Sorbonne University
Doctoral School of Informatics, Telecommunications and Electronics of Paris
EURECOM

On enforcing Network Slicing in the new generation of Radio Access Networks

Presented by **Karim Boutiba**

Dissertation for Doctor of Philosophy in Information and Communication Engineering

Directed by **Prof. Adlen Ksentini**

The Jury committee is composed of:

Prof. Philippe Martins	(Télécom Paris)	Reviewer
Prof. Marco Fiore	(IMDEA, Spain)	Reviewer
Prof. Géraldine Texier	(IMT Atlantique)	Examiner
Dr. Bessem Sayadi	(Nokia Bell Labs, France)	Examiner
Prof. Jérôme Harri	(Eurecom)	Jury President
Prof. Adlen Ksentini	(Eurecom)	Thesis Director

Abstract

The emerging 5G networks and beyond promise to support new advanced use cases such as the immersive holographic communication, the Internet of Skills, and the 4D Interactive mapping [1]. These use cases require stringent requirements in terms of Quality of Service (QoS), such as low latency, high Downlink (DL)/Uplink (UL) data rates and low energy consumption. The 3rd Generation Partnership Project (3GPP) specifications introduced many features in 5G New Radio (NR) to improve the physical efficiency of 5G to meet the stringent and heterogeneous requirements of the beyond 5G use cases. Among the key 5G NR features, wemention the numerology, the BandWidth Part (BWP), the dynamic Time Duplex Division (TDD) and the Connected-mode Discontinuous Reception (C-DRX). However, the 3GPP specifications does not provide how to configure the next Generation Node B (gNB)/User Equipment (UE) in order to optimize the use of the 5G NR features. We configure the latter by applying Machine Learning (ML), particularly Deep Reinforcement Learning (DRL), to fill this gap. Indeed, Artificial Intelligence (AI)/ML is playing a vital role in communications and networking [2] thanks to its ability to provide self-configuring and self-optimizing networks.

In this thesis, we propose novel solutions to enable a smart and autonomous configuration of the Radio Access Network (RAN). We divided the solutions into three different parts. The first part concerns RAN slicing featuring numerology and BWPs. In contrast, the second part tackles dynamic TDD, and the last part goes through different RAN optimizations to support Ultra-Reliable and Low-Latency Communication (URLLC) services.

In the first part, we propose two contributions. First, we introduce NRflex, a RAN slicing framework aligned with Open RAN (O-RAN) architecture. NRflex dynamically assigns BWPs to the running slices and their associated UEs, aiming to fulfill the slices' required QoS. Simulation results show the superiority of NRflex to meet network slice requirements while optimizing the 5G RAN resources compared to other existing solutions. Then, we model the RAN slicing problem as a Mixed-Integer Linear Programming (MILP) problem. To our best knowledge, this is the first MILP modeling of the radio resource management featuring network slicing, taking into account (i) Mixed-numerology, (ii) both latency and throughput requirements (iii) multiple slices attach per UE (iv) Inter-Numerology Interference (INI). After showing that solving the problem takes an exponential time, we consider a new approach to solve it in a polynomial time, which is highly required when scheduling radio resources. The new approach consists of formalizing this problem using a DRL-based solver. We evaluate the use of DRL to solve the problem for different network configurations and compare its performance with the optimal solution obtained by solving the MILP problem.

In the second part of this thesis, we propose a DRL-based solution to enable dynamic TDD in a single 5G NR cell. The proposed DRL algorithm monitors UL and DL buffers at the gNB to derive the optimal TDD pattern concerning the current traffic configuration. Simulation results demonstrate the efficiency of the proposed algorithm to avoid buffer overflow and ensure generality by reacting to traffic pattern changes. The solution is implemented in OAI and tested using real UEs. Then, we extend the solution by leveraging Multi-Agent Deep Reinforcement Learning (MADRL) to support multiple cells, considering cross-link interference between cells. Based on the simulation results, the algorithm effectively prevents buffer overflows, avoids cross-link interference, and adapts to changes in the traffic pattern, ensuring its versatility. We compare

our solution with the optimal solution and different static TDD configurations. We found that our solution outperforms the static TDD configurations. We finally discuss the algorithm's limitations in terms of the number of cells, traffic variance, and cross-link interference probability.

