DEGREE PROJECT IN ELECTRICAL ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2020

Reinforcement Learning for Link Adaptation in 5G-NR Networks

EVAGORAS MAKRIDIS

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Reinforcement Learning for Link Adaptation in 5G-NR Networks

EVAGORAS MAKRIDIS

Master of Science Autonomous Systems
Date: November 3, 2020
Supervisor: Alexandre Proutiere, Euhanna Ghadimi, Soma Tayamon
Examiner: Mikael Johansson
School of Electrical Engineering and Computer Science
Host company: Ericsson AB
Swedish title: Förstärkningslärande för länkanpassning i 5G-NR-nätverk

Abstract

The Adaptive Modulation and Coding (AMC) scheme in the link adaptation is a core feature in the current cellular networks. In particular, based on Channel Quality Indicator (CQI) measurements that are computed from the Signal-to-Interference-plus-Noise Ratio (SINR) level of User Equipment (UE), the base station (e.g., Next Generation NodeB (gNB)) selects a Modulation and Coding Scheme (MCS) to be used for the next downlink transmission. However, communication channels are inherently variant due to changes in traffic load, user mobility, and transmission delays and thus the estimation of the SINR levels at the transmitter side usually deviates from the actual value. The Outer-Loop Link Adaptation (OLLA) technique was proposed to improve the channel quality estimation by adjusting the value of SINR by an offset dependent on whether previous transmissions were decoded successfully or not captured by Hybrid Automatic Repeat Request (HARQ) feedback. Although this technique indeed improves the user throughput, it typically takes several Transmission Time Intervals (TTIs) to converge to a certain SINR value that fulfills a predefined target Block Error Rate (BLER). As a result, the slow convergence speed of the OLLA mechanism causes inaccurate MCS selection specially for users with bursty traffic, while it needs to be *a priori* tuned with a fixed BLER target. These factors lead to degraded network performance, in terms of throughput and spectral efficiency. To cope with these challenges, in this project we propose a reinforcement learning (RL) framework where an agent takes observations from the environment (e.g., from UEs and the network) and learns proper policies that adjust the estimated SINR, such that a reward function (*i.e.,* the UE normalized throughput) is maximized. This framework was designed and developed in a radio network system-level simulator, while for the agents using RL (hereafter called RL agents), Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) models were trained accordingly. Both models showed significant increment of about 1.6% - 2.5% and 10% - 17% on the average throughput for mid-cell and cell-edge users respectively, over the current state-of-the-art OLLA mechanism. Finally, setting *a priori* a fixed BLER target is not needed, and hence the RL-based link adaptation performs well in diverse radio conditions.

Sammanfattning

Adaptive Modulation and Coding (AMC)-schemat i länkanpassning är en central funktion i nutida mobilnätverk. Baserat på Channel Quality Indicator (CQI)-mätningar som är beräknade från Signal-till-Störning-plus- Brusförhållande (SINR)-nivån av User Equipment (UE), väljer basstationen (t.ex., Next Generation NodeB (gNB)) ett Modulerings och kodningsschema (MKS) som används till nästa nedlänksöverföring. Kommunikationskanaler uppvisar dock variationer av sig själva på grund av förändringar i trafikbelastning, användarmobilitet, och överföringsfördröjningar. Detta gör att uppskattningen av SINR-nivåer i sändarsidan avviker från det faktiska värdet. Outer-Loop Link Adaptation (OLLA)-metoden föreslogs för att förbättra uppskattningen av kanalkvaliteten genom att justera värdet på SINR med en förskjutning beroende på om tidigare sändningar avkodades framgångsrikt eller alternativt om de inte fångades av återkoppling från Hybrid Automatic Repeat Request (HARQ). Även om denna teknik förbättrar användares genomströmning, tar det vanligtvis flera sändningstidsintervall (TTI) för att konvergera till ett visst SINR-värde som uppfyller en fördefinierad målfelsfrekvens (BLER). Som ett resultat orsakar OLLA-mekanismens långsamma konvergenshastighet ett felaktigt MCS-val, speciellt för användare med tuff trafik. OLLA-mekanismen måste även anpassas efter ett fast BLER-mål. Dessa faktorer leder till försämrad nätverksprestanda när det gäller genomströmning och spektral effektivitet. För att klara av dessa utmaningar föreslår vi i detta projekt en förstärkningsinlärningsram (RL) där en agent tar observationer från miljön (t.ex. från UE:er och nätverket) och lär sig riktiga policies som justerar den uppskattade SINR:en, så att en belöningsfunktion (dvs. UE-normaliserad genomströmning) maximeras. Denna ram utformades och utvecklades i en radiosimulator på systemnivå. För de agenter som använde RL (hädanefter RL-agenter) utbildades Deep Q-Network (DQN) och Proximal Policy Optimization (PPO)-modeller på lämpligt sätt. Båda modellerna visade en signifikant ökning på cirka 1,6% - 2,5% och 10% - 17% av den genomsnittliga genomströmningen för mellancellsanvändare respektive cellkantsanvändare, jämfört med den nuvarande toppmoderna OLLA mekanismen. Slutligen är det inte nödvändigt att apriori sätta ett fast BLER-mål, och därför fungerar den RL-baserade länkanpassningen bra under olika radioförhållanden.

Acknowledgements

Firstly, I would like to thank Dr. Euhanna Ghadimi who served as my internship primary advisor and RAN Data Scientist in Ericsson AB, for the valuable support, and his guidance he provided throughout the entire period of my internship. Together with Dr. Soma Tayamon and Dr. Pablo Soldati that I would also like to thank, they provided me valuable feedback, and they were updated regarding the progress of my project. It is worth mentioning that this work would not have been possible and exciting without their presence and support.

I would also like to thank Mr. Ulf Norholm, line manager of the BB Analytics Section of Ericsson AB, for his support throughout my internship, and for his willing to help not only regarding the provision of hardware and software licensing, but also for his management skills under the whole period of my internship and especially during the COVID-19 outbreak and the adaptation to remote working it implied.

In addition, I would like to thank Mr. Christian Skarby, Mr. Panagiotis Fotiadis and Mr. Yu Wang for their technical support they provided me when needed. I thank also all the members of the BB Analitics Section for their welcoming in the team and for ideas they gave me during short discussions we had.

I would like to express my greatest gratitude to my academic supervisor Professor Alexandre Proutiere for his feedback he provided me and the answers he gave to all of questions during my master thesis.

Many thanks to my examiner Professor Mikael Johansson for his interest in the master thesis project and for his time he spent to provide me his valuable feedback and suggestions to improve the quality of this work.

Finally, I would like to thank my former academic advisor at Aalto University, Professor Themistoklis Charalambous for his continuous support during my studies, and for his willingness to provide me feedback on this thesis report.

Stockholm, September 2020
Evagoras Makridis

Contents

List of Figures

List of Tables

Acronyms

3GPP	Third Generation Partnership Project
5G-NR	5^{th} Generation New Radio
AI	Artificial Intelligence
AMC	Adaptive Modulation and Coding
BLER	Block Error Rate
CDMA	Code Division Multiple Access
CN	Core Network
CQI	Channel Quality Indicator
CRC	Cyclic Redundancy Check
DDPG	Deep Deterministic Policy Gradient
DNN	Deep Neural Network
DP	Dynamic Programming
DQN	Deep Q-Network
eMBB	Enhanced Mobile Broadband
gNB	Next Generation NodeB
H2H	Human-to-Human
HARQ	Hybrid Automatic Repeat Request
ILLA	Inner-Loop Link Adaptation
LA	Link Adaptation
LTE	Long-Term Evolution
M2M	Machine-to-Machine
MAB	Multi-Armed Bandits
MCS	Modulation and Coding Scheme
MDP	Markov Decision Process
mMTC	Massive Machine-Type Communication
ng-eNB	Next Generation E-UTRAN NodeB
OFDM	Orthogonal Frequency Division Multiplexing
OLLA	Outer-Loop Link Adaptation
PPO	Proximal Policy Optimization
PRB	Physical Resource Block
QoS	Quality of Service

RAN	Radio-Access Network
RL	Reinforcement Learning
SINR	Signal-to-Interference-plus-Noise Ratio
TBS	Transport Block Size
TTIs	Transmission Time Intervals
UE	User Equipment
uMAB	Unimodal Multi-Armed Bandits
URLLC	Ultra-Reliable and Low-Latency Communication
WCDMA	Wideband Code Division Multiple Access

Chapter 1

Introduction

The idea of communication between people is not a recent need. Centuries ago, people managed to find ways to communicate such as morse signals, fires, etc. Since then, the communication systems evolved and have proved to enable core technologies that help the communication between people as well as between devices and systems. The evolution from 1G to 5G changed the way we, and our devices communicate by enabling services such as telephony, short messages, browsing in the internet, and smart interconnected systems. As a result, during the last few years, the need for more reliable and faster communications has been raised. Approaching the next technological revolution, notions like smart cities, autonomous self-driving cars, autonomous unmanned aerial vehicles (UAVs), and industrial automation, started to sound more realistic. This is not only due to the fact that fields like Artificial Intelligence (AI), algorithmics, control theory and many others are getting the interest of researchers in academia and industry, but also because of core technologies such as the 5^{th} Generation New Radio (5G-NR) mobile communication.

While the number of users and devices is growing, the whole network should be ready to cover the demand, and perform resource management techniques in order to meet certain predefined Quality of Service (QoS) requirements. However, due to the increasing number of heterogeneous user equipment (UE) in the network, the dynamics of such systems are complex and difficult to be modelled, and thus the techniques that can be designed and developed are limited. From the other side, the fact of having huge amounts of data directly available in the Radio-Access Network (RAN) enables the study and design of data-driven approaches such as reinforcement learning. As a result, learning-based approaches for radio resource management are gaining the interest of researchers and practitioners in order to provide more effective solutions in the whole radio network.

1.1 Thesis Statement

This thesis focuses on the problem of downlink link adaptation in 5G-NR network using reinforcement learning. In particular, the purpose of this work is to study different approaches that could potentially find more flexible and effective solutions without the need of predefined fixed mapping for the link adaptation problem in 5G-NR. The need for effective and flexible solutions, comes from the several challenges that are implied as a result of limited resources and highly-varying radio conditions due to increased demand in Human-to-Human (H2H), and Machine-to-Machine (M2M) communications [1]. Consequently, current link adaptation techniques cannot cope with these challenges that 5G-NR services bring. For this reason, in this work we design, develop and evaluate a Reinforcement Learning (RL) framework to cope with the challenges of the link adaptation use case. These challenges introduce more and more difficulties to the modelling procedures of the radio network behavior. Although current state-of-the-art algorithms can improve the performance in terms of throughput, they introduce more constraints on the mechanism design since they need to set predefined fixed mappings that correspond to certain theoretical bounds for certain channel conditions that are usually time-varying and driven with noise. Hence, a data-driven approach is needed to generate generic policies that will solve the link adaptation problem for different channel conditions and thus improve the average throughput of the radio network. This framework is developed in a radio network system-level simulator provided by Ericsson, to generate data and train RL models by interfacing between the simulator and open-source RL agents provided by RLlib [2] package developed in Python language. It is important to mention that this work is extremely important since the system-level simulator represents a detailed implementation of a real radio network with all constraints and difficulties that are involved during the development phase of the work. To this end, this work provides useful insights for the development of new data-driven algorithms for 5G-NR usecases which triggers new developments in the field with the use of the last generation of communication systems, the 5G-NR. Thus, the goal of this work is not only to show potential RL methods that could solve the problem of link adaptation, but also to provide details for a realistic scenario accompanied by the challenges expected during actual implementations.

1.2 Organization

Chapter 2 gives a brief overview of the 5G-NR mobile communications. The key technologies and landmark features of different generation mobile networks are described. This chapter also introduces the basic concepts of the transmission structure and functionalities of radio resource management that are used throughout this thesis. Finally it introduces the link adaptation problem. The basic principles, problem formulation, algorithms, related work and existing problems about link adaptation are presented as well.

Chapter 3 begins with defining the Reinforcement Learning problem, its elements and different algorithms and models that were used in this work to solve the problem of link adaptation in 5G-NR cellular networks.

Chapter 4 describes the framework that was designed and developed to solve the problem of link adaptation. In particular, the Markov Decision Process (MDP) design and the experimental setup are described to connect the theory to the actual experiments that were performed.

