathcal{M}| \cdot \sum_{m=1}^{|\mathcal{M}|} (R^m)^2}, \tag{5.41}$$

where $R^m = \sum_{s=1}^{|\mathcal{S}^m|} R_T^{(s,m)}$ is the total throughput of all $|\mathcal{S}^m|$ stations of the m^{th} AP.

Reward-cooperative ϵ-greedy MA-MAB

In addition to the previous cooperative algorithm, we propose a cooperative approach based on the classical ϵ-greedy strategy [7] that takes into account in the action's reward update a percentage of the average reward of other agents. This algorithm is described in Algorithm 5.2.

Algorithm 5.2 Reward-cooperative ϵ-greedy MA-MAB

Initialize $\epsilon_m(t=0) = \epsilon_0$, $Q_{m,k}(t=0) \leftarrow 0$, $N_{m,k}(t=0) \leftarrow 0$ and β.
for environment step $t \leftarrow 1$ **to** T **do**
  for each agent m **do**
    Execute action $a_m(t)$:

$$a_m(t) = \begin{cases} \texttt{argmax}_{k=1,...,K} r_{k,i}(t) & \text{with probability} 1 - \epsilon_m(t) \\ k \sim \mathcal{U}(0, K) & \text{o.w} \end{cases}$$

    Calculate reward $r_{AP}^m(t)$ based on feedback of the environment
    Update $Q_{m,k}(t+1) = Q_{m,k}(t) + \frac{1}{N_m(t)}[(r_{AP}^m + \beta \cdot \frac{1}{M-1} \sum_{m=1}^{M-i} r_{AP}^m) - Q_{m,k}(t)]$
    Update $N_m \leftarrow N_m(t) + 1$;
    Update $\epsilon_m \leftarrow \frac{\epsilon_m(t)}{\sqrt{t}}$
  end for
end for

Finally, in the next subsection, we present the details of the deep transfer reinforcement learning (DTRL) scheme to improve SR adaptation in dynamic environments.

Sample Average Uncertainty-Sampling MA-CMAB based Deep Transfer Reinforcement Learning

Typically, RL agents learn their best policy based on the feedback received from the environment in a T horizon time. However, in real-world scenarios, the environment conditions

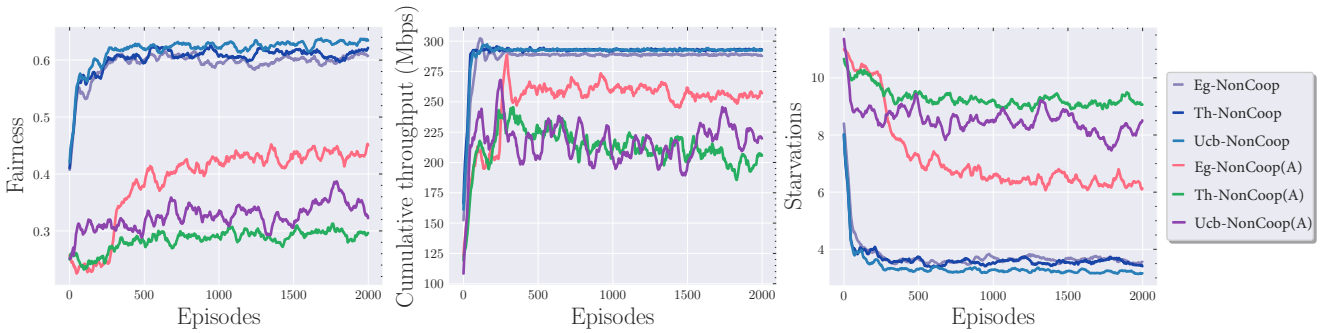

Figure 5.3: Convergence performance of ϵ-greedy (Eg-NonCoop), Thompson Sampling (Th-NonCoop) and UCB (Ucb-NonCoop) MA-MABs under non-cooperative and distributed regimen. (A) indicates the usage of the full set of actions.

can change at $T+1$, and thus, adapting to the updated environment is necessary [178]. In such cases, the "outdated" agent's policy might not be optimal to address the new conditions efficiently. For instance, a modification in the stations' distribution over the APs can cause the SR-related parameters chosen by the "outdated" agent's policy to affect network performance.

To address the previous situation, we propose two main solutions: **1.** If the agent detects a change in the environment indicated by a singularity, it will decide to correct its configuration by forgetting the policy already learned (**forget**) or **2.** adapting the agent's policy to the new conditions via a transfer learning technique. A singularity is defined as an anomalous behavior of the KPIs of interest after the policy of each CMAB agent has converged. In this work, we don't delve into how to detect a singularity, and we assume the existence of an anomaly detector in our system [179]. In Algorithm 5.3, we present the transfer learning algorithm depicting the second proposed solution. At $t = 0$, each SAU-Sampling MA-CMAB agent will reset their weights and biases and start learning as part of Algorithm 14. At $t = S1$, where $S1$ corresponds to the time when an anomaly is detected and the transfer procedure is activated (Algorithm 5.3, `line 7`). In our setup, we transfer $l = 2$ and reset $l = 1$ (Algorithm 5.3, `line 11`), where l corresponds to the layer of the neural network utilized in the SAU-Sampling MA-CMAB agent. However, as indicated (Algorithm 5.3, `line 13`), the transfer is not constrained to one layer but more generally to a set of layers. The set of transferred layers is considered a hyperparameter to be tuned. The partial transfer of a model avoids negative transfer by giving the agent room to adapt to the new context since it mitigates model overfitting.

Algorithm 5.3 SAU-Sampling MA-CMAB Transfer Learning

Function `Detect_Singularity`($\mathcal{K}$) # returns True if anomaly is detected in network KPIs data $\mathcal{K}$ at time t, and False otherwise.
Let $\mathcal{L} = \{l | l \in \mathbb{N}, l > 0\}$ the set of layers of model $\hat{\boldsymbol{\theta}}^{l}_{m,k}$ and $\mathcal{M} \subset \mathcal{L}$ the subset of layers to be transferred.
Run algorithm `SAU-Sampling MA-CMAB (Algorithm 14)`
while environment step $t < T$ **do**
  if $\neg$`Detect_Singularity` **then**
    continue
  else
    Reset exploration parameters $S^2_{m,k}, n_{m,k}$
    Reinitialize weights w and biases b of the l^{th} layer of $\hat{\boldsymbol{\theta}}^{l \notin \mathcal{M}}_{m,k}$ via:
    $\nu_l = \left(\sqrt{|\hat{\boldsymbol{\theta}}^{l \notin \mathcal{M}}_{m,k}|} \right)^{-1}$
    $\hat{\boldsymbol{\theta}}^{l \notin \mathcal{M}}_{m,k}(w, b) \rightarrow w_l \sim \mathcal{U}(-\nu_l, \nu_l), b_l \sim \mathcal{U}(-\nu_l, \nu_l)$
    Transfer weights and biases via:
    $\hat{\boldsymbol{\theta}}^{l \in \mathcal{M}}_{m,k}(w, b) \rightarrow \hat{\boldsymbol{\theta}}^{l \in \mathcal{M}'}_{m,k}(w, b)$
  end if
end while

5.1.5 Performance Evaluation

In this subsection, we intend to demonstrate the positive impact on network KPIs of the reduced action set derived in subsection 5.1.3, as presented in Subsections 5.1.5 and 5.1.5. In addition, we show a comparison among the cooperative and non-cooperative versions of our proposed algorithm and compare with two baselines in Subsection 5.1.5. Furthermore, we leverage a transfer learning approach to avoid starvation in dynamic environments in Subsection 5.1.5. Finally, in subsection 5.1.5, we present a complexity analysis of the MA-CMAB algorithm.

Simulation Settings

We consider two scenarios in our simulations. The first one considers stationary users, while the second scenario considers mobile users to model dynamic scenarios (see subsection 5.1.5). In addition, stations and APs are two-antenna devices supporting up to two spatial streams in transmission and reception. In this work, we assume a frequency of 5 GHz with an 80 MHz channel bandwidth in a Line of Sight (LOS) setting. The propagation loss model is the Log Distance propagation loss model with a constant speed propagation delay. Additionally, an adaptive rate data mode is considered with UDP downlink traffic. The number of runs using different seeds corresponds to 10. We implement our proposed solutions using ns-3, and we also use OpenAI Gym to interface between ns-3 and the MA-MAB solution [131]. In Table 5.2 and Table 5.3, we present the learning hyperparameters and network settings parameters, respectively.

Reduced set of actions vs. all actions

In subsection 5.1.3, we presented a mathematical analysis to obtain a reduced set of optimal actions to decrease exploration time and consequently improve convergence time. As concluded in Fig. 5.2, high TP and low CCA threshold values maximize the network capacity in the simulation scenario under study. Therefore, we selected a fixed value of CCA threshold ($P_{cs} = -82.0$ dBm) and a reduced set of TP ($P_{tx} \in 15, 16, 17, 18, 19, 20, 21$ dBm) and observed the performance against the full set of possible actions described in 5.1.4.

In Fig. 5.3, we present the convergence performance of three MA-MAB algorithms under UDP traffic of 0.056 Gbps in non-cooperative and cooperative settings (indicated with "**Non-coop**"). The algorithms correspond to ϵ-greedy (Eg-NonCoop), UCB (Ucb-NonCoop), and Thompson Sampling (Th-NonCoop) MA-MABs. For each algorithm, we

Table 5.2: Learning hyperparameters

Parameter	**Value**
ϵ-greedy MAB Thompson Sampling MAB Upper Confidence Bound MAB SAU-Sampling	Annealing ϵ: $\sqrt{T}$ Prior distribution: Beta Level of exploration, $c = 1$ Number of hidden layers, $N_{hl} = 2$ Number of neurons per hidden layer, $N_h = 100$ Number of inputs, $\|\boldsymbol{x}_m(t)\|= 3$ and number of outputs, $N_o = K$ Batch size, $B_s = 64$ Optimizer : RMSProp (8e-3) Weight decay : 5e-4 Activation function : ReLU
Gym environment step time	0.05 s

plotted three convergence graphs in terms of fairness, cumulative throughput, and station starvation representing the behavior when a reduced set of actions and the full action set, indicated with (A) are used, respectively. For the case of the set of optimal actions, we can observe that the performance is similar with a slight improvement when utilizing MAB-Thompson Sampling. On the other hand, when utilizing the full action set, the behavior shows a noticeable improvement with the MAB ϵ-greedy algorithm concerning the others. In [181], the authors study the unreasonable behavior of greedy algorithms when K is sufficiently large. They concluded that when K increases above 27 arms, intelligent algorithms are greatly affected by the exploration stage. The former results validate ours based on the fact that $K = |A_{cs}|\cdot|A_{tx}|= 21$[2]. Finally, it can be noted that the impact of utilizing reduced optimal actions in terms of convergence time and KPI maximization. The set of optimal tasks allows reducing station starvation when compared with the best performer Eg-NonCoop by an average of two starving users. However, to obtain such a set, prior knowledge of stations and APs' geographical locations is required. In the following subsection, we compare the results of ϵ-greedy MA-MAB and a default typical configuration without machine learning.

[2]We assume that all APs are configured to use one channel out of the available 11. This is a practical selection to create dense deployment scenarios.

Table 5.3: Network settings

Parameter	Value
Number of APs	6
Number of Stations	15
Number of antennas (AP)	2
Max Supported Tx Spatial Streams	2
Max Supported Tx Spatial Streams	2
Channel Number [2]	1
Propagation Loss Model	Log Distance Propagation Loss Model
Wi-Fi standard	802.11 ax
Frequency	5 GHz
Channel Bandwidth	80 MHz
Traffic Model - UDP application	$[0.011, 0.056, 0.11$ [180], $0.16]$ Gbps
Maximum & minimum Transmission Power	$P_{tx}^{max} = 21.0$ dBm & $P_{tx}^{min} = 1.0$ dBm
Maximum & minimum CCA threshold	$P_{cs}^{max} = -62.0$ dbm & $P_{cs}^{min} = -82.0$ dbm $K_{cs} = 1$ and $K_{tx} = 1$

Distributed ϵ-greedy MA-MAB vs. default configuration performance results

In this subsection, we present the comparative results and advantages of utilizing a distributed intelligent solution such as MAB ϵ-greedy over the default CCA threshold and TP configuration with no ML. In Fig. 5.4, we show the performance under four different UDP data traffic regimes: $\{0.011, 0.056, 0.11, 0.16\}$ Gbps. We considered two typical configurations of the CCA threshold: -82.0 dBm and -62.0 dBm. In both cases, the AP's TP is 16.0 dBm. It can be observed that MAB ϵ-greedy achieves a significant improvement over the default configuration ($P_{cs} = -82.0$ dBm) with an average gain over all the considered traffic of 44.4% in terms of cumulative throughput, 70.9% in terms of station starvation, 12.2% in terms of fairness, 138.0% in terms of latency, and 94.5% in terms of packet loss ratio (PLR), respectively. Additionally, a gain over the default configuration ($P_{cs} = -62.0$ dBm) with an average gain over all the considered traffics of 53.9% in terms of cumulative throughput, 138.4% in terms of station starvation, 43.0% in terms of fairness, 84.0% in terms of latency, and 105.4% in terms of packet loss ratio (PLR) is shown, respectively.

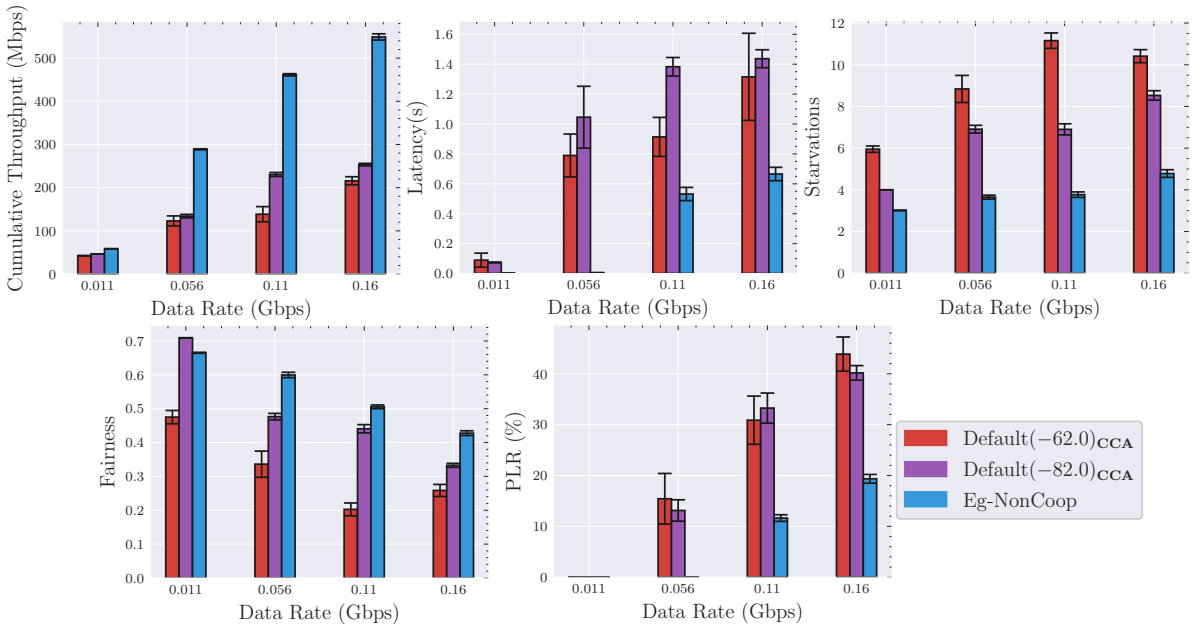

Figure 5.4: Performance results: ϵ-greedy MAB w/ optimal set vs. default configuration with $P_{cs} \in \{-62.0, -82.0\}$ dBm.

Cooperation vs. non-cooperation performance results

In the two past subsections, we have shown the results considering the set of optimal actions. In this subsection, we assume the non-existence of stations and APs location information and thus, we must rely on the full set of actions. Consequently, we investigate if cooperation can improve the KPIs of interest by utilizing the cooperative (indicated by "**Coop**") proposal of the MAB ϵ-greedy algorithm (Rew-Coop) and the SAU-Sampling MA-CMAB algorithm (SAU-Coop). Additionally, we present two non-cooperative algorithms: SAU-NonCoop, which corresponds to the non-cooperative version of the SAU-Sampling MA-CMAB, and Eg-NonCoop, which refers to the MAB ϵ-greedy algorithm utilized in the previous section.

As observed in Fig. 5.5, simulations show that SAU-Coop improves Eg-NonCoop over all the data traffic with an average of 14.7% in terms of cumulative throughput, 21.3% in terms of station starvation, 4.64% in terms of network fairness, 36.7% in terms of latency, and 32.5% in terms of PLR. Similarly, SAU-NonCoop presents a better performance over Eg-NonCoop, indicating that context is beneficial to solving the current optimization problem. Additionally, SAU-Coop presents a better performance over its non-cooperative version,

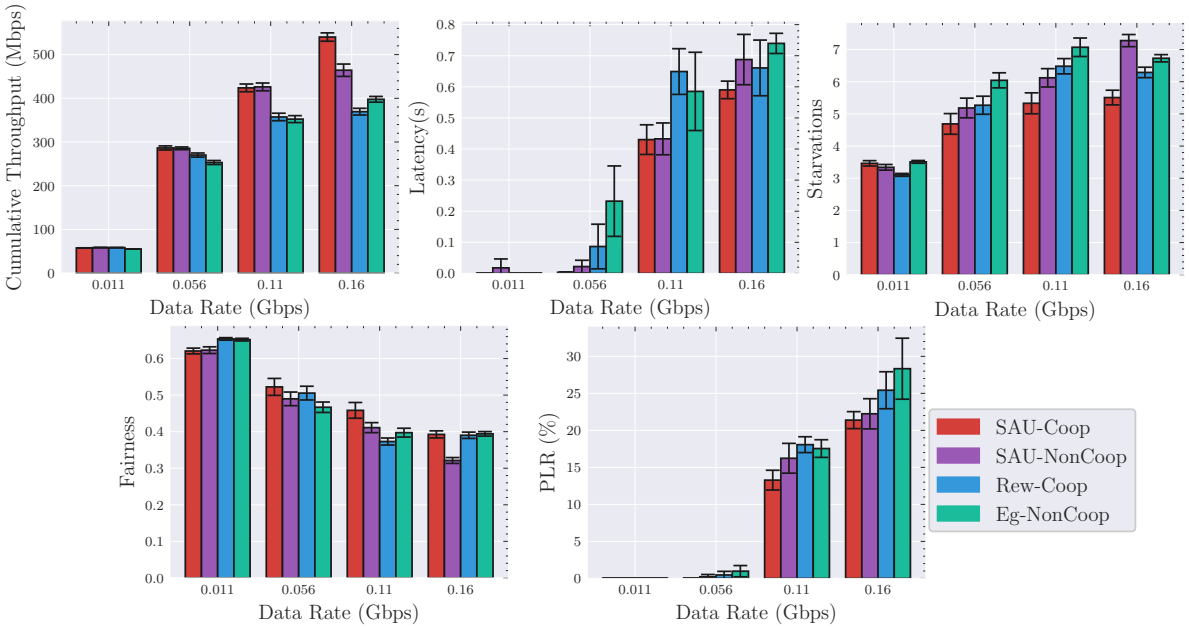

Figure 5.5: Performance results of cooperative algorithms: ϵ-greedy MA-MAB (Rew-Coop), SAU-Sampling MA-CMAB (SAU-Coop) and non-cooperative versions of the previous algorithms SAU-NonCoop and Eg-NonCoop under full-set of actions.

especially when the data rate increases up to 0.16 Gbps where it is observed a gain of 14.1% in terms of cumulative throughput, 32.1% in terms of station starvation, 18.2% in terms of network fairness, 16.5% in terms of latency, and 4% in terms of PLR. To sum up, cooperative approaches contribute positively to the improvement of SR in Wi-Fi over non-cooperative approaches. In addition, in cases where cooperation is not possible, it is advisable to utilize contextual multi-armed bandits over stateless multi-armed bandits.

Deep Transfer Learning in Adaptive SR in Dynamic scenarios results

To model a dynamic scenario, we design a simulation where the users move across the simulation area and attach to the AP that offers the best signal quality. Consequently, the user load in each AP will change, reflecting the dynamics of the environment. We model this scenario with 3 APs and 15 users, where the load will change twice throughout the simulation. As depicted in Table 5.4, the user load of the m^{th} AP, denoted as $\mathcal{L}_m$, will change at two instances in time: 3 and 6 minutes, respectively.

In Fig. 5.6, we present the network behavior in terms of fairness and station starvation

Table 5.4: Dynamic scenario load distribution

	$t = 0$ min	$t = 3$ min	$t = 6$ min
$\mathcal{L}_1$	8	5	2
$\mathcal{L}_2$	5	5	2
$\mathcal{L}_3$	2	5	11

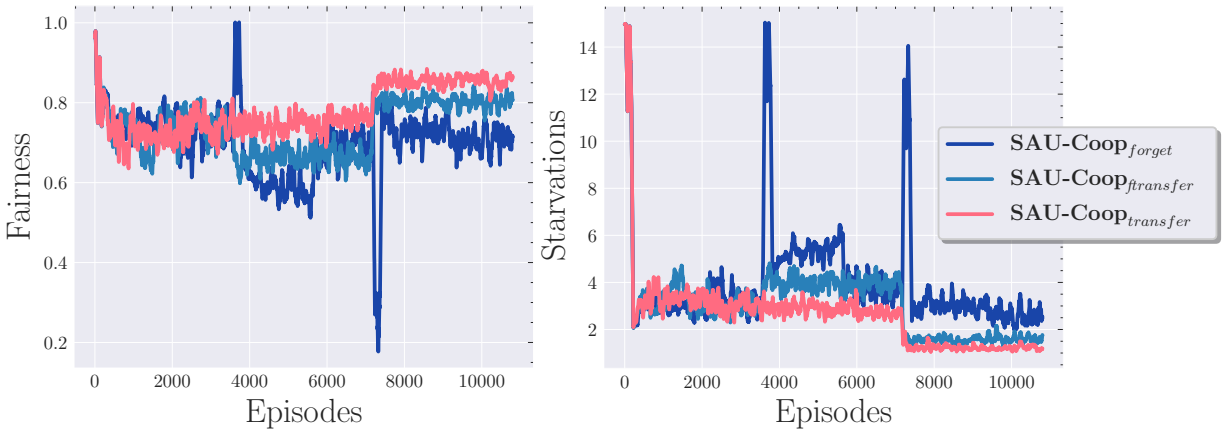

Figure 5.6: Network response in terms of fairness and station starvation when utilizing the forget, full transfer (ftransfer), and transfer strategies.

under the scenario depicted in Table 5.4. In addition to the two methods previously mentioned (**forget** and **transfer**), we introduce the performance of a third approach called **full transfer** (ftransfer), where the complete transfer of the model is considered. During the first interval ($0-3$ min), the performance is similar for all three methods, as expected. However, after the two changes in the network load, singularities in each graph become visible in terms of fairness and starvation.

Specifically, the **forget** method exhibits the worst behavior, with a 54.3% decrease in station starvation and an 11.7% decrease in fairness compared to the **transfer** method. The **forget** method shows peaks at the moments of singularities, representing 60% of the total users with a service drop. This behavior is inherently related to the agents' process of restarting learning and cannot be avoided. From a quality of service perspective, such disturbances are highly undesirable.

Meanwhile, the **full transfer** method underperforms the **transfer** method, showing

an 18.7% decrease in station starvation and a 6% decrease in fairness. Interestingly, in the second interval under study ($3-6$ min), the **forget** method can outperform the **full transfer** method by the end of the period. This is due to negative transfer resulting from transferring the entire model. Partial transfer learning not only significantly reduces the peaks in performance of the **forget** method but also achieves better adaptation than the **full transfer** method. In all methods, the cumulative throughput is similar, but as observed in Fig. 5.6, station starvation, and consequently fairness, are affected.

Complexity analysis

The SAU-Sampling MA-CMAB variants (Coop and NonCoop) are built upon [76], where it has demonstrated an empirical reduction in running time and comparable complexity with other MAB techniques such as TS [182]. More specifically, the SAU metric τ_k^2 introduced in this work resembles TS according to `Proposition 2, page 5` in [76] with an empirical behavior similar to TS as well. This might not be an improvement when compared with the TS MAB from the regret theoretical analysis, but this approach can be adapted to any action-value function since it does not require access to the uncertainty of the expected reward as TS does.

Each agent in the SAU-Sampling MA-CMAB is comprised of a neural network that predicts the reward given the observed context. Such a predictor is composed of 2 hidden layers, the input, and the output layer. Consequently, three matrices are needed to represent the weight relationships of the four layers: W_{ab}, W_{ca}, W_{dc}, where a, b, c, d are the number of nodes of each layer. The propagation from layer a to b can be written in the following fashion:

$$S_{at} = W_{ab} * Z_{bt} \qquad \# \texttt{ Propagation from layer a to b} \tag{5.42}$$

$$Z_{at} = f(S_{at}) \qquad \# \texttt{ Activation function} \tag{5.43}$$

The operation complexity in (5.42) corresponds to $\mathcal{O}(a * b * t)$, meanwhile (5.43) is $\mathcal{O}(a * t)$, which gives a total complexity of $\mathcal{O}(a * b * t)$. Analogously, we can proceed with matrices W_{ca} and W_{dc} and obtain a total time complexity of $\mathcal{O}(n * t * (ab + ca + dc))$, which can be further reduced to $\mathcal{O}(ab + ca + dc)$ since $n = t = 1$ in our MA-CMAB proposal. This corresponds to a low time complexity, thus, having no implications for the performance of the algorithm.

5.2 Meta-Bandit: Spatial Reuse Adaptation via Meta-Learning in Distributed Wi-Fi 802.11ax

Wi-Fi technology has evolved dramatically in the last 10 years with a $16x$ factor increase in terms of throughput from 802.11n to 802.11ax standards. Moreover, in 2022 the number of global Wi-Fi devices in use remounts to nearly 18 billion [183]. This experienced growth altogether with the variety of use cases such as cloud computing, multigigabit streaming, augmented and virtual reality (AR/VR), and telepresence in combination with high dense user deployments has imposed several challenges to Wi-Fi technology [184]. IEEE 802.11ax [185] reduces the impact of the aforementioned issues with an interest in spatial reuse (SR). Improving SR decreases packet collisions among stations and it allows determination of channel access rights. In addition, SR enables a dynamic Clear Channel Assessment (CCA) threshold and Transmission Power (TP) of the Access Point (AP) equipment to improve adaptability. However, dynamic adjustment of such a setting remains an open issue. Furthermore, highly populated areas with mobile end users such as shopping malls, stadiums, public institutions or transportation hubs require effective mechanisms to select TP and CCA thresholds rapidly to maintain Quality of Service (QoS) while improving SR.

Despite the advantages of transfer learning techniques in RL, they are susceptible to negative transfer in which the target task underperforms after receiving knowledge from the source task [110]. To address the aforementioned issue, newer techniques such as meta-learning allow us to quickly adapt to unseen tasks given sufficient source tasks expertise [77].

The rest of this work is organized as follows. Subsection 5.2.1 introduces the system model followed by the proposed MA-CMAB. Subsection 5.2.2 describes the meta-reinforcement learning algorithm, which is followed by the performance evaluation of the meta-learning approach along with discussions in subsection 5.2.3. Finally, subsection 5.2.4 concludes this work.

5.2.1 Multi-Agent Contextual Multi-Armed Bandit algorithm

In this subsection, we start with the introduction of the system model followed by the context definition, action space, and reward function for the MA-CMAB algorithm utilized in this work. In Table 5.5 we present the list of notations.