In the last part, we propose three different solutions to optimize the RAN to support URLLC services. First, we propose a two-step ML-based solution to predict Radio Link Failure (RLF). We combine Long Short-Term Memory (LSTM) and Support Vector Machine (SVM) algorithms to find the correlation between radio measurements and RLF. The RLF prediction model was trained with real data obtained from a 5G testbed. The validation process of the model showed an accuracy of 98% when predicting the connection status (i.e., RLF). In the second contribution, we propose a DRL-based solution to reduce UL latency. Our solution dynamically allocates the future UL grant by learning from the dynamic traffic pattern. Simulation results demonstrate the efficiency of the proposed algorithm to reduce the UL latency down to 1 ms and ensure generality by reacting to different traffic patterns. In the last contribution, we introduce a DRL-based solution to balance latency and energy consumption by jointly deriving the C-DRX parameters and the BWP configuration. Simulation results demonstrate the effectiveness of the proposed methodology in reducing the PC (i.e., 50-95% power gain) while avoiding latency overflow for a different number of connected UEs (i.e., 1 to 20 UEs).

Résumé

Les réseaux 5G émergents et au-delà promettent de prendre en charge de nouveaux cas d'utilisation tels que la communication holographique immersive, l'internet des compétences et la cartographie interactive 4D [1]. Ces cas d'usage ont des exigences strictes en termes de Quality of Service (QoS), telles qu'une faible latence, un débit Downlink (DL)/Uplink (UL) élevé, ainsi qu'une faible consommation d'énergie. Les spécifications du groupe de normalisation 3GPP ont introduit de nombreuses fonctionnalités aux système radio 5G (5G NR), dans le but d'améliorer l'efficacité de la 5G et de répondre aux exigences strictes et hétérogènes des services de la 5G et au-delà. Parmi les principales fonctionnalités de la 5G NR, on peut citer la numérologie, les BandWidth Parts (BWPs), le multiplexage temporel (TDD) dynamique et la Connected-mode Discontinuous Reception (C-DRX). Toutefois, les spécifications 3GPP n'indiquent pas comment configurer la next Generation Node B (gNB)/User Equipment (UE) afin d'optimiser l'utilisation des fonctionnalités 5G NR. Afin de combler ce manque, nous proposons de nouvelles solutions qui mettent en œuvre des fonctionnalités 5G NR en appliquant les techniques de l'apprentissage automatique ou Machine Learning (ML), en particulier l'apprentissage profond par renforcement ou Deep Reinforcement Learning (DRL). En effet, les outils de l'intelligence artificielle jouent un rôle essentiel dans l'optimisation des systèmes de communication et des réseaux [2] grâce à leurs capacités à rendre le réseau capable de s'auto-configurer et s'auto-optimiser.

Dans cette thèse, nous proposons plusieurs solutions pour permettre une configuration intelligente du réseau d'accès radio (RAN). Nous avons divisé les solutions en trois parties distinctes. La première partie couvre le découpage du RAN en tranches (ou slicing en tirant parti des concepts de numerologie et de BWP. La deuxièmes partie traite le TDD dynamique. Tandis que la dernière partie passe en revue les plusieurs optimisations du RAN pour prendre en charge les services Ultra-Reliable and Low-Latency Communication (URLLC).

Dans la première partie, nous proposons deux contributions. Tout d'abord, nous présentons NRflex, une solution de découpage du RAN en tranches (ou slicing), aligné sur l'architecture Open RAN (O-RAN). NRflex attribue dynamiquement des BWP aux tranches en cours d'exécution et aux UEs qui leur sont associées afin de satisfaire la QoS requise par ces tranches. Les résultats de la simulation ont démontré la supériorité de NRflex par rapport à d'autres solutions à répondre aux exigences des tranches de réseau tout en optimisant les ressources radio. Par la suite, nous modélisons le