Chapter 5 presents the results of this work. In particular, the resutls reveal some basic behavior of the radio network used to train the RL models. Finally, the section discusses the results obtained from the current state-of-the-art approaches but also from the RL-based ones and compares them in terms of average user throughput. Finally, the results show significant improvements for RL-based link adaptation techniques over the current state-of-the-art for mid-cell and cell-edge users.

Chapter 6 summarises the work of this thesis, draws the conclusions and also discusses potential future work that would extend this work.

Chapter 2

Link Adaptation in 5G-NR

Fifth generation (5G) mobile communication is already expanding the capabilities of mobile networks, enabling new opportunities for smart interconnected systems with higher data rates [3]. Thus new functionalities are being introduced in several fields such as transportation, smart cities, and other mission critical applications. Hence, increased demands and heterogeneous devices generate more complex and time-varying channel conditions within the radio network. To this end, flexible and generic Link Adaptation (LA) techniques to cope with these challenges are needed.

2.1 Evolution of Mobile Communications

Since 1980, the world has witnessed five major generations of mobile communication (see Figure 2.1), transforming the communications from analog to digital and from voice to high-speed data exchange.

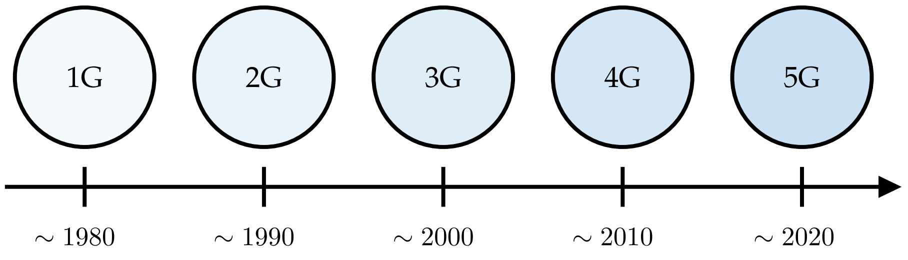

Figure 2.1: Mobile communications evolution timeline

The first generation of mobile communication (1G), appeared around the mid-1980, and it was based on analog transmission. Although the mobile communication systems based on the 1G technology were limited to voice calls, it was the first time for ordinary people to use mobile telephony.

In the early-1990, the second generation of mobile communication (2G), or global system for mobile communication (GSM), was introduced by replacing the analog transmissions to digital ones on the radio link. Although at the beginning the target service of 2G was voice, another non-voice application called short message service (SMS) was introduced in late-1990. In addition to the non-voice applications, the use of digital transmission of the GSM, enabled mobile data services even though the data rate was limited. It is important to mention that, even today the GSM is the core and major technology (and sometimes the only available) in many places in the world.

The third generation of mobile communication (3G), or universal mobile telecommunications system (UMTS), was introduced in the early-2000. This technology enabled the use of high-quality mobile broadband, multimedia message service (MMS) and even video streaming content. Since 3G is based on Wideband Code Division Multiple Access (WCDMA) as the channel access method, it allows several users to share a band of frequencies, which leads to efficient use of the spectrum.

Moving from 3G to the fourth generation of mobile communication (4G), known as Long-Term Evolution (LTE), there were several advances providing higher network efficiency and enhanced mobile-broadband experience in terms of higher user data rates. This technology replaced the transmission from Code Division Multiple Access (CDMA) to Orthogonal Frequency Division Multiplexing (OFDM), enabling wider transmission bandwidths and more advanced multi-antenna technologies. By introducing the 4G, all mobile-network operators converged to the use of a unique global technology for mobile communication, which then led to the transition from 4G to the fifth generation of mobile communication (5G).

2.2 5G-NR Overview

Despite the advancements that 4G-LTE brought to mobile communications, the Third Generation Partnership Project (3GPP) initiated the development of a new generation for mobile communication, the 5G-NR. Although the term 5G-NR is used to refer to the new radio-access technology, the same term is also used to describe a wide range of new services. These services are provided for different applications in different disciplines and sectors such as, cloud applications, autonomous driving, smart cities, industrial automation, and others, while they are classified into three different main use cases:

- *Enhanced Mobile Broadband (eMBB)* which corresponds to an evolution of the current mobile broadband services, by supporting higher data rates for a further enhanced user experience.

- *Massive Machine-Type Communication (mMTC)* which corresponds to a massive number of interconnected devices such as remote sensors, agents, and actuators that require low device cost and low device energy consumption, since high-data rates are not that important for such applications.
- *Ultra-Reliable and Low-Latency Communication (URLLC)* which corresponds to services that require very low latency and extremely high reliability.

2.3 System Architecture

The overall system architecture of 5G-NR, consists of two different networks, the RAN and the Core Network (CN). The RAN enables all radio-related functionality of the network and more specifically the radio access and the radio resource management such as, scheduling, coding, retransmission mechanisms and many others. On the other hand, the CN is responsible for other necessary functions that are not related to the radio access. Such functions include authentication, charging functionality, and setup of end-to-end connections [4]. The focus of this work is on the radio resource management tasks which is included in the RAN functionality, and thus the CN is not being discussed further.

2.3.1 Radio-Access Network (RAN)

A RAN is a major element of mobile communication systems since it provides radio access, and coordination of network resources across UEs. Due to the diversity of 5G-NR services, the RAN must be able to adapt to the requirements of the services by means of channel bandwidths and propagation conditions, and scale appropriately with respect to number of UEs [5]. In general, the RAN has two types of nodes connected to the 5G core network: (a) gNB, which serves NR devices; or Next Generation E-UTRAN NodeB (ng-eNB), which serves LTE devices. These nodes (*i.e.,* gNB, and ng-eNB) are logical nodes responsible for all radio-related functionality in the cells, such as radio resource management, and many others that are not within the context of this work. Note that, a single gNB can handle several cells and that is the reason why it is considered as a logical unit, and not a physical one. Instead, the base station (BS) can be a possible physical implementation of gNB.

2.3.2 Transmission Structure

After several proposals regarding the waveform for transmission, 3GPP agreed to adopt orthogonal frequency division multiplexing (OFDM) with a cyclic-prefix (CP-OFDM) for both downlink and uplink transmissions. It has proven to be suitable for 5G-NR due to its robustness to time dispersion and ease of exploiting both the time and frequency domains when defining the structure for different channels and signals [4]. Regarding the spectrum, 5G-NR supports operations within two different frequency ranges defined in 3GPP Release 15 [6] by:

- Frequency range 1 (FR1): 450 MHz – 6 GHz.
- Frequency range 2 (FR2): 24.25 GHz – 52.6 GHz.

In 5G-NR the physical time and frequency resources correspond to OFDM symbols (time) and subcarriers (frequency) respectively. As shown in Figure 2.2 physical radio resources in a given frame (or subframes) can be considered as a resource grid made up of OFDM subcarriers in the frequency domain, and OFDM symbols in the time domain. The smallest element of this grid is a resource element (RE) which corresponds to a single OFDM subcarrier in frequency and a single OFDM symbol in time. A physical resource block (PRB) consists of 12 OFDM subcarriers. A *radio frame* has a duration of 10ms and it consists of 10 *subframes* having 1ms duration each always. A subframe is formed by one or more adjacent *slots* (depending on the numerology), while each slot has 14 OFDM symbols as shown in Figure 2.3.

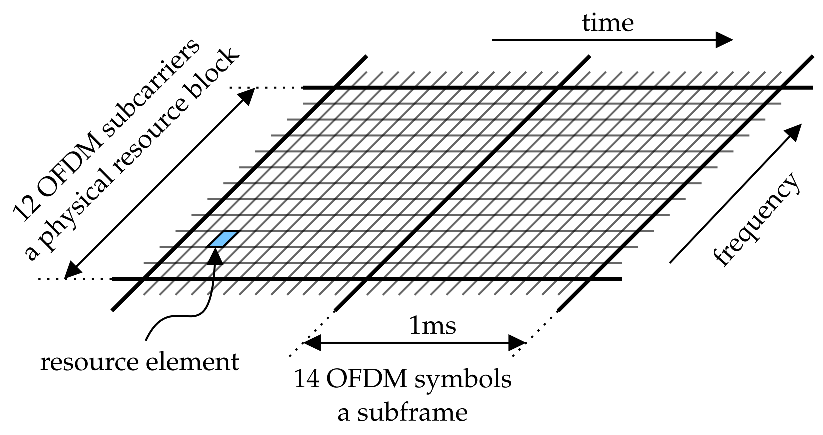

Figure 2.2: Physical resources structure.

One main difference from 4G-LTE, is that 5G-NR supports multiple options for subcarrier spacing (*i.e.*, numerology) and cyclic prefix length. While, in 4G-LTE there is only one available subcarrier spacing (*i.e.*, 15kHz), in 5G-NR the selection of numerology Δf defines the useful symbol length T_u (and hence the slot length) and the cyclic prefix length T_{CP}.

$$T_u = \frac{1}{\Delta f} \tag{2.1}$$

subcarrier spacing Δf	useful symbol length T_u	cyclic prefix length T_{CP}
15 kHz	66.7 μs	4.7 μs
30 kHz	33.3 μs	2.3 μs
60 kHz	16.7 μs	1.2 μs
120 kHz	8.33 μs	0.59 μs
240 kHz	4.17 μs	0.29 μs

Table 2.1: Supported subcarrier spacings by 5G-NR.

The flexible subcarrier spacing selection that 5G-NR supports is beneficial since having a larger subcarrier spacing leads to lower negative impact from frequency errors and phase noise. However, the selection of the subcarrier spacing needs to be carried out in such a way that requirements for different services (*i.e.,* URLLC and eMBB) are met.

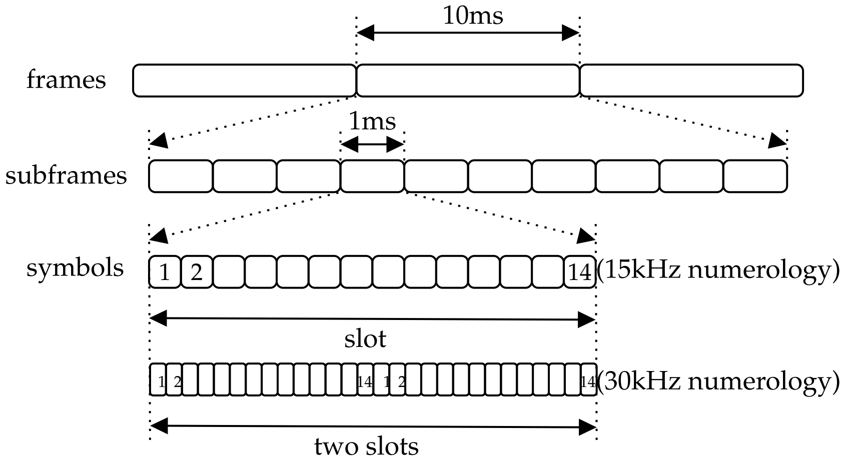

Figure 2.3: Frame structure.

2.4 Downlink Link Adaptation

This section presents the problem of downlink link adaptation in 5G-NR networks. Link adaptation is a fundamental functionality in a channel affected by fading, providing suggestions for the optimal transmitting parameters (*i.e.,* modulation and coding scheme). An overview of the main functionality of the downlink link adaptation mechanism is described in the following sections. However, for more details of the link adaptation mechanism, one can refer to [7]. In 4G-LTE and 5G-NR cellular technologies, link adaptation techniques such as AMC have proved to be core features. The reason is that higher data rates can be achieved and reliably transmitted by automatically adapting the modulation and coding scheme (MCS) [8]. The link adaptation mechanism consists of two different feedback loops called the Inner-Loop Link Adaptation (ILLA) and the OLLA. These loops receive information about the channel quality of the UEs in order to generate the MCS index in the gNB side and send it back to the UEs.

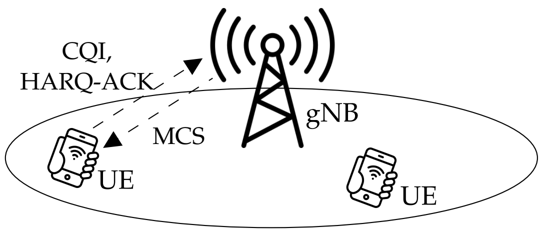

Figure 2.4: Link adaptation paradigm.

In particular, during the downlink AMC process, a user equipment (UE) reports the channel quality indicator (CQI) of the link to the gNB, as shown in Figure 2.4. This CQI is associated with a particular estimated instantaneous signal to interference plus noise ratio (SINR). Subsequently, the gNB receives the CQI index value that is mapped to an estimated instantaneous SINR which corresponds to certain SINR intervals (defined by lower and upper limits).