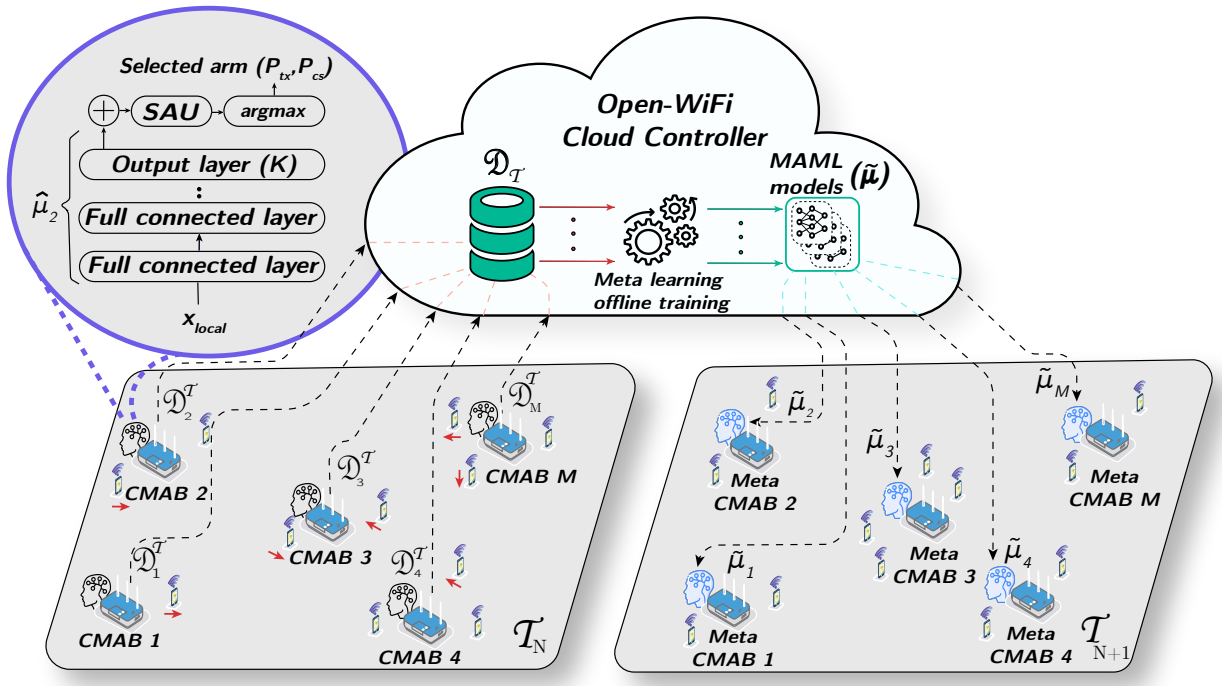

Figure 5.7: Meta-Agent training and deployment workflow in an Open-WiFi architecture.

System Model

We consider a Wi-Fi 802.11ax scenario composed of M APs and S stations that are randomly positioned. In each AP, a CMAB agent resides and can communicate back and forth with the Open-WiFi Cloud Controller. In addition, each station and AP are equipped with two antennas supporting up to two spatial streams in transmission and reception. In this work, we assume 5 GHz frequency with a 80 MHz channel bandwidth in a Line of Sight (LOS) setting. The propagation loss is modeled based on the Log Distance propagation loss model with a constant speed propagation delay. Finally, an SINR-based adaptive rate data model is considered with downlink UDP traffic.

Context definition

Our MA-CMAB solution is based on a contextual multi-arm bandit algorithm introduced in [76]. More specifically, we employ a cooperative multi-agent version of the previous algorithm named Cooperative Sample Average Uncertainty-Sampling (SAU-Coop) [42]. Note

Table 5.5: Notations and definitions.

Notation	Definition
s and $\mathcal{S}$	Index and set of stations
m and $\mathcal{M}$	Index and set of APs
P_{cs}^{m}	CCA threshold of m^{th} AP
P_{tx}^{m}	TP of m^{th} AP
L_{cs} and L_{tx}	Number of levels to be quantized the CCA threshold and TP values
R_{s}^{m}	Throughput of s^{th} STA of m^{th} AP
$R_{s,A}^{m}$	Achievable throughput of s^{th} STA of m^{th} AP
$\lvert\Psi_{m}^{AP}\rvert$, N_{m}^{AP} and ω	Number of starving stations, number of stations attached to the m^{th} AP and a fraction of $R_{s,A}^{m}$ in which STAs are considered in starvation
$\overline{I}_{m}^{AP}$	Average RSSI of m^{th} AP
$\overline{\Upsilon}_{m}^{AP}$	Average Noise of m^{th} AP

that the methods presented in this work are extensible to non-cooperative implementations. The context is composed of the local observations of APs, as such:

1. Number of starving stations, $|\Psi_{m}^{AP}|$ where m stands for the index of an AP under ω fraction of their attainable throughput during the t-th episode.
2. Average RSSI, $\overline{I}_{m}^{AP}$ where m denotes the index of an AP during the t-th episode.
3. Average Noise, $\overline{\Upsilon}_{m}^{AP}$ where m represents the index of an AP during the t-th episode.

Furthermore, each context element is normalized as follows:

$$\psi_{m}^{AP} = |\Psi_{m}^{AP}|/N_{m}^{AP}, \tag{5.44}$$

where N_m^{AP} corresponds to the total of STAs attached to the m^{th} AP. Here, τ_m^{AP} is:

$$\tau_m^{AP} = \begin{cases} 0, & -50 \text{ dBm} \leq \overline{I}_m^{AP} \leq -60 \text{ dBm}, \\ 0.25, & -60 \text{ dBm} \leq \overline{I}_m^{AP} \leq -70 \text{ dBm}, \\ 0.5, & -70 \text{ dBm} \leq \overline{I}_m^{AP} \leq -80 \text{ dBm}, \\ 0.75, & -80 \text{ dBm} \leq \overline{I}_m^{AP} \leq -90 \text{ dBm}, \\ 1, & -90 \text{ dBm} \geq \overline{I}_m^{AP} \end{cases} \tag{5.45}$$

$$\hat{\Upsilon}_m^{AP} = \overline{\Upsilon}_m^{AP}/100 \tag{5.46}$$

where normalized values of the average RSSI and Noise, τ_m^{AP} and $\hat{\Upsilon}_m^{AP}$ are chosen based on the discretization of their maximum values presented in [186]. Finally, the context of each CMAB agent is defined as:

$$\boldsymbol{x}_m(t) = [\psi_m^{AP}(t), \tau_m^{AP}(t), \hat{\Upsilon}_m^{AP}(t)] \tag{5.47}$$

Action space

The action space per m^{th} AP corresponds to the number of combinations of CCA threshold (P_{cs}^m) and TP values (P_{tx}^m) which in the context of MABs translates to the number of arms for each MAB agent. The action space is defined as shown in Eq. 5.48.

$$A_{cs} = \{P_{cs}^{min}, P_{cs}^{min} + \frac{P_{cs}^{max} - P_{cs}^{min}}{L_{cs} - 1}, ..., P_{cs}^{max}\}, \tag{5.48}$$

$$A_{tx} = \{P_{tx}^{min}, P_{tx}^{min} + \frac{P_{tx}^{max} - P_{tx}^{min}}{L_{tx} - 1}, ..., P_{cs}^{max}\}, \tag{5.49}$$

where P_{cs}^{min}, P_{cs}^{max} and P_{tx}^{min}, P_{tx}^{max} are the maximum and minimum values of CCA threshold and TP values, respectively. L_{cs} and L_{tx} correspond to the number of levels to be quantized the CCA threshold and TP values, respectively. Finally, the number of arms corresponding to the action space for the m^{th} agent K_m^{AP} becomes $|A_{cs}^{(m)}| \cdot |A_{tx}^{(m)}|$. Note that the quantization value has an impact on the convergence speed and the agent's capability of achieving the optimal policy. For instance, a small set of actions can truly provide faster convergence but can potentially lead to suboptimal decisions. On the other hand, a large set of actions due to the quantization could produce long training time and suboptimal decisions due to the exploration stage. In our future work, we plan to investigate the impact of quantization.

Reward definition

The following two terms contribute to the reward function: a local reward per AP, r_m^{AP} and the Jain's fairness of the network, $r_{\mathcal{J}}$:

$$r_m^C = r_m^{AP} + r_{\mathcal{J}}, \tag{5.50}$$

$$r_m^{AP} = \frac{|\Psi_m^{AP}| \prod_{j \in \Psi_m^{AP}} \frac{R_m^s}{\omega R_{s,A}^m} + |N_m^{AP} \setminus \Psi_m^{AP}| (N_m^{AP} + \prod_{j \in N_m^{AP} \setminus \Psi_m^{AP}} \frac{R_m^s}{R_{s,A}^m})}{N_m^{AP}(N_m^{AP}+1)}, \tag{5.51}$$

$$r_{\mathcal{J}} = \mathcal{J}(r_1^{AP}, ..., r_M^{AP}) = \frac{(\sum_{m=1}^{M} r_m^{AP})^2}{M \cdot \sum_{m=1}^{M} (r_m^{AP})^2}, \tag{5.52}$$

where Ψ_m^{AP} is the set of starving stations attached to the m^{th} AP, $r_{\mathcal{J}}$ stands for the overall network's Jain's fairness index, and N_m^{AP} denotes the set of stations attached to the m^{th} AP. Finally, R_s^m and $R_{s,A}^m$ are the effective and achievable throughput of the s^{th} station attached to the m^{th} AP, respectively. The cooperative multi-agent SAU-Sampling algorithm is described as follows in Algorithm 14.

Algorithm 5.4 Cooperative Sample Average Uncertainty-Sampling MA-CMAB (SAU-Coop)

Initialize network $\hat{\boldsymbol{\theta}}_{i,a}$, $S_{i,a}^2$ and $n_{i,a}(t=0) = 0$ for all actions $a \in K_m$.
for environment step $t \leftarrow 1$ **to** T **do**
  for each agent i **do**
    Observe context $\boldsymbol{x}_m(t) = [\psi_m^{AP}(t), s_m^{AP}(t), \hat{\Upsilon}_m^{AP}(t)]$
    for $a = 1, ..., K_m$ **do**
      Calculate reward prediction $\hat{\mu}_{i,t}(t) = \mu(x_m|\hat{\boldsymbol{\theta}}_i)$ and $\tau_{i,a}^2(t) = S_{i,a}^2/n_{i,a}$
      $\tilde{\mu}_{i,a} \sim \mathcal{N}(\hat{\mu}_{i,a}, n_{i,a}^{-}\tau_{i,a}^2)$
    end for
    Compute $a_i(t) = \texttt{argmax}_a(\{\tilde{\mu}_{i,a}(t)\}_{a \in K_m})$ if $t > K_m$, otherwise $a_i(t) \sim \mathcal{U}(0, K)$;
    Select action $a_m(t)$, observe reward r_m^C;
    Update $\hat{\boldsymbol{\theta}}_{i,a}$ using SGD with gradients $\partial l_m/\partial\theta$ where $l_m = 0.5(r_m^C - \hat{\mu}_{i,a}(t))$;
    Update $S_{i,a}^2 \leftarrow S_{i,a}^2 + e_m^2$ using prediction error $e_m = r_m^{AP}(t) - \hat{\mu}_{i,a}(t)$ and $n_{i,a} \leftarrow n_{i,a} + 1$;
  end for
end for

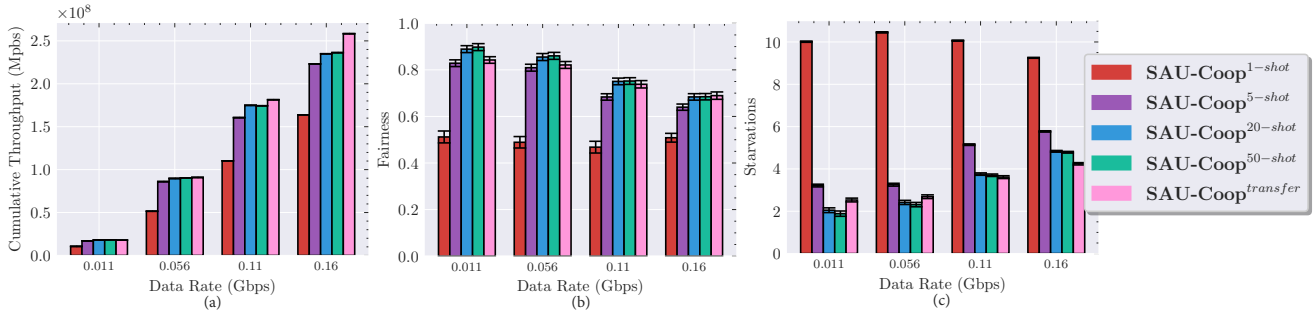

Figure 5.8: Network Key Performance Indicators for $0.001, 0.056, 0.11, 0.16$ Gbps traffic regimes in terms of a) Cumulative Throughput, b) Fairness, and c) User starvation. Each figure shows the performance of one-shot, 5-shot, 20-shot, and 50-shot for the meta-bandit and the transfer learning baseline.

5.2.2 Meta-Reinforcement Learning Algorithm

In this section, we introduce our Meta-Bandit scheme that leverages the previously presented MA-CMAB and MAML. The proposed algorithm consists of two offline and one online stage: **1.** Meta-Task datasets generation, **2.** Meta-Training and **3.** Meta-Adaptation.

Meta-Task dataset generation

Meta-learning requires a large number of meta-tasks to train the meta-agent [77]. In our case study, the Open-cloud controller should gather enough experience based on the numerous independent learning environments to become ready for the Meta-Training stage. In this work, the meta-tasks are involved in scenarios that face different station distributions and thus, different loads in terms of attached stations to each AP. We leverage the internal structure of the SAU-Sampling CMAB and store the context $\boldsymbol{x}_t = (\boldsymbol{x}_{1,t}, ..., \boldsymbol{x}_{M,t})$, action selected $\boldsymbol{a}_t = (\boldsymbol{a}_{1,t}, ..., \boldsymbol{a}_{M,t})$ and predicted reward $\tilde{\boldsymbol{\mu}}_t = (\mu(\boldsymbol{x}_{1,t}|\hat{\boldsymbol{\theta}}_1) + \texttt{SAU}_1)$,..., $\mu(x_{M,t}|\hat{\boldsymbol{\theta}}_M) + \texttt{SAU}_M)$ per agent in each simulated load. The internal structure is comprised of a neural network that receives an environment context $x_{M,t}$ and maps such context to a reward via $\mu(x_{M,t}|\hat{\boldsymbol{\theta}}_M)$. Three datasets per agent are utilized in the Meta-Training stage: *train* defined as $\boldsymbol{\mathcal{D}}^{(m)}_{train} = \{\mathcal{D}^{(m)}_1, ..., \mathcal{D}^{(m)}_{Tr}\} | \boldsymbol{\mathcal{D}}^{(m)}_{train} \subset \boldsymbol{\mathcal{D}}, Tr = |\boldsymbol{\mathcal{D}}^{(m)}_{train}|$, *test* $\boldsymbol{\mathcal{D}}^{(m)}_{test} = \{\mathcal{D}^{(m)}_1, ..., \mathcal{D}^{(m)}_{Te}\} | \boldsymbol{\mathcal{D}}^{(m)}_{test} \subset \boldsymbol{\mathcal{D}}, Te = |\boldsymbol{\mathcal{D}}^{(m)}_{test}|$ and *validation* $\boldsymbol{\mathcal{D}}^{(m)}_{val} = \{\mathcal{D}^{(m)}_1, ..., \mathcal{D}^{(m)}_{Val}\} | \boldsymbol{\mathcal{D}}^{(m)}_{val} \subset \boldsymbol{\mathcal{D}}, Val = |\boldsymbol{\mathcal{D}}^{(m)}_{val}|$ where $\boldsymbol{\mathcal{D}}$ is the database of all datasets. Such datasets are assumed to be available at the Open-WiFi controller for further training in the second stage.

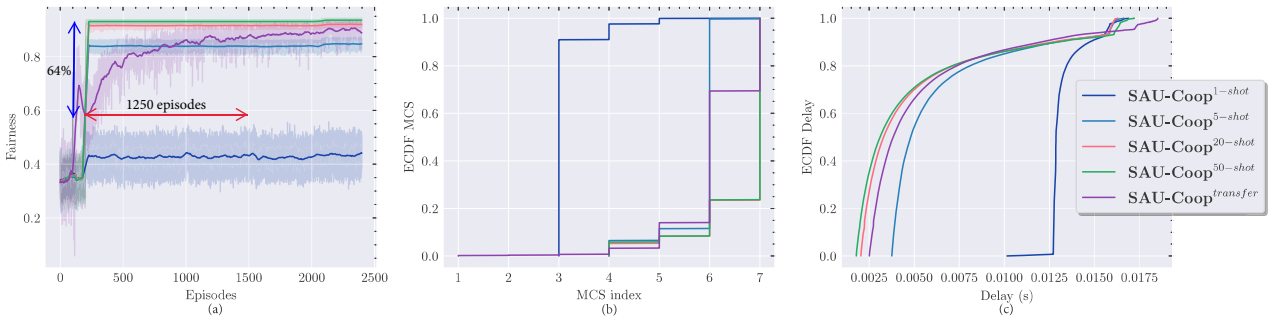

Figure 5.9: Network Key Performance Indicators for 0.001 Gbps traffic regimen for the meta-agent and transfer learning baseline: a) Fairness convergence graph, b) Empirical Cumulative Distribution Function (ECDF) for the Modulation Coding Scheme (MCS) index c) ECDF for the Delay. The red double arrow in a) indicates the improvement in terms of convergence time of the meta-agents over the transfer learning.

Meta-Training

The SAU-Sampling CMAB maps its context onto the rewards via NNs. In addition, it considers the action taken by the m-th agent and uses it as a one-hot vector at training time. The network structure utilized for the meta-agent is defined as follows:

$$f_{\psi_m}(\boldsymbol{x}) = h^{out}_{\psi_m} \circ h_{\psi_{m,K}} \circ \cdots \circ h_{\psi_{m,2}} \circ h_{\psi_{m,1}(\boldsymbol{x})} \tag{5.53}$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}^p$, $h_{\psi_{m,K}} : \mathbb{R}^{n_m} \rightarrow \mathbb{R}^{p_m}$, $h_{\psi_{m,K}}(\boldsymbol{x}) = \text{ReLU}(w_i\boldsymbol{x} + b_i)$. $h_{\psi_{m,1}}(\boldsymbol{x})$ stands for the input layer, n denotes the dimension of the context $\boldsymbol{x}$, p the dimension corresponding to the number of each agent's K arms.

In this case, a supervised multi-output regression is still in effect. Thus, the general form of the loss function is defined as the mean-squared error taking the form of:

$$\mathcal{L}(f_\theta) = \sum_{\boldsymbol{x},\boldsymbol{a},\tilde{\boldsymbol{\mu}} \sim \mathcal{D}_d} ||\boldsymbol{a} f_\theta(\boldsymbol{x}) - \hat{\boldsymbol{\mu}}||_2^2, \tag{5.54}$$

where $\boldsymbol{x}, \boldsymbol{a}, \tilde{\boldsymbol{\mu}}$ are the context, action, and predicted output sampled from the d^{th} dataset and m^{th} agent. The output of the algorithm corresponds to a meta-agent represented by the parameterized function $\psi^{(m,i)}$ and θ (see Algorithm 5.5, `line 10`) capable of adapting with few examples. The meta-learning algorithm named Meta-Bandit is described in Algorithm 5.5.

Algorithm 5.5 Meta-Bandit Algorithm

Require: Database $\boldsymbol{\mathcal{D}}$, Tr, Te, Val meta batch sizes, step sizes α, α_{meta}
Initialize: $\theta \rightarrow w_l \sim \mathcal{U}(-\nu_l, \nu_l), b_l \sim \mathcal{U}(-\nu_l, \nu_l)$, l number of layers
while not `done` **do**
  Draw train, test and validation meta-batches: $\boldsymbol{\mathcal{D}}_{train}^{(m)} \sim \boldsymbol{\mathcal{D}}$ of Tr datasets, $\boldsymbol{\mathcal{D}}_{test}^{(m)} \sim \boldsymbol{\mathcal{D}}$ of Te datasets and $\boldsymbol{\mathcal{D}}_{val}^{(m)} \sim \boldsymbol{\mathcal{D}}$ of Val datasets
  for $\mathcal{D}_{train}^{(m,i)}$ in $\boldsymbol{\mathcal{D}}_{train}^{(m)}$ **do**
    Sample P datapoints $\hat{\mathcal{D}}_{train}$ from $\mathcal{D}_{train}^{(m,i)}$
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{D}_{train}^{(m,i)}}(f_\theta)$ using P and $\hat{\mathcal{D}}_{train}$
    Compute $\psi^{(m,i)} = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{D}_{train}^{(m,i)}}(f_\theta)$
  end for
  Update $\theta \leftarrow \theta - \alpha_{meta} \nabla_\theta \sum_i^{\boldsymbol{\mathcal{D}}_{test}^{(m)}} \sum_{P' \sim \boldsymbol{\mathcal{D}}_{test}^{(m,i)}} \mathcal{L}_{\boldsymbol{\mathcal{D}}_{test}^{(m)}}(\psi^{(m,i)})$
end while
 `# Evaluate the loss against the validation dataset`
for $\mathcal{D}_{val}^{(m,i)}$ in $\boldsymbol{\mathcal{D}}_{val}^{(m)}$ **do**
  Sample P'' datapoints $\hat{\mathcal{D}}_{val}$ from $\mathcal{D}_{val}^{(m,i)}$
  Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{D}_{val}^{(m,i)}}(f_\theta)$ using P'' and $\hat{\mathcal{D}}_{val}$
end for

Meta-Adaptation

The third stage of the algorithm, named Meta-Adaptation, is performed in an online manner. After each agent of the MA-CMAB scheme experiences F samples, a k-shot sampling procedure will follow to be further utilized in the meta-adaptation. Finally, each meta-agent will adapt its model as follows:

$$\theta \leftarrow \theta - \alpha_{meta} \nabla_{\theta} \sum_{k \sim \mathcal{F}^{(m)}} \mathcal{L}_{\mathcal{F}^{(m)}}(\psi^{(m,i)}), \tag{5.55}$$

where $\mathcal{F}^{(m)}$ is the set of F samples from the m-th AP.

Complexity analysis

The complexity analysis must be performed separately for the different stages of our proposal. In addition, we will consider regret, as the standard metric to compare bandits. As mentioned before, our SAU-Coop algorithm is built upon [76] where it has been proved an empirical reduction in running time and comparable complexity with other MAB techniques such as TS [182]. More specifically, the SAU metric τ_a^2 introduced in this work, resembles TS according to (`Proposition 2, page 5`) [76] with an empirical behavior similar to TS as well. This might not be an improvement when compared with the TS MAB from the regret theoretical analysis but this approach can be adapted to any action-value function since does not require access to the uncertainty of the expected reward as TS does. The expected regret is logarithmically bounded as:

$$\sum_{a:\mu_a<\mu_*} \Delta_a \left(\frac{96 \log n}{\Delta_a^2} + 6 \right), \tag{5.56}$$

where μ_a and μ_* corresponds the expected reward for action a and $\mu_* = max\{\mu_1, ..., \mu_K\}$, respectively. $a \in K$ refers to the action taken among the possible K arms and n the number of times where the SAU chooses an action. Finally, Δ_a is defined as $\Delta_a = \mu_* - \mu_a$.

On the other hand, the MAML's time complexity corresponds to $\mathcal{O}(d^2)$ where d is the problem dimension. The quadratic term indicates the outer and inner loop updates of the MAML algorithm which can be quite costly if the set of the model's parameter dimension is large. However, the MAML training is performed offline, thus we can ignore it. On the other hand, in the online stage described in subsection 5.2.2, the meta-adaptation is performed via gradient descent with a complexity of $\mathcal{O}(knd)$ where k, n and d corresponds

to the number of iterations, number of samples and number of features, respectively. If d increases the complexity will become an issue, however, this is not the case and consequently, performance is not affected. Finally, the time complexity analysis of our proposed SAU-Coop scheme can be found in the **Complexity Analysis** subsection of 5.1.5.

5.2.3 Performance Evaluation

In this subsection, we present the simulation settings and baseline model under consideration.

Simulation Settings

The dataset is generated by simulating S random scenarios with different user loads per AP via the discrete network simulator, ns-3. OpenAI Gym [131] is used to interface between ns-3 and the MA-CMAB solution. In Table 5.6 and Table 5.7, we present the learning hyperparameters and network settings for the dataset generation and online meta-adaptation.

Baseline Model

We consider a transfer learning-based baseline model that is already proposed in [42]. Specifically, we utilize partial transfer learning where specific layers of the CMAB agents are transferred from task $\mathcal{T}_N$ to $\mathcal{T}_{N+1}$ instead of transferring the full model. The selection of the CMAB neural network hidden layers $\boldsymbol{l}_t$ to be transferred is considered as an additional hyperparameter. The previous approach reduced the model overfitting allowing an improved adaptive behavior.

5.2.4 Simulation results

In this subsection, we evaluate the main network KPIs for the meta-bandit under different k-shot sampling strategies to select the optimal k-shot configuration and the transfer learning baseline while $k \in \{1, 5, 20, 50\}$. Each meta-bandit is trained with $S = 200$ random tasks where the train, test, and validation split is $0.6 : 0.2 : 0.2$. In the online stage, 50 runs per traffic mode are performed with 10 random unseen scenarios per run to observe the performance of the trained meta-bandit. Similarly, the transfer learning strategy uses the

Table 5.6: Network settings

Parameter	**Value**
Number of APs	6
Number of Stations	Variable - Max 20 users
Number of antennas (AP)	2
Max Supported Tx Spatial Streams	2
Max Supported Rx Spatial Streams	2
Channel Number	1
Propagation Loss Model	Log Distance Propagation Loss Model
Wi-Fi standard	802.11 ax
Frequency	5 GHz
Channel Bandwidth	80 MHz
Traffic Model - UDP application	$[0.011, 0.056, 0.11, 0.16]$ Gbps
Maximum & minimum TP	$P_{tx}^{max} = 21.0$ dBm & $P_{tx}^{max} = 1.0$ dBm
Maximum & minimum CCA threshold	$P_{cs}^{max} = -62.0$ dbm & $P_{cs}^{max} = -82.0$ dbm
	$K_{cs} = 1$ and $K_{tx} = 1$

Table 5.7: Learning hyperparameters (determined based on hyperparameter search)

Parameter	**Value**
SAU-Sampling	Number of hidden layers, $N_h = 2$ Number of neurons per hidden layer, $n_h = 100$ Number of inputs, $N_m = 3$ and number of outputs, $N_o = K$ Batch size, $B_s = 64$ Optimizer : Adam (8e-3) Weight decay : 5e-4 Activation function : ReLU
Transfer learning	Transferred layers, $l_h \in \{2\}$
Meta-learning	Number of training tasks, $S = 200$ Total samples before meta-adaptation, $F = 100$ Inner learning rate $\alpha_{in} = 1$ Meta-learning rate $\alpha_{meta} = 0.001$
Gym environment step time	0.05 s
Number of runs in the online stage	50
Number of simulation/run	10

first trained policy per run as the transfer policy to the rest of the nine unseen scenarios. In Fig. 5.8, we present the results for the meta-bandit and the transfer learning baseline in the online stage under different traffic modes regimen at 95% confidence level.

Figure 5.8(a) depicts the network fairness performance of the techniques under study. One-shot is outperformed in all cases whereas the best performance can be achieved under 50-shot or the transfer learning technique. It is worth noting that the transfer learning technique achieves better throughput performance for two reasons: 1) it does not modify the internal NN structure of the CMAB agents, and 2)it has longer fine-tuning steps. A slightly different behavior is observed in Fig. 5.8(b) and 5.8(c) for fairness and user starvation, respectively. For traffic load of 0.011 and 0.056 Gbps, the transfer learning exhibits the worst performance, especially in terms of starvation, as a product of negative transfer learning. As previously mentioned, negative transfer occurs following the transfer of a source policy to a target policy resulting in degraded performance of the target policy. Apparently, the selection of the source policy has a significant impact on the transfer learning algorithm. In this work, we select a random policy to be transferred which may lead to the negative transfer phenomena under some scenarios. Selection of the best source policy remains an open issue that can be tackled with intelligent RL solutions [187].

Figure 5.9 presents the impact of the meta-learning approach against the transfer learning baseline in terms of fairness convergence, MCS index and delay at 95% confidence level and 0.001 Gbps data traffic. In Fig. 5.9(a), it can be seen that the *one*-shot sampling strategy is not capable of adapting successfully. On the other hand, for $k \in \{5, 20, 50\}$, successful adaptation can be observed. In addition to the previous results, we show a red and a blue arrow indicating the improvement in terms of convergence and adaptability over the transfer learning baseline, respectively. We observe that in terms of convergence, an average of less than 1,250 environment steps are needed, for the proposed techniques. Furthermore, in terms of fairness and starvation, 64% and 80% improvement over the transfer learning baseline is achieved, respectively. In Fig. 5.9(b) the ECDF of the MCS index indicates that the best performance is obtained with a 50-shot sampling strategy. In the same fashion, Fig. 5.9(c) depicts the behavior of the ECDF delay overall strategies, with 50-shot standing out as the best strategy. Other convergence graphs under different traffic regimes are not presented; however, their behavior is similar, with the caveat that the impact of negative transfer can be observed.