However, using fixed look-up tables to map the received CQI with the instantaneous SINR is not a good practice due to transmission delays and link conditions that are inherently variant. For this reason, a feedback loop technique called OLLA was proposed to cope with the time-varying link conditions and the transmission delays, by adjusting the instantaneous SINR value by adding or subtracting an offset, using positive or negative acknowledgement signals (*i.e.,* ACK or NACK respectively). The offset is updated continuously based on the Hybrid Automatic Repeat reQuest (HARQ) acknowledgement feedback, such that the average BLock Error Rate ($aBLER$) converges to a predefined target ($BLER_T$). More details for the outer loop link adaptation follow in Section 2.6.

2.4.1 Signal to Interference plus Noise Ratio (SINR)

Signal to interference plus noise ratio (SINR) is defined as the power of a certain signal divided by the sum of the interference power (from all the other interfering signals), and the power of some background noise [9]. The SINR experienced by a UE is represented by γ and is given by:

$$\gamma = \frac{G_0 P_0}{\sum_{j=1}^{N} G_j P_j + \sigma_n^2} \tag{2.2}$$

where G_0 is the channel gain for the desired signal with power P_0, G_j is the channel gain for the interfering signal with power P_j, σ_n^2 is the thermal noise power, and N is the number of interfering cells. Then, multiple SINRs within the subframe could be compressed into an effective SNR. Such a method to achieve SINR compression was presented in [10] called Effective Exponential SNR Mapping (EESM).

2.4.2 Channel Quality Indicator (CQI)

The CQI report is a 4-bit word representing indices ranging from 0 to 15, as shown in Table A.1 in the Appendix. Each index of the CQI word gives a measure of the radio channel quality, and provides an estimated recommendation of the MCS that a UE can reliably receive from the gNB. In other words, given a CQI index, the BS tunes for modulation order and code rate such that a predefined block error rate target (*i.e.,* BLER_T) is maintained below a certain value (*i.e.,* in 4G-LTE usually 0.1, while in 5G-NR can be varied) [11]. Note that, the higher the value of the CQI index, the higher the modulation order and coding rate. The gNB can select among two different types of CQI report schemes: a) wideband CQI, the UE reports only one wideband CQI value for the whole system bandwidth; b) subband CQI, the UE reports the CQI for each subband (different contiguous resource blocks). In this work we consider the case where the gNB selects the wideband CQI feedback scheme.

2.4.3 Hybrid Automatic Repeat Request (HARQ)

Hybrid automatic repeat request (Hybrid ARQ or HARQ) is a combination of high-rate forward error-correcting coding (FEC) and automatic repeat request (ARQ) error-control.

Forward error correction (FEC) is a technique where the sender encodes the message in a redundant way using error-correcting code (ECC). This is done to allow the receiver to detect errors (up to a number of errors depending on the code that is being used) that may occur anywhere in the message, and often to correct these errors without re-transmission. Thus, since the receiver does not need to request re-transmission of the data, a reverse channel (back-channel) is not required. Hence, FEC is suitable where re-transmissions are costly or even impossible [12].

Automatic Repeat reQuest (ARQ) is an error control method for data transmission. It uses error-detection codes, acknowledgment (ACK) or negative acknowledgment (NACK) messages, and timeouts to maintain the reliability of data transmissions. An acknowledgment is a feedback signal sent by the receiver (*i.e.,* UE in downlink) which indicates that a data frame has been correctly received. When the transmitter does not receive an acknowledgment within a reasonable period of time after sending the data frame (*i.e.,* timeout), it retransmits the data frame. This procedure is repeated until it receives an ACK or until the number of consecutive NACKS is bigger than the predefined number of retransmissions. In other words, the receiver has up to a predefined number of retransmission trials to receive the data frame correctly, otherwise the data frame is dropped.

To summarize the HARQ process, the FEC is used to detect and correct expected errors that may occur anywhere in the message, while the ARQ method is used as a backup strategy to correct errors that cannot be corrected by the FEC redundancy sent in the initial transmission. However, there is a drawback regarding the HARQ process which is the fact that it imposes an additional delay on the transmission, called HARQ Round Trip Time (RTT) [13].

2.4.4 Modulation and Coding Scheme (MCS)

Two of the major key components of the 5G-NR physical layer are the modulation and channel coding schemes (MCS). In particular, 5G-NR supports five different modulation schemes for the uplink and four for the downlink, similarly to 4G-LTE. For both uplink and downlink, the 5G-NR supports quadrature phase shift keying (QPSK), 16 quadrature amplitude modulation (16QAM), 64QAM and 256QAM modulation formats, while there is an extra modulation scheme called $\pi/2$-BPSK, for the uplink case. The 5-bit MCS index describes the different modulation schemes which are described by the number of bits per symbol (Q_m) used for modulation, and by the target code rate (R). MCS depends on radio link quality and it defines the number of bits (either useful or parity bits) that can be transmitted per Resource Element (RE).

The better the quality of the link, the higher the MCS and the more payload can be transmitted. Contrarily, the worse the link's quality, the lower MCS and thus less useful data can be transmitted. In other words MCS depends on the CQI which is reported by the UE. However, experiencing bad link quality implies higher error probability. As mentioned in Section 2.4.2, a block error rate target, which is a design parameter, influences the link adaptation performance based on different QoS agreements and radio link conditions, and it is typically set to a constant threshold 0.1. To maintain this error probability below this threshold, the MCS index should be dynamically adjusted accordingly. In 4G-LTE and 5G-NR, this is done once per TTI (1 ms) individually for each active user. For more information regarding the values of the modulation and coding schemes refer to Table A.2 and Table A.3.

2.5 Inner Loop Link Adaptation (ILLA)

The inner loop of the link adaptation mechanism, called ILLA is used to determine the resources to be used for a transmission and the corresponding transport format (*i.e.,* MCS, TBS), to serve a scheduling entity with a given buffer size and channel quality. Figure 2.5 illustrates the link adaptation mechanism with inner-loop functionality. As it can be seen, ILLA selects the optimal MCS index to be used for this user for a future transmission time interval, out of a predefined fixed table, given the estimated SINR $\hat{\gamma}_m$ of a user for a given transmission time interval and a given chunk of Physical Resource Block (PRB) in the frequency domain.

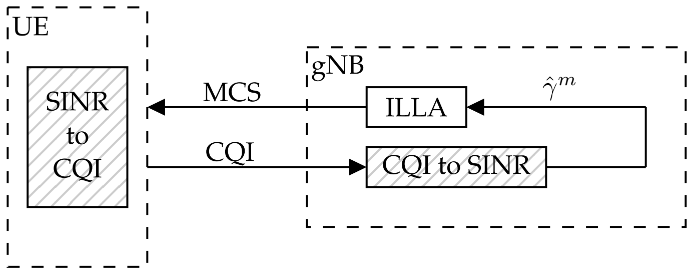

Figure 2.5: Inner-loop link adaptation block diagram.

Using the allocation size, numer of PRBs and the MCS index, the Transport Block Size (TBS) is computed and it corresponds to the maximum data rate that the user can currently achieve [14]. Note that, for initial transmissions, the desired number of bits of the scheduling entity and min/max TBS restrictions are given as input, while for retransmissions, the initial transmission TBS is given as input to the link adaptation mechanism. To get a clear picture on the ILLA mechanism, we present the following main steps for new transmissions, however for a more detailed description of the TBS determination refer to [15].

1. determine the MCS based on the channel resource efficiency.
2. determine the number of resource elements N_{RE} .
3. calculate the initial number of information bits N_0^{info} using the MCS and N_{RE}.
4. calculate TBS based on the conditions given in TBS determination section in [15].

2.6 Outer Loop Link Adaptation (OLLA)

Although the ILLA techniques proved to increase the average user throughput, a work proposed in [16], proved that another outer loop mechanism for the link adaptation can be added to improve the throughput even more. In particular the authors proposed the well-known Outer Loop Link Adaptation technique (OLLA) to adjust the SINR thresholds by an offset (Δ^{OLLA}) which is updated online based on a feedback representing the accuracy of the transmitted bits, so that the average BLER converges (marginally) to a predefined BLER target (BLER$_T$). In other words, OLLA generates a correction term which is the accumulation of predefined fixed up and down steps (*i.e.*, Δ^{up} and Δ^{down} respectively) corresponding to HARQ ACKs and NACKs, respectively, received from the UE. The ratio of the step sizes for ACK and NACK determines the BLER target to which the feedback loop tries to converge. Note that, the convergence speed of the OLLA is determined from the up and down step size. A graphical representation of the downlink link adaptation problem using OLLA is shown in Figure 2.6.

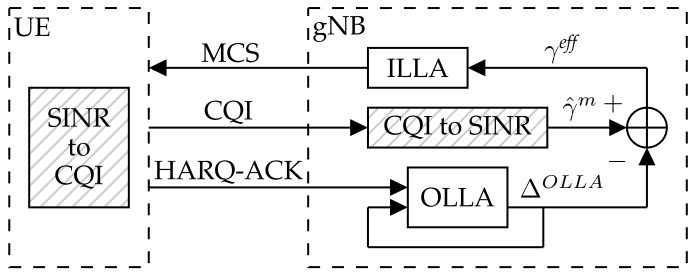

Figure 2.6: Outer-loop link adaptation block diagram.

Each Transmission Time Interval (TTI), herein denoted as k, a positive acknowledgment (ACK) or a negative acknowledgement (NACK) is received by the UE at the gNB, if the transmitted bits are recovered accurately or not respectively. The accuracy of the transmission is verified by a Cyclic Redundancy Check (CRC) which corresponds to a dichotomous random variable (it takes one of only two possible values when measured) whose average is the BLER. Thus the evolution of the discrete time OLLA offset is given by:

$$\Delta_k^{OLLA} = \Delta_{k-1}^{OLLA} + \Delta^{up} \cdot e_k - \Delta^{down} \cdot (1 - e_k) \tag{2.3}$$

where $e[k] = 0$ for ACK, and $e[k] = 1$ for NACK. Equation 2.3 can be also written in this form to give a better understanding how ACK and NACK affect the evolution of the Δ_k^{OLLA}:

$$\Delta_k^{OLLA} = \begin{cases} \Delta_{k-1}^{OLLA} - \Delta^{down}, & \text{if ACK } (e_k = 0) \\ \Delta_{k-1}^{OLLA} + \Delta^{up}, & \text{if NACK } (e_k = 1). \end{cases} \tag{2.4}$$

Note that, $\Delta^{OLLA}k = 10 \cdot \log_{10}(\Delta_k^{OLLA})$. In addition, Δ^{up} and Δ^{down} are expressed in decibels (dB) while their values should satisfy the following relationship to meet the predefined BLER target:

$$BLER_T = \frac{1}{1 + \frac{\Delta^{up}}{\Delta^{down}}}. \tag{2.5}$$

or to express it in a ratio of Δ^{down} and Δ^{up}:

$$\frac{\Delta^{down}}{\Delta^{up}} = \frac{BLER_T}{1 - BLER_T}. \tag{2.6}$$

The estimated SINR compensated by the OLLA mechanism is given by the following relationship in the logarithmic domain [17]:

$$\gamma_k^{eff} = \hat{\gamma}_k^m - \Delta_k^{OLLA} \tag{2.7}$$

Slow convergence of the traditional OLLA has a negative impact on the performance of LTE networks [18].

2.7 Related Work

A self-optimization algorithm was proposed in [19] for outer loop link adaptation of LTE downlink channel. Based on recordings of connection traces, their algorithm adapts the initial OLLA offset value to the median value observed in a certain connection. They have shown that tuning the initial OLLA offset parameter in the downlink is beneficial since the OLLA convergence is faster, the throughput is increased and the retransmission rates are reduced.

The authors in [20] proposed a scheme based on sequential hypothesis testing for outer loop link adaptation (OLLA). In particular, they investigated the rate of convergence under scenarios with large changes in the SINR inaccuracies, and with normal changes during the steady-state. Their proposed scheme addressed these scenarios by aggressive control mode and a more conservative control mode, based on the sequential hypothesis testing, respectively. They illustrated the efficacy of their proposed scheme under these scenarios using numerical simulations showing increased average throughput.

More recently, Saxena *et al.* in [21] proposed an online machine learning approach based on contextual Multi-Armed Bandits (contextual MAB), to select the optimal MCS for outer loop link adaptation in cellular communication systems. They formulated the problem of selecting MCS for link adaptation as an online stochastic policy optimization problem, and they solved it using the contextual MAB. Their approach has demonstrated up to 25% increase in average link throughput, and faster convergence compared to the traditional OLLA approach.

The authors in [22] designed a reinforcement learning (RL) framework based on the Q-learning algorithm for MCS selection in order to increase the spectral efficiency and maintain low block error rate (BLER). Their proposed framework learns to decide a suitable MCS that maximizes the spectral efficiency. In particular, at each time instance, the base station (BS) receives CQI measurements and selects a MCS which maximizes a certain reward. The goal of the RL algorithm is to find the best policy using the Q-function.

In [23] the authors proposed a new approach based on logistic regression to enhance the traditional OLLA algorithm. This approach dynamically adapts the step size of the control based on the channel state and updates the OLLA offset parameter independently of the reception conditions (data packet received or not). Several simulation scenarios and comparisons were carried out to show that their proposed approach (eOLLA) outperforms the traditional OLLA.

Chapter 3

Reinforcement Learning

Reinforcement Learning (RL) is an emerging area of machine learning which is studied in many other fields, such as control theory, game theory, information theory, multi-agent systems, operations research, and genetic algorithms. Looking back to the literature, one can say that RL borrows ideas from optimal control for finding optimal sequential decisions and artificial intelligence for learning through observation and experience. Indeed, what is called reinforcement learning, in optimal control theory is called approximate dynamic programming. However in this work, we follow the terminologies and notation from RL and specifically from [24] to avoid any confusions.

3.1 The Reinforcement Learning Problem

The reinforcement learning problem is defined as the problem of learning from interaction between an agent and an environment to achieve a specific goal. An agent is a decision-maker and it interacts with an environment (usually unknown) that is actually everything outside the agent. In RL, the agent tries to determine the optimal policy (the policy that maximizes the future rewards) to achieve a specific goal, through interaction with an unknown environment, based on a reward signal indicating the quality of the action taken.

A brief overview of the RL process is presented in Figure 3.1. In particular, at each time instance, $t = 1, 2, 3, \ldots, T$, an agent receives an observation $O_t \in \mathcal{O}$ of the environment's current state $S_t \in \mathcal{S}$, where $\mathcal{O}$ and $\mathcal{S}$ are sets of possible observations and states respectively. Note that, in this work we assume that the observation O_t is an exact copy of the state S_t (*i.e.*, $O_t = S_t$). Then, the agent generates a policy $\pi_t(a \mid s)$, which is a mapping from states to probabilities of selecting each possible action). In other words, this mapping is the probability that $A_t = a$ given that $S_t = s$. Based on this policy, the

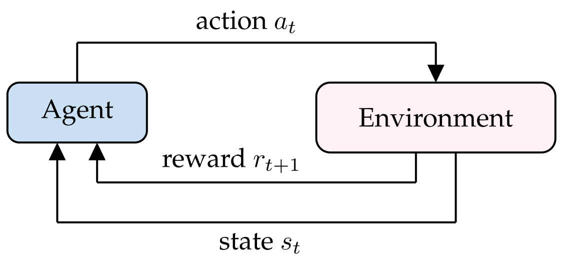

Figure 3.1: Interactions between agent-environment.

agent selects an action $A_t \in \mathcal{A}(S_t)$ where $\mathcal{A}(S_t)$ is the set of actions available in state s_t. After one complete loop (time instance), the agent receives a reward R_{t+1} while the state of the environment provoked to a new state S_{t_1} as a consequence of action A_t. The same process is repeated until the the environment reaches a terminal state S_T. Each subsequence of agent-environment interactions between the initial and terminal states, is called an episode.

3.2 Elements of Reinforcement Learning

In the RL problem, beyond the two main elements (*i.e.,* agent, and environment), there exist four other main subelements *i.e.,* policy, reward signal, value function, and a model of the environment (if known).

A policy, π, or $\pi(a \mid s)$, defines the way the agent has to behave (what action to take) based on the environment's state at a certain time instance. A policy can be deterministic or stochastic depending on the nature of the problem/environment. In other words a policy might not lead to a certain action confidently but randomly. Although, usually it is a simple function or a lookup table, it might be a search process. In the reinforcement learning control problem, the policy is updated online (during an episode) or offline (after the end of an episode) in order to find the one that maximizes the value function.

A reward signal, R_t, defines a direct feedback from the environment to the agent, containing information of how good was a certain action that the agent has taken at a certain state. The reward is received by the reinforcement learning agent from the environment at each time instance. The goal of the reinforcement learning agent is to maximize the total reward it receives over the long run (during an episode). Usually, reward signals are divided into three different types: a) positive reward, showing that the action taken at a certain state was good; b) negative reward, showing that the action taken at a certain state was bad; and c) zero, showing that the action taken at a certain state did not have any effect. The reward signal guides the decision making process of changing a policy. For example, if an action selected by a policy leads to low reward, then it would be beneficial for the reinforcement learning agent to change the current policy in order to increase the future rewards by selecting other actions.

A value function, $V_\pi(s)$, is a measure of the overall expected reward assuming that the agent is in state s and follows a policy π until the end of the episode. An action-value function, $Q_\pi(s,a)$, also called Q-Value or Q-function (where Q is abbreviation from the word Quality), is a measure of the overall expected reward assuming that the agent is in state s, takes an action a, and follows a policy π until the end of the episode. It is worth mentioning, that typically when the state and action spaces are small enough, the value and action-value functions can be represented in a tabular form, since exact solutions can be found.

Another important yet optional element of reinforcement learning problem is the model of an environment. A model is the element which captures the dynamics and mimics the behavior of the environment. In other words, given a state and action, a model can estimate the next state and reward based on its dynamics and behavior. Reinforcement learning problems can be tackled using model-based methods that use models and planning to predict the next state and reward, or model-free methods that do not use models but instead they use trial-and-error methods to learn optimal policies.

3.3 Markov Decision Processes (MDP)

Typically, reinforcement learning problems are instances of the more general class of Markov Decision Processes (MDPs), which are formally defined using a tuple $(\mathcal{S}\mathcal{A}PR)$:

- **States:** a finite set $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ of the n possible states in which the environment can be;

- **Actions:** a finite set $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ of the m admissible actions that the agent may apply;

- **Transition:** a transition matrix P over the space $\mathcal{S}$. The element $P\left(s, a, s'\right)$ of the matrix provides the probability of making a transition to state $s' \in \mathcal{S}$ when taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. Note that s denotes s_t and s' denotes s_{t+1};

- **Reward:** a reward function R that maps a state-action pair to a scalar value τ, which represents the immediate payoff of taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$

The goal of an MDP is to train an agent in order to find a policy π that will maximize the total amount of rewards (cumulative rewards) it receives from taking a series of actions in one or more states. The total reward is calculated over an infinite horizon for *continuing-tasks* and over a finite horizon for *episodic-tasks*. In continuing-tasks, the interaction between agent-environment is kept alive without limit. In episodic-tasks, there exists the notion of final time instance T in which an episode ends. In this case, the agent-environment interaction breaks in a special state called terminal state, followed by a reset to an initial state.

For continuing-tasks, if an agent follows a policy π starting from state s at time t, the return or sum of rewards over an infinite time horizon is given by:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{3.1}$$

where $\gamma \in [0,1]$ is a discount factor that weights future rewards also called *discount rate*. Discount rate with $\gamma \to 0$ leads to "myopic" evaluation, that is, the return G_t takes into account the immediate reward R_{t+1}. Discount rate with $\gamma \to 1$ leads to "far-sighted" evaluation, that is, the return G_t takes into account all future rewards more strongly. For episodic-tasks, the return or sum of rewards over a finite time horizon is given by:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T. \tag{3.2}$$

Although this work focuses on episodic tasks, for reasons of completeness, a unified notation for the return of both continuing-tasks and episodic-tasks is given by:

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k. \tag{3.3}$$

Note that there is a constraint where the final time instance can be infinity, $T = \infty$, and the discount rate can be one, $\gamma = 1$, but not both.

3.4 Action-Value Methods

A policy maps the observed states of the environment to actions to be taken when in those states. The goal of the agent is to find the policy that increases the overall amount of rewards. However, in reality, knowing *a priori* the sum of future rewards is usually not possible, since a reward expresses only the immediate feedback from a certain state, and thus what happens in the future is unknown. For example it might be possible that the environment is at a state with a high positive reward but then multiple states with low or negative rewards follow. Hence, the agent needs to take a long-term view of the future rewards in consideration, and to do so it needs to estimate these rewards by means of value functions or action-value functions. Estimating value functions or action-value functions (*i.e.,* functions of states or state-action pairs respectively) is necessary in order to get an approximation of how big is the expected return when in a certain state or how big is the expected return when applying an action when in a certain state respectively.

The *value function* v_π of a state expresses the overall reward that is expected to be obtained when using that state as the initial state. The values are really important for planning since the policy that chooses actions is based on the values. Although high values promise optimal policies in order to reach the final goal, in reality it is challenging to plan the sequence of steps to reach it. Recall that a policy, π, is a mapping from each state, $s \in \mathcal{S}$, and action, $a \in \mathcal{A}(s)$, to the probability $\pi(a \mid s)$ of taking action a when in state s. The *value function* when following a policy π is denoted by $v_\pi(s)$, and is given by:

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s\right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right], \tag{3.4}$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expected value of a random variable given that the agent follows policy π. Note that the value of the terminal state, if any, is always zero.

For the control problem, an alternative to the value-function, is the *action-value function* $q_\pi(s, a)$ which is the expected return starting from a state s, taking an action a, and following the same policy π for the remaining duration of the episode. The action-value function is given by:

$$q_\pi(s,a) = \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a\right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]. \tag{3.5}$$

As mentioned before, solving a reinforcement learning is equivalent to finding a policy that achieves the highest reward in the long run. This policy is called *optimal policy* π_* and it is better than or equal to a policy π', if its expected return is greater than or equal to that of π' for all states,

$$v_{\pi^*}(s) \geq v_{\pi'}(s), \quad \forall s \in \mathcal{S}, \forall \pi. \tag{3.6}$$

Note that, there is always an optimal policy, and sometimes there may be several optimal policies that share the same optimal value function $v_*(s)$ or the same optimal action-value function $q_*(s, a)$, defined as:

$$v_*(s) = \max_\pi v_\pi(s), \quad \forall s \in \mathcal{S} \tag{3.7}$$

and

$$q_*(s,a) = \max_\pi q_\pi(s,a), \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s) \tag{3.8}$$

respectively. For the state-action pair (s,a), this function gives the expected return for taking action a in state s and thereafter following an optimal policy.

As mentioned before, reinforcement learning uses concepts from dynamic programming. One of this concepts is the property of value functions to satisfy recursive relationships between the value of a state and the values of its successor states. Hence, using this property, the so called *Bellman equation*, we get the optimal value-function and the optimal action-value function:

$$v_*(s) = \max_a \sum_{s',r} p\left(s',r|s,a\right)\left[r + \gamma v_*\left(s'\right)\right], \tag{3.9}$$

and

$$q_*(s,a) = \sum_{s',r} p\left(s',r|s,a\right)\left[r + \gamma \max_{a'} q_*\left(s',a'\right)\right], \tag{3.10}$$

respectively.

3.5 Exploration - Exploitation

The trade-off between exploration of unknown policies and exploitation of the current best policy is one of the most important factors to achieve optimal performance for a reinforcement learning problem. The basic idea to balance the exploration-exploitation came from multi-armed bandits and specifically by using the *greedy-action* as the action-selection method. The greedy-action is the action which gives the highest estimated value. Briefly, we say that we perform exploitation when we select a greedy action, and exploration when we select non-greedy actions to explore the state space. It makes sense that to maximize the expected reward for one step, we need to exploit, however, to achieve a long-run higher total reward, we need to explore.

Although there are several advanced exploration techniques (see [25] for more information), in this work we consider the *ϵ-greedy* action selection method. As we said before, the simplest action selection method is to select the action with the highest estimated action value. The greedy action selection method can be written as

$$A_t = \operatorname*{argmax}_a Q_t(S_t, a), \tag{3.11}$$

where A_t takes the value of a at which the expression $Q_t(S_t,a)$ is maximized. To introduce the notion of exploration, we focus on an alternative action selection method, that is the *ϵ-greedy*. In this action selection method, the agent performs exploitation (*i.e.,* selects the greedy action) with probability $1 - \epsilon$, and exploration (*i.e.,* selects a random action) with probability ϵ. A good practice for this action selection method is to initialize ϵ to a probability close to one at the beginning of training and then gradually to reduce ϵ to a lower value. This is logical because at the beginning the agent has limited knowledge about the environment and hence it is encouraged to perform exploration. Contrarily, after many iterations, the agent gains knowledge of the environment and hence it is encouraged to perform exploitation.