5.3 RL meets Multi-Link Operation in IEEE 802.11be: Multi-Headed Recurrent Soft-Actor Critic-based Traffic Allocation

Wireless Fidelity (Wi-Fi) technology has experienced a rapid evolution in the last four years with the introduction of Wi-Fi 6 and Wi-Fi 6E in 2019 and 2021, respectively. This technological development momentum seems to keep going and IEEE 802.11be — Extremely High Throughput (EHT)— known commercially as Wi-Fi 7 is expected to become reality by 2024. Even more, IEEE 802.11 has started in 2022 the first conversations about devising the next generation of Wi-Fi (named Wi-Fi 8) believed to be released by 2028 [188]. The still-under-development IEEE 802.11be amendment is considered with the goal of dealing with highly congested network environments with stringent requirements in terms of high throughput and low latency. To do so, the amendment proposes to increase the channel bandwidth up to 320 MHz, a higher modulation rate up to 4096 QAM and different than all other Wi-Fi generations, Wi-Fi 7 introduces Multi-Link Operation (MLO) and Multiple Resource Units (MRU) capabilities.

In this work, we focus our interest on MLO in 802.11be. Specifically, MLO allows Multi-Link Devices (MLD)s to concurrently use their available interfaces for multi-link communications. In this context, we intend to optimize the traffic allocation policy over the available interfaces with the aid of RL. RL has proven its effectiveness in dealing with challenging problems in wireless networks [46]. Thus, in this work, we propose to utilize a Soft-Actor Critic (SAC) algorithm named Multi-Headed Recurrent SAC (MH-RSAC). We choose the SAC algorithm over other Actor-Critic methods such as *Advantage Actor Critic* (A2C) or *Proximal Policy Optimization* (PPO) due to its consistent performance in many RL challenging tasks [189]. Differently from others, the SAC algorithm maximizes its rewards altogether with the entropy which benefits the agent's exploration. We propose for the first time, to the best of our knowledge, a novel SAC-based traffic-to-link allocation policy in 802.11be MLO capable networks. In Fig. 5.10, we show the scenario and system overview of our proposed scheme. Here, each agent residing in the Access Point (AP) equipment, decides the incoming traffic distribution percentage (a1, a2, a3) according to the MLO capabilities. For instance, if the number of interfaces of the end station corresponds to $n_f > 1$, the agent will propose a set of actions for the case of two and three interfaces, respectively. Finally, the function $U(n_f)$ selects the final action based on the actual interface number. Note that, when $n_f = 1$ the agent is not involved since all the traffic is passed to the only available interface. This design allows to have one agent per AP instead of two. In addition, we consider the non-Markovian behavior of the scenario and

include a recurrent neural network in the model design. Moreover, we improve even further the performance of our agent by utilizing a modified reward function named *rewards with hindsight* that considers the goals of the baselines. Finally, we utilize in our simulations, three traffic flow types: Web Browsing (WB), High Definition (HD)/ Ultra High Definition (4K), and Virtual Reality (VR). For the latter, we derive the VR CDFs based on [190] and present the given equations to be utilized in any wireless simulator.

We compare our results with two non-machine learning type baselines presented in [115] named *Single Link Less Congested Interface* (SLCI), *Multi Link Congestion-aware Load balancing at flow arrivals* (MCAA), respectively. Our results show an improvement in terms of Throughput Drop Ratio (TDR) with a gain of up to 35.2% and 6% when compared with SLCI, MCAA non-RL baselines, respectively. TDR is defined as the percentage of traffic in terms of throughput dropped after a traffic allocation decision. Finally, we observed that our scheme can respond more efficiently to high throughput and dynamic traffic such as VR and Web Browsing (WB) with an improvement in terms of Flow Satisfaction (FS) up to 25.6% and 6% for the SCLI and MCAA, respectively.

The rest of this work is organized as follows. The system model is demonstrated in subsection 5.3.1. subsection 5.3.2 describes the MH-RSAC scheme and the considerations taken on its design, including the Markov Decision Process (MDP) and the description of the baselines. Subsection 5.3.3 depicts the traffic considerations in this work. Finally, subsection 5.3.4 presents our proposed scheme's performance evaluation and comparison with the baselines.

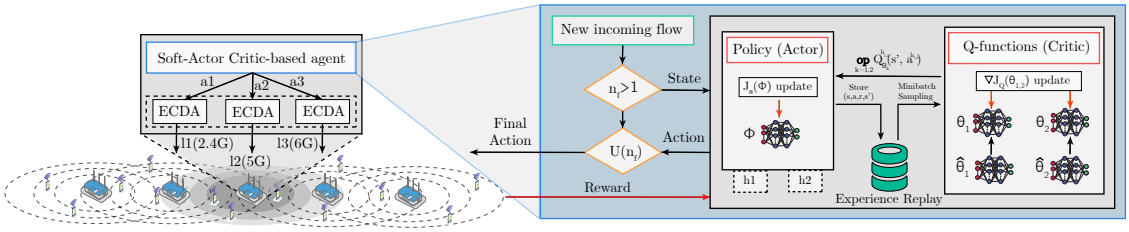

Figure 5.10: Overview of the MH-RSAC based traffic allocation policy in IEEE 802.11be MLO: Upon an incoming flow the agent will decide the percentage (a1, a2, a3) of traffic flow allocated to each available interface (l1, l2, l3) based on the observed state.

5.3.1 System Model

In this work, we utilize an IEEE 802.11be network with a predefined set of M APs and N stations attached per AP. In addition, we consider all APs to be MLO-capable with

a number of available interfaces $n_f = 3$ whereas in the case of the stations, the MLO capability can vary. This design decision is taken based on the fact that single-device and multi-device terminals will coexist in real scenarios due to terminal manufacturer diversity. Furthermore, stations are positioned in two ways to their attached AP: 80% of the users are positioned randomly in a radius $r \sim [1 - 8]$ m and the rest with a radius $r \sim [1 - 3]$ m. All APs and stations support up to a maximum of 16-SU multiple-input multiple-output (MIMO) spatial streams. The path loss model corresponds to the enterprise model described in [191]. In addition, we consider an adaptive control rate based on Signal to Noise Ratio (SNR) with a maximum 4096 QAM modulation rate.

5.3.2 Multi-Headed Recurrent Soft-Actor Critic

In the current subsection, we discuss the details and considerations taken in the design of the Multi-Headed Recurrent Soft-Actor Critic (MH-RSAC) agent.

Soft-Actor Critic

The SAC agent introduced initially in [189] is a maximum entropy, model-free, and off-policy actor-critic method that outperformed most of the *state-of-the-art* RL algorithms such as *Twin Delayed Deep Deterministic Policy Gradient* (TD3) and PPO. The success of SAC relies on the inclusion of a policy entropy term into the reward function to encourage exploration. In addition, it reutilizes the successful experience of the minimum operator in the selection of the double Q-Functions that comprise the critic in the *Deep Deterministic Policy Gradient* (DDPG) algorithm [56].

In this work, we utilize a discrete SAC agent presented in [192] that provides an adaptation of the original SAC agent to the discrete action space. Moreover, we take into consideration the results obtained recently in [193]. In such work, the authors study the discrete SAC and find that such an algorithm is affected by Q-value underestimation due to the usage of the minimum operator. In consequence, they propose to substitute the minimum operator with an average operator which allows the reduction of the bias of the lower bound of the double critics. In addition to the previous considerations, we modify the structure of the critic and the actor of the discrete SAC to allow multi-output. To do so, we diverge the model into two heads as indicated by Fig. 5.11: one head (Head 1) producing the actions when two interfaces are available at the station end and another (Head 2) when three are available. A detailed description of the algorithm is found in Algorithm 5.6. The reason behind this design decision corresponds to the practical assumption, briefly mentioned in the introduction of this section, that terminals with diverse

MLO capabilities will coexist. Thus, instead of utilizing two agents handling each case, we propose to utilize one agent capable of deciding in both circumstances. In such a way, we can reduce the computational load required to train and run two agents simultaneously.

Dealing with Non-Markovian environments in RL

Reinforcement Learning faces some challenges when dealing with non-Markovian environments. The reason is that RL main goal is to maximize the reward R_{t+1} given a certain state S_t and action A_t as $R = E[R_{t+1}|S_t, A_t]$. When the previous relationship is not fulfilled due to partial observability of the Markovian state, the observation state should include more than one input and utilize a portion of the interaction history [194, 195].

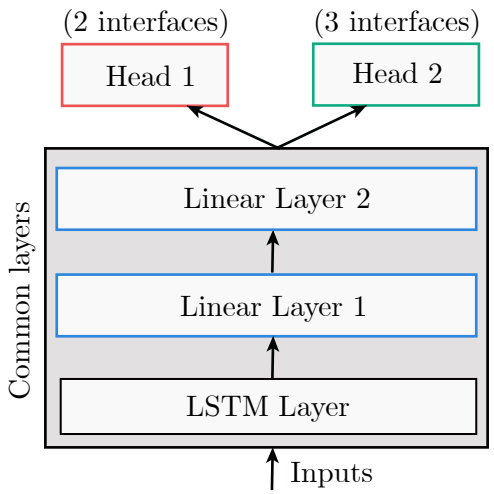

Figure 5.11: Network structure used in the Multi-Headed Recurrent Soft-Actor Critic agent.

In Fig. 5.12, we show an example of the non-Markovian behavior of our environment. For instance, the reward obtained upon the arrival of Flow 1, $r1$ is affected by the arrival of Flow 2. That means that the reward obtained given the action taken upon Flow 1 arrival is affected as well by the action taken upon Flow 2 arrival as well. Such behavior violates the assumption of being in the presence of an MDP. To solve the aforementioned issue, we include as the first layer of the model's structure of the MH-RSAC a *Long Short-Term Memory* (LSTM) neural network as shown in Fig. 5.11. In addition, we add two more linear layers after the LSTM layer.

Algorithm 5.6 Multi-Headed Recurrent Soft-Actor Critic

Initialize actor's policy ϕ, critic's Q-function parameters θ_1, θ_2 and experience replay buffer $\mathcal{D}$. Set critic's Q-target function equal to main parameters $\hat{\theta}_1 \leftarrow \theta_1$ and $\hat{\theta}_2 \leftarrow \theta_2$. $\boldsymbol{op} \in \{\texttt{min, avg}\}$ corresponds to the operator, L_{per} the update periodicity, n_{up} number of updates per learning step and $\rho \in [0, 1]$ is the Polyak factor.
for environment step $t \leftarrow 1$ **to** T **do**
  Observe state $s = [C_1^o, C_2^o, C_3^o, O_f, T_{id}]$ and select action for each head $a^{h_{1,2}} \sim \pi_\phi^{h_{1,2}}(\cdot|s)$
  Execute a in the environment
  Observe next state s' and reward r
  Store (s, a, r, s') in experience replay buffer $\mathcal{D}$
  if t **mod** $L_{per} = 0$ **then**
    for $n \leftarrow 1$ to n_{up} **do**
      Randomly sample a batch of transitions with size B from $\mathcal{D}$
      Compute targets for the Q-functions (Critic) for each head $h_{1,2}$:
      $y_{1,2}(r, s') = r + \gamma \left(\boldsymbol{op}_{k=1,2} Q_{\hat{\theta}_{1,2}}^{h_{1,2}}(s', \tilde{a}'^{h_{1,2}}) - \alpha \log \pi_\phi^{h_{1,2}}(\tilde{a}'^{h_{1,2}}|s') \right)$ where $\tilde{a}'^{h_{1,2}} \sim \pi_\phi^{h_{1,2}}(\cdot|s')$
      Update Q-functions (Critic) considering each head $h_{1,2}$:
      $L_1^{h_{1,2}}(\theta_1, \mathcal{D}) = E_{(s,a,r,s')\in B}[(Q_{\theta_1}^{h_{1,2}}(s, a^{h_{1,2}}) - y_1(r, s'))^2]$
      $L_2^{h_{1,2}}(\theta_2, \mathcal{D}) = E_{(s,a,r,s')\in B}[(Q_{\theta_2}^{h_{1,2}}(s, a^{h_{1,2}}) - y_2(r, s'))^2]$
      $L(\theta^{h_{1,2}}, \mathcal{D}) = L_1^{h_{1,2}}(\theta_1, \mathcal{D}) + L_2^{h_{1,2}}(\theta_2, \mathcal{D})$
      Update policy (Actor) considering each head $h_{1,2}$:
      $L_1^{h_{1,2}}(\phi^{h_1}, \mathcal{D}) = E_{(s)\in B}[\boldsymbol{op}_{k=1,2} Q_{\hat{\theta}_{1,k}}^{h_{1,2}}(s, \tilde{a}'^{h_{1,2}}) - \alpha \log \pi_\phi^{h_{1,2}}(\tilde{a}'^{h_{1,2}}|s')]$
      $L_2^{h_{1,2}}(\phi^{h_2}, \mathcal{D}) = E_{(s)\in B}[\boldsymbol{op}_{k=1,2} Q_{\hat{\theta}_{1,k}}^{h_{1,2}}(s, \tilde{a}'^{h_{1,2}}) - \alpha \log \pi_\phi^{h_{1,2}}(\tilde{a}'^{h_{1,2}}|s')]$
      $L(\phi^{h_{1,2}}, \mathcal{D}) = L_1^{h_{1,2}}(\phi^{h_1}, \mathcal{D}) + L_2^{h_{1,2}}(\phi^{h_2}, \mathcal{D})$
      Finally, update Q-functions target networks with Polyak averaging:
      $\hat{\theta}_i \leftarrow \rho\hat{\theta}_i + (1 - \rho)\theta_i$ for $i = 1, 2$
    end for
  end if
end for

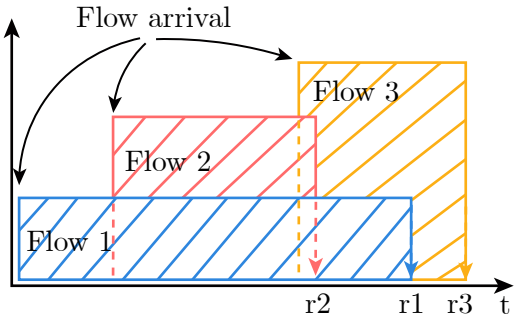

Figure 5.12: Non-Markovian nature of the scenario: Rewards are not only dependent on the current state but affected by other arrival flows.

State space selection

As discussed in previous subsections, the environment is modeled as a Partially Observable Markov Decision Process (POMDP), thus a sequence of the observation space is considered instead of one instance. The state space is defined as:

$$\begin{aligned} \boldsymbol{s}_t = [&\{C^o_{1,t-W}, C^o_{2,t-W}, C^o_{3,t-W}, O_{f,t-W}, T_{id,t-W}\}, \\ &\{C^o_{1,t-W-1}, C^o_{2,t-W-1}, C^o_{3,t-W-1}, O_{f,t-W-1}, \\ &T_{id,t-W-1}\}, ..., \{C^o_{1,t}, C^o_{2,t}, C^o_{3,t}, O_{f,t}, T_{id,t}\}], \end{aligned} \tag{5.57}$$

where W corresponds to the size of the interaction window, C^o_1, C^o_2, C_3 the occupancy of 2.4 GHz, 5 GHz and 6 GHz interface, respectively, O_f the ratio of active flows and T_{id} corresponds to the type of the upcoming flow. Furthermore, we define O_f and T_{id}, respectively as:

$$O_f = A_f / N_A, \tag{5.58}$$

where A_f corresponds to the number of stations receiving traffic flows and N_A is the number of stations attached to the corresponding AP.

$$T_{id} = \begin{cases} 0.33 & \text{if } \textit{UHD/4K traffic}, \\ 0.66 & \text{if } \textit{VR traffic}, \\ 1 & \text{else } \textit{WB traffic} \end{cases} \tag{5.59}$$

Action space selection

In this subsection, we proceed to describe the action space of the MH-RSAC scheme. As discussed in subsection 5.3.2, the MH-RSAC structure is comprised of two heads: one providing the action output when two interfaces are available and another for the case when three are. Thus, the action space can be defined as:

$$\boldsymbol{a}_t = \begin{cases} \boldsymbol{a}_{h1,t} = [\{a_{1,t-W}, a_{2,t-W}\}, \\ \{a_{1,t-W-1}, a_{2,t-W-1}\} \\ , ..., \{a_{1,t}, a_{2,t}\}], & \text{if } n_f = 2, \\ \boldsymbol{a}_{h2,t} = [\{a_{1,t-W}, a_{2,t-W}, a_{3,t-W}\}, \\ \{a_{1,t-W-1}, a_{2,t-W-1}, a_{3,t-W-1}\} \\ , ..., \{a_{1,t}, a_{2,t}, a_{3,t}\}], & \text{if } n_f = 3 \end{cases} \tag{5.60}$$

where $\boldsymbol{a}_{k\in\{1,2,3\}}$ refers to the fraction of the total flow traffic to be allocated to each available interface.

Consequently, the action space size of each head is calculated as the number of permutations of the possible fractions that sum to one. The previous number is obtained using the well-known combinatronics "stars and bars" technique [196]. The size of each head can be calculated as:

$$|A|_{h2} = \frac{(n+1)!}{n!}, \tag{5.61}$$

$$|A|_{h3} = \frac{(n+2)!}{2!\,n!}, \tag{5.62}$$

where $n = 10$. The previous n value is chosen based on the discretization of the maximum flow traffic distribution value using a 0.1 interval. After the substitution of n, we obtain $|A|_{h2} = 11$ and $|A|_{h3} = 66$.

Reward function

The reward function in the t-*th* episode is defined as:

$$R_t = 1 - D_t^{avg}, \tag{5.63}$$

where D_i^{avg} corresponds to the average throughput drop ratio observed by the m^{th} AP. In addition, we scale the reward to $[-1, 1]$.

Reward function with hindsight

In [197], the authors present a buffer technique called *Hindsight Experience Replay* (HER) that allows the successful application of model-free RL algorithms in environments characterized by sparse rewards. The idea involves the usage of *goals states* to improve the sample efficiency and convergence. Differently from the previous work, we propose to utilize goals not in the state space but in the reward function itself. Thus, we define the term *goal reward* as the threshold in which given a reward from the environment a penalization is applied to the composite reward function as described as follows:

$$R_t^h = \begin{cases} R_t & \text{if } D_t^{avg} < D_{TOL}, \\ -1 & otherwise, \end{cases} \tag{5.64}$$

where D_{TOL} corresponds to a hindsight reward based on baseline results or expert knowledge. The intuition behind the before mentioned formulation relies on the more number of times the agent obtains a reward worse than the *goal reward* given an action $\boldsymbol{a}$, the stronger the indication that the action taken was not positive given an observed state. Eventually, the agent will be able to learn from those undesired actions and start learning from the desired ones.

Complexity Analysis

The complexity of our proposed MH-RSAC scheme corresponds to the complexity of the neural network comprising the SAC base algorithm. Thus, the training complexity is $O(\mathcal{H} \cdot \mathcal{W} \cdot \frac{|\mathcal{D}|}{B} \cdot T)$, where $\mathcal{H}$ represents the learning update periodicity, $\mathcal{W}$ the number of updates per session, $|\mathcal{D}|$ the buffer size, B the batch size and $O(T)$ denotes the time complexity of a single iteration. This means that increasing the frequency of updates (decreasing $\mathcal{H}$) or the number of updates per session $\mathcal{W}$ increases the overall complexity. A larger buffer $|\mathcal{D}|$ increases complexity linearly, while larger batch sizes B reduce complexity proportionally. More training steps T increase the complexity linearly.

Baselines: SLCI and MCAA

In this work, we compare our proposed scheme with two non-RL approaches described in [115].

Single Link Less Congested Interface (SLCI): The traffic is allocated to the less congested interface.

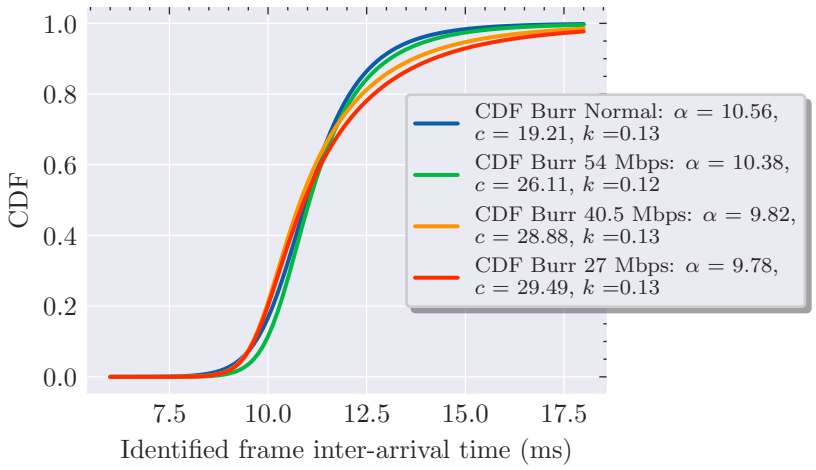

Figure 5.13: Identified frame inter-arrival time CDF

Multi-Link Congestion-aware Load balancing at flow arrivals (MCAA): The traffic is distributed among the available interfaces given the observed occupancy per interface at the AP.

Note that, in a recent letter [116] the authors have presented a dynamic traffic allocation proposal. However, different than the baselines and our proposed approach, such a policy could modify the flow distribution over the available interfaces periodically without the constraint of only doing so upon an incoming flow.

5.3.3 Traffic considerations

As previously mentioned, IEEE 802.11be aims to respond to the increasing demand for high-throughput services. Among them, we encounter Virtual Reality (VR). In [190], the authors present an empirical model based on two campaigns of VR gaming data measurements. The first campaign is run using a VR game ("Beat Saber") on a local server and the second on a cloud server, respectively. In both cases, the frame size and frame inter-arrival time are measured using Wireshark. Results showed that the frame size behaved as a log-logistic distribution and the frame inter-arrival time as a Burr distribution. In this work, we utilize the cloud server model and derive the Cumulative Distribution Functions (CDF) to be utilized as one of our simulation traffic flows. The derivation of the inverse of the CDF for the frame inter-arrival time and frame size is described below:

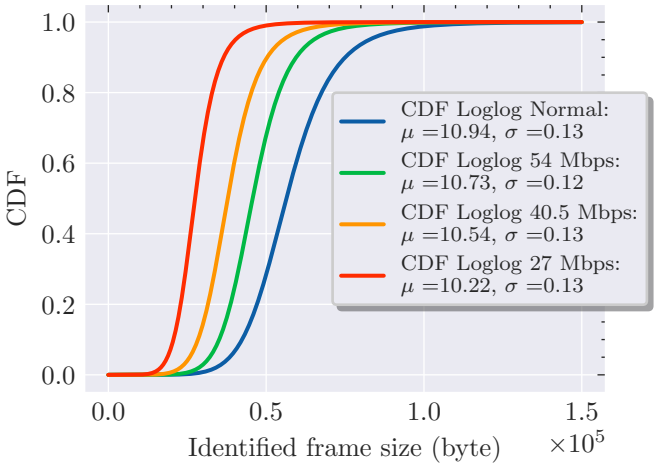

Figure 5.14: Identified frame size CDF

$$F_{fs}(x) = \int_0^{\infty} \frac{e^{\frac{\ln(x)-\mu}{\sigma}}}{\sigma x(1+e^{\frac{\ln x-\mu}{\sigma}})^2}\,dx = -\frac{1}{e^{\frac{\ln(x)-\mu}{\sigma}}+1} + C, \tag{5.65}$$

where F_{fs} corresponds to the frame size CDF, μ and σ are the statistical model parameters of the loglogistic distribution and C a constant related to the mathematical integration. Finally, the inverse of the CDF in bytes is:

$$F_{fs}^{-1}(y) = e^{\mu}(-\frac{y}{y-1})^{\sigma}[bytes]. \tag{5.66}$$

The frame inter-arrival time can be described as follows:

$$F_{ft}(x) = \int_0^{\infty} \frac{\frac{kc}{a}(\frac{x}{a})^{c-1}}{\left(1+(\frac{x}{a})^c\right)^{k+1}}\,dx = -\frac{1}{\left((\frac{x}{a})^c+1\right)^k} + C, \tag{5.67}$$

where F_{ft} corresponds to the frame inter-arrival time CDF, k, a and c are the statistical model parameters of the Burr distribution. The inverse of the CDF in ms is:

$$F_{ft}^{-1}(y) = a\sqrt[c]{\sqrt[k]{\frac{1}{1-y}}-1}[ms]. \tag{5.68}$$

Finally, the VR flow size in Mbps can be calculated as:

$$f_{VR}(y) = \frac{F_{fs}^{-1}(y)}{F_{ft}^{-1}(y)}[Mbps], \tag{5.69}$$

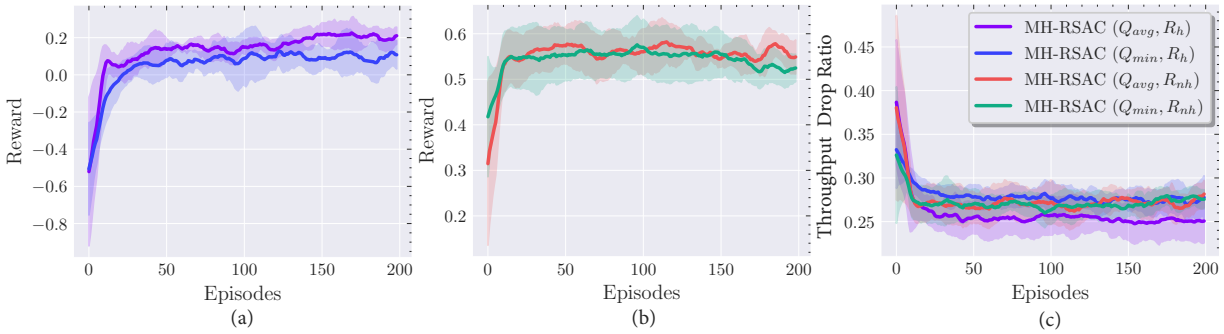

Figure 5.15: Reward and TDR convergence for the "**avg**", "**min**" operator of the MH-RSAC's variants in the **U2** scenario: **(a)** Reward with hindsight (R_h), **(b)** Reward without hindsight (R_{nh}) and **(c)** TDR convergence.

where y is randomly sampled as $y \sim \mathcal{U}(0, 1]$. In Fig. 5.13 and 5.14, we present the VR CDFs for the frame inter-arrival and frame size considering AP bandwidth throttling of 54 Mbps, 40.5 Mbps and 27 Mbps and no AP throttling (Normal), respectively. Note that only downlink traffic is considered and all traffic flows are modeled as a single Constant Bit Ratio (CBR) flow. For this work purpose, the VR traffic generated corresponds to the non-throttling case (Normal). Besides the VR traffic, we assume the coexistence of two more traffic flows. One depicting an HD/4K flow with a size of $f_{4K} \sim \mathcal{U}(7, 25)$ Mbps and a web browsing traffic with a size of $f_{WB} \sim \mathcal{U}(1, 3)$ Mbps. At the beginning of each simulation, the data flow types are distributed among the stations attached to their AP as follows: {WB: 0.8, V4K: 0.1, VR: 0.1}.

5.3.4 Performance evaluation

Simulations are performed using the flow-level simulator Neko 802.11be [115] and Pytorch-based RL agents. The communication between the simulator and the RL agents is done using the ZMQ broker library.

Simulation Settings

Simulation settings and RL parameters utilized in this work are depicted in Table 5.8 and 5.9, respectively. Furthermore, we consider two load distribution scenarios: **U1** with number of users attached to its corresponding AP, $N_A \sim \mathcal{U}(15, 20)$ and **U2** with $N_A \sim$

Table 5.8: Network settings

Parameter	Value
IEEE protocol	802.11be
Channel Bandwidth	20 MHz/40 MHz/80 MHz/160 MHz
Carrier Frequency (f_c)	2.437 GHz/5.230 GHz/6.295 GHz
Max Modulation/Max Modulation Coding Scheme	4096 QAM/MCS 13
Number of APs	5
Number of Stations per AP	**U1** : $N_A \sim \mathcal{U}(15, 20)$, **U2** : $N_A \sim \mathcal{U}(20, 25)$
Max Spatial Streams	16
Propagation Loss Model	$P_l(d) = 40.05 + 20\log(f_c/2.4) + 20\log(\min(d, 10))+$ $(d > 10) * 35\log(d/10) + 7W$
Data Flow Types	Web browsing (WB), Video 4K (V4K), Virtual Reality (VR)
Data Flow User Distribution	WB: 0.8, V4K: 0.1, VR: 0.1
AP/STA Noise Figure	7 dB
AP/STA Transmission Power	20/15 dBm
CCA threshold	-82 dBm
Packet Error Rate	10%

$\mathcal{U}(20, 25)$. In the next subsection, we will discuss the performance results of the proposed scheme.