3.6 Temporal-Difference Learning

Temporal-difference (TD) is a core idea in reinforcement learning since its methods consider model-free learning (*i.e.,* environment's dynamics are not needed), while it learns directly from raw experience. This algorithm uses ideas from Dynamic Programming (DP) to estimate the state and action values based on estimates of subsequent values. In other words, TD methods use *bootstrapping,* that is they can determine the increment to $V(S_t)$ after the next time instance. At time $t + 1$ they immediately form a target and make a useful update using the observed reward R_{t+1} and the estimate $V(S_{t+1})$. This way of updating the value (one-step update) is called TD(0) and is the simplest TD method. However, this idea can be generalized for cases where the value update is done based on multiple steps (*i.e.,* TD(λ)). In other words, to have more information from the environment, we can wait for more than one step, or even for the whole episode to be finished, and then to update the weights. For the TD(0), the value update is given by:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{3.12}$$

where α is the learning rate, and $V(\cdot)$ denotes the current estimate of $v_*(\cdot)$,

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi \left(S_{t+1}\right) | S_t = s\right]. \tag{3.13}$$

The difference between the agent's value approximation at time $V\left(S_t\right)$, and its discounted approximation of the successor state $\gamma V\left(S_{t+1}\right)$, plus the reward R_{t+1}, is called the TD-error and is given by:

$$\delta_t = R_{t+1} + \gamma V\left(S_{t+1}\right) - V\left(S_t\right). \tag{3.14}$$

The algorithm of the tabular TD is presented in Algorithm 1. First, the TD(0) uses the policy π to be evaluated, and initializes the value estimate $V(s)$ for all states that belong to (S). Then, for each episode, the state S is initialized. After the initialization of the state, for each time instance until the environment reaches the terminal state S_T, the policy gives an action A

at state S. When this action is applied, the reward R will be observed, and the environment will be moved to the next state S'. Based on these measurements, the estimated value will be updated as shown in Equation (3.12), and the whole process will be repeated for each time instance until the end of the episodes.

Algorithm 1: Temporal-Difference (TD) algorithm

```
Input: the policy $\pi$ to be evaluated
Initialize: $V(s)$ arbitrarily (e.g., $V(s) = 0, \forall s \in \mathcal{S}^+$ );
repeat
    Initialize $S$;
    repeat
        $A \leftarrow$ action given by $\pi$ for $S$
        Take action $A$; observe reward $R$, and next state $S'$
        $V(S) \leftarrow V(S) + \alpha \left[R + \gamma V\left(S'\right) - V(S)\right]$
        $S \leftarrow S'$;
    until reach of terminal state $S$;
until end of episodes;
```

3.7 Q-Learning Control

Along with the TD estimation, another important contribution to the reinforcement learning field, was the development of an off-policy control algorithm, the Q-learning [26]. The goal of Q-learning algorithm is to find the best action to take when at a certain state. In other words Q-learning tries to find the optimal policy that maximizes the total reward. The estimation of the total reward is given by the q-table which is updated with the q-values after each episode. The agent uses the q-table to select the action that has the highest q-value at a certain state. Note that, the agent is initialized arbitrarily with some weights that represent the agent's current q-values of the q-function that is to be approximated. Following the initialization, the agent interacts with the environment by applying the action with the highest estimated q-value, receiving the reward and moves to the next state. Algorithm 2 presents the basic steps of the Q-Learning algorithm. First a q-table of size $n_s \times n_a$, where n_s, and n_a represent the number of states and actions respectively, needs to be generated. Next, and until the end of episodes, an action when at a certain state is chosen based on the q-table which is arbitrarily initialized (*i.e.,* usually the q-values are set to 0). At the beginning the agent will try to explore the environment since the epsilon rate (*i.e.,* of the $\epsilon - greedy$) starts with high value, and thus random actions will be chosen. The intuition behind this is that at the beginning the agent does not have information about the environment and hence it will try to explore. However, as the agent learns more about the environment via exploration, the epsilon rate will start decreasing and thus the agent will start selecting the greedy action (*i.e.,* as mentioned in Section 3.5). At this point, the agent updates the q-values for being at a certain state and applying a certain action using the Bellman equation. In other words using the update of the Bellmain equation, the q-table is updated and the action-value function Q which gives the expected future reward of an action when at a certain state is maximized.

Algorithm 2: Q-Learning algorithm

```
Initialize arbitrarily: $Q(s,a)\ \forall s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, and $Q(S_t;\cdot) = 0$;
repeat
    Initialize $S$;
    repeat
        $A \leftarrow$ action from $S$ using policy derived from Q (e.g., $\epsilon$-greedy)
        Take action $A$; observe reward $R$, and next state $S'$
        $Q(S,A) \leftarrow Q(S,A) + \alpha\,[R + \gamma \max_a Q(S',a) - Q(S,A)]$
        $S \leftarrow S'$;
    until reach of terminal state $S$;
until end of episodes;
```

3.8 Deep Q-Learning and Deep Q-Network (DQN)

Although Q-learning was considered a good candidate to solve difficult tasks, most applications were limited to the ones with small state space. However, in [27] the authors showed that Q-learning could be used in combination with a Deep Neural Network (DNN) to solve problems with larger state space approaching human level performance.

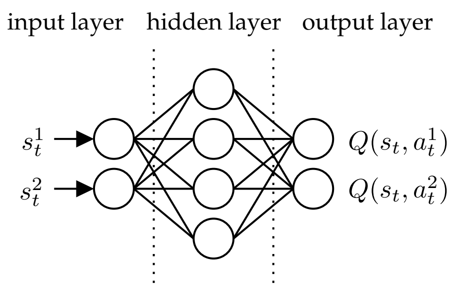

Figure 3.2: Q-Network example.

In particular, the main idea behind Deep Q-learning, is the use of a deep neural network, called Deep Q-Network (DQN), (see Figure 3.2 for a simple example), which is used as a function approximator of the optimal state-value function given by:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots \mid S_t = s, A_t = a, \pi\right], \tag{3.15}$$

where $Q^*(s,a)$ is the maximum expected cumulative reward discounted by parameter γ (*i.e.,* discount factor) while following the policy π. However, an issue which arises from the use of a nonlinear function approximator such as the DQN, is the convergence [28]. In particular, an agent usually receives sequences of states that are highly correlated which leads to instability or even divergence of the RL problem. The authors in [27] proposed the use of *experience replay* to cope with these problems. This method stores the experience of the agent at each time instance, $e_t = (s_t, a_t, r_t, s_{t+1})$, in a buffer of predefined size (*i.e.,* batch size), which is then sampled randomly to eliminate the correlation of the state-action-reward sequences. Finally, for the DQN to converge, a fixed reference point which is called *target network* needs to be updated periodically. A summarize of these steps is presented in Algorithm 3.

Algorithm 3: Deep Q-learning with experience replay.

```
Initialization;
for episode = 1 to M do
    Initialize state $S_1$;
    for step t = 1 to T do
        Apply $A_t$ based on $\epsilon$ - greedy policy;
        Observe reward $R_t$, and next state $S_{t+1}$;
        Store $E_t = (S_t, A_t, R_t, S_{t+1})$ in memory
        Sample random mini-batch from memory
        if $S_{j+1}$ terminal then
            $y_j = R_j$;
        else
            $y_j = R_j + \gamma \max_a Q(S_{j+1}, a; \theta')$
        end
        Perform GD step on $(y_j - Q(S_j, A_j; \theta))^2$;
        Update $S_t \leftarrow S_{t+1}$;
        Update weights, $\theta' \leftarrow \theta$, every $C$ steps;
    end
end
```

3.9 Proximal Policy Optimization (PPO)

Proximal Policy Optimization method, falls into the policy-based methods which is an alternative method to the aforementioned DQN which falls into the action-value-based methods. The fundamental difference is that in policy-based methods the policy is updated directly instead of being deduced from the value function [29]. In PPO, we maintain two models, the policy $\pi_\theta(a \mid s)$ and the action-value function $Q_w(s,a)$, which corresponds to the actor and the critic respectively. The former, $\pi_\theta(s,a)$, is updated in the direction suggested by the latter, $Q_w(s,a) \simeq Q^{\pi_\theta}(s,a)$ which is used to estimate the action-value function under the policy π_θ. These methods learn the probabilities of taking a certain action from the action space leading to a stochastic policy. In general, policy gradient methods' goal is to optimize the following loss:

$$L(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta \left(a_t \mid s_t \right) \hat{A}_t \right], \tag{3.16}$$

where $\hat{A}_t$ denotes the advantage function at time instance t, and π_θ is a stochastic policy. By performing stochastic gradient ascent on the aforementioned objective function, the policy is updated and the gradient estimator is given by:

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi \left(a_t \mid s_t, \theta \right) \hat{A}_t \right]. \tag{3.17}$$

The advantage estimate $\hat{A}_t$ involves a k-step estimate of the approximate value function subtracted by the returns, given by:

$$\hat{A}_t^{(k)} = -V\left(s_t\right) + r_t + \gamma r_{t+1} + \ldots + \gamma^{k-1} r_{t+k-1} + \gamma^k V\left(s_{t+k}\right). \tag{3.18}$$

Note that, as $k \to \infty$, the advantage estimate becomes equivalent to the approximate value function subtracted by the empirical return. However, in this work we consider only the case of $k = 1$, which reduces the advantage estimate to:

$$\hat{A}_t^{(1)} = -V\left(s_t\right) + r_t + \gamma V\left(s_{t+1}\right). \tag{3.19}$$

The authors in [30] proposed an extension of the PPO, called clipped surrogate objective. The objective is to assure a probability ratio $r_t(\theta)$ given in Equation 3.20 is clipped within a range near unity.

$$r_t(\theta) = \frac{\pi_\theta \left(a_t \mid s_t \right)}{\pi_{\theta_{\text{old}}} \left(a_t \mid s_t \right)} \tag{3.20}$$

This is used to keep the gradient updates within the clipping range and thus to avoid large updates of the gradient. Hence the new clipped surrogate objective is given by:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \operatorname{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right) \right] \tag{3.21}$$

where ϵ defines the clipping range. The algorithm of the PPO using clipped surrogate objective is given in Algorithm 4.

Algorithm 4: Proximal Policy Optimization (PPO) with clipped surrogate objective.

```
for iteration $= 1, 2, \cdots$ do
    for actor $= 1, 2$ to $N$ do
        Run policy $\pi_{\theta_{\text{old}}}$ in environment for $T$ time instances;
        Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$;
    end
    Optimize surrogate $L$ wrt $\theta$, with $K$ epochs and mini-batch size
      $M \leq NT$;
    $\theta_{\text{old}} \leftarrow \theta$;
end
```

Chapter 4

Methodology

In this chapter we introduce the training and inference phase of the reinforcement learning approaches that we presented in the previous section. In particular, we present the architecture of our reinforcement learning-based downlink adaptation approach in 5G networks. Note that, in this work we consider the problem of decentralized downlink link adaptation, where each user is considered as an episode of a global agent that is trained to optimally adjust the SINR backoff in order to converge to the actual estimated SINR of the user. The proposed method of this work was developed, implemented and evaluated using Ericsson's system-level radio network simulator herein referred as *the simulator*, and the RLlib library [2] written in Python as the platform for developing and configuring the DQN and PPO models. To communicate between the simulator (i.e., client) and the RLlib libraries (i.e., server), an interface was developed to establish the required protocols for sending and receiving requests and responses respectively. As shown in Figure 4.1, the simulator's elements represent the environment and they are in light red colour while the RLlib library represents the agent and it is in light blue color.

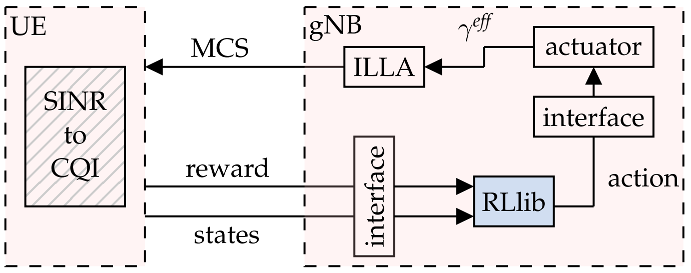

Figure 4.1: RL-based link adaptation framework.

Although there are several successful deep reinforcement learning algorithms that were recently developed, and even more adaptations of them, the selection was steered based on several considerations. One of these considerations is that the agents used and developed in this work can handle the problem of link adaptation despite the stochastic transitions and rewards coming from the environment, i.e., the radio network. Another strict constraint for the selection of the aforementioned agents, is the extremely detailed nature of the simulator, which implies slow simulations, and the periodicity of different measures such as the throughput, which adds an extra delaying factor to the already detailed simulator. Simulating one second of a radio network could take several minutes in real-life. Although DQN is characterized by its high data efficiency due to the feature of experience replay, the aforementioned factors imply difficulties regarding the online training procedures of the DQN agent, and thus the hyperparameter tuning, made the training phase time consuming. For this reason other types of agents such as the PPO were considered in order to simplify the hyperparameter tuning procedure.

4.1 Problem Formulation

Consider the downlink link adaptation problem in a 5G radio network consisting of C cells, indexed by $c = 1, 2, \dots, C$, and U users, indexed by $u = 1, 2, \dots, U$. Each cell serves a set of users $\mathcal{U}_c$ with cardinality $U_c = |\mathcal{U}_c|$, hence $U = \sum_c U_c$. The network bandwidth is denoted with B, and the cells within the network operate in this frequency bandwidth, which is divided into K equally sized time-frequency REs. As mentioned in Section 2.4, users within the same cell do not interfere with each other since the OFDM scheme is used to eliminate intra-cell interference. However, interference inter-cell interference is present as a consequence of transmissions in other neighbouring cells. Note that, the transmitters and receivers of the network are assumed to have a single omni-directional antenna.

The modulation and coding schemes (MCS) supported in 5G-NR are given in Table A.3, and they are denoted by indices $MCS \in \{MCS_0, MCS_1, \dots, M_m\}$, where $m = 28$ is the maximum MCS index. At any given time instant, the ILLA algorithm has to choose the most suitable index among MCS for each UE being scheduled for downlink data reception. For this decision, ILLA takes a compensated SINR estimate adapted by either the OLLA algorithm (with predefined BLER target), or by the RL-agent. As a result, the RL-agent needs to have information about the state of its serving UE, and thus the agent measures several metrics, mentioned in prior chapters, such as the CQI, HARQ-ACK, and the current SINR backoff. In other words, the UE feeds back to the gNB (i.e., agent), its CQI, the HARQ-ACK of a certain transmission of the UE, and its current measured SINR backoff. The CQI are given in

Table A.1 and they are denoted by indices $CQI \in \{CQI_0, CQI_1, \ldots, CQI_q\}$, where $q = 15$ is the maximum CQI index. The HARQ-ACK which represents whether a transmission was succesfully decoded or not (i.e., ACK or NACK respectively) is denoted by $HARQ - ACK \in \{0, 1\}$. The SINR correction value is denoted by Δ^{OLLA} when using the OLLA algorithm, and by Δ^{RL} when using the RL-agent. Note that, the values that the Δ^{OLLA} can take, depend on the BLER target and consequently on the Δ^{up} and Δ^{down} as defined in Equation (2.6). In contrast, Δ^{RL} is more flexible and can take values defined in the action space depending on its resolution, and thus there is no need for setting a predefined BLER target.

To illustrate the aforementioned problem, refer to Figure 4.2. As shown in the figure, a gNB sends and applies an MCS index for a link (i.e., UE). Next, the UE feeds back to the gNB (i.e., agent) its current CQI index, and the HARQ-ACK of its latest transmission. Based on this information, the previous SINR backoff value Δ, and the reward taken from previous time instance, the agent decides how to adjust the estimated SINR $\hat{\gamma_m}$ reducing its value by a new Δ. As a result, the compensated SINR γ_{eff} will eventually converge to the actual SINR and consequently, the ILLA algorithm will select the most appropriate MCS index for the transmission.

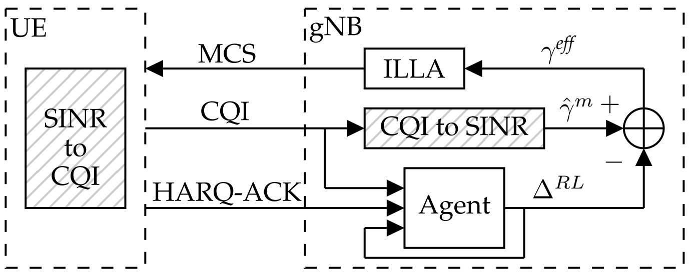

Figure 4.2: RL-based link adaptation block diagram.

However, as it is shown in Figure 4.2, the agent does not control explicitly the MCS index. Instead, it controls the SINR backoff value Δ, and implicitly the MCS index changes accordingly. The MCS selection of a UE is denoted by:

$$MCS^u = f\left(CQI^u\right) + g\left(\mathcal{I}_{HARQ-ACK_i}\right) \tag{4.1}$$

where $\mathcal{I}_{HARQ-ACK_i}$ denotes the history of HARQ-ACK feedbacks $\forall i \in (1, 2, \ldots, N)$ where N denotes the total number of transmissions within a time interval. In practice, $g\left(\mathcal{I}_{HARQ-ACK_i}\right)$ denotes a function which defines the offset value which is subtracted from the estimated SINR $\hat{\gamma}_m$. As mentioned earlier, although we do not select explicitly the MCS index for a link, we try to give the optimal compensated SINR γ_{eff} using the sequential decisions made from the RL-agents.

4.2 RL Algorithm Selection

Although there are many RL approaches such as Deep Deterministic Policy Gradient (DDPG) [31], Multi-Armed Bandits (MAB) [32] and Unimodal Multi-Armed Bandits (uMAB) [33], that can potentially solve the problem of downlink link adaptation by formulating it accordingly, in this work we consider the DQN and the PPO. The reasoning of this selection comes not only from the fact that these algorithms are proved to perform well in different problems similar to our use case, but also because of the nature of the simulator and the interface developed to interact with the agents. In addition, the selection was based on the measurements that are readily available from the simulator such as the CQI and HARQ-ACK feedback but also the average user throughput. The upcoming sections focus on the description of the experimental setup used in this work, and the design of the RL algorithms used to evaluate their performance.

4.3 Markov Decision Process Design

To get a better idea of the RL algorithms, and solve the aforementioned problem, we need to define the MDP and introduce the configuration used. The interaction between the agent (i.e., gNB) and the environment (i.e., UE in the network) is modeled by a Markov Decision Process (MDP) as mentioned in Section 3.3. In this section we design the MDP in order to solve the problem of downlink link adaptation using the agents (*i.e.,* DQN, and PPO) discussed in the previous chapter. To support the selection of states, actions and reward function, we integrate ideas from related work in the literature, our own experience gained from simulations, but also from the nature of the RL models used. Although the MDP is a tuple consisting states, actions, state transition function, and reward, we consider model-free algorithms which do not use the transition probability distribution. In other words, the agent does not have any prior information about the environment dynamics (i.e., state transition), but instead it estimates a value function or a policy based on interaction between the agent and the environment.

4.3.1 State Space

As a first step, we need to design the state representation, that contains valuable information regarding the state of the environment at each time instance. In particular, the state vector is denoted with $S_t = [s_t^1, s_t^2, s_t^3, s_t^4]^T$, where t denotes the time instance, and the number in the superscript denotes the index of the dimension. More specifically, s_t^1 represents the CQI index, s_t^2 represents the error between the CQI index at time t and the CQI index at time

$t-1$. A weighted time average of the HARQ-ACK of transmissions within a time interval is denoted with s_t^3. Finally, the Δ backoff value at time instance t in dB is denoted with s_t^4.

4.3.2 Action Space

Although the traditional OLLA algorithm uses fixed predefined steps, based on a fixed BLER target that adjust the SINR estimate, in this work we introduce a more flexible framework. In particular, the action that an RL agent selects and applies to a UE, does not depend on BLER targets which is an important and fundamental difference compared to the OLLA algorithm. In fact, we consider two different ways of applying the actions selected by the agent. The first way is to add a value a_t in dB on the previous SINR correction Δ_{t-1}^{RL}, while the other is to set explicitly the value a_t of the SINR correction Δ_t^{RL}. In contrast to the OLLA algorithm, the RL-based algorithms use a periodic way of applying the action for both aforementioned ways. In particular, the action period is 5 TTIs which corresponds to 5ms.

For the first case, there are five different discrete actions: add high step, add low step, do nothing, subtract low step, and subtract high step on the previous Δ value. In this work, the aforementioned actions were defined intuitively and by trial-and-error, as follows: $a_t \in \mathcal{A}^1 = \{-1.0, -0.1, 0.0, +0.1, +1.0\}$. Note that these values are expressed in dB which will adjust the value of Δ as shown in the following equation:

$$\Delta_t^{RL} = \Delta_{t-1}^{RL} + a_t, \text{where } \Delta_0^{RL} = 0. \tag{4.2}$$

For the second case, we need to first define the resolution of the action space since the selected action will apply directly a corresponding and explicit value of the Δ_t^{RL}, as shown in Eq. (4.3).

$$\Delta_t^{RL} = a_t. \tag{4.3}$$

As it can be seen, in this case, the algorithm can set directly the Δ^{RL}, where $a_t \in \mathcal{A}^2 = \{\psi k \mid k \in [k^{min}, \dots, k^{max}]\}$, where $\psi = 0.5$ denotes the resolution of the actions, and $k^{max} = 10$, $k^{min} = -10$ which denote the maximum and minimum values that Δ^{RL} can take in dB respectively. Although this way of applying the Δ^{RL} is not of high resolution, it can change its value in one only time instance. However, there is a drawback which is related to the size of action space which is big enough and it might be difficult for a RL agent to be trained. Note that, by increasing the resolution of the actions, the action space increases as well.

4.3.3 Reward Signal

The goal of the downlink adaptation technique, is to maximize the link throughput by selecting the most appropriate MCS index based on CQI and HARQ-ACK measurements. Thus, the choice of reward function for the RL problem comes naturally and it is related to the average downlink throughput within a time interval t (*i.e.,* RL action period), which is computed periodically and after taking action a_t when in state s_t. The reason behind the selection of such a reward signal is because we need to give more weight on the latest instantaneous throughput. In addition, when a transmission is not able to be decoded at the receiver side, the instantaneous throughput is 0, while it is still used to compute the time-weighted average throughput which is the actual reward signal. Thus, the reward signal is given by:

$$r_t = \frac{1}{BWk} \sum_{i=1}^{k} \phi_i, \tag{4.4}$$

where ϕ is the instantaneous throughput of a transmission k which represents the total number of bits transmitted within its time frame, and BW is a constant representing the bandwidth. The reward is 0 if there are no transmissions within a time interval t, or if the data from all the k transmissions within the time interval t were not able to be decoded at the receiver (*i.e.,* k consecutive NACKs).

4.4 Experimental Setup

In this section we present the experimental setup architecture, the details regarding the simulations and necessary configurations made in order to develop and evaluate our algorithms. As discussed in the previous section, to train our models we consider the average downlink throughput as the reward signal and thus, we need to have access of this measurement not only online but also to have logs for post-processing in order to evaluate and compare the different methods. In particular, we use the empirical cumulative distribution function (CDF) of the downlink throughput to illustrate in a statistical way the average throughput of the network while pointing to the performance of different groups of UEs (*i.e.,* cell-center, and cell-edge users). Higher probability in the CDF plot represent cell-center users (*i.e.,* users that are near the center of the cell) experiencing higher throughput compared to mid-cell users (*i.e.,* users that are near the middle of the cell radius), and cell-edge users (*i.e.,* users that are near the edge of the cell), that are represented with lower probabilities in the CDF plot.