Simulation Results

We present the performance results of our proposed scheme in terms of convergence, FS per interface, and TDR. Figure 5.15 shows the convergence behavior of the **U2** scenario for the SAC variants proposed in this work. The figures that depict the scenario **U1** are not shown in this work, as they show similar information. Figure 5.15 (a) shows how the "**avg**" operator outperforms the "**min**" operator when the reward with hindsight is considered. Conversely, the previous behavior does not repeat when the rewards without hindsight are utilized as shown in Fig. 5.15 (b). Finally, we show the convergence in terms of TDR for the four variants in Fig. 5.15 (c) where the best convergence is achieved when the

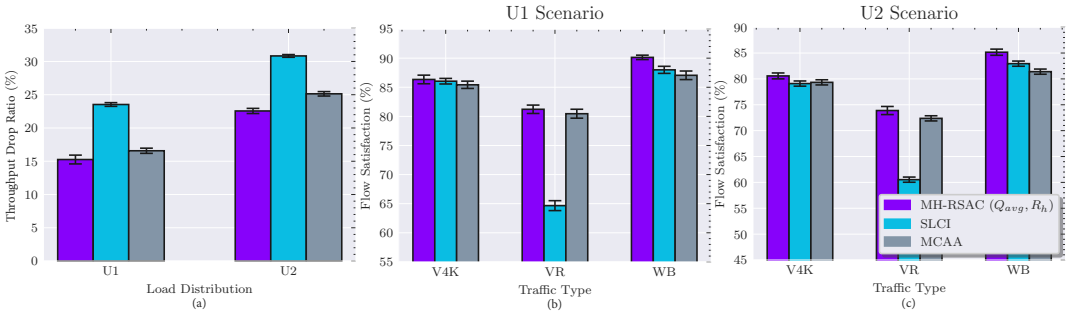

Figure 5.16: **(a)** Throughput Drop Ratio (TDR) per load distribution **U1** and **U2** and Flow Satisfaction (FS) per traffic for Video HD/4K (V4K), Virtual Reality (VR) and Web Browsing (WB) per load distribution **(b) U1** and **(c) U2**

"**avg**" operator with rewards with hindsight is used. The previous results confirm that the *goal rewards* strategy proposed in this work affects positively the learning process in non-Markovian scenarios.

Furthermore, in Fig. 5.16 we present the results in terms of TDR and FS per traffic type for **U1** and **U2** scenarios, respectively. In Fig. 5.16 (a), we show the TDR performance of the MH-RSAC scheme with "**avg**" operator and rewards with hindsight and the two baselines: SLCI and MCAA. Consequently, we observe that the MH-RSAC scheme offers a gain of 34.2% and 35.2% when compared with the SLCI algorithm and 2.5% and 6% when compared with the MCAA algorithm for the **U1** and **U2** scenarios, respectively. Additionally, we present a comparison in terms of FS per traffic type in Fig. 5.16(b) and (c). For instance, in Fig. 5.16 (b) we can see that the MH-RSAC scheme's gains for the V4K traffic around 1% but when it comes to the VR traffic which is a more throughput-hungry service SLCI is not capable to perform positively with a reduction of 25.6% in terms of FS. Moreover, for the dynamic WB traffic, the MH-RSAC scheme shows an improvement of 2.5% and 4% over the SLCI and MCAA algorithms. In a similar fashion in Fig. 5.16 (c), we obtain a gain of 21% and 3% for the VR traffic and 3.2% and 6% for the WB traffic of the MH-RSAC algorithm over the two baselines SLCI and MCAA, respectively. To sum up, we can see that MH-RSAC is capable of distributing high throughout traffic over multiple links and also responding efficiently to more dynamic traffic types like Web Browsing.

Table 5.9: Reinforcement Learning settings

Parameter	**Value**
Actor learning rate	1e−4
Actor final layer activation function	Softmax
Critic learning rate	1e−3
Critic final layer activation function	None
Linear hidden layers	$[64, 64]$
Window (W)	10
Tau (τ)	5e−3
Discount rate	0.99
Gradient clipping norm	1
Weight initializer	Xavier
Batch size	512
Learning updates periodicity ($\mathcal{H}$)	50 steps
Learning updates per learning session ($\mathcal{W}$)	10
Automatic entropy tuning	Enabled

5.4 Channel Selection for Wi-Fi 7 Multi-Link Operation via Optimistic-Weighted VDN and Parallel Transfer Reinforcement Learning

Recent IEEE 802.11 Wireless Fidelity (Wi-Fi) documentation introduces a new standard named 802.11be EHT –Extremely High Throughput– or Wi-Fi 7, which will become the replacement for the former 802.11ax amendment. The release of the final draft is expected by May 2024 [155] adding to the new set of modifications a modulation rate increase of up to 4096 QAM, channel bandwidth up to 320 MHz, and more importantly the addition of Multi-Link Operation (MLO) and Multiple Resource Units (MRU) capabilities [198]. These new features will allow Wi-Fi technology to adapt to the continuously increasing demand in terms of throughput and low-latency services such as Virtual Reality, Live Streaming, and Ultra-High Definition video in dense deployments.

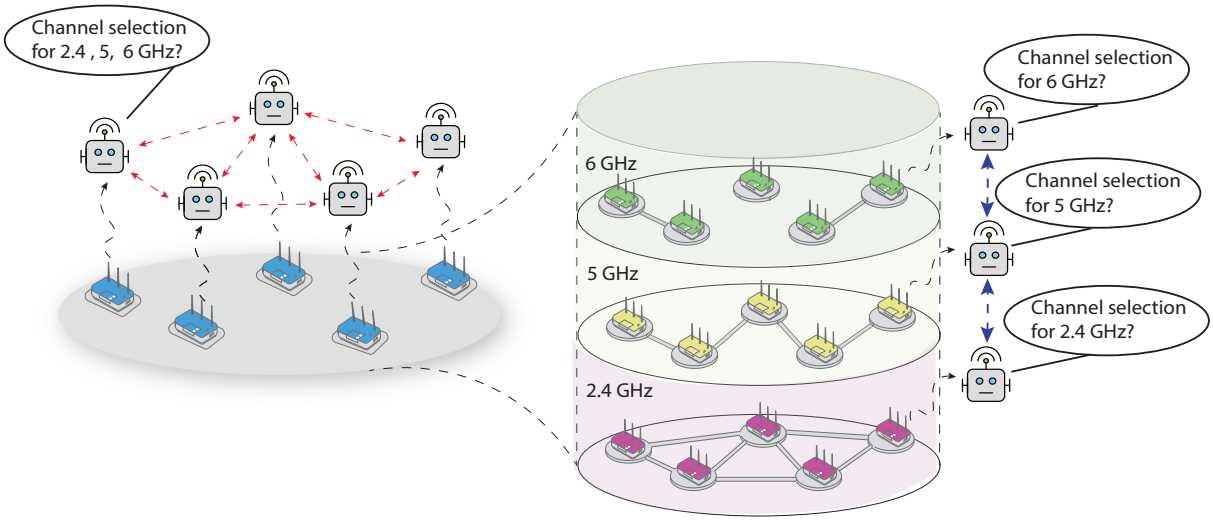

Figure 5.17: Instead of utilizing one Multi-Agent System (MAS) where each agent provides a joint action selection for all interfaces, in the context of MLO, we divide it into three MASs. Each MAS provides an action selection per interface and parallel transfer learning is leveraged to improve learning.

In this work, with the aid of Fig. 5.17 we raise the following question: Is it beneficial to transfer knowledge in a parallel fashion among MASs? To do so, we propose dividing a complex problem into less complex ones to enable the utilization of Parallel Transfer

Reinforcement Learning (PTRL) methods. Thus, we study PTRL to improve channel selection in the context of MLO in IEEE 802.11be. MLO adds an extra dimension to channel selection in IEEE 802.11 networks since it permits Multi-Link Devices (MLD)s such as terminals and Access Points (APs) to use their available interfaces for multi-link communications. This means that not only channel selection should be performed concurrently in one interface, say 2.4 GHz; but this procedure needs to be repeated for the rest of the interfaces. Specifically, we propose leveraging Multi-Agent Reinforcement Learning (MARL) and PTRL to optimize channel selection in IEEE 802.11be MLO-capable networks. MARL has been brought to the attention of the wireless networks research community due to its realistic approach [29]. In addition, transfer learning has also proven its applicability and positive implications in improving learning convergence and Key Importance Indicators (KPIs) in wireless networks based on RL multi-agent solution [44]. However, in this work, we utilize a less explored area of transfer learning called parallel transfer learning that proposes concurrent knowledge transfer without the source/target paradigm of the well-known sequential transfer learning. More specifically we put our interests on parallel transfer reinforcement learning among Multi-Agent Systems (MASs). To enable Machine Learning (ML)-based technology in Wi-Fi, a project named OpenWiFi [78] proposes disaggregating the Wi-Fi technology stack by utilizing open-source software for the Access Point (AP) firmware operating system. Such technology enables the integration of diverse cloud-based services in ML-based applications like the one proposed in this work.

In subsection 5.4.1 a brief system model is described. Subsection 5.4.2 introduces the proposed scheme Parallel Transfer Reinforcement Learning Optimistic-Weighted VDN (referred to as oVDN) and its design considerations, including the PTRL methods and the Markov Decision Process (MDP). Finally, subsection 5.4.3 presents a study of the proposed scheme's performance.

5.4.1 System Model

In this work, we utilize an IEEE 802.11be network with a predefined set of M APs and N stations attached per AP. All APs are MLO-capable with a number of available interfaces n_f. All APs and stations support up to a maximum of 16-SU multiple-input multiple-output (MIMO) spatial streams. The path loss model corresponds to the enterprise model described in [191]. In addition, we consider an adaptive control rate based on Signal-to-Interference-Noise Ratio (SINR) with a maximum 4096 QAM modulation rate.

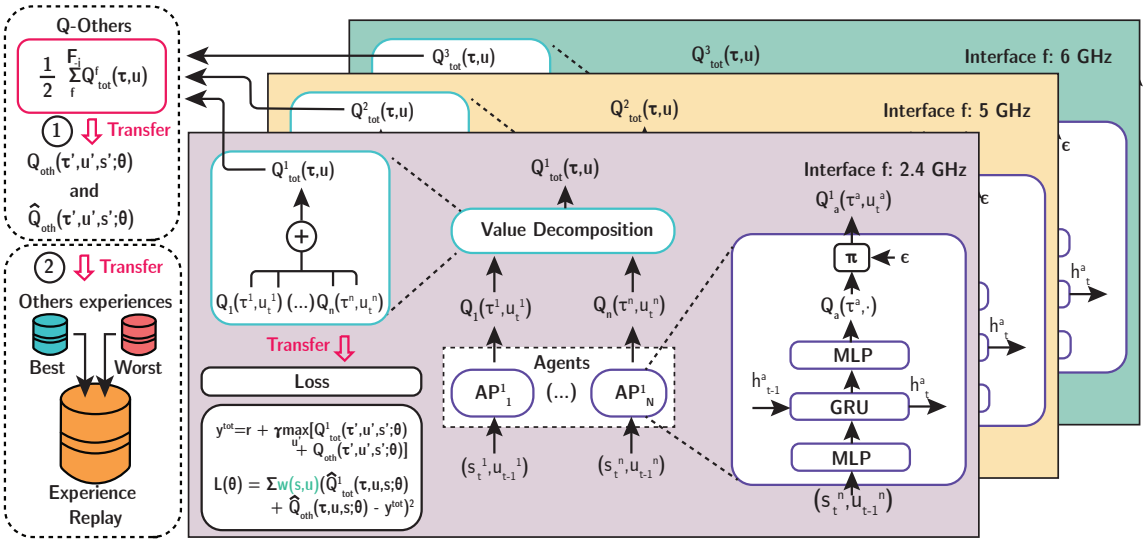

Figure 5.18: Overview of the oVDN algorithm

5.4.2 Parallel Transfer Reinforcement Learning Optimistic Weighted VDN (oVDN)

In this subsection, we discuss the details and considerations taken in the design of the Parallel Transfer Reinforcement Optimistic Weighted VDN (oVDN) architecture.

Optimistic Weighted VDN

In their paper, Rashid et al. [71] propose two weighting schemes to improve the QMIX algorithm, which was originally introduced in [199]. QMIX is a fully cooperative algorithm that performs value function factorization among agents, similar to VDN. However, instead of using an additive approach for mixing strategies, QMIX employs hypernetworks that enforce monotonic Q-function behavior. One limitation of QMIX is that it assumes equal importance among actions, which can lead to suboptimal policies. To address this issue, the authors introduce Optimistically-Weighted QMIX, which assigns weights to each Q-function, allowing for better joint actions. Based on this idea, we modify the VDN algorithm to add importance among agents following the weighting as described in previously in equation 2.13.

Multi-Agent Parallel Transfer Reinforcement Learning (MA-PTRL)

In this work, to the best of our knowledge, we introduce for the first time PTRL for cooperative inter-multi-agent systems. We identify two PTRL types: Intra-PTRL that corresponds to the scenario where agents in a MAS concurrently transfer knowledge and inter-PTRL where transfer learning occurs concurrently among MAS systems. Employing this technique can have several benefits and direct implications on the definition of the RL problem such as reduction of the action and state space as well as convergence and learning improvement. However, the MASs involved in any PTRL application must have an inner relationship to obtain the expected behavior as needed in sequential transfer learning. Moreover, a careful design must be taken in the MDP definition to avoid negative transfer. For example, we realize that scaling rewards among MAS systems does have an impact on the effectiveness of Multi-Agent Parallel Transfer Reinforcement Learning (MA-PTRL). In the next subsection, we introduce the PTRL methods used in this work.

Parallel Transfer Learning (PTRL) Methods

PTRL is a challenging technique due to its inherent online characteristics. Despite the benefits previously mentioned, the lack of a source-target task relationship enforces a careful transfer design. Differently from other works, we utilize PTRL to improve convergence and alleviate the action/state space large dimensionality that a centralized multi-agent RL problem definition could suffer.

Let's consider the existence of $\mathcal{M}$ MASs that will concurrently transfer knowledge among them. As observed in Figure 5.18 and Algorithm 5.7, we utilize two PTRL techniques in the context of cooperative MARL: **(1)** MAS Joint Q-function Transfer and **(2)** MAS Best/Worst Experience Transfer. Each of the previously mentioned techniques can be summarized as follows:

MAS Joint Q-function Transfer: As the name indicates the average joint Q-functions (Q_{tot}) of the set $\{\mathcal{M} \setminus m\}$ are used in the training stage of each MAS. Thus, the loss function for the m^{th} MAS can be written as follows:

$$y = r + \gamma(Q^m_{tot} + \sigma \frac{1}{|\mathcal{M} \setminus m|} \sum_i^{\mathcal{M}\setminus m} Q^i_{tot}), \tag{5.70}$$

$$\mathcal{L} = w(o,u)([\hat{Q}^m_{tot} + \sigma \frac{1}{|\mathcal{M} \setminus m|} \sum_i^{\mathcal{M}\setminus m} \hat{Q}^i_{tot}] - y)^2, \tag{5.71}$$

Algorithm 5.7 Parallel Transfer Reinforcement Learning Optimistic-Weighted VDN (oVDN)

Let $\mathcal{M}$ be the set of multiagent VDN systems. For each $m^{th} \in \mathcal{M}$ VDN, initialize the agents' target and evaluation recurrent policies θ and θ', respectively. Set initial hyperparameters and set experience buffers $\mathcal{D}$ and $\mathcal{D}_{ep}$.
for environment step $t \leftarrow 1$ **to** T **do**
  $\mathbf{u}_t = \{\}$
  for each VDN m **do**
    for each agent i **do**
      Observe state $s_t^{m,i} = \{a_t^{m,i}, i^m\}$ and append to $\tau_t^{m,i,a} = \tau_{t-1}^{m,i,a} \cup \{(s_t^{m,i}, u_{t-1}^{m,i})\}$
      $\epsilon_t^{m,i} = \epsilon_t^{m,i} - \epsilon_d$ **if** $\epsilon_t^{m,i} - \epsilon_d > \epsilon_{min}$ **else** ϵ_{min}
      Select joint action:

$$u_t^{m,i,a} = \begin{cases} argmax_{u_t^{m,i,a}}[\pi^{m,i}(\tau_t^{m,i}, u_t^{m,i,a}; \theta^{m,'})] & \textbf{if } \epsilon \geq \mathcal{N}(0,1), \\ \text{random action from } \mathcal{I}(0, |U|^m); & \\ |U|^m \subset \mathbb{Z}^+ & otherwise \end{cases}$$

    end for
    $\mathbf{u}_t = \mathbf{u}_t \cup \mathbf{u}_t^{m,a}$
  end for
  Execute $\mathbf{u}_t$ in the environment
  for each VDN m **do**
    for each agent i **do**
      Observe next state $s_{t+1}^{m,i}$ and reward $r^{m,i}$
    end for
    $\mathcal{D}^m = \mathcal{D}^m \cup \{(\mathbf{o}_t^m, \mathbf{u}_t^{m,a}, r_t^m, \mathbf{o}_{t+1}^m)\}$
    $\mathcal{D}_{ep}^m = \mathcal{D}_{ep}^m \cup \{(\mathbf{o}_t^m, \mathbf{u}_t^{m,a}, r_t^m, \mathbf{o}_{t+1}^m)\}$
  end for
  if t **mod** $n_{steps} = 0$ **then**
    for each VDN m **do**
      # (1) Transfer Q_{tot} among MAS.
      $\boldsymbol{y}_B = r_B + \gamma(\boldsymbol{Q}_{B,tot}^m + \sigma \frac{1}{2} \sum_n^{\mathcal{M} \setminus m} \boldsymbol{Q}_{B,tot}^n)$
      $\mathcal{L}(\theta^m) = \sum_{b=1}^B (\boldsymbol{w}(\boldsymbol{o}_b, \boldsymbol{u}_b)([\hat{Q}_{b,tot}^m + \sigma \frac{1}{2} \sum_n^{\mathcal{M} \setminus m} \hat{Q}_{b,tot}^n] - y_b)^2$
      $\theta^m \leftarrow \theta^{m,'}$
      for each VDN $n \in \{\mathcal{M} \setminus m\}$ **do**
        #(2) Transfer best and worst experiences among MAS.
        $\{(\mathbf{s}_t^m, \mathbf{u}_t^{m,a}, r_t^m, \mathbf{s}_{t+1}^m)\}_g = \texttt{sort}_g(\mathcal{D}_{ep}^n, n_g)$
        $\{(\mathbf{s}_t^m, \mathbf{u}_t^{m,a}, r_t^m, \mathbf{s}_{t+1}^m)\}_b = \texttt{sort}_b(\mathcal{D}_{ep}^n, n_b)$
        $\mathcal{D}^m = \mathcal{D}^m \cup \{(\mathbf{s}_t^m, \mathbf{u}_t^{m,a}, r_t^m, \mathbf{s}_{t+1}^m)\}_g \cup \{(\mathbf{s}_t^m, \mathbf{u}_t^{m,a}, r_t^m, \mathbf{s}_{t+1}^m)\}_b$
      end for
      if $\mathcal{D}^m > B_s$ **then**
        Randomly sample a batch of transitions $\boldsymbol{B}$ with size B_s from $\mathcal{D}^m$
        for each batch $b \in \boldsymbol{B}$ **do**
          $\boldsymbol{Q}_{\{1..|\mathcal{M}|\}_b}^m = \pi^m(\tau_b^m, u_b^m; \theta^{m,'})$
          $Q_{b,tot}^m = \mu(\boldsymbol{Q}_{\{1..|\mathcal{M}|\}_b}^m)$
          $\hat{\boldsymbol{Q}}_{\{1..|\mathcal{M}|\}_b}^m = \pi^m(\tau_b^m, u_b^m; \theta^m)$
          $\hat{Q}_{b,tot}^m = \mu(\hat{\boldsymbol{Q}}_{\{1..|\mathcal{M}|\}_b}^m)$
        end for
        Reset all D_{ep} buffers
      end if
    end for
  end if
end for

where σ is a tunable transfer weight, Q_{tot} and $\hat{Q}_{tot}$ are target and evaluation Q-functions, respectively. Transferring the Q-function values can help to improve convergence since it suggests a consensus on the set of actions that are being selected by the MASs comprising the PTRL scheme.

MAS Best/Worst Experience Transfer: This approach relies on the nature of offline RL algorithms that utilize an experience replay as part of their policy training. Intuitively, the best and worst experiences contribute to the final agent's action selection policy because the best action will receive higher rewards than the worst ones. Specifically, we gather during each episode, for each m^{th} MAS, n_e number of sample experiences and store them in a temporal buffer $\mathcal{D}_{ep}$. As indicated in Equations (5.72)-(5.73), we sort $(n_g, n_b) < n_e$ samples ranking the best and worst experiences based on the ones with higher and lower rewards, respectively.

$$\{(\mathbf{s}_t^m, \mathbf{u}_t^m, r_t^m, \mathbf{s}_{t+1}^m)\}_g = \texttt{sort}_g(\mathcal{D}_{ep}^m, n_g), \tag{5.72}$$

$$\{(\mathbf{s}_t^m, \mathbf{u}_t^m, r_t^m, \mathbf{s}_{t+1}^m)\}_b = \texttt{sort}_b(\mathcal{D}_{ep}^m, n_b), \tag{5.73}$$

$$\mathcal{D}^{\mathcal{M}\setminus m} = \mathcal{D}^{\mathcal{M}\setminus m} \cup \{(\mathbf{s}_t^m, \mathbf{u}_t^{m,a}, r_t^m, \mathbf{s}_{t+1}^m)\}_g \cup \{(\mathbf{s}_t^m, \mathbf{u}_t^{m,a}, r_t^m, \mathbf{s}_{t+1}^s)\}_b, \tag{5.74}$$

where $\mathbf{s}_t^m$, $\mathbf{s}_{t+1}^m$, r_t^m and $\mathbf{u}_t^m$ corresponds to the state, next state, team reward, and joint action, respectively. Finally, we append both experiences to the global buffer as indicated by Equation (5.74).

State space selection

The state space is defined as follows:

$$s_{(t,f)}^n = \{a_{(t-1),f}^n, h_{(t-2),f}^n, AP_{id}^n\}, \tag{5.75}$$

where $a_{(t-1),f}^n$ corresponds to the last action taken, $h_{(t-2),f}^n$ the history action taken before it and the AP_{id} corresponds to the ID of n^{th} AP. The first two terms help to alleviate partial observability via the utilization of the Gated Recurrent Unit (GRU) layer in the Q-function as observed in Figure 5.18. The previous definition refers to the action space of a single agent in any of the MASs of our algorithm.

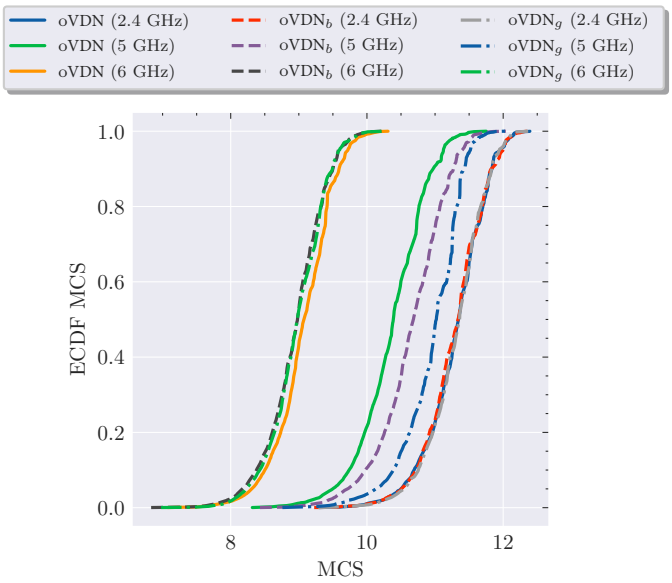

Figure 5.19: MCS ECDFs comparison per interface frequency of three variants of transfer: oVDN, oVDN$_b$ and oVDN$_g$, indicating transfer best and worst, only best and only worst experiences, respectively.

Action space selection

In this subsection, we present the proposed general action space:

$$a^n_{(t,f)} = \{1, ..., c_{(max,f)}\}, \tag{5.76}$$

where $c_{(max,f)}$ corresponds to the maximum number of channels per interface frequency and $f \in \{2.4, 5, 6\}$ GHz indicates the frequency of each of the MASs utilized in this work. As specified before, the current definition indicates the action space of a single agent of any of the defined MASs. Finally, the action space's size corresponds to $|a_{t,f}|!$.

Reward function

The general reward function for the m^{th} MAS can be defined as follows:

$$r^m_t = min(\texttt{mcs}^1_f, ..., \texttt{mcs}^{|\mathcal{N}|}_f), \tag{5.77}$$

where $\texttt{mcs} \in \{0, 1, ..., 13\}$ indicates the corresponding mapped Modulation Code Selection (MCS) index given the measured SINR. The SINR comprises the effect of all APs and

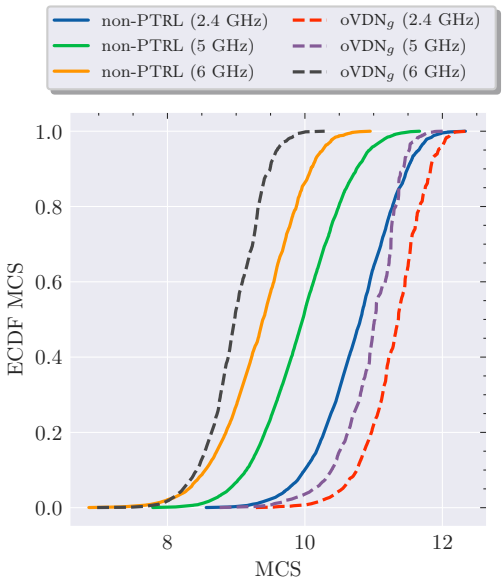

Figure 5.20: MCS ECDFs comparison per interface frequency of the non-PTRL and our proposal using best experience transfer.

stations considered during the simulations. The mentioned mapping is done according to IEEE 802.11be which employs a similar MCS-SINR mapping as IEEE 802.11ax. In addition, we scale the reward to $[-1, 1]$.

Special consideration is taken in the context of channel selection in IEEE 802.11be for the MA-PTRL proposed scheme regarding the definition of the rewards. Evidently, Q-functions depend on the rewards acquired during the interaction of the environment, and as discussed in subsection 5.4.2, Q-functions are used as part of the information transferred. Thus, different reward scales can provoke some of the MASs comprising the MA-PTRL algorithm can impose their action selection policy. Specifically, we realize that MCS selection according to the SINR ranges varies depending on the interface frequency and channel bandwidth. Thus, we scale the rewards received by each MASs. Notice that this consideration must only be taken when there is a scaling difference between MASs' rewards in any PTRL use case.

5.4.3 Performance evaluation

Simulations are performed using the flow-level simulator Neko 802.11be [115]. The bidirectional communication between the environment simulated in Neko and the Pytorch-based

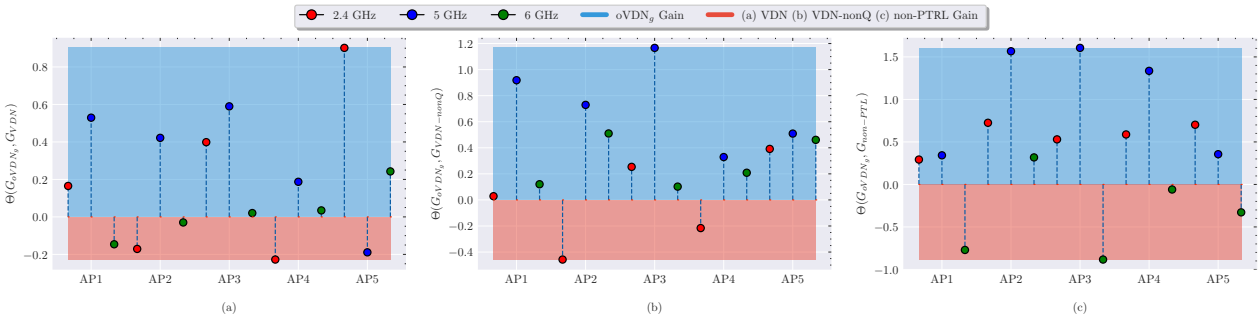

Figure 5.21: Gain Θ comparison per AP and interface frequency **(a)** oVDN$_g$ and VDN, **(b)** oVDN$_g$ and VDN-nonQ, **(c)** oVDN$_g$ and non-PTRL. The blue and red area indicates what technique offers the best performance.

RL scheme is performed via the ZMQ broker library.

Simulation Settings

RL parameters and simulation settings are described in Table 6.2 and 6.3, respectively. Furthermore, we consider the scenario where there are $|\mathcal{N}| \sim \mathcal{U}(25, 40)$ users attached to each AP, and the AP's locations are unknown, thus solely relying on RF feedback from the environment. In addition, 80% of the users are positioned randomly in a radius $r \sim [1-8]$ m and 20% within a radius of $r \sim [1-3]$ m. In the next subsection, we will discuss the performance results of the proposed scheme.