4.4.1 Radio Network Simulator

The simulator uses a 5G-NR network which consists of 3 cells, and 10 users placed in random initial locations within the cells. The traffic model adopted for the simulations is the full-buffer due to it simplicity as the number of users in a cell is constant and the buffers of the users have always unlimited data to transmit [34]. Users in the network use a random mover mobility scheme with a constant speed of 0.833m/s. Each cell has a radius of 577.33m. The transmission scheme adopted in the simulator is the time division duplex (TDD) which uses a single carrier frequency of 3.5GHz, total bandwidth of 36MHz and a subcarrier spacing of 30KHz which is used to separate the transmissions and receptions apart by multiplexing on a time basis. The parameters discussed in this section are also presented in Table 4.1. Note that to generate multiple networks in the simulator we set a range of seeds in order to generate copies of the network with different parameters that involves randomness.

parameter	value
total no cells	3
cell radius	577.33m
bandwidth	36MHz
carrier frequency	3.5GHz
subcarrier spacing	30KHz
traffic model	full buffer
total no users	10
user mobility	random
user speed	0.833m/s (fixed)

Table 4.1: Simulation parameters of the radio network.

4.4.2 Training the Reinforcement Learning Models

The RL models developed in this work are included in the RLlib library which is supported in Python programming language. However, the simulator is written in a different language and thus an interface for interaction with external environments such as the simulator, was deployed as well to establish the communication between agent-environment in order to share states, actions, and rewards. For both the models used in this work (*i.e.,* DQN and PPO), the training data are generated from 2000 radio networks (*i.e.,* 2000 seeds), where each network is simulated for 0.5s which is a sufficient duration to train RL models for the link adaptation problem, since the action period is set to 0.005s which we call herein as time interval or sampling period. This configuration for training results in a total of 200,000 data samples.

The hyperparameters used for the trainining process of the DQN, are listed in Table 4.2.

parameter	value
discount factor	0.7
target net. update frequency	500 (iterations)
time instances per iteration	200
batch size	16
learning rate	0.0005
optimizer	Adam
exploration max	1
exploration min	0.02

Table 4.2: Hyperparameters for the DQN model.

The input layer of the DQN is of size 4 which corresponds to the size of the state vector. The input layer is connected with two hidden layers of size 256 and a rectified linear unit (ReLU) as activation function for each hidden layer. Finally, the output layer is of size 5 which corresponds to the cardinality of the action space. In addition, we use a discount factor γ =0.7 which corresponds to a reasonable balance between the agent's action and the future reward. In other words, we need to give some weighting for cases where an action may affect future rewards. This happens in cases where the adjustment of the Δ^{RL} does not affect only the immediate throughput but also the future trend of the throughput due to the nature of the time-varying channel conditions. In addition, at each training iteration, a batch of experiences is sampled from the replay memory to update the weights of the DQN, using the Adam optimizer firstly introduced in [35] with a learning rate of 0.0005.

The hyperparameters used in the experiments for the PPO are listed in Table 4.3. It is important to mention that using the explicit adjustment on the Δ^{RL} using the action space $\mathcal{A}^2$, the models were not able to be trained appropriately and thus we do not consider this action space for the next chapters.

Following the aforementioned configurations of the simulator and the RL models, we run the training phase to capture the mean cumulative episode reward over all the agents, shown in Figure 4.3. Note that the mean cumulative reward shown in Figure 4.3, corresponds to the normalized value of the throughput over the fixed bandwidth.

We can see that both the RL models increase the mean cumulative reward from about 110 to a converged value of about 125. Note that these results were obtained from the configurations above, although several experiments with different hyperparameters were performed. The lighter line colors for

parameter	value
discount factor	0.8
learning rate	0.0004
optimizer	Adam
entropy coefficient	0.01
batch size	200
clipping (ϵ)	0.2

Table 4.3: Hyperparameters for the PPO model.

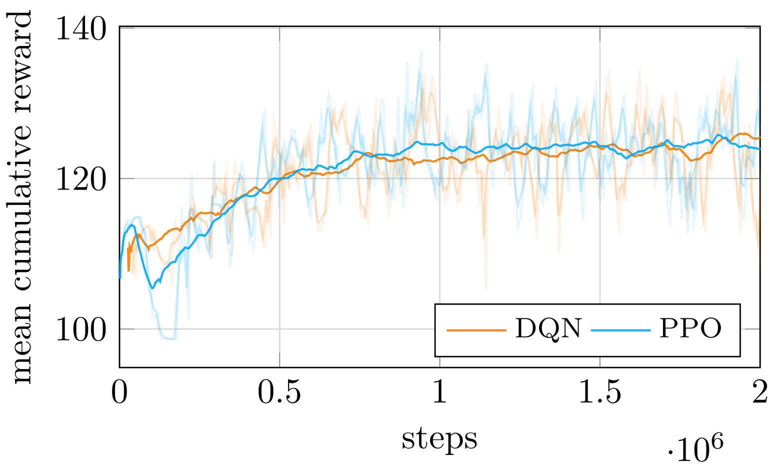

Figure 4.3: Training plots. Mean cumulative episode reward over all agents.

both models in the Figure 4.3 illustrate different weights of moving averaging of the mean cumulative reward, while the bold ones represent a moving averaging filter with 0.95 weighting.

Chapter 5

Results

In this section we mainly present and analyse the results from the inference phase of the RL agents. However, firstly we present some results using the basic configuration of the radio network without any RL-agent involved, in order to get a clear picture on different factors that affect the decisions made not only using the OLLA algorithm but also using the RL-based algorithms.

Figure 5.1, Figure 5.2, and Figure 5.3 depict the median CQI index, the average CQI index, the average downlink throughput respectively for an increasing number of users. Note that in these experiments we keep the same radio network configuration as mentioned in the previous chapter, while we change the number of users to plot the behavior of each individual measure with respect to the number of users. It can be seen that when the number of users in the network increases, the median and mean value of the CQI of the users decreases accordingly. Recalling Equation 2.2, this happens because by adding users to the network, more interference is generated between the cells which affects the SINR and consequently the CQI indices since it corresponds to a quantized version of the SINR. In addition to this, the average throughput is also affected and more specifically it decreases with an increasing number of users in the network.

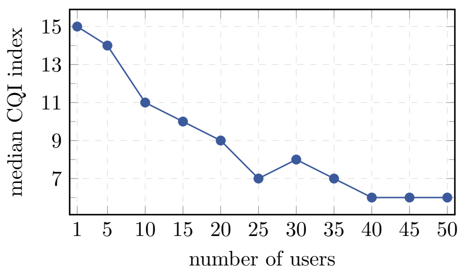

Figure 5.1: Median CQI index with respect to the number of users.

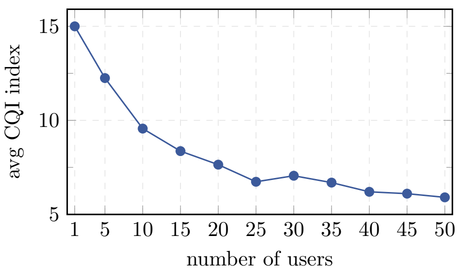

Figure 5.2: Average CQI with respect to the number of users.

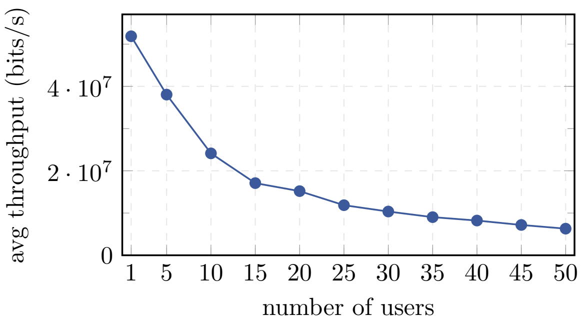

Figure 5.3: Mean HARQ throughput with respect to the number of users.

Since the variability of the CQI measurement is not only affected by the number of users but with their speed as well, we illustrate this behavior in Figure 5.4. In this figure, the lines with the markers correspond to the mean CQI index of the 10 users of the radio network, while the error bars represent the standard deviation. In particular, it can be observed that when users (*i.e.,* UEs) move in space with higher speed, the CQI not only decreases, but it also fluctuates more. This can be seen with the error bars as they are of higher value for the case where the users move with higher speed (*i.e.,* 20m/s).

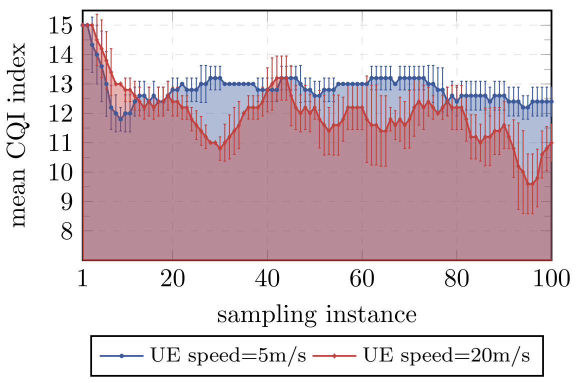

Figure 5.4: CQI of users with random mobility and different speeds.

5.1 Link Adaptation

For the link adaptation experiments we keep the same basic network configuration as mentioned in Section 4.4, although the experiements' duration was set to 2s for 200 different radio networks (*i.e.,* seeds). Recall that the action period is set to 0.005s for the RL-based methods (*i.e.,* DQN, and PPO) while for OLLA the action is sent aperiodically when a HARQ-ACK feedback is available. Note that, through the whole duration of the inference phase, the agents do not update the weights of the neural networks, and the reward signals are not considered. It is also important to mention that during the inference phase, we use seeds different from the ones used in the training phase, in order to evaluate the system's performance with scenarios that were not experienced before during the training phase.

Figure 5.5 depicts the cumulative distribution function (CDF) of the downlink throughput using both the state-of-the art approaches, as well as the RL-based ones. To evaluate our proposed approaches we compare not only with OLLA tuned at 10% BLER target, but also with a BLER target of 90%. The reasoning behind this is that several studies, [36], [37], showed that changing the BLER target would be beneficial for different radio conditions in 5G-NR. However, changing the BLER target dynamically does not imply higher throughput in any case, since as it can be seen from Figure 5.5, when OLLA is tuned for 90% BLER, the cell-center users suffer with low throughput comparing to the RL-based algorithms.

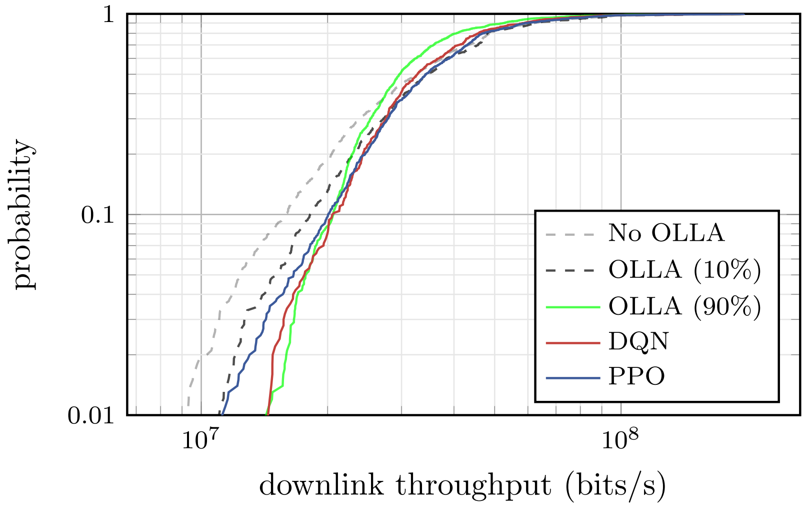

Figure 5.5: Cumulative distribution function of the downlink throughput.

In Figure 5.5 we can see directly that the performance in terms of throughput for the whole network using no outer loop link adaptation (No OLLA) is low, although for cell-center users the performance is relatively close to the other methods. That is logical since when the channel conditions are good, link adaptation techniques are not necessary to improve the estimation of the SINR. Note that higher probabilities in the CDF plot correspond to cell-center users while lower probabilities correspond to cell-edge users. Next we can see that the RL-based methods (*i.e.*, DQN and PPO) improves the average downlink throughput for cell-edge users since the curves lie to the right part of the figure for probabilities lower than 30%. For almost the same set of cell-edge users (*i.e.,* lower than 20%), the OLLA algorithm tuned at 90% target BLER, the throughput is improved as well. However, when OLLA is tuned for 90% target BLER, the mid-cell and cell-center users suffer with low values of throughput. In contrast, when OLLA is tuned for 10% target BLER, the average user throughput is relatively high for mid-cell and cell-center users, although it is low for cell-edge users. In regards to the RL-based methods, both of them increase the average user throughput for mid-cell and cell-edge users while they also maintain a sufficiently high throughput for cell-center users. We can also see that DQN performs slightly better than PPO at the low probabilities of the distribution.

For the same experiments, we present some numerical results with different statistical measures in Table 5.1. In particular we put an emphasis on the mid-cell and cell-edge users since for these particular groups of users link adaptation is needed.

measure	5^{th} percentile	30^{th} percentile	50^{th} percentile
OLLA (10%) / No OLLA	21.065%	13.663%	6.695%
DQN / No OLLA	41.613%	15.434%	0.556%
DQN / OLLA (10%)	16.973%	1.558%	-5.751%
PPO / No OLLA	33.210%	16.506%	5.402%
PPO / OLLA (10%)	10.032%	2.501%	-1.211%
PPO / DQN	-5.934%	0.929%	4.817%

Table 5.1: Throughput gains.

We can see that indeed RL-based methods increase the average user throughput of the network in 5^{th} and 30^{th} percentile comparing to the state-of-the-art OLLA mechanism. In particular, DQN gains 16.973% and 1.558% increase on the average user throughput comparing to the OLLA tuned at 10% BLER target, for the aforementioned percentiles. For the same percentiles, PPO gains 10.032% and 2.501% over the OLLA tuned at 10% BLER target. However, at the 50^{th} percentile, we observe that both DQN and PPO has negative gains of -5.751% and -1.211% respectively.

Although RL-based methods manage to improve the mid-cell and cell-edge users' downlink throughput, there is a sacrifice of its mean. This is depicted in Table 5.2. Although DQN and PPO improves the mean throughput compared to the No OLLA and OLLA tuned at 90% BLER target, both RL-based methods have a lower throughput than OLLA tuned at 10% BLER target in average. However, it is important to mention that trading-off resources from cell-center users to provide resources to mid-cell and cell-edge

measure	mean
No OLLA	3.7213E7
OLLA (10% BLER)	3.9452E7
OLLA (90% BLER)	3.4267E7
DQN	3.7768E7
PPO	3.8823E7

Table 5.2: Mean downlink throughput.

users is the main objective of link adaptation techniques.

Chapter 6

Conclusions and Future Work

In this work we focus on the problem of downlink link adaptation in 5G-NR networks using reinforcement learning techniques. These techniques were studied, designed and developed in Ericsson's system-level radio network simulator, by interfacing with RLlib models written in Python language. In particular, a DQN and a PPO model were trained online using data generated from the radio network simulator to solve the problem of downlink link adaptation. During the training phase, several state vector representations with different measurements logged by the radio network simulator were tried in order to optimize the learning procedures. The objective of these models was to train a generic agent that maximizes the downlink throughput for all users in the radio network. Both methods were evaluated and compared to the current state-of-the-art algorithm for outerloop downlink link adaptation called OLLA.

The results have shown a significant increment of the downlink throughput for cell-edge and mid-cell users. In addition, it is important to mention that using the RL-based methods, an important constraint that OLLA algorithms were depended on (*i.e.,* BLER target) can be eliminated and thus the whole link adaptation mechanism for 5G-NR networks does not depend on predefined fixed parameters. This is an extremely important step that enables more flexible link adaptation schemes without any kind of tuning for certain scenarios. In other words, using the proposed RL-based methods, tuning *a priori* for different BLER targets is not needed even with time-varying channel conditions. In addition to this, another important factor of this work is that the framework developed maintains the minimal radio network intervention, which is important for future developments in real-life applications.

6.0.1 Future Work

This work can be extended in several aspects. First and most important is to design and extend this work by means of selecting directly an MCS index based on the feedback from the receiver without any inner functions for estimating the SINR at the transmitter side. Although this work considered the problem of link adaptation by adjusting the SINR estimate at the transmitter side, it could be more effective to neglect the current state-of-the-art inner loop and outer loop link adaptation, and instead use a reinforcement learning agent to learn a policy to select the most appropriate MCS index for a certain environment state.

In addition, since the scheduling and link adaptation mechanism affect each other's decision, a potential extension would be the co-design of these two mechanisms within a reinforcement learning context for optimal radio resource management. Another extension of this work would be the consideration and design of other reward functions such as the spectral efficiency which would embed valuable information of the scheduled resources as well. Finally, keeping similar configuration as in this work, other approaches such as multi-armed bandits (MAB) or more specifically unimodal multi-armed bandits [33] where the reward is a unimodal function over partially ordered arms which correspond to MCS indices.

Appendix A

Tables

CQI index	modulation	code rate $\times$ 1024	efficiency
0	out of range	out of range	out of range
1	QPSK	78	0.1523
2	QPSK	193	0.3770
3	QPSK	449	0.8770
4	16QAM	378	1.4766
5	16QAM	490	1.9141
6	16QAM	616	2.4063
7	64QAM	466	2.7305
8	64QAM	567	3.3223
9	64QAM	666	3.9023
10	64QAM	772	4.5234
11	64QAM	873	5.1152
12	256QAM	711	5.5547
13	256QAM	797	6.2266
14	256QAM	885	6.9141
15	256QAM	948	7.4063

Table A.1: CQI indices (4-bit).

modulation scheme	number of bits (Q)
$\pi/2$-BPSK, BPSK	1
QPSK	2
16 QAM	4
64 QAM	6
256 QAM	8

Table A.2: Modulation schemes.

MCS index I_{MCS}	modulation (Q_m)	target code rate $\times$ 1024 (R)	spectral efficiency
0	2	120	0.2344
1	2	157	0.3066
2	2	193	0.3770
3	2	251	0.4902
4	2	308	0.6016
5	2	379	0.7402
6	2	449	0.8770
7	2	526	1.0273
8	2	602	1.1758
9	2	679	1.3262
10	4	340	1.3281
11	4	378	1.4766
12	4	434	1.6953
13	4	490	1.9141
14	4	553	2.1602
15	4	616	2.4063
16	4	658	2.5703
17	6	438	2.5664
18	6	466	2.7305
19	6	517	3.0293
20	6	567	3.3223
21	6	616	3.6094
22	6	666	3.9023
23	6	719	4.2129
24	6	772	4.5234
25	6	822	4.8164
26	6	873	5.1152
27	6	910	5.3320
28	6	948	5.5547

Table A.3: MCS index table for PDSCH.

Bibliography

[1] Mykhailo Klymash et al. "Method for optimal use of 4G/5G heterogeneous network resourses under M2M/IoT traffic growth conditions". In: *2017 International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo)*. IEEE. 2017, pp. 1–5.

[2] Eric Liang et al. "RLlib: Abstractions for distributed reinforcement learning". In: *International Conference on Machine Learning*. 2018, pp. 3053–3062.

[3] Cheng-Xiang Wang et al. "Cellular architecture and key technologies for 5G wireless communication networks". In: *IEEE communications magazine* 52.2 (2014), pp. 122–130.

[4] Erik Dahlman, Stefan Parkvall, and Johan Skold. *5G NR: The next generation wireless access technology*. Academic Press, 2018.

[5] Patrick Marsch et al. "5G radio access network architecture: Design guidelines and key considerations". In: *IEEE Communications Magazine* 54.11 (2016), pp. 24–32.

[6] Amitabha Ghosh et al. "5G evolution: A view on 5G cellular technology beyond 3GPP release 15". In: *IEEE Access* 7 (2019), pp. 127639–127651.

[7] Gwanmo Ku and John MacLaren Walsh. "Resource allocation and link adaptation in LTE and LTE advanced: A tutorial". In: *IEEE communications surveys & tutorials* 17.3 (2014), pp. 1605–1633.

[8] Seong Taek Chung and Andrea J Goldsmith. "Degrees of freedom in adaptive modulation: a unified view". In: *IEEE Transactions on Communications* 49.9 (2001), pp. 1561–1571.

[9] Wikipedia contributors. *Signal-to-interference-plus-noise ratio — Wikipedia, The Free Encyclopedia*. [Online; accessed 11-March-2020]. 2019. URL: `https://en.wikipedia.org/w/index.php?title=Signal-to-interference-plus-noise_ratio&oldid=920301448`.

[10] Sushruth N Donthi and Neelesh B Mehta. “An accurate model for EESM and its application to analysis of CQI feedback schemes and scheduling in LTE”. In: *IEEE Transactions on Wireless Communications* 10.10 (2011), pp. 3436–3448.

[11] 3GPP. *5G; NR; Physical layer procedures for data*. Technical Specification (TS) 38.214. Version 15.2.0. 3rd Generation Partnership Project (3GPP), July 2018. URL: `https://www.etsi.org/deliver/etsi_ts/138200_138299/138214/15.02.00_60/ts_138214v150200p.pdf`.

[12] Charles Wang, Dean Sklar, and Diana Johnson. “Forward error-correction coding”. In: *Crosslink* 3.1 (2001), pp. 26–29.

[13] Nils Strodthoff et al. “Enhanced machine learning techniques for early HARQ feedback prediction in 5G”. In: *IEEE Journal on Selected Areas in Communications* 37.11 (2019), pp. 2573–2587.

[14] Pierre Bertrand, Jing Jiang, and Anthony Ekpenyong. “Link adaptation control in LTE uplink”. In: *2012 IEEE Vehicular Technology Conference (VTC Fall)*. IEEE. 2012, pp. 1–5.

[15] 3GPP. *5G; NR; Physical layer procedures for data*. Technical Specification (TS) 38.214. Version 15.3.0. 3rd Generation Partnership Project (3GPP), Oct. 2018. URL: `https://www.etsi.org/deliver/etsi_ts/138200_138299/138214/15.03.00_60/ts_138214v150300p.pdf`.

[16] Ashwin Sampath, P Sarath Kumar, and Jack M Holtzman. “On setting reverse link target SIR in a CDMA system”. In: *IEEE 47th Vehicular Technology Conference*. Vol. 2. IEEE. 1997, pp. 929–933.

[17] Klaus I Pedersen et al. “Frequency domain scheduling for OFDMA with limited and noisy channel feedback”. In: *2007 IEEE 66th Vehicular Technology Conference*. IEEE. 2007, pp. 1792–1796.

[18] Víctor Buenestado et al. “Analysis of throughput performance statistics for benchmarking LTE networks”. In: *IEEE Communications letters* 18.9 (2014), pp. 1607–1610.

[19] A Duran et al. “Self-optimization algorithm for outer loop link adaptation in LTE”. In: *IEEE Communications letters* 19.11 (2015), pp. 2005–2008.

[20] Ramón A Delgado et al. “Fast convergence outer loop link adaptation with infrequent updates in steady state”. In: *IEEE 86th Vehicular Technology Conference (VTC-Fall)*. IEEE. 2017, pp. 1–5.

[21] Vidit Saxena et al. "Contextual Multi-Armed Bandits for Link Adaptation in Cellular Networks". In: *Proceedings of the 2019 Workshop on Network Meets AI & ML*. 2019, pp. 44–49.

[22] Mateus P Mota et al. "Adaptive Modulation and Coding based on Reinforcement Learning for 5G Networks". In: *arXiv preprint arXiv:1912.04030* (2019).

[23] Francisco Blanquez-Casado et al. "eOLLA: an enhanced outer loop link adaptation for cellular networks". In: *EURASIP Journal on Wireless Communications and Networking* 2016.1 (2016), p. 20.

[24] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018.

[25] Yuxi Li. "Deep reinforcement learning: An overview". In: *arXiv preprint arXiv:1701.07274* (2017).

[26] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[27] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[28] John N Tsitsiklis and Benjamin Van Roy. "Analysis of temporal-diffference learning with function approximation". In: *Advances in neural information processing systems*. 1997, pp. 1075–1081.

[29] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge, 1998.

[30] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[31] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[32] Michael N Katehakis and Arthur F Veinott Jr. "The multi-armed bandit problem: decomposition and computation". In: *Mathematics of Operations Research* 12.2 (1987), pp. 262–268.

[33] Richard Combes and Alexandre Proutiere. "Unimodal bandits: Regret lower bounds and optimal algorithms". In: *International Conference on Machine Learning*. 2014, pp. 521–529.

[34] Pablo Ameigeiras et al. "Traffic models impact on OFDMA scheduling design". In: *EURASIP Journal on Wireless Communications and Networking* 2012.1 (2012), p. 61.

[35] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[36] Ahmed Al Amin et al. "Analysis of modulation and coding scheme for 5th generation wireless communication system". In: *2016 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE. 2016, pp. 1545–1549.

[37] Romano Fantacci et al. "Adaptive modulation and coding techniques for OFDMA systems". In: *IEEE Transactions on Wireless Communications* 8.9 (2009), pp. 4876–4883.

TRITA-EECS-EX-2020:909