Simulation Results

We present the performance results of our proposed scheme in terms of MCS ECDF, ECDF gain, and convergence, among the baselines and the best PTRL variant. Figure 5.19 shows the MCS ECDFs of three different variants based on what type of experiences are being transferred. The results show that the ECDF of the scenario, where only the best experiences are transferred (oVDN$_g$), contributes more positively to the channel selection and thus, higher MCS is obtained. Interestingly, only transferring the worst examples offers better performance than transferring both types of experiences. The reason behind this behavior could be explained by the fact that we utilize a fixed number of worst and bad experiences and a weighted approach could be the best alternative to use. Additionally, in Fig. 5.20, we present the results in terms of MCS ECDFs for the centralized baseline

Table 5.10: Reinforcement Learning Settings.

Parameter	**Value**
Training steps	1.5e4
Initial ϵ	1
ϵ decay steps	500 steps
ϵ minimum	5e$-$2
Buffer ($\mathcal{D}$) size	2e3
Batch size	64
Learning rate	8e$-$3
Discount factor (γ)	0.99
Recurrent Layer hidden dimension	64
MultiLayer Perceptron hidden dimension	64
Weight initializer	Orthogonal
Deep Q-Network structure	Double Q-networks
Optimistic Weight α	0.1
Number of MASs	$3, (2.4, 5, 6)$ GHz
Reduced buffer ($\mathcal{D}_{ep}$) size	50
Best/Worst transferred experiences	$20\%\lvert\mathcal{D}_{ep}\rvert$
Transfer weight (σ)	1

Table 5.11: Network settings

Parameter	**Value**
IEEE protocol	802.11be
Channel Bandwidth	20 MHz/40 MHz/80 MHz/160 MHz
Carrier Frequency (f_c)	2.437 GHz/5.230 GHz/6.295 GHz
Max Modulation/Max Modulation Coding Scheme	4096 QAM/MCS 13
Number of APs per MAS ($\mathcal{N}$)	5
Number of Stations per AP	$N_A \sim \mathcal{U}(25, 40)$
Max Spatial Streams	16
Propagation Loss Model	$P_l(d) =$ $40.05 + 20\log(f_c/2.4) + 20\log(\min(d, 10))+$ $(d > 10) * 35\log(d/10) + 7W$
AP/STA Noise Figure	7 dB
AP/STA Transmission Power	20/15 dBm
CCA threshold	-82 dBm
Packet Error Rate	10%

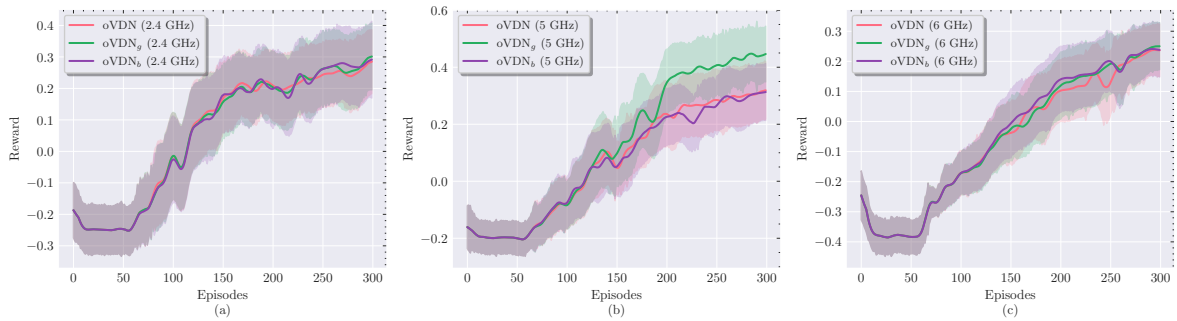

Figure 5.22: Reward convergence for oVDN, oVDN$_g$ and oVDN$_b$ **(a)** 2.4 GHz, **(b)** 5 GHz and **(b)** 6 GHz

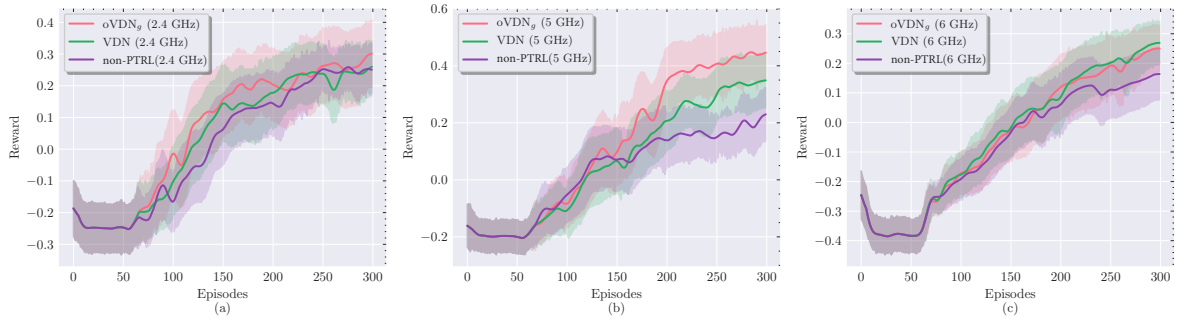

Figure 5.23: Reward convergence for oVDN$_g$, VDN and non-PTRL **(a)** 2.4 GHz, **(b)** 5 GHz and **(b)** 6 GHz

and the best transfer variant (oVDN$_g$). It can be seen that our parallel transfer solution offers an important improvement in terms of MCS over the non-parallel baseline in two interfaces (2.4 and 5 GHz) with MCS in the range of $(10-13)$ over $(8-11)$ and a slight degradation in the 6 GHz interface.

In Fig. 5.21, we illustrate the obtained gain per AP and interface frequency of oVDN$_g$ over three baselines: VDN, VDN-nonQ, and non-PTRL, indicating both types of experience transfer, absence of Joint Q-function transfer, and the non-PTRL case, respectively. The gain $\Theta(G_X, G_Y)$ corresponds to the area difference among ECDFs as follows:

$$\begin{aligned}\Theta(G_X, G_Y) &= \int \{G_Y(x) - G_X(x)\}dx \\ &= \int \{[1 - G_Y(x)] - [1 - G_X(x)]\}dx \\ &= \mu_X - \mu_Y,\end{aligned} \tag{5.78}$$

where G_Y and G_X correspond to the ECDFs of the compared variants and μ_X, μ_Y are their means. The upper blue colored area indicates where the oVDN$_g$ performs better than the baselines and the red area otherwise. Consequently, we observe that the oVDN$_g$ offers a gain up to 3%, 7.2%, and 11% when compared with (a) VDN, (b) VDN-nonQ and (c) non-PTRL baselines, respectively.

Additionally, we present in Fig. 5.22 a comparison in terms of convergence per frequency interface among oVDN's PTRL variants. In (a) and (c) the behavior is similar among the studied techniques, however in (b) where we show the reward convergence of the 5GHz interface, we observe an improvement of 33.3% of oVDN$_g$ over oVDN$_b$ and oVDN, respectively. Finally, in Fig. 5.23, we can see the convergence behavior of our best performer PTRL algorithm oVDN$_g$, VDN, and a non-PTRL alternative. Our proposal outperforms the non-PTRL baseline in terms of convergence speed up to 40 episodes and rewards up to 135%. The previous results can be explained by the non-PTRL or centralized variant's action space size. For instance, as discussed in subsection 6.1.2, the action space size of each MASs in the PTRL algorithm corresponds to $|a_f|^N, f \in \{2.4, 5, 6\}$, meanwhile the centralized to $|a_{2.4} \times a_5 \times a_6|^N$, where a indicates the set of possible actions, the $\times$ operator corresponds to the Cartesian product and N the number of agents. Thus, if $|a_f|= 3$, PTRL offers a reduction of action space in the order of 9^N which is quite considerable, especially when N increases.

To sum up, we can foresee the advantages that can potentially offer a PTRL approach among MASs such as oVDN$_g$. As observed in our presented results, faster convergence and better action selection are some of the most immediate benefits. When the decomposition of a MAS problem is feasible, we advise utilizing PTRL to accelerate and improve RL performance since typically MARL suffers from slow convergence and non-optimal action selection.

Chapter 6

Multi-Agent Reinforcement Learning-based XR Codec Adaptation

6.1 Extended Reality (XR) Codec Adaptation in 5G using Multi-Agent Reinforcement Learning with Attention Action Selection

The sixth generation (6G) wireless systems are anticipated to drive several new Internet of Everything (IoE) applications. 6G will be identified mainly by two fundamental characteristics: self-sustainability and proactiveness [200]. The previous terms refer to the capability of 6G to perform efficient adaptability and fulfillment of the extreme requirements of the emerging IoE services. Cloud Gaming (CG) and Extended Reality (XR), including Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR) are considered new emergent applications in 6G. XR technology has been around for some time, however, its impact and potential applications have been extended in 6G from the well-known gaming and smartphone applications to the healthcare industry and other verticals [201]. For this reason, 3GPP and others have shown their interest in the standardization of XR in 5G since 2016. More specifically, 3GPP release-16 studies VR Quality of Experience (QoE) relevant metrics for user experience [202]. Furthermore, 3GPP presents, in [203] an introductory report concerning XR in 5G where 23 possible use cases are presented, altogether with a standardized 5G QoS Identifier (5QI) mapping to Quality of Service (QoS). In this work, as indicated in figure 6.1, we utilize a 5G network with User Equipments (UEs) receiving diverse XR traffic (AR, VR) and CG in the downlink direction. The users report the Key

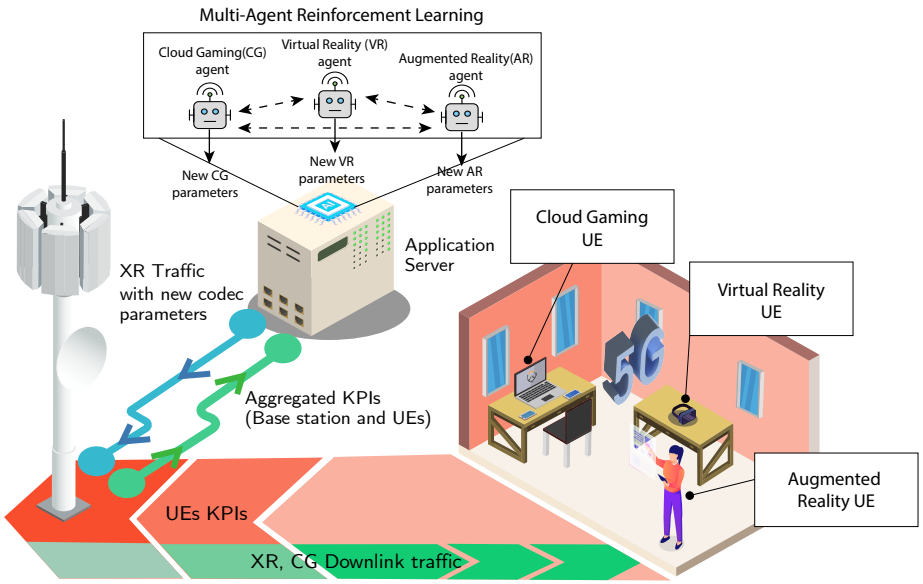

Figure 6.1: An AI-powered application server exchanges aggregated KPIs from the BS and UEs to decide on the proper XR and CG codec parameters to satisfy XR/CG QoE requirements.

Performance Indicator (KPI) metrics to the base station (BS) to be further aggregated with BS's local KPIs and shared with an application server. Afterward, the application server decides the new optimized codec parameters for the future XR traffic. We propose the usage of Multi-Agent Reinforcement Learning (MARL) to optimize the codec selection using cross-layer information. More specifically, we model this problem as a multi-agent scenario where three agents (AR, VR) and CG will act as a team to maintain fairness and quality of experience among the XR users. We choose the QMIX (Mixture of Q-Values) algorithm over other centralized-training decentralized-execution (CTDE) methods due to its proven excellent performance. More specifically, we utilized a modified QMIX algorithm named Optimistic QMIX and leveraged an attention technique to improve the learning performance of agents. We compare our attention-based QMIX and oQMIX algorithms with a state-of-the-art XR loopback mechanism called Adjust Packet Size (APS) [124]. Our results show average gains of oQMIX and QMIX over APS of 30.1%, 15.6%, 16.5% 50.3%, and 17.6%, 13.2%, 11.2%, 7.86% with respect XR index, jitter, delay, and PLR, respectively.

In subsection 6.1.1 a brief system model is described. Subsection 6.1.2 introduces some background on Quality of Experience in XR and presents the MDP preliminaries of the

proposed algorithm and attention mechanism. Additionally, subsection 6.1.4 presents the simulation results of our proposed scheme as well as the presented baseline.

6.1.1 System Model

In this work, we utilize a 5G network with N XR users receiving downlink traffic from a BS. The XR traffic is generated from an application server that receives every t_w window of time, feedback from the BS. The BS aggregates its own and users' KPIs and exchanges it with the application server. Upon the new feedback, the application server decides the future XR codec parameters to maintain QoE metrics. Our scenario consists of 3 different XR users attached to one BS. To simulate a loaded network, some parameters of the simulation are adjusted such as the transmission Radio Link Control buffer's (Unacknowledged Mode) capacity and the available bandwidth. Furthermore, UEs are located within three rings as in figure 6.2 (a). Finally, the modeled XR traffic complies with the 3GPP release-17 study [122].

6.1.2 XR Codec Adaptation-based Multi-Agent Reinforcement Learning

In this subsection, we introduce the details and considerations in the design of the XR Codec Adaptation Multi-Agent Reinforcement Learning algorithm. We begin by introducing some background on Quality of Experience in the context of XR and follow with the specifics of the QMIX algorithm and attention action selection.

Quality of Experience in XR

Quality of Experience (QoE) is a well-known metric in multimedia-related topics. It is subjective to the observer and thus, difficult to measure objectively. For instance, as mentioned in [204, 205], QoE can be defined as "the degree of delight or annoyance of the user of an application or service".

The Mean Opinion Score (MOS) is one of the methods utilized in the literature to measure QoE. This metric is calculated by showing different video streams to a group of observers. The opinions are classified into a rank of five levels where 5 indicates an excellent quality (imperceptible impairment) and 1 bad quality (very annoying). Despite the subjectiveness of the metric, some efforts have been made to quantify QoE with Key

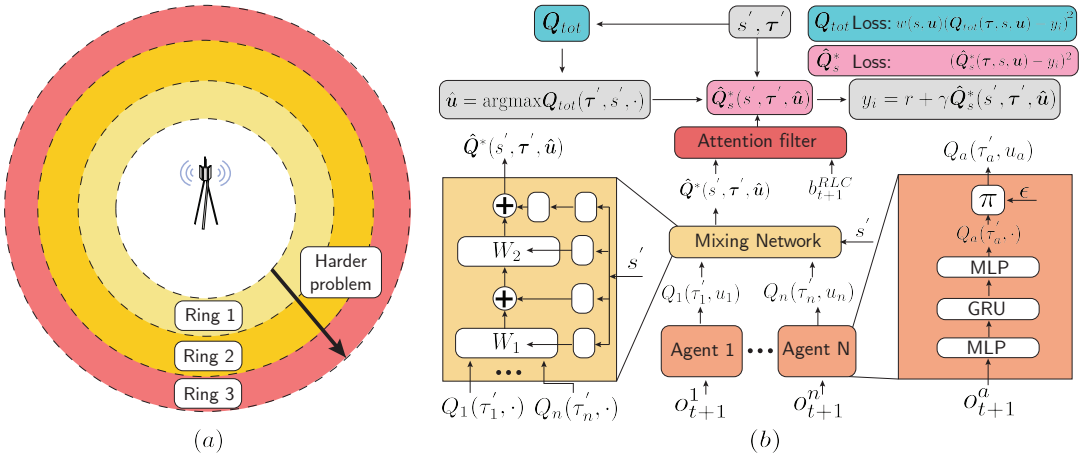

Figure 6.2: **(*a*)** Illustration of user locations in three coverage. regions. When users are located in outer rings the solution becomes harder due to the reduced action set that satisfies QoE requirements. **(*b*)** Overview of the oQMIX algorithm with action prohibition.

Performance Metrics at the Radio Access Network (RAN) level. For instance, in [203] 3GPP identifies the QoS requirements for different XR services. Furthermore, according to the agreements in 3GPP RAN1 meeting [202], some combinations of Packet Success Ratio (PSR) and Packet Delay Budget (PDB) should be evaluated to measure QoE in VR/AR traffic. Based on the previous proposals, in [3] the authors propose a coarse-grained XR Quality Index (XQI) that consists of a mapping between a subjective metric such as MOS and 3GPP's latest documentation. In the following subsection, we introduce the preliminaries of the attention-based CTDE algorithm oQMIX.

Attention action selection and Slate-Markov Decision Process in QMIX

Action space size corresponds to one of the main challenges at the exploration and exploitation stages in MARL. This well-known issue is more noticeable in fully-centralized methods, yet it affects greatly CTDE methods as well. Some techniques have been discussed in the literature to reduce such dimensionality. Among them, attention mechanisms offer the capacity to give different levels of importance to different parts of the state space when deciding on the best action to take [206]. In addition, we model our problem as a special type of Markov Decision Process (MDP) called Slate-MDP [207]. Such MDP allows one to select the slate or a set of actions simultaneously rather than selecting a single action

at a time. The combination of previous techniques allows us to formalize our proposal as:

Definition 1. Let $\mathcal{M} = \langle S, U, P, R, Z, O, \gamma \rangle$ be an MDP that defines a fully cooperative Decentralized Partially Observable Markov Decision Process (Dec-POMDP). Let $\kappa : \mathcal{S} \times U^l \rightarrow U$. At each environment step, each agent $n \in N$ chooses an action $u^n \in U^l$, forming a joint action $\mathbf{u}^l \in \mathbf{U}^l$. $s \in S$ describes the state of the environment. After selecting all actions, this causes a transition on the environment as $P'(s'|s, \mathbf{u}^l) : S \times \mathbf{U}^l \times S \rightarrow [0, 1]$. The team reward is calculated as $R'(s, \mathbf{u}^l) : S \times \mathbf{U}^l \rightarrow \mathbb{R}$ and $\gamma \in [0, 1)$ is a discount factor. Each agent draws individual observations $z \in Z$ according to observation function $O(s, a) : S \times N \rightarrow Z$. Now, the tuple $\left\langle S, U^l, P', R', Z, O, \gamma \right\rangle$ is called a slate dec-POMDP with an underlying MDP $\mathcal{M}$ and action-execution κ.

We leverage the attention mechanism by selecting a portion of the agent's observation to provide an available slate of actions. This can be seen in Fig. 6.2 (b), where we incorporate an attention filter after the mixing network of the oQMIX algorithm. More specifically, the transmission RLC buffer capacity ratio $\boldsymbol{(b)}$ is utilized to create a mapping between certain predefined buffer occupancy ranges and subsets of the action set $\mathbb{A}_t^n$ of the agent n^{th} at each time step t as follows:

$$f : \mathbb{A}_t^n \rightarrow \boldsymbol{b}_t \tag{6.1}$$

This mapping is used in Algorithm 6.1, line 13 by the function Is_Action_Disabled. This function looks in each time step for the set of disabled actions given the observed RLC buffer capacity. If the chosen action by the policy is in such a set, then the action selection procedure will repeat. The subsets are adaptively formed by the triggering of the done condition. The done condition is a well-common Boolean variable that informs when a game becomes unsolvable in RL. In our case, the done condition becomes true when any of the XR flows' throughput becomes zero upon the agent's action selection.

State space selection

The state space of the n^{th} agent of the multi-agent system is defined as follows:

$$s_t^n = \{\boldsymbol{a}_{(t-1)}, b_t^{RLC}, p_t^{XR}\}, \tag{6.2}$$

where $\boldsymbol{a}_{(t-1)}$ is a vector containing the team's previous codec parameter selection, b_t^{RLC} the RLC buffer occupancy at the BS and the p_t^{XR} corresponds to Packet Delivery Ratio (PDR) of of any of the traffic flows where $XR \in \{AR, VR, CG\}$. The first term helps to alleviate partial observability via the utilization of the Gated Recurrent Unit (GRU) layer in the Q-function as observed in Figure 6.2. The previous definition refers to the action space of a single agent in any of the MASs of our algorithm.

Action space selection

In this subsection, we present the proposed general action space:

$$a_t^{XR} = \{d_{min}^{XR}, d_{min}^{XR} + \frac{d_{max}^{XR} - d_{min}^{XR}}{K_o - 1}, ..., d_{max}^{XR}\}, \tag{6.3}$$

where d_{min}^{XR} and d_{max}^{XR} are predefined maximum and minimum values of the codec data rate values per XR traffic, respectively. The maximum and minimum values are defined in table 6.3.

Reward function

The reward function is carefully designed to satisfy QoE requirements. It is modeled as a team reward and can be defined as follows:

$$r_t = min(\hat{\mathcal{R}}_t^{XR}), \tag{6.4}$$

where $\hat{\mathcal{R}}_t^{XR}$ is a vector comprising the reward of all the agents forming the team, $XR \in \{AR, VR, CG\}$ corresponds to the three types of codec adaptation agents.

The individual reward of any of the XR codec agents can be defined in time t as:

$$\mathcal{R}_t^{XR} = \begin{cases} r_t^{XQI} & \text{if } \nexists x \in \hat{\mathbb{T}}_t : x = 0 \\ -1 & \textit{otherwise}, \end{cases} \tag{6.5}$$

$$r_t^{XQI} = \begin{cases} 1 & \text{if } p_t^{XR} \geq 99\% \text{ and } d_t^{XR} \leq 7 \\ 0.75 & \text{if } p_t^{XR} \geq 99\% \text{ and } 7 \leq d_t^{XR} \leq 10 \\ 0.5 & \text{if } p_t^{XR} \geq 95\% \text{ and } d_t^{XR} \leq 13 \\ 0.25 & \text{if } p_t^{XR} \geq 95\% \text{ and } 13 \leq d_t^{XR} \leq 20 \\ 0 & \text{otherwise.} \end{cases} \tag{6.6}$$

We present a reward that considers the XR Quality Index metric referred in [3] and section 6.1.2. The previously mentioned values are summarized in Table 6.1.

In addition, as indicated in equation (6.5) we penalize the event when after a selected combination of codec parameters by each agent any of the XR throughput flows (vector $\hat{\mathbb{T}}_t$) becomes 0. Furthermore, p_t^{XR} and d_t^{XR} represent the Packet Delivery Ratio and the average delay in ms during the observation window of each XR flow.

Table 6.1: KPI Mapping for XR [3]

XR Quality Index (XQI)	**PDR (%), PDB (ms)**
5	$(99, 7)$
4	$(99, 10)$
3	$(95, 13)$
2	$(95, 20)$
1	PDR < 95 or PDB > 20

6.1.3 Baselines: Adjust Packet Size Algorithm (APS) and QMIX

In this work, we compare our proposed scheme with one analog XR loopback threshold-based algorithm approach described in [124] and QMIX presented in [199].

As observed in algorithm 6.2, the codec parameter a_t^{XR} is adjusted by observing the Packet Loss Ratio (PLR), p_t^{XR}. The configuration variables $l_{dec}^s, l_{dec}^q, l_{inc}$ correspond to PLR thresholds and a_{max}, a_{min} to the maximum and minimum value of the codec parameter a_t^{XR}, respectively. In addition to the algorithmic baseline APS, we also include the results obtained by the QMIX algorithm in [199]. QMIX differently from oQMIX does not consider any Q-value function weighting strategy in the loss calculation.

6.1.4 Performance evaluation

Simulations are performed using the ns-3 New Radio (NR) module in its version 2.3 of April, 2023 [208]. The previous module known also as NR-Lena is built on top of the ns-3 simulator and provides simulation for 3GPP NR non-standalone cellular networks. In addition, we use as an interface between the ns-3 and the Python-based agents the module ns-3 gym [131].

Simulation Settings

RL parameters and simulation settings are described in Table 6.2 and 6.3, respectively. To study the impact of distance on the proposed algorithm and existent baseline, the users are randomly positioned in three different rings $r \in \{100 - 200, 200 - 300, 300 - 400\}$ m as observed in figure 6.2 (a). The users are randomly positioned within the inner radius of each of the rings and they maintain random mobility during the length of the simulation

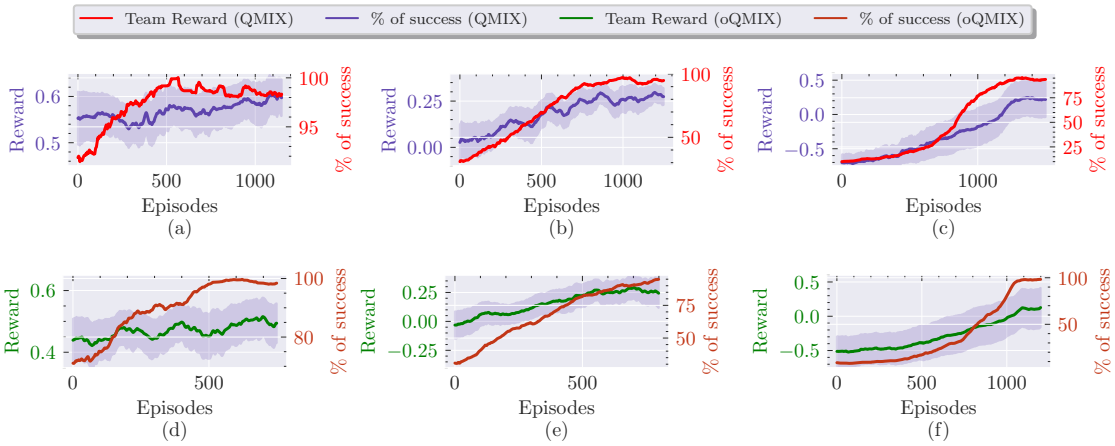

Figure 6.3: Convergence performance for QMIX: **(*a*)** 200 m, **(*b*)** 300 m and **(*c*)** 400 m and oQMIX: **(*d*)** 200 m, **(*e*)** 300 m and **(*f*)** 400 m

with a speed of 3 m/s. The transmission RLC buffer's capacity is limited to $6e4$ Bytes. It is considered an Urban Macro (UMa) channel model with all the nodes in non-line-of-sight and one bandwidth part with a central frequency of 4 GHz and bandwidth of 20 MHZ. In the next subsection, we will discuss the performance results of the proposed scheme.

Simulation Results

We present the performance results of our proposed schemes in terms of RL convergence, % of success, and KPIs as XR index, jitter, delay and PLR versus distance. For each distance, the data of 10 runs is collected and a 95% confidence interval is considered. In addition, we show flow-based graphs of the throughput and goodput. It is worth mentioning, that the proposed algorithm in [124], APS performs the best among all schemes in that work. Interestingly, the results obtained in such previous research show that when channel conditions are bad, the algorithm sends bigger frames. This behavior is undesired and must be avoided since increasing the XR packet size when the channel conditions are not favorable due to low Signal-to-Interference-Noise Ratio (SINR) is not the best strategy. Figure 6.3 shows the convergence and % of success for three different distances for algorithms QMIX and oQMIX, respectively. The success rate is defined as the percentage of completion of the simulation without triggering the done condition as in algorithm 6.1. As expected, when UEs are closer to the gNB as in figure 6.3 (a) and (d), the success rate is almost maximum with a short number of episodes. Meaning, that solving the problem

Table 6.2: Reinforcement Learning Settings.

Parameter	**Value**
Maximum training steps	30e4
Initial ϵ	1
ϵ decay steps	$15e3$ steps
ϵ minimum	5e$-$2
Buffer ($\mathcal{D}$) size	2e3
Batch size	64
Learning rate (α)	8e$-$3
Discount factor (γ)	0.99
Recurrent Layer hidden dimension	64
MultiLayer Perceptron hidden dimension	64
Weight initializer	Orthogonal
Deep Q-Network structure	Double Q-networks
Optimistic Weight (α_{opt})	0.1
The number of layers of hyper-network	2
The dimension of the hidden layer of the hyper-network	64
Number of agents in the MAS	3
Transmission RLC buffer ranges ($\boldsymbol{b}$)	$\{0-0.96, 0.96-0.97, 0.97-98$ $0.98-0.99, 0.99-1\}$
Observation window (t_w)	0.5 s

Table 6.3: Network Settings

Parameter	**Value**
Wireless network	New Radio (NR)
Channel Bandwidth	40 MHz
Central Frequency (f_c)	4 GHz
Number of UEs (N)	3
Number of gNB	1
Propagation Loss Model	UMa nLos
Numerology	2
gNB Noise Figure	5 dB
gNB Transmission Power	43 dBm
gnB Antenna configuration	4x8
UE Noise Figure	7 dB
UE Transmission Power	26 dBm
UE Antenna configuration	1x1
Max Transmission Buffer Size	60 KBytes
XR and CG traffic characteristics	AR (3 flows), VR (1 flow), CG (1 flow)
Codec data rate [min,max]	AR: [0.5,10], VR: [10,30], CG: [10,30] Mbps

when the signal-to-interference and noise ratio (SINR) is low becomes easy for our scheme. On the other hand, when the distance increases, the problem solution complexity also grows. This is related to the fact that only a small set of actions will be the ones that satisfy the requirements imposed by the reward function when stringent requirements are set. In figure 6.3 (b,e) and (c,f) can be observed that the reward convergence time and success rate increase considerably for both schemes. However, note that oQMIX offers a faster convergence with respect to QMIX.

Furthermore, in figure 6.4 we present the results in terms of XR index, jitter, delay, and PLR. We observe that when UEs are in the 200 and 300 m range, APS, QMIX, and oQMIX perform similarly in terms of the XR index with a small gain of 1.7% of the oQMIX algorithm. Conversely, oQMIX and QMIX offer an improvement when the distance increases in terms of average XR index up to 30.1% and 17.6%, respectively. Concerning other KPIs of interest, oQMIX and QMIX provide average gains over APS of 15.6%, 16.5% 50.3%, and 13.2%, 11.2%, 7.86% with respect jitter, delay, and PLR, respectively. In addition, we show in figure 6.4 a detailed comparison of throughput and goodput per XR flow versus distance. It can be seen in figure 6.4 (a) that the baseline APS, shows a more aggressive behavior by increasing the throughput. On the contrary, oQMIX and QMIX present a more conservative decision-making. Moreover, it can be observed in figure 6.4 (b) the performance in terms of goodput of the baseline APS and the proposed RL schemes. Note that goodput is calculated at the application layer meanwhile, throughput is calculated at the transport layer. APS shows a slightly better goodput performance, however this is due to the aggressive behavior of APS increasing the data rate and causing PLR performance degradation when distance increases. Evidently, the effects of increasing data rate and the consequent growth of PLR will be more noticeable when the number of XR users increases.

Algorithm 6.1 Optimistic Weighted QMIX with attention action selection

1: Let $\mathbb{A}^m_{b,t}$ be a vector of the set of disabled actions per m^{th} agent, size $|\boldsymbol{b}|$ in step t and $\hat{\mathbb{T}}_t$ be the vector of the observable throughput of all the XR flows at step t. Let $\boldsymbol{b}$ be a vector of predefined transmission RLC buffer's capacity ranges. Initialize θ and θ' indicating the target and evaluation recurrent policies, the parameters of mixing network, agent networks, and hypernetwork. Set the learning rate α and replay buffer $\mathcal{D} = \{\}$
2: episode $= 0, \theta^{'} = \theta$
3: **for** environment episode $e \leftarrow 1$ **to** E **do**
4: $t = 0, o_0 =$ initial state
5: **while** *done* $\neq True$ and $t <$ steps/episode limit **do**
6: **for** each agent m **do**
7: $\tau^m_t = \tau^m_{t-1} \cup \{(o_t, u_{t-1})\}$
8: $\epsilon =$ epsilon-schedule(step)
9: **if** $\epsilon \geq \mathcal{N}(0,1)$ **then**
10: $randint(1, |U|^m); |U|^m \subset \mathbb{Z}^+$
11: **end if**
12: $u^m_t = \begin{cases} {}_{u^m_t}Q(\tau^m_t, u^m_t) & \text{with probability } (1-\epsilon) \\ randint(1, |U|^m); |U|^m \subset \mathbb{Z}^+ & \text{with probability } \epsilon \end{cases}$
13: **if** Is_Action_Disabled$(u^m_t, \mathbb{A}^m_{b,s})$ **then**
14: Go to line 9.
15: **end if**
16: **end for**
17: Get reward r_t and next state s_{t+1}
18: $\mathcal{D} = \mathcal{D} \cup \{(s_t, \mathbf{u}_t, r_t, s_{t+1})\}$
19: steps $=$ steps $+ 1$
20: **if** $\nexists x \in \hat{\mathbb{T}}_t : x = 0$ **then**
21: *done* $= True$
22: Update $\mathbb{A}^m_{b,s}$; $\mathbb{A}^m_{b,s} = \mathbb{A}^m_{b,s} \cup u^m_t$
23: **end if**
24: **end while**
25: **if** $|\mathcal{D}| >$ batch-size **then**
26: b $\leftarrow$ random batch of episodes from $\mathcal{D}$
27: **for** each timestep t in each episode in batch b **do**
28: $Q_{tot} =$ *Mixing-network* $(Q_1(\tau^1_t, u^1_t), \ldots, Q_n(\tau^n_t, u^n_t)$;
29: *Hypernetwork*$(s_t; \theta))$
30: Calculate target Q_{tot} using *Mixing-network* with *Hypernetwork*$(s_t; \theta^{'}))$
31: **end for**
32: $\mathcal{L}(\theta) = \sum_{i=1}^{b} \left[w\left(s, \mathbf{u}\right) \left(y^{tot}_i - Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, s; \theta) \right)^2 \right]$
33: $\Delta\theta = \nabla_\theta (\mathcal{L}(\theta))^2$
34: $\theta = \theta - \alpha \Delta\theta$
35: **end if**
36: **if** hard update steps have passed **then**
37: $\theta^{'} = \theta$
38: **end if**
39: **end for**

Algorithm 6.2 Adjust Packet Size algorithm

Inputs: p_t^{XR} and a_t^{XR}
if $p_t^{XR} > l_{dec}^{s}$ and $p_t^{XR} < l_{dec}^{q}$ **then**
  $a_t^{XR} = \max(a_{t-1}^{XR} * \alpha_{dec}^{s}, a_{min})$
else if $p_t^{XR} \geq l_{dec}^{q}$ **then**
  $a_t^{XR} = \max(a_{t-1}^{XR} * \alpha_{dec}^{q}, a_{min})$
else if $p_t^{XR} < l_{inc}^{q}$ **then**
  $a_t^{XR} = \min(a_{t-1}^{XR} * \alpha_{inc}, a_{max})$
end if

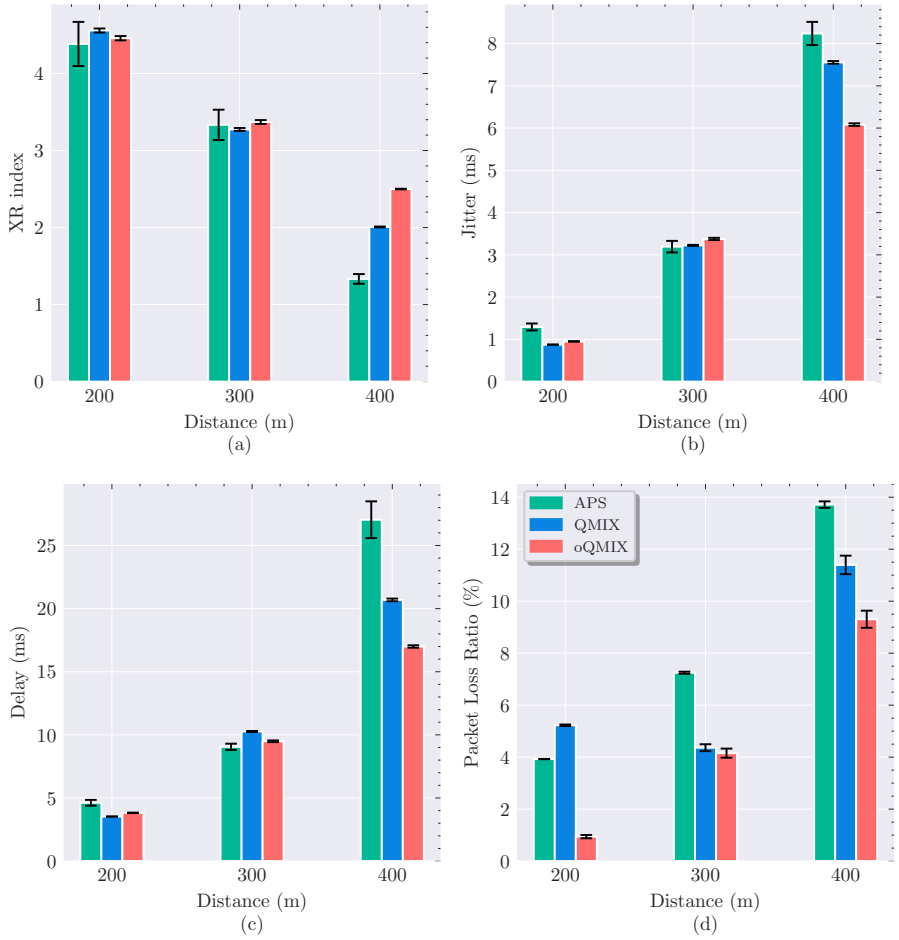

Figure 6.4: Key Performance Indicators of interest vs. Distance (***a***) XR index, (***b***) Jitter, (***c***) Delay, and (***d***) Packet Loss Ratio

Chapter 7

Conclusion

The ever-growing applications and advancements in Reinforcement Learning underscore its enduring relevance as a dynamic and influential paradigm in artificial intelligence, promising continuous innovation and adaptation to address contemporary challenges. On that note, we have presented in this thesis, the capabilities and opportunities of RL in the optimization of complex systems such as Wireless Access Networks. Moreover, we have proposed advanced state-of-the-art RL algorithms to improve processes in two main areas of Wireless Networks: cellular networks and Wi-Fi. In each case, our algorithms improved existing solutions and opened the door for new and interesting research areas. In the following, we summarize the fundamental discoveries of our research.

7.1 RL-based Load balancing and Handover in 4G and 5G Networks

In chapter 3 and the first section, we presented a Clipped Double Q-Learning strategy that performs load balancing with awareness of QoS metrics. As the main difference from previous works, our RL method uses a state-of-the-art algorithm and a QoS-aware load balancing approach enhancing the overall KPI metrics such as throughput, delay, jitter, and packet delivery ratio. We compared our proposed scheme with two baselines: the traditional A3 handover algorithm and a resource block utilization-based handover scheme, named ReBuHa. The results showed an average improvement up to 6.1% and 9.5% in terms of throughput, 49.8% and 52.9% in terms of delay, 55% and 51% in terms of jitter,

and 34% and 55.2% in terms of PLR, in comparison to the A3 and ReBuHa algorithms, respectively. Additionally, we evaluated the performance of our scheme under mobility. The results revealed the importance of performing load balancing while maintaining latency and CQI metrics. Despite the performance of CDQL in load balancing, we introduced a multi-agent alternative considering the competition among BSs. Thus, in section 2 we presented a Multi-Agent Deep Deterministic Policy Gradient with Adaptive Policies and ranked buffer strategy where BSs compete to achieve load balancing with awareness of QoS metrics. Our work differentiates from previous approaches mainly with the proposal of a state-of-the-art adaptive competitive MARL with a continuous action state scheme with self-supervision capabilities. We compared our proposed method with a baseline (CDQL) utilized in a previous work that has overcome the performance of traditional handover algorithms. Although no evident improvement was obtained in terms of throughput, significant gain was obtained in terms of average delay with 25.7% and PLR with 28.11%, in comparison to CDQL scheme. Additionally, an average improvement was achieved in the execution stage of 70.49% concerning the CDQL scheme.

Later, in section 3 we presented two Reinforcement Learning (RL)-based dual connectivity (DC) solutions that further improve latency metrics in inter-RAT handover. Additionally, we propose leveraging an initial access cell search algorithm that utilizes context information to reduce the latency in gNB-UE's measurement collection. We compared two RL solutions named Hierarchical Deep Q-Learning (HiDQL) and Clipped Double Q-Learning (CDQL) with two previously proposed baselines, fixed Threshold to Time (TTT) and dynamic TTT, obtaining a significant gain in both cases in terms of latency with a 47.6% and 26.1%, for Digital-Analog BF, 17.1% and 21.6%, for Hybrid-Analog BF and 24.7% and 39% for Analog-Analog BF when compared with the baselines' best results. Finally, we presented a context-aware mechanism under line-of-sight assumptions and obtained a significant improvement in terms of latency when compared with the case of no context. As future work, we foresee the integration of the presented context-aware mechanism and reinforcement learning to further improve the DC handover mechanism.

7.2 Multi-Agent Team Learning in Virtualized Open Radio Access Networks (O-RAN)

In chapter 4, we discuss the potential of using MATL to organize intelligence in O-RAN. We first give an overview of RAN evolution from C-RAN to vRAN to O-RAN, and then we provide a summary of MAS and MATL and their applications in O-RAN. Additionally, we present a case study on team learning in the O-RAN architecture by comparing

three schemes, sequential multi-agent deep reinforcement learning (SMADRL), concurrent multi-agent deep reinforcement learning (CMADRL), and team multi-agent deep reinforcement learning (TMADRL). Simulation results show that team learning via the TMADRL scheme offers a gain in terms of energy utilization of 29.7% and 59.04% when compared to SMADRL and CMADRL, respectively. Additionally, TMADRL overperforms CMADRL and SMADRL in terms of throughput with a gain of 1% and 4.56%, respectively. Finally, we provide a detailed discussion on challenges, open issues, and future directions for MATL in O-RAN. As future work, we intend to keep exploring the dynamics of team learning in the O-RAN architecture and 6G.

7.3 Multi-agent Deep Reinforcement Learning for Next Generation Wi-Fi Networks

New-generation Wi-Fi networks will serve user-dense and dynamic environments. To support this reality, the newest amendments include several novel features such as coordinated spatial reuse and multi-link operation MLO. In chapter 5, we present several RL-based alternatives for the optimization of spatial reuse, traffic allocation, and channel selection in MLO capable networks.

In the first and second sections, we study specifically spatial reuse. Firstly, we have proposed Machine Learning (ML)-based solutions to optimize Spatial Reuse (SR) in distributed Wi-Fi 802.11ax/802.11be scenarios. We have studied if cooperation is indeed needed for spatial reuse optimization and presented a solution based on transfer learning to improve the adaptive behavior of our proposal. The proposed algorithm MA-CMAB, named SAU-Sampling in the cooperative setting, contributes positively to an increase in throughput and fairness and a reduction of PLR when compared with non-cooperative approaches. Under dynamic scenarios, transfer learning benefits the SAU-Sampling algorithm to overcome service drops for at least 60% of the total users when utilizing the forget method. In conclusion, the utilization of the cooperative version of the MA-CMAB to improve SR in Wi-Fi scenarios is preferable since it outperforms the presented ML-based solutions and prevents service drops in dynamic environments via transfer learning. In the second section, we have proposed meta-learning-based improvement for spatial reuse in distributed cooperative Wi-Fi 802.11ax/be networks by leveraging a Multi-Agent Contextual Multi-Armed Bandit (MA-CMAB). The results have shown that meta-learning can improve spatial reuse adaptation in terms of convergence for highly dense and dynamic scenarios when compared to a transfer learning baseline.

In the third section of this chapter, we delve into traffic allocation in MLO capable devices. We presented a Soft-Actor Critic (SAC) Reinforcement Learning (RL) agent to improve performance metrics such as Throughput Drop Ratio (TDR) and Flow Satisfaction (FS) in IEEE 802.11be MLO capable networks. More specifically, we proposed an agent named Multi-Headed Recurrent Soft-Actor Critic (MH-RSAC) that is capable of dealing with different types of Multi-Link Devices (MLD)s. We included the modifications required in its design such as the "**avg**" operator to reduce underestimation in the SAC algorithm. In addition, we acknowledged the non-Markovian nature of the scenario under study and included two main techniques that involve the usage of Long Short-Term Memory (LSTM) neural networks and rewards with hindsight. Moreover, we presented the derivation of a Virtual Reality (VR) traffic to be further utilized in any wireless simulator. We compared the proposed RL variants in terms of convergence and observed the best performance for the case of the MH-RSAC with "**avg**" operator and utilization of rewards with hindsight (MH-RSAC(Q_{avg}, R_h)). Furthermore, we presented the simulation results of our proposed scheme and two previously defined baselines named *Single Link Less Congested Interface* (SLCI) and *Multi Link Congestion-aware Load balancing at flow arrivals* in terms of TDR and FS per traffic type. Results showed that MH-RSAC(Q_{avg}, R_h) outperforms in terms of TDR the SLCI baseline with an average gain of 34.2% and 35.2% and the MCAA baseline with 2.5% and 6% in the **U1** and **U2** proposed scenarios, respectively. Finally, we observed that our scheme can respond more efficiently to high throughput and dynamic traffic such as VR and Web Browsing (WB) when compared with the baselines. Results showed an improvement of the MH-RSAC scheme in terms of FS of up to 25.6% and 6% for the SCLI and MCAA, respectively.

Finally, in the last section of this chapter, we presented a Parallel Transfer Reinforcement Learning (PTRL) and cooperative optimistic-weighted VDN algorithm to improve radio and Reinforcement Learning (RL) related metrics such as Modulation Code Selection (MCS) and convergence speed in IEEE 802.11be channel selection. To the best of our knowledge, we studied for the first time, PTRL techniques in the context of knowledge transfer among Multi-Agent Systems (MASs). Specifically, we proposed two techniques named: **MAS Joint Q-function Transfer** and **MAS Best/Worst Experience Transfer**. The previous techniques consist of transferring the joint Q-Function of cooperative MARL algorithms such as VDN and transferring the best and worst experiences during each episode. Moreover, we presented a modification to the VDN algorithm taken from another technique named QMix that allows to weight of the contribution of each agent to the joint Q-function. Furthermore, we presented the simulation results of our proposed scheme and analyzed how each PTRL method affects its behavior. Results showed that the oVDN$_g$ offers an important improvement in terms of MCS over the non-parallel base-

line in two interfaces (2.4 and 5 GHz) with MCS in the range of $(10-13)$ over $(8-11)$. Additionally, we present a metric gain to calculate the MCS improvement. We observed that the oVDN$_g$ variant offers a gain up to 3%, 7.2%, and 11% when compared with VDN, VDN-nonQ, and non-PTRL baselines. In terms of convergence speed among oVDN alternatives, we show a reward convergence gain in the 5GHz interface of 33.3% over oVDN$_b$ and oVDN.

7.4 Multi-Agent Reinforcement Learning-based XR codec adaptation

In chapter 6, we presented a Multi-Agent Reinforcement Learning (MARL) algorithm with Attention Action Selection to improve Extended Reality (XR) Key Performance Indicators (KPIs) of interest. More specifically, the proposed algorithm named Optimistic QMIX (oQMIX) uses attention action selection to reduce the action set comprised by the codec parameters of the Cloud Gaming (CG) and XR traffics: Augmented Reality (AR), Virtual Reality (VR). We presented the results in terms of convergence and % of success of three different scenarios with varying distances. Results showed that when the distance between the UE and the base station increases, the problem difficulty grows and convergence time also presents the same behavior. Furthermore, we presented the simulation results of our proposed scheme and a state-of-the-art baseline Adjust Packet Size (APS). Results show that oQMIX overperforms APS with an average gain of 30.1.%, 15.6%, 16.5% 50.3% with respect to XR index, jitter, delay, and PLR, respectively. On the other hand, we observed that APS presented a more aggressive behavior with the tendency of higher throughput in all XR flows, increasing packet collisions and packet loss when distance increased. Conversely, oQMIX presented a more conservative behavior reducing PLR and maintaining a similar behavior in terms of goodput for both under-study algorithms.

7.5 Challenges and Open Issues

Reinforcement Learning (RL) has shown immense potential for many applications in the real world. From robotics, game playing, autonomous vehicles, finance, healthcare, Natural Language Processing (NLP), Recommendation Systems, Energy Management, Marketing and Advertising, Telecommunications, and many others, RL has provided efficient and brilliant solutions for many of the most challenging problems in each of those fields. In the

area of Telecommunications and more specifically, Wireless Area Networks such as cellular and Wi-Fi, some challenges and open issues need to be addressed to fully use the capacity of the several existent RL techniques. Some of the challenges and open issues are listed below:

(i) **Inherent RL issues**: There are several inherent open issues related to the application of RL in any domain. The impact of these issues can vary based on the environment and the specific technique applied. Throughout the various applications addressed in this thesis, we have identified general issues within cellular and Wi-Fi network environments.
`(General issues)`: For instance, high-dimensional and continuous action spaces often require careful tuning of hyperparameters related to exploration strategies, such as the exploration noise level, to ensure effective learning. Typically, researchers sacrifice the number of actions by discretizing the continuous action space or significantly reducing the action space, losing some potential actions to explore. Another important issue is sample efficiency, especially in wireless networks where data collection is expensive or time-consuming. It remains an ongoing challenge. The exploration vs. exploitation dilemma represents a fundamental open issue in the wireless access networks environment due to the complexity and specific characteristics of such an environment. This issue is closely related to the size of the action and state space, as exploring too little could leave actions unexplored that could potentially be optimal. Reward shaping and formulation correspond to another challenge in RL-based Wireless Access Networks applications, as it depends on the application and is hardly reusable in other problems. In the following, we introduce some specific issues.

`(Specific issues)`:RL Model Selection in Wireless Networks is still an open question since a priori any model can offer good performance. Selecting and learning a model of the environment is not typically clear and could vary according to many unrelated factors such as computational capability and time requirements. Also, different RL algorithms (e.g., DQN, double Deep Q-Network (DDQN), MADDPG, etc.) have specific hyperparameters. Developing a unified approach for tuning hyperparameters across diverse RL algorithms is still an open challenge.

(ii) **Standardized wireless network environments**: Reproducibility in wireless simulations is crucial for validating research findings, comparing algorithms, and promoting transparency in scientific experiments. However, in many cases where the pseudocode of RL algorithms is depicted in existing research and their code is available on open platforms, the provision of a standardized playground or environment is

not guaranteed. Typically, research papers only mention network parameters without providing additional information on the simulator's implementation. This lack of detail can create an issue as it may lead to a misperception of an algorithm's performance. Oversimplified simulators fail to depict real-world scenarios, rendering proposals less representative of actual networks. Another consequence of oversimplification is that RL algorithms are expected to be robust in noisy environments. Training them under such conditions may result in oversimplified results that could prove catastrophic when applied in real networks. Despite these challenges, substantial efforts have been made in this area, and one solution could be the standardization of open-source simulators throughout academia, such as ns-3 or OMNet++, which have been validated by thousands of researchers.

(iii) **Scarcity of real Wireless Networks datasets**:Advancement in the field of RL for wireless communications and networks relies heavily on the availability of datasets for testing purposes. This is especially true for offline RL. However, creating reference or standard datasets from practical simulations or experimental testbeds can be challenging and expensive for research organizations. The limited accessibility of such datasets can serve as a significant bottleneck in this research. Moreover, commercial datasets from telecommunication operators are typically out of reach for most researchers. Therefore, generating standard reference datasets for research, which cover low-level physical layer measurements to telecommunication network analyses, remains a costly and challenging task.

(iv) **Long convergence time of RL algorithms in Wireless Networks**: The amount of time required for training RL algorithms in Wireless Networks is a major concern. The complexity of the environment raises questions about how long it will take for the algorithm to converge. In dynamic and unpredictable environments, models trained for a long time may not be useful and might not work effectively. Therefore, to make RL applicable in Wireless Networks, adaptive and robust methods must be adopted. Some techniques, such as transfer learning, parallel transfer learning, meta-learning, and federated learning, have proven to be effective in accelerating learning, increasing robustness, and providing adaptability. However, there is a need for further research to determine how to select the appropriate knowledge to avoid unwanted behaviors like negative transfer.

(v) **Reproducibility of Multi-Agent Reinforcement Learning algorithms in Wireless Networks**: Many efforts have been made in the MARL community to standardize performance reporting in the game and robotics fields. However, a benchmarking tool for such algorithms in Wireless Networks applications is still lacking. Such a

tool could, in principle, enable the reproducibility and standardization of MARL in well-studied applications such as Radio Resource Management (RRM) or beamforming.

Appendix A

Proofs

A.1 Useful properties

Let us assume that action 1 refers to the optimal matching given an optimal team policy $\mu_{k^{**}}$, where $\mu_1 = \mu_{k^{**}}$ and k^{**} corresponds to the optimal bipartite matching. Similar to [76] we define $r_{k,1}, r_{k,2}, \cdots, r_{k,n_k}$ be the random variables referring to the rewards yielded by action k of successive n_k plays. Also, we fix n and avoid the usage of such subscript. The notation "k, j" means that the number of plays of action k is j. X_k is a random variable drawn from the normal distribution $\mathcal{N}(\hat{\mu}_{k,n_k}, \tau_k^2/n_k)$, where

$$\hat{\mu}_{k,n_k} = \frac{1}{n_k}\sum_{j=1}^{n_k} r_{k,j} \quad \text{and } \tau_k^2 = \frac{1}{n_k}\sum_{j=1}^{n_k}(r_{k,j} - \hat{\mu}_{a,j-1})^2.$$

Thus X_k becomes:

$$X_k = \hat{\mu}_{k,n_k} + \delta_{k,n_k}, \quad \text{where } \delta_{k,n_k} \sim \mathcal{N}(0, \tau_k^2/n_k). \tag{A.1}$$

Gaussian Tail bound: Based on the definition in the SAU-Sampling algorithm in [76], δ_{k,n_k} in probability. By the Gaussian tail bound [209, Definition 2.1], for $\alpha > 0$,

$$\Pr\left\{\delta_{k,n_k} \geq \alpha | \tau_k^2\right\} \leq \exp\left\{-\frac{\alpha^2 n_k}{2\tau_k^2}\right\} \leq \exp\left\{-n_k\alpha^2/2\right\}, \tag{A.2}$$

where the second inequality is from that $\tau_k^2 \leq 1$. Eq. (A.2) follows that

$$\Pr\left\{\delta_{k,n_k} \geq \alpha\right\} \leq \exp\left\{-n_k\alpha^2/2\right\}. \tag{A.3}$$

A.1.1 Proof of Theorem 5.1

Proof. Let us define first the team regret based on the suboptimal selection of bipartite matching k*:

$$R(n) = \sum_{k*:\mu_{k*}<\mu_{k^{**}}} \Delta_{k^{**}} \mathcal{E}[T_{k*}(n)] = \Delta_{k^{**}} \sum_{k*:\mu k*<\mu k^{**}} \mathcal{E}[T_{k*}(n)] \tag{A.4}$$

$$\leq \Delta_{k^{**}} \sum_{i=1}^{N} \sum_{j=1}^{K} \mathcal{E}[\tilde{T}_{ij}(n)] \tag{A.5}$$

where $\tilde{T}_{ij}(n) \in \mathbb{R}^{K\times N}$ corresponds to a counter that gets incremented by 1 when a non-optimal matching is selected by the SAU-Sampling MA-CMAB algorithm. Additionally, the tuple (i, j) indicates the j^{th} arm selected by the i^{th} agent. $\Delta_{k^{**}}$ is defined as $\Delta_{k^{**}} = \mu_{n,k^{**}}(n) - \mu_{n,k}(n)$.

Denote $c_{1,n} = N\sqrt{\frac{(N+1)\log n}{n_1}}$ and $c_{k,n} = N\sqrt{\frac{(N+1)\log n}{n_k}}$. Let

$$P_1 = \{\hat{\mu}_{1,n_1} > \mu_1 - \frac{1}{N}c_{1,n}\}, \text{ and } P_k = \{\hat{\mu}_{k,n_k} < \mu_k + \frac{1}{N}c_{k,n}\} \text{ for } k = 2, \cdots, K.$$

where $\bar{P}_1$ and $\bar{P}_a$ be the complements of P_1 and P_k respectively. $\Pr\left\{\bar{P}_1\right\}$ and $\Pr\left\{\bar{P}_a\right\}$ are bounded from the Azuma-Hoeffding inequality [209, Corollary 2.1]:

$$\Pr\left\{\bar{P}_1\right\} = \Pr\left\{\hat{\mu}_{1,n_1} \leq \mu_1 - \frac{1}{N}c_{1,n}\right\} \leq \exp(-2(M+1)\log n) = n^{-2(M+1)}; \tag{A.6}$$

$$\Pr\left\{\bar{P}_a\right\} = \Pr\left\{\hat{\mu}_{k,n_k} \geq \mu_k + \frac{1}{N}c_{k,n}\right\} \leq \exp(-2(M+1)\log n) = n^{-2(M+1)}. \tag{A.7}$$

Define for $\eta \in \mathbb{R}$

$$Q_{k,n_k}(\eta) = \Pr(X_k \geq \eta).$$

Following Eq. (A.4) and taken into account [169], the following lemma is used:

$$R(n) \leq \sum_{k*:\mu_{k*}<\mu_{k^{**}}} \Delta_{k**}(R_a + R_b), \tag{A.8}$$

where

$$R_a = \sum_{n_1=0}^{n-1} \mathcal{E}\left[\min\left\{\frac{1}{NQ_{1,n_1}(\eta)} - 1, n\right\}\right] \text{ and } R_b = \sum_{n_k=0}^{n-1} \Pr[NQ_{k,n_k}\eta) > 1/n] + 1.$$

We set

$$\eta_{k**} = \mu_{k**} + \frac{\Delta_{k**}}{2} = \mu_1 - \frac{\Delta_{k**}}{2}. \tag{A.9}$$

First, we split R_a into two terms $R_a^{(1)}$ and $R_a^{(2)}$ and bound them by applying Theorem [169, Theorem 1]. Now in the first step, we derive the upper bound on $R_a^{(1)}$. Denote $\bar{n}_{k**} = \frac{24N^2 \log n}{\Delta_k^2}$. Noting $Q_{1,0}(\eta_k) = 1$ and $\lceil \bar{n}_{k**} \rceil$ is the smallest integer not less than $\bar{n}_{k**}$.

$$\begin{aligned} R_a &= \sum_{n_1=1}^{n-1} \mathcal{E}\left[\min\left\{\frac{1}{NQ_{1,n_1}(\eta_{k**})} - 1, n\right\}\right] \\ &= \sum_{n_1=1}^{\lceil \bar{n}_{k**} \rceil - 1} \mathcal{E}\left[\min\left\{\frac{1}{NQ_{1,n_1}(\eta_{k**})} - 1, n\right\}\right] + \sum_{n_1=\lceil \bar{n}_{k**} \rceil}^{n-1} \mathcal{E}\left[\min\left\{\frac{1}{NQ_{1,n_1}(\eta_{k**})} - 1, n\right\}\right] \\ &=: R_a^{(1)} + R_a^{(2)}, \end{aligned} \tag{A.10}$$

The law of total probability implies that, when $n_1 \geq \bar{n}_{k**}$,

$$\begin{aligned} Q_{1,n_1}(\eta_{k**}) =& \Pr\{X_1 > \eta_{k**}\} = 1 - \Pr\{X_1 < \eta_{k**}\} \\ =& 1 - \Pr(P_1)\Pr\{X_1 < \eta_{k**}|P_1\} - \Pr(\bar{P}_1)\Pr\left\{X_1 < \eta_{k**}|\bar{P}_1\right\} \\ >& 1 - \Pr\{\hat{\mu}_{1,n_1} + \delta_{1,n_1} < \mu_1 - \Delta_{k**}/2|P_1\} - \Pr(\bar{P}_1) \\ \geq& 1 - \Pr\left\{\delta_{1,n_1} \leq -\Delta_{k**}/2 + \frac{1}{N}c_{1,n}\right\} - \Pr(\bar{P}_1) \\ >& 1 - \Pr\left\{\delta_{1,n_1} \leq -\frac{\sqrt{2}}{N}c_{1,n}\right\} - \Pr(\bar{P}_1) \\ \geq& 1 - n^{-(N+1)} - n^{-2(N+1)}, \end{aligned} \tag{A.11}$$

where the 1st inequality is from the facts that $\Pr(P_1) < 1$ and $\Pr\left\{X_1 < \eta_{k**}|\bar{P}_1\right\} < 1$, the 2nd inequality is from the definition of P_1, the 3rd inequality is from $\Delta_{k**}/2 - \frac{1}{N}c_{1,n} \geq \frac{\sqrt{2}}{N}c_{1,n}$ as $n_1 \geq \bar{n}_{k**}$, and the last inequality is from Eqs. (A.3) and (A.6).

Eq. (A.11) implies that for $n \geq 3$ and $N \geq 1$,

$$R_a^{(1)} = \sum_{n_1=\lceil \bar{n}_{k**} \rceil}^{n-1} \mathcal{E}\left[\min\left\{\frac{1}{NQ_{1,n_1}(\eta_{k**})} - 1, n\right\}\right] = \sum_{n_1=\lceil \bar{n}_{k**} \rceil}^{n-1} \min\left\{\frac{1}{NQ_{1,n_1}(\eta_{k**})} - 1, n\right\}$$
$$< \sum_{n_1=\lceil \bar{n}_{k**} \rceil}^{n-1} \frac{1}{N(1 - n^{-(N+1)} - n^{-2(N+1)})} - 1 < 2N \sum_{n_1=\lceil \bar{n}_{k**} \rceil}^{n-1} (n^{-(N+1)} + n^{-2(N+1)}) < \frac{3N}{N+1}, \tag{A.12}$$

where the 1st inequality fulfills the condition $\frac{1}{Q_{1,n_1}(\eta_{k**})} - 1 < \frac{1}{1-n^{-(N+1)}-n^{-2(N+1)}} - 1 < n$, the 2nd inequality is from that $\frac{1}{1-n^{-(N+1)}-n^{-2(N+1)}} - 1 < 2(n^{-(N+1)} + n^{-2(N+1)})$ when $n \geq 3$ and $N \geq 1$.

It follows the calculation of the lower bound of R_a, $R_a^{(2)}$ in Eq. (A.10). A second lower bound can be derive from $Q_{1,n_1}(\eta_{k**})$ as follows: Let

$$P_1^L = \{\hat{\mu}_{1,n_1} \geq \mu_1\}.$$

Denote $\bar{P}_1^L$ be the complements of P_1^L. We have that

$$\Pr\left\{P_1^L\right\} = 1/2; \quad \Pr\left\{\bar{P}_1^L\right\} = 1/2 \tag{A.13}$$

Similarly as Eq. (A.11), we have that

$$\begin{aligned} Q_{1,n_1}(\eta_{k**}) =& \Pr\left\{X_{1,n_1} > \eta_{k**}\right\} \\ =& 1 - \Pr(P_1^L)\Pr\left\{X_{1,n_1} < \eta_{k**} | P_1^L\right\} - \Pr(\bar{P}_1^L)\Pr\left\{X_{1,n_1} < \eta_{k**} | \bar{P}_1^L\right\} \\ >& 1 - 1/2\Pr\left\{\hat{\mu}_{1,n_1} + \frac{1}{N} c_{1,n}\delta_{1,n_1} < \mu_1 - \Delta_{k**}/2 | P_1^L\right\} - \Pr(\bar{P}_1^L) \\ \geq& 1/2 - 1/2\Pr\left\{\frac{1}{N} c_{1,n}\delta_{1,n_1} \leq -\Delta_{k**}/2\right\} \\ \geq& \frac{1}{2}\left[1 - \exp\left(-\frac{n_1\Delta_{k**}^2}{8N^2 \log n}\right)\right], \end{aligned} \tag{A.14}$$

where the 1st inequality is from the facts that $\Pr\left\{X_{1,n_1} < \eta_{k**} | \bar{P}_1^L\right\} < 1$, and the 2nd inequality is from the definition of P_1^L and Eq. (A.13), and the last inequality is from Eq. (A.3).

We have that (1) $\log\left(1 - \frac{2}{n+1}\right) \geq \frac{-3}{n+1}$ when $n \geq 4$; (2) $\frac{-3}{n+1} \geq \frac{-3}{n}$; and (3) $-\frac{\Delta_{k**}^2}{8N^2 \log n} \leq \frac{-3}{n+1}$ when $\frac{n}{\log n} \geq \frac{24N^2}{\min_{k**} \Delta_{k**}^2}$. The three inequalities imply that when $n \geq \max\{\frac{24N^2 \log n}{\min_{k**} \Delta_{k**}^2}, 4\}$,

$$1 - \exp\left(-\frac{\Delta_{k**}^2}{8N^2 \log n}\right) \geq \frac{2}{n+1}.$$

Thus, Eq. (A.14) follows that

$$\begin{aligned}
R_a^{(1)} =& \sum_{n_1=1}^{\lceil \bar{n}_{k**} \rceil - 1} \mathcal{E}\left[\min\left\{\frac{1}{NQ_{1,n_1}(\eta_{k**})} - 1, n\right\}\right] \\
\leq& \sum_{n_1=1}^{\lceil \bar{n}_{k**} \rceil - 1} \left[\frac{2}{1 - \exp\left(-\frac{n_1 \Delta_{k**}^2}{8N^2 \log n}\right)} - 1\right] \\
<& \sum_{n_1=1}^{\lceil \bar{n}_{k**} \rceil - 1} \left[4\exp\left(-\frac{n_1 \Delta_{k**}^2}{8N^2 \log n}\right) - 1\right] \\
<& 3\bar{n}_{k**}.
\end{aligned} \tag{A.15}$$

Therefore, inserting Eqs. (A.12) & (A.15) into Eq. (A.10),

$$R_a^{(1)} \leq 3\bar{n}_k + 4. \tag{A.16}$$

In the next step, we derive the upper bound on $R_a^{(2)}$. When $n_{k**} \geq \bar{n}_{k**}$,

$$\Delta_{k**}/2 - c_{k,n} \geq \sqrt{2} c_{k,n}. \tag{A.17}$$

From the definition of event P_k, the law of total probability implies that

$$\begin{aligned}
\Pr\left\{\frac{1}{N} Q_{k,n_k}(\eta_k) > 1/n\right\} =& \Pr(P_k)\Pr\left\{\frac{1}{N} Q_{k,n_k}(\eta_k) > 1/n | P_k\right\} + \Pr(\bar{P}_a)\Pr\left\{\frac{1}{N} Q_{k,n_k}(\eta_k) > 1/n | \bar{P}_a\right\} \\
\leq& \Pr\left\{\frac{1}{N} Q_{k,n_k}(\eta_k) > 1/n | P_k\right\} + \Pr(\bar{P}_a).
\end{aligned} \tag{A.18}$$

When $n_k \geq \bar{n}_k$, given event P_k,

$$\begin{aligned}
Q_{k,n_k}(\eta_k) =& \Pr\left\{X_k > \eta_k | P_k\right\} = \Pr\left\{\hat{\mu}_a + \delta_{k,n_k} > \Delta_k/2 + \mu_k | P_k\right\} \\
\leq& \Pr\left\{\delta_{k,n_k} \geq \Delta_k/2 - \frac{1}{N} c_{k,n}\right\} \\
\leq& \Pr\left\{\delta_{k,n_k} \geq \frac{\sqrt{2}}{N} c_{k,n}\right\} \\
\leq& \exp\left\{-(N+1)\log n\right\} = n^{-(N+1)}.
\end{aligned} \tag{A.19}$$

where the 1st inequality is from the definition of event P_k, the 2nd inequality is from Eq. (A.17), the 3rd inequality is from Eq. (A.3).

Eq. (A.19) follow that when $n_k \geq \bar{n}_k$,

$$\Pr\left\{\frac{1}{N}Q_{k,n_k}(\eta_k) > 1/n | P_k\right\} = 0. \tag{A.20}$$

Inserting Eq. (A.20) into Eq. (A.18),

$$\Pr[\frac{1}{N}Q_{k,n_k}(\eta_k) > 1/n] \leq \Pr(\bar{P}_a) \leq n^{-(N+1)}, \tag{A.21}$$

where the last step is from Eq. (A.7).

We have that

$$\begin{aligned}
R_b =& \sum_{n_k=0}^{n-1} \Pr[NQ_{k,n_k}(\eta_k) > 1/n] + 1 \\
\leq& \sum_{n_k=0}^{\lceil \bar{n}_k \rceil} \Pr[NQ_{k,n_k}(\eta_k) > 1/n] + 1 + \sum_{n_k=\lceil \bar{n}_k \rceil}^{n-1} \Pr[NQ_{k,n_k}(\eta_k) > 1/n] + 1 \\
\leq& N\bar{n}_k + \sum_{n_k=\lceil \bar{n}_k \rceil}^{n-1} \Pr[NQ_{k,n_k}(\eta_k) > 1/n] + 1 \\
<& N\bar{n}_k + (N+1) \\
<& N(\bar{n}_k + 1) + 1,
\end{aligned} \tag{A.22}$$

Finally, we substitute R_a and R_b terms in Eq. (A.8) having:

$$R(T) \leq N^2 K \Delta_{k^{**}} \left(\frac{96 \log n}{\Delta_{k^{**}}^2} + \frac{1}{1+N}\right) \tag{A.23}$$

□

Bibliography

[1] L. G. Giordano, G. Geraci, M. Carrascosa, and B. Bellalta, "What Will Wi-Fi 8 Be? A Primer on IEEE 802.11bn Ultra High Reliability," *arXiv preprint arXiv:2303.10442*, 2023.

[2] A. Nauman, T. N. Nguyen, Y. A. Qadri, Z. Nain, K. Cengiz, and S. W. Kim, "Artificial Intelligence in Beyond 5G and 6G Reliable Communications," *IEEE Internet of Things Magazine*, vol. 5, no. 1, pp. 73–78, 2022.

[3] S. Dou, S. Liao, J. Wu, K. Wu, E. Chen, W. Chen, H. Shen, and N. Li, "XR Quality Index: Evaluating RAN Transmission Quality for XR services over 5G and beyond," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, pp. 1–6, 2021.

[4] O-RAN-SC, "RIC Message Router – RMR." `https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-lib-rmr/en/latest/rmr.7.html`. Accessed: 20 November 2021.

[5] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Commun. Surv. Tutor.*, vol. 23, no. 2, pp. 1226–1252, 2021.

[6] M. Usama, J. Qadir, A. Raza, H. Arif, K.-l. A. Yau, Y. Elkhatib, A. Hussain, and A. Al-Fuqaha, "Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges," *IEEE Access*, vol. 7, pp. 65579–65615, 2019.

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2nd ed., 2018.

[8] M. E. Morocho-Cayamcela, H. Lee, and W. Lim, "Machine Learning for 5G/B5G Mobile and Wireless Communications: Potential, Limitations, and Future Directions," *IEEE Access*, vol. 7, pp. 137184–137206, 2019.

[9] M. Elsayed and M. Erol-Kantarci, "AI-Enabled Future Wireless Networks: Challenges, Opportunities, and Open Issues," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 70–77, 2019.

[10] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The Roadmap to 6G: AI Empowered Wireless Networks," *IEEE Commun. Mag.*, vol. 57, no. 8, pp. 84–90, 2019.

[11] A. Dogra, R. K. Jha, and S. Jain, "A Survey on Beyond 5G Network With the Advent of 6G: Architecture and Emerging Technologies," *IEEE Access*, vol. 9, pp. 67512–67547, 2021.

[12] M. N. Mahdi, A. R. Ahmad, Q. S. Qassim, H. Natiq, M. A. Subhi, and M. Mahmoud, "From 5G to 6G Technology: Meets Energy, Internet-of-Things and Machine Learning: A Survey," *Applied Sciences*, vol. 11, no. 17, 2021.

[13] L. Qiao, Y. Li, D. Chen, S. Serikawa, M. Guizani, and Z. Lv, "A survey on 5G/6G, AI, and Robotics," *Computers and Electrical Engineering*, vol. 95, p. 107372, 2021.

[14] H. Fourati, R. Maaloul, and L. Chaari, "A survey of 5G network systems: challenges and machine learning approaches," *International Journal of Machine Learning and Cybernetics*, vol. 12, no. 2, pp. 385–431, 2021.

[15] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang, and L.-C. Wang, "eep Reinforcement Learning for Mobile 5G and Beyond: Fundamentals, Applications, and Challenges," *IEEE Veh. Technol. Mag.*, vol. 14, no. 2, pp. 44–52, 2019.

[16] F. Mogyorósi, P. Revisnyei, A. Pašić, Z. Papp, I. Törös, P. Varga, and A. Pašić, "Positioning in 5G and 6G Networks-A Survey," *Sensors*, vol. 22, no. 13, 2022.

[17] A. Mekrache, A. Bradai, E. Moulay, and S. Dawaliby, "Deep reinforcement learning techniques for vehicular networks: Recent advances and future trends towards 6G," *Vehicular Communications*, vol. 33, p. 100398, 2022.

[18] Y. Shi, L. Lian, Y. Shi, Z. Wang, Y. Zhou, L. Fu, L. Bai, J. Zhang, and W. Zhang, "Machine Learning for Large-Scale Optimization in 6G Wireless Networks," *IEEE Commun. Surv. Tutor.*, vol. 25, no. 4, pp. 2088–2132, 2023.

[19] S. J. Nawaz, S. K. Sharma, S. Wyne, M. N. Patwary, and M. Asaduzzaman, "Quantum Machine Learning for 6G Communication Networks: State-of-the-Art and Vision for the Future," *IEEE Access*, vol. 7, pp. 46317–46350, 2019.

[20] F. Tang, B. Mao, Y. Kawamoto, and N. Kato, "Survey on Machine Learning for Intelligent End-to-End Communication Toward 6G: From Network Access, Routing to Traffic Control and Streaming Adaption," *IEEE Commun. Surv. Tutor.*, vol. 23, no. 3, pp. 1578–1598, 2021.

[21] M. M. Azari, S. Solanki, S. Chatzinotas, O. Kodheli, H. Sallouha, A. Colpaert, J. F. Mendoza Montoya, S. Pollin, A. Haqiqatnejad, A. Mostaani, E. Lagunas, and B. Ottersten, "Evolution of Non-Terrestrial Networks From 5G to 6G: A Survey," *IEEE Commun. Surv. Tutor.*, vol. 24, no. 4, pp. 2633–2672, 2022.

[22] O. Nassef, W. Sun, H. Purmehdi, M. Tatipamula, and T. Mahmoodi, "A survey: Distributed Machine Learning for 5G and beyond," *Computer Networks*, vol. 207, p. 108820, 2022.

[23] J. Tanveer, A. Haider, R. Ali, and A. Kim, "An Overview of Reinforcement Learning Algorithms for Handover Management in 5G Ultra-Dense Small Cell Networks," *Applied Sciences*, vol. 12, no. 1, 2022.

[24] 3GPP, "TS 36.331: Radio Resource Control (RRC); Protocol specification (Release 15)." `https://www.etsi.org/deliver/etsi_ts/136300_136399/136331/15.03.00_60/ts_136331v150300p.pdf`, 2018. (Accessed on 20-12-2020).

[25] P. E. Iturria-Rivera and M. Erol-Kantarci, "QoS-Aware Load Balancing in Wireless Networks using Clipped Double Q-Learning," in *2021 IEEE 18th International Conference on Mobile Ad-Hoc and Smart Systems, MASS 2021*, pp. 10–16, 2021.

[26] P. E. Iturria-Rivera and M. Erol-Kantarci, "Competitive Multi-Agent Load Balancing with Adaptive Policies in Wireless Networks," in *IEEE 19th Annual Consumer Communications Networking Conference (CCNC)*, pp. 796–801, 2022.

[27] M. Polese, M. Giordani, M. Mezzavilla, S. Rangan, and M. Zorzi, "Improved Handover Through Dual Connectivity in 5G mmWave Mobile Networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 9, pp. 2069–2084, 2017.

[28] P. E. Iturria-Rivera, M. Elsayed, M. Bavand, R. Gaigalas, S. Furr, and M. Erol-Kantarci, "Hierarchical Deep Q-Learning Based Handover in Wireless Networks with Dual Connectivity," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 6553–6558, 2022.

[29] P. E. Iturria-Rivera, H. Zhang, H. Zhou, S. Mollahasani, and M. Erol-Kantarci, "Multi-Agent Team Learning in Virtualized Open Radio Access Networks (O-RAN)," *Sensors*, vol. 22, no. 14, p. 5375, 2022.

[30] 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. Surv. Tutor.*, vol. 24, no. 3, pp. 1843–1893, 2022.

[31] D. Atzeni, D. Bacciu, D. Mazzei, and G. Prencipe, "A Systematic Review of Wi-Fi and Machine Learning Integration with Topic Modeling Techniques," *Sensors*, vol. 22, no. 13, 2022.

[32] E. Mozaffariahrar, F. Theoleyre, and M. Menth, "A Survey of Wi-Fi 6: Technologies, Advances, and Challenges," *Future Internet*, vol. 14, no. 10, 2022.

[33] N. Singh, S. Choe, and R. Punmiya, "Machine Learning Based Indoor Localization Using Wi-Fi RSSI Fingerprints: An Overview," *IEEE Access*, vol. 9, pp. 127150–127174, 2021.

[34] V. Bellavista-Parent, J. Torres-Sospedra, and A. Pérez-Navarro, "Comprehensive Analysis of Applied Machine Learning in Indoor Positioning Based on Wi-Fi: An Extended Systematic Review," *Sensors*, vol. 22, no. 12, 2022.

[35] A. Nessa, B. Adhikari, F. Hussain, and X. N. Fernando, "A Survey of Machine Learning for Indoor Positioning," *IEEE Access*, vol. 8, pp. 214945–214965, 2020.

[36] J. Rojo, G. M. Mendoza-Silva, G. Ristow Cidral, J. Laiapea, G. Parrello, A. Simó, L. Stupin, D. Minican, M. Farrés, C. Corvalán, F. Unger, S. M. López, I. Soteras, D. C. Bravo, and J. Torres-Sospedra, "Machine Learning applied to Wi-Fi fingerprinting: The experiences of the Ubiqum Challenge," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8, 2019.

[37] M. Morshedi and J. Noll, "A Survey on Prediction of PQoS Using Machine Learning on Wi-Fi Networks," in *International Conference on Advanced Technologies for Communications (ATC)*, pp. 5–11, 2020.

[38] M. Ali, P. Hendriks, N. Popping, S. Levi, and A. Naveed, "A Comparison of Machine Learning Algorithms for Wi-Fi Sensing Using CSI Data," *Electronics*, vol. 12, no. 18, 2023.

[39] H. F. Thariq Ahmed, H. Ahmad, and A. C.V., "Device free human gesture recognition using Wi-Fi CSI: A survey," *Engineering Applications of Artificial Intelligence*, vol. 87, p. 103281, 2020.

[40] S. Yousefi, H. Narui, S. Dayal, S. Ermon, and S. Valaee, "A Survey on Behavior Recognition Using WiFi Channel State Information," *IEEE Commun. Mag.*, vol. 55, no. 10, pp. 98–104, 2017.

[41] A. E. Aminanto and M. E. Aminanto, "Deep Learning Models for Intrusion Detection in Wi-Fi Networks: A Literature Survey," *Sustainable Architecture and Building Environment: Proceedings of ICSDEMS 2020*, pp. 115–121, 2022.

[42] P. E. Iturria-Rivera, M. Chenier, B. Herscovici, B. Kantarci, and M. Erol-Kantarci, " Cooperate or Not Cooperate: Transfer Learning With Multi-Armed Bandit for Spatial Reuse in Wi-Fi ," *IEEE Trans. Mach. Learn. Commun. Net.*, vol. 2, pp. 351–369, 2024.

[43] P. E. Iturria-Rivera, M. Chenier, B. Herscovici, B. Kantarci, and M. Erol-Kantarci, "Meta-Bandit: Spatial Reuse Adaptation via Meta-Learning in Distributed Wi-Fi 802.11ax," *IEEE Networking Letters*, vol. 5, no. 4, pp. 179–183, 2023.

[44] P. E. Iturria-Rivera, M. Chenier, B. Herscovici, B. Kantarci, and M. Erol-Kantarci, "RL meets Multi-Link Operation in IEEE 802.11be: Multi-Headed Recurrent Soft-Actor Critic-based Traffic Allocation," in *Proceedings - IEEE Int. Conf. on Comput. Commun. Net.*, pp. 4001–4006, 2023.

[45] P. E. Iturria-Rivera, M. Chenier, B. Herscovici, B. Kantarci, and M. Erol-Kantarci, "Channel Selection for Wi-Fi 7 Multi-Link Operation via Optimistic-Weighted VDN and Parallel Transfer Reinforcement Learning," in *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–6, 2023.

[46] P. E. Iturria-Rivera and M. Erol-Kantarci, "Competitive Multi-Agent Load Balancing with Adaptive Policies in Wireless Networks," in *IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 796–801, 2022.

[47] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," in *Advances in Neural Information Processing Systems*, 2017.

[48] D. Bouneffouf, I. Rish, and C. Aggarwal, "Survey on Applications of Multi-Armed and Contextual Bandits," in *IEEE Congr. Evol. Comput. (CEC)*, 2020.

[49] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge University Press, 2020.

[50] A. Slivkins, *Introduction to multi-armed bandits*. Now Publishers, Inc., 2019.

[51] L. Zhou, "A survey on contextual multi-armed bandits," *arXiv preprint arXiv:1508.03326*, 2015.

[52] R. Agrawal, "Sample mean based index policies by O(log n) regret for the multi-armed bandit problem," *Adv. Appl. Probab.*, 1995.

[53] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, "A Tutorial on Thompson Sampling," *Found. Trends Mach. Learn.*, 2018.

[54] V. Mnih, K. Kavukcuoglu, D. Silver, and et al., "Human-level Control through Deep Reinforcement Learning," *Nature*, vol. 518, p. 529–533, Jan. 2015.

[55] H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," in *30th AAAI Conference on Artificial Intelligence*, 2016.

[56] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," in *35th International Conference on Machine Learning, ICML 2018*, 2018.

[57] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous Control with Deep Reinforcement Learning," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[58] T. D. Kulkarni, K. R. Narasimhan, A. Saeedi, and J. B. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[59] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, Programmable, and Virtualized 5G Networks: State-of-the-Art and the Road Ahead," *Computer Networks*, vol. 182, p. 107516, 2020.

[60] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud RAN for Mobile Networks—A Technology Overview," *IEEE Commun. Surv. Tutor.*, vol. 17, no. 1, pp. 405–426, 2015.

[61] A. Garcia-Saavedra and X. Costa-Perez, "O-RAN: Disrupting the Virtualized RAN Ecosystem," *IEEE Commun. Stand. Mag.*, vol. 5, no. 4, pp. 96–103, 2021.

[62] O-RAN Working Group 2, "O-RAN AI/ML Workflow Description and Requirements - v1.03." `https://orandownloadsweb.azurewebsites.net/download?id=158`, October 2021. (Accessed on 2021-11-25).

[63] B. Balasubramanian, E. S. Daniels, M. Hiltunen, R. Jana, K. Joshi, R. Sivaraj, T. X. Tran, and C. Wang, "RIC: A RAN Intelligent Controller Platform for AI-Enabled Cellular Networks," *IEEE Internet Comput.*, 2021.

[64] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and Learning in O-RAN for Data-Driven NextG Cellular Networks," *IEEE Commun. Mag.*, vol. 59, no. 10, pp. 21–27, 2021.

[65] M. Dryjański and A. Kliks, "The O-RAN Whitepaper 2022 RAN Intelligent Controller, xApps and rApps," *White Paper*, Feb 2022.

[66] M. Dryjański, Kułacz, and A. Kliks, "Toward Modular and Flexible Open RAN Implementations in 6G Networks: Traffic Steering Use Case and O-RAN xApps," *Sensors*, vol. 21, no. 24, 2021.

[67] Y. Cao, S. Y. Lien, Y. C. Liang, and K. C. Chen, "Federated Deep Reinforcement Learning for User Access Control in Open Radio Access Networks," in *IEEE International Conference on Communications*, pp. 1–6, 2021.

[68] S. Gronauer and K. Diepold, "Multi-agent Deep Reinforcement Learning: a Survey," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 895–943, 2022.

[69] C. S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. S. Torr, M. Sun, and S. Whiteson, "Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge?," *arXiv preprint arXiv:2011.09533*, 2020.

[70] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward," in *Proceedings of the 17th International Conference on Autonomous Agents*

and MultiAgent Systems, AAMAS '18, (Richland, SC), p. 2085–2087, International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[71] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted QMIX: Expanding Monotonic Value Function Factorisation," in *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, vol. 33, pp. 10199–10210, 2020.

[72] S. Amin, M. Gomrokchi, H. Satija, H. van Hoof, and D. Precup, "A Survey of Exploration Methods in Reinforcement Learning," *arXiv preprint arXiv:2109.00157*, 2021.

[73] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey, "Meta-Learning in Neural Networks: A Survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 09, pp. 5149–5169, 2022.

[74] A. E. Kalør, O. Simeone, and P. Popovski, "Prediction of mmWave/THz Link Blockages Through Meta-Learning and Recurrent Neural Networks," *IEEE Wirel. Commun. Lett.*, vol. 10, no. 12, pp. 2815–2819, 2021.

[75] T. Li, Z. Hong, L. Liu, Z. Wen, and L. Yu, "Meta-WF: Meta-Learning-Based Few-Shot Wireless Impersonation Detection for Wi-Fi Networks," *IEEE Commun. Lett.*, vol. 25, no. 11, pp. 3585–3589, 2021.

[76] R. Zhu and M. Rigotti, "Deep Bandits Show-Off: Simple and Efficient Exploration with Deep Networks," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 17592–17603, 2021.

[77] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, PMLR, 06–11 Aug 2017.

[78] TIP, "OpenWiFi Release 2.4 GA." `https://openwifi.tip.build/`, 2022. (Accessed on 10-5-2022).

[79] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[80] T. Scott, K. Ridgeway, and M. C. Mozer, "Adapted Deep Embeddings: A Synthesis of Methods for k-Shot Inductive Transfer Learning," in *Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[81] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A Comprehensive Survey on Transfer Learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[82] L. Vu, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Deep Transfer Learning for IoT Attack Detection," *IEEE Access*, vol. 8, pp. 107335–107344, 2020.

[83] M. Elsayed, M. Erol-Kantarci, and H. Yanikomeroglu, "Transfer Reinforcement Learning for 5G New Radio mmWave Networks," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 5, pp. 2838–2849, 2021.

[84] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A Survey on Deep Transfer Learning," in *IEEE Int. Conf. Artif. Neural Netw. (ICANN)*, pp. 270–279, 2018.

[85] Z. Zhu, K. Lin, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *arXiv preprint arXiv:2009.07888*, 2020.

[86] M. E. Taylor and P. Stone, "Transfer Learning for Reinforcement Learning Domains: a Survey," *Journal of Machine Learning Research*, vol. 10, no. 7, 2009.

[87] A. Taylor, I. Dusparic, and E. Galván-López, "Transfer Learning in Multi-agent Systems through Parallel Transfer," *Proceedings of the 30th International Conference on Machine Learning*, vol. 28, 2013.

[88] R. Ahmad, E. A. Sundararajan, N. E. Othman, and M. Ismail, "Handover in LTE-advanced Wireless Networks: State of Art and Survey of Decision Algorithm," *Telecommunication Systems*, vol. 66, pp. 533–558, 2017.

[89] M. Tayyab, X. Gelabert, and R. Jantti, "A Survey on Handover Management: From LTE to NR," *IEEE Access*, vol. 7, pp. 118907–118930, 2019.

[90] V. Yajnanarayana, H. Rydén, and L. Hévizi, "5G Handover using Reinforcement Learning," in *2020 IEEE 3rd 5G World Forum (5GWF)*, pp. 349–354, IEEE, 2020.

[91] Z.-H. Huang, Y.-L. Hsu, P.-K. Chang, and M.-J. Tsai, "Efficient Handover Algorithm in 5G Networks using Deep Learning," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, Dec 2020.

[92] Y. Xu, W. Xu, Z. Wang, J. Lin, and S. Cui, "Load Balancing for Ultradense Networks: A Deep Reinforcement Learning-Based Approach," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9399–9412, 2019.

[93] K. Attiah, K. Banawan, A. Gaber, A. Elezabi, K. Seddik, Y. Gadallah, and K. Abdullah, "Load Balancing in Cellular Networks: A Reinforcement Learning Approach," in *2020 IEEE 17th Annual Consumer Communications and Networking Conference, CCNC 2020*, 2020.

[94] H. Choi, T. Kim, H.-s. Park, and J. K. Choia, "A Cooperative Online Learning-Based Load Balancing Scheme for Maximizing QoS Satisfaction in Dense HetNets," *IEEE Access*, vol. 9, pp. 92345–92357, 2021.

[95] T. Mai, H. Yao, Z. Xiong, S. Guo, and D. T. Niyato, "Multi-agent Actor-Critic Reinforcement Learning Based In-network Load Balance," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2020.

[96] G. Alsuhli, H. A. Ismail, K. Alansary, M. Rumman, M. Mohamed, and K. G. Seddik, "Deep Reinforcement Learning-based CIO and Energy Control for LTE Mobility Load Balancing," in *IEEE 18th Annual Consumer Communications and Networking Conference (CCNC)*, pp. 1–6, 2021.

[97] O. N. C. Yilmaz, O. Teyeb, and A. Orsino, "Overview of LTE-NR Dual Connectivity," *IEEE Commun. Mag.*, vol. 57, no. 6, pp. 138–144, 2019.

[98] J. F. Monserrat, F. Bouchmal, D. Martin-Sacristan, and O. Carrasco, "Multi-Radio Dual Connectivity for 5G Small Cells Interworking," *IEEE Commun. Stand. Mag.*, vol. 4, pp. 30–36, 2020.

[99] 3GPP TS 37.340, "Universal Mobile Telecommunications System (UMTS), LTE, 5G, NR, Multi-connectivity, Overall description, Stage-2," tech. rep., 2019.

[100] M. Agiwal, H. Kwon, S. Park, and H. Jin, "A Survey on 4G-5G Dual Connectivity: Road to 5G Implementation," *IEEE Access*, vol. 9, pp. 16193–16210, 2021.

[101] Y. Geng, E. Liu, R. Wang, and Y. Liu, "Hierarchical Reinforcement Learning for Relay Selection and Power Optimization in Two-Hop Cooperative Relay Network," *IEEE Trans. Commun.*, vol. 70, pp. 171–184, 2021.

[102] S. Liu, J. Wu, and J. He, "Dynamic multichannel sensing in cognitive radio: Hierarchical reinforcement learning," *IEEE Access*, vol. 9, pp. 25473–25481, 2021.

[103] H. Peng and X. Shen, "Deep Reinforcement Learning Based Resource Management for Multi-Access Edge Computing in Vehicular Networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, pp. 2416–2428, 2020.

[104] Z. Pan, Z. Qu, Y. Chen, H. Li, and X. Wang, "A Distributed Assignment Method for Dynamic Traffic Assignment Using Heterogeneous-Adviser Based Multi-Agent Reinforcement Learning," *IEEE Access*, vol. 8, pp. 154237–154255, 2020.

[105] H. Yang, J. Yuan, C. Li, G. Zhao, Z. Sun, Q. Yao, B. Bao, A. V. Vasilakos, and J. Zhang, "BrainIoT: Brain-Like Productive Services Provisioning With Federated Learning in Industrial IoT," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 2014–2024, 2022.

[106] A. Bardou, T. Begin, and A. Busson, "Improving the Spatial Reuse in IEEE 802.11ax WLANs," *Proc. 24th Int. ACM Conf. Model. Anal. Simul. Wireless Mobile Syst.*, pp. 135–144, 2021.

[107] A. Bardou and T. Begin, "INSPIRE: Distributed Bayesian Optimization for Improv-iNg SPatIal REuse in Dense WLANs," in *Proc. 25th Int. ACM Conf. Model. Anal. Simul. Wireless Mobile Syst.*, pp. 133–142, 2022.

[108] F. Wilhelmi, C. Cano, G. Neu, B. Bellalta, A. Jonsson, and S. Barrachina-Muñoz, "Collaborative Spatial Reuse in Wireless Networks via Selfish Multi-Armed Bandits," *Ad Hoc Networks*, vol. 88, pp. 129–141, 2019.

[109] F. Wilhelmi, J. Hribar, S. F. Yilmaz, E. Ozfatura, K. Ozfatura, O. Yildiz, D. Gündüz, H. Chen, X. Ye, L. You, *et al.*, "Federated Spatial Reuse Optimization in Next-Generation Decentralized IEEE 802.11 WLANs," *arXiv preprint arXiv:2203.10472*, 2022.

[110] E. S. Olivas, J. D. M. Guerrero, M. Martinez Sober, J. R. Magdalena Benedito, and A. J. Serrano López, *Handbook of research on machine learning applications and trends: Algorithms, methods, and techniques*. IGI global, 2009.

[111] H. Lee, H.-S. Kim, and S. Bahk, "LSR: Link-aware Spatial Reuse in IEEE 802.11ax WLANs," in *IEEE Wireless Commun. Netw. Conf. (WCNC)*, pp. 1–6, 2021.

[112] H. Kim and J. So, "Improving Spatial Reuse of Wireless LAN Uplink Using BSS Color and Proximity Information," *Appl. Sci.*, vol. 11, no. 22, 2021.

[113] G. Lacalle, I. Val, O. Seijo, M. Mendicute, D. Cavalcanti, and J. Perez-Ramirez, "Analysis of Latency and Reliability Improvement with Multi-Link Operation Over 802.11," in *IEEE International Conference on Industrial Informatics (INDIN)*, 2021.

[114] M. Carrascosa, G. Geraci, E. Knightly, and B. Bellalta, "An Experimental Study of Latency for IEEE 802.11be Multi-link Operation," in *IEEE International Conference on Communications (ICC)*, pp. 2507–2512, 2022.

[115] A. Lopez-Raventos and B. Bellalta, "IEEE 802.11be Multi-Link Operation: When the best could be to use only a single interface," in *19th Mediterranean Communication and Computer Networking Conference (MedComNet)*, pp. 1–6, 2021.

[116] A. Lopez-Raventos and B. Bellalta, "Dynamic Traffic Allocation in IEEE 802.11be Multi-Link WLANs," *IEEE Wirel. Commun. Lett.*, vol. 11, no. 7, pp. 1404–1408, 2022.

[117] T. Yang, Z. Jiang, R. Sun, N. Cheng, and H. Feng, "Maritime Search and Rescue Based on Group Mobile Computing for Unmanned Aerial Vehicles and Unmanned Surface Vehicles," *IEEE Trans. Ind. Inform.*, vol. 16, pp. 7700–7708, dec 2020.

[118] M. Carrascosa-Zamacois, G. Geraci, L. Galati-Giordano, A. Jonsson, and B. Bellalta, "Understanding Multi-link Operation in Wi-Fi 7: Performance, Anomalies, and Solutions," in *IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–6, IEEE, 2023.

[119] A. Taylor, I. Dusparic, M. Gueriau, and S. Clarke, "Parallel Transfer Learning in Multi-Agent Systems: What, when and how to transfer?," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 1–8, 2019.

[120] J. Wang, W. Yang, P. Du, and T. Niu, "A Novel Hybrid Forecasting System of Wind Speed based on a Newly Developed Multi-objective Sine Cosine Algorithm," *Energy Conversion and Management*, vol. 163, pp. 134–150, May 2018.

[121] Y. Kishimoto, X. Wang, and M. Umehira, "Reinforcement Learning based Joint Channel/Subframe Selection Scheme for Fair LTE-WiFi Coexistence," in *16th International Conference on Mobility, Sensing and Networking (MSN)*, pp. 367–372, 2020.

[122] 3GPP, "Study on XR (Extended Reality) Evaluations for NR." `https://www.3gpp.org/ftp/Specs/archive/38_series/38.838/38838-h00.zip`, Dec. 2021. Version 17.0.0.

[123] S. Lagen, B. Bojovic, K. Koutlia, X. Zhang, P. Wang, and Q. Qu, "QoS Management for XR Traffic in 5G NR: A Multi-Layer System View End-to-End Evaluation," *IEEE Commun. Mag.*, vol. 61, no. 12, pp. 192–198, 2023.

[124] B. Bojovic, S. Lagen, K. Koutlia, X. Zhang, P. Wang, and L. Yu, "Enhancing 5G QoS Management for XR Traffic Through XR Loopback Mechanism," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 6, pp. 1772–1786, 2023.

[125] S. Sengupta, N. Ganguly, S. Chakraborty, and P. De, "HotDASH: Hotspot Aware Adaptive Video Streaming Using Deep Reinforcement Learning," in *Proceedings - International Conference on Network Protocols, ICNP*, pp. 165–175, 2018.

[126] K. Tang, N. Kan, J. Zou, C. Li, X. Fu, M. Hong, and H. Xiong, "Multi-User Adaptive Video Delivery over Wireless Networks: A Physical Layer Resource-Aware Deep Reinforcement Learning Approach," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 2, pp. 798–815, 2021.

[127] B. O. Turkkan, T. Dai, A. Raman, T. Kosar, C. Chen, M. F. Bulut, J. Zola, and D. Sow, "GreenABR: Energy-Aware Adaptive Bitrate Streaming with Deep Reinforcement Learning," in *Proceedings of the 13th ACM Multimedia Systems Conference*, MMSys '22, (New York, NY, USA), pp. 150—163, Association for Computing Machinery, 2022.

[128] A. Lutu, D. Perino, M. Bagnulo, E. Frias-Martinez, and J. Khangosstar, "A Characterization of the COVID-19 Pandemic Impact on a Mobile Network Operator Traffic," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, pp. 19–33, 2020.

[129] B. Bojovic and N. Baldo, "A New Channel and QoS Aware Scheduler to Enhance the Capacity of Voice over LTE Systems," in *IEEE 11th International Multi-Conference on Systems, Signals and Devices (SSD)*, pp. 1–6, 2014.

[130] N. Baldo, M. Miozzo, M. Requena-Esteso, and J. Nin-Guerrero, "An Open Source Product-oriented LTE Network Simulator based on ns-3," in *MSWiM'11 - Proceedings of the 14th ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, pp. 293–298, 2011.

[131] P. Gawłowicz and A. Zubow, "Ns-3 meets OpenAI Gym: The playground for machine learning in networking research," in *MSWiM 2019 - Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 113–120, 2019.

[132] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A Survey and Critique of Multiagent Deep Reinforcement Learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, 2019.

[133] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell, "Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient," in *33rd Conference on Artificial Intelligence (AAAI)*, vol. 33, pp. 4213–4220, 2019.

[134] Y. Wang and F. Wu, "Policy Adaptive Multi-Agent Deep Deterministic Policy Gradient," in *Proceedings of the 23rd International Conference on Principles and Practice of Multi-Agent Systems (PRIMA)*, pp. 165–181, Nov. 2020.

[135] Z. Ye, Y. Chen, G. Song, B. Yang, and S. Fan, "Experience Augmentation: Boosting and Accelerating Off-Policy Multi-Agent Reinforcement Learning," *arXiv preprint arXiv:2005.09453*, 2020.

[136] D. Zha, K.-H. Lai, K. Zhou, and X. Hu, "Simplifying Deep Reinforcement Learning via Self-Supervision," *arXiv preprint arXiv:2106.05526*, 2021.

[137] 3GPP TR 36.842, "Study on small cell enhancements for E-UTRA and E-UTRAN, v12.0.0 ," tech. rep., 2013.

[138] R. M. Corless, G. H. Gonnet, D. E. Hare, D. J. Jeffrey, and D. E. Knuth, "On the Lambert W function," *Adv. Comput. Math.*, no. 5, p. 329–359, 1996.

[139] M. Mezzavilla, M. Zhang, M. Polese, R. Ford, S. Dutta, S. Rangan, and M. Zorzi, "End-to-End Simulation of 5G mmWave Networks," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2237–2263, 2018.

[140] H. Yin, P. Liu, K. Liu, L. Cao, L. Zhang, Y. Gao, and X. Hei, "Ns3-Ai: Fostering Artificial Intelligence Algorithms for Networking Research," in *WNS3 2020: Proceedings of the 2020 Workshop on ns-3*, pp. 57–64, 2020.

[141] M. Giordani, M. Mezzavilla, and M. Zorzi, "Initial Access in 5G mmWave Cellular Networks," *IEEE Commun. Mag.*, vol. 54, no. 11, pp. 40–47, 2016.

[142] P. Trakadas, L. Sarakis, A. Giannopoulos, S. Spantideas, N. Capsalis, P. Gkonis, P. Karkazis, G. Rigazzi, A. Antonopoulos, M. A. Cambeiro, S. Gonzalez-Diaz, and L. Conceição, "A Cost-Efficient 5G Non-Public Network Architectural Approach: Key Concepts and Enablers, Building Blocks and Potential Use Cases," *Sensors*, vol. 21, no. 16, 2021.

[143] P. Yang, L. Kong, and G. Chen, "Spectrum Sharing for 5G/6G URLLC: Research Frontiers and Standards," *IEEE Commun. Mag.*, vol. 5, Jun 2021.

[144] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo, "Thirty Years of Machine Learning: The Road to Pareto-Optimal Wireless Networks," *IEEE Commun. Surv. Tutor.*, vol. 22, no. 3, pp. 1472–1514, 2020.

[145] C.-L. I., S. Kuklinski, T. Chen, and L. L. Ladid, "A perspective of O-RAN integration with MEC, SON, and network slicing in the 5G era," *IEEE Netw.*, vol. 34, no. 6, pp. 3–4, 2020.

[146] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-Agent Systems: A Survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.

[147] O-RAN Working Group 1, "O-RAN Architecture Description 6.00," O-RAN.WG1.O-RAN-Architecture-Description-v06.00," tech. rep., March 2022.

[148] M. Polese, L. Bonati, S. D'oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *IEEE Commun. Surv. Tutor.*, vol. 25, no. 2, pp. 1376–1411, 2023.

[149] A. Giannopoulos, S. Spantideas, N. Kapsalis, P. Gkonis, L. Sarakis, C. Capsalis, M. Vecchio, and P. Trakadas, "Supporting Intelligence in Disaggregated Open Radio Access Networks: Architectural Principles, AI/ML Workflow, and Use Cases," *IEEE Access*, vol. 10, pp. 39580–39595, 2022.

[150] M. Elsayed and M. Erol-Kantarci, "Reinforcement Learning-based Joint Power and Resource Allocation for URLLC in 5G," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2019.

[151] "Playing Atari with Deep Reinforcement Learning, author=Mnih, Volodymyr and Kavukcuoglu, Koray and Silver, David and Graves, Alex and Antonoglou, Ioannis and Wierstra, Daan and Riedmiller, Martin, journal=arXiv preprint arXiv:1312.5602, year=2013,"

[152] B. Yongacoglu, G. Arslan, and S. Yuksel, "Reinforcement Learning for Decentralized Stochastic Control," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 5556–5561, IEEE, 2019.

[153] Cisco Systems Inc., "Cisco Annual Internet Report (2018–2023)." `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`, 2020. (Accessed on 2-5-2021).

[154] Wi-Fi Alliance, "Wi-Fi® by the numbers: Technology momentum in 2023." https://www.wi-fi.org/beacon/the-beacon/wi-fi-by-the-numbers-technology-momentum-in-2023, 2023. (Accessed on 11-08-2023).

[155] IEEE 802.11, "Official IEEE 802.11 Working Group Project Timelines." https://www.ieee802.org/11/Reports/802.11_Timelines.htm. (Accessed on 11-08-2023).

[156] F. Ye, S. Yi, and B. Sikdar, "Improving Spatial Reuse of IEEE 802.11 Based Ad Hoc Networks," in *IEEE Global Telecommunications Conference*, vol. 2, pp. 1013–1017, 2003.

[157] F. Wilhelmi, S. Barrachina-Muñoz, C. Cano, I. Selinis, and B. Bellalta, "Spatial Reuse in IEEE 802.11ax WLANs," *Comput. Commun.*, vol. 170, pp. 65–83, 2021.

[158] C. Thorpe and L. Murphy, "A Survey of Adaptive Carrier Sensing Mechanisms for IEEE 802.11 Wireless Networks," *IEEE Commun. Surv. Tutor.*, vol. 16, no. 3, pp. 1266–1293, 2014.

[159] T. Huehn and C. Sengul, "Practical Power and Rate Control for Wi-Fi," *IEEE Int. Conf. on Comput. Commun. Net.*, pp. 1–7, 2012.

[160] D. Nunez, F. Wilhelmi, S. Avallone, M. Smith, and B. Bellalta, "TXOP sharing with Coordinated Spatial Reuse in Multi-AP Cooperative IEEE 802.11be WLANs," in *Proceedings - IEEE Consumer Commun. and Net. Conf., CCNC*, pp. 864–870, 2022.

[161] C. Khosla and B. S. Saini, "Enhancing Performance of Deep Learning Models with different Data Augmentation Techniques: A Survey," in *Proc. Int. Conf. Ind. Eng. Oper. Manage.*, pp. 79–85, 2020.

[162] P. E. Iturria-Rivera, M. Chenier, B. Herscovici, B. Kantarci, and M. Erol-Kantarci, "Channel Selection for Wi-Fi 7 Multi-Link Operation via Optimistic-Weighted VDN and Parallel Transfer Reinforcement Learning," in *Proceedings - IEEE 34th Annual Int. Symp. on Pers., Indoor and Mob. Radio Commun. (PIMRC)*, pp. 1–6, 2023.

[163] S. Hossain, E. Micha, and N. Shah, "Fair Algorithms for Multi-Agent Multi-Armed Bandits," in *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 24005–24017, 2021.

[164] P. C. Landgren, *Distributed Multi-Agent Multi-Armed Bandits*. PhD thesis, 2019.

[165] G. Lugosi and A. Mehrabian, "Multiplayer Bandits Without Observing Collision Information," *Math. Oper. Res.*, pp. 1247–1265, 2022.

[166] W. Huang, R. Combes, and C. Trinh, "Towards Optimal Algorithms for Multi-Player Bandits without Collision Sensing Information," in *Proceedings of Thirty Fifth Conference on Learning Theory* (P.-L. Loh and M. Raginsky, eds.), vol. 178 of *Proceedings of Machine Learning Research*, pp. 1990–2012, PMLR, 02–05 Jul 2022.

[167] P.-A. WANG, A. Proutiere, K. Ariu, Y. Jedra, and A. Russo, "Optimal algorithms for multiplayer multi-armed bandits," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics* (S. Chiappa and R. Calandra, eds.), vol. 108 of *Proceedings of Machine Learning Research*, pp. 4120–4129, PMLR, 26–28 Aug 2020.

[168] Y. Gai, B. Krishnamachari, and R. Jain, "Learning multiuser channel allocations in cognitive radio networks: a combinatorial multi-armed bandit formulation," in *2010 IEEE Symposium on New Frontiers in Dynamic Spectrum, DySPAN 2010*, pp. 1–9, 2010.

[169] B. Kveton, C. Szepesvári, S. Vaswani, Z. Wen, M. Ghavamzadeh, and T. Lattimore, "Garbage in, reward out: Bootstrapping exploration in multi-armed bandits," in *36th International Conference on Machine Learning, ICML 2019*, pp. 3601–3610, 2019.

[170] S. Agrawal and N. Goyal, "Further optimal regret bounds for thompson sampling," in *Journal of Machine Learning Research*, 2013.

[171] P. Landgren, V. Srivastava, and N. E. Leonard, "Distributed Cooperative Decision Making in Multi-Agent Multi-Armed Bandits," *Automatica*, p. 109445, 2021.

[172] Y. Xiao, "IEEE 802.11e: QoS Provisioning at the MAC Layer," *IEEE Wirel. Commun.*, vol. 11, no. 3, pp. 72–79, 2004.

[173] M. Derakhshani, X. Wang, D. Tweed, T. Le-Ngoc, and A. Leon-Garcia, "AP-STA Association Control for Throughput Maximization in Virtualized Wi-Fi Networks," *IEEE Access*, pp. 45034–45050, 2018.

[174] G. Holland, N. Vaidya, and P. Bahl, "A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks," in *Proc. Annu. Int. Conf. Mobil. Comput. Net. (MOBICOM)*, pp. 236–251, 2001.

[175] G. F. Riley and T. R. Henderson, *The ns-3 Network Simulator*, pp. 15–34. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[176] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 3, pp. 535–547, 2000.

[177] 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 *Proc. Annu. Int. Conf. Mobil. Comput. Net. (MOBICOM)*, pp. 366–377, 2006.

[178] S. Padakandla, "A Survey of Reinforcement Learning Algorithms for Dynamically Varying Environments," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–25, 2021.

[179] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A Review on Outlier/Anomaly Detection in Time Series Data," *ACM Comput. Surv.*, vol. 54, no. 3, pp. 1–33, 2021.

[180] 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," *J. Netw. Comput. Appl.*, vol. 127, pp. 26–42, 2019.

[181] M. Bayati, N. Hamidi, R. Johari, and K. Khosravi, "Unreasonable Effectiveness of Greedy Algorithms in Multi-Armed Bandit with Many Arms," in *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 1713–1723, 2020.

[182] C. Riquelme, G. Tucker, and J. Snoek, "Deep Bayesian bandits showdown: An empirical comparison of Bayesian deep networks for Thompson sampling," in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

[183] Wi-Fi Alliance, "Wi-Fi Alliance® 2022 Wi-Fi® trends | Wi-Fi Alliance." `https://www.wi-fi.org/news-events/newsroom/wi-fi-alliance-2022-wi-fi-trends`. (Accessed on 2022-08-25).

[184] Wi–Fi Alliance, "Wi-Fi 6E: Wi-Fi® in the 6 GHz band." `https://www.wi-fi.org/downloads-registered-guest/Wi-Fi_6E_paper_202112.pdf/37285`, 2021. (Accessed on 2022-08-26).

[185] "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 Amendment 1: Enhancements for High-Efficiency WLAN," *IEEE Std 802.11ax-2021 (Amendment to IEEE Std 802.11-2020)*, pp. 1–767, 2021.

[186] R. Stacey, "Specification Framework for TGax." https://mentor.ieee.org/802.11/dcn/15/11-15-0132-16-00ax-spec-framework.docx, 2016. (Accessed on 2022-08-15).

[187] T. V. Phan, S. Sultana, T. G. Nguyen, and T. Bauschert, "Q-TRANSFER: A Novel Framework for Efficient Deep Transfer Learning in Networking," in *International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pp. 146–151, 2020.

[188] J. Choi, "Discussion on Next Generation Wi-Fi." https://mentor.ieee.org/802.11/dcn/22/11-22-0685-00-0wng-discussion-on-next-generation-wi-fi.pptx, May 2022. (Accessed on 2022-10-16).

[189] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *35th International Conference on Machine Learning (ICML)*, pp. 1861–1870, 2018.

[190] S. Zhao, H. Abou-zeid, R. Atawia, Y. S. K. Manjunath, A. B. Sediq, and X.-P. Zhang, "Virtual Reality Gaming on the Cloud: A Reality Check," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2021.

[191] IEEE P802.11, "TGax Simulation Scenarios." https://mentor.ieee.org/802.11/dcn/14/11-14-0980-16-00ax-simulation-scenarios.docx, 2015.

[192] P. Christodoulou, "Soft Actor-Critic for Discrete Action Settings," *arXiv preprint arXiv:1910.07207*, 2019.

[193] H. Zhou, Z. Lin, J. Li, D. Ye, Q. Fu, and W. Yang, "Revisiting Discrete Soft Actor-Critic," *arXiv preprint arXiv:2209.10081*, 2022.

[194] M. Hausknecht and P. Stone, "Deep Recurrent Q-Learning for Partially Observable MDPs," in *AAAI Fall Symposium - Technical Report*, 2015.

[195] X. Li, L. Li, J. Gao, X. He, J. Chen, L. Deng, and J. He, "Recurrent Reinforcement Learning: A Hybrid Approach," *arXiv preprint arXiv:1509.03044*, 2015.

[196] K. Krickeberg and W. Feller, *An Introduction to Probability Theory and Its Applications; Vol. 1.* JSTOR, 1969.

[197] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight Experience Replay," in *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[198] A. Garcia-Rodriguez, D. Lopez-Perez, L. Galati-Giordano, and G. Geraci, "IEEE 802.11be: Wi-Fi 7 Strikes Back," *IEEE Commun. Mag.*, vol. 59, no. 4, pp. 102–108, 2021.

[199] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning," *Journal of Machine Learning Research*, vol. 21, no. 178, pp. 1–51, 2020.

[200] L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong, "Digital-Twin-Enabled 6G: Vision, Architectural Trends, and Future Directions," *IEEE Commun. Mag.*, vol. 60, no. 1, pp. 74–80, 2022.

[201] H. F. Ahmad, W. Rafique, R. U. Rasool, A. Alhumam, Z. Anwar, and J. Qadir, "Leveraging 6G, Extended Reality, and IoT Big Data Analytics for Healthcare: A Review," vol. 48, p. 100558, 2023.

[202] 3GPP, "5G; QoE parameters and metrics relevant to the Virtual Reality (VR) user experience ." `https://www.etsi.org/deliver/etsi_tr/126900_126999/126929/16.01.00_60/tr_126929v160100p.pdf`, Nov. 2020. Version 16.1.0.

[203] 3GPP, "5G; Extended Reality (XR) in 5G." `https://www.etsi.org/deliver/etsi_tr/126900_126999/126928/16.00.00_60/tr_126928v160000p.pdf`, Nov. 2020. Version 16.0.0.

[204] K. Brunnström, S. A. Beker, K. De, A. Dooms, S. Egger, M.-N. Garcia, T. Hossfeld, S. Jumisko-Pyykkö, C. Keimel, and M.-C. Larabi, "Qualinet White Paper on Definitions of Quality of Experience, Output from the fifth Qualinet meeting, Novi Sad," *European Network on Quality of Experience in in Multimedia Systems and Services (COST Action IC 1003)*, 2013.

[205] G. Kougioumtzidis, V. Poulkov, Z. D. Zaharis, and P. I. Lazaridis, "A Survey on Multimedia Services QoE Assessment and Machine Learning-Based Prediction," *IEEE Access*, vol. 10, pp. 19507–19538, 2022.

[206] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[207] P. Sunehag, R. Evans, G. Dulac-Arnold, Y. Zwols, D. Visentin, and B. Coppin, "Deep Reinforcement Learning with Attention for Slate Markov Decision Processes with High-Dimensional States and Actions," *arXiv preprint arXiv:1512.01124*, 2015.

[208] N. Patriciello, S. Lagen, B. Bojovic, and L. Giupponi, "An E2E Simulator for 5G NR Networks," *Simulation Modelling Practice and Theory*, vol. 96, p. 101933, 2019.

[209] M. J. Wainwright, *High-dimensional statistics: A non-asymptotic viewpoint*, vol. 48. Cambridge University Press, 2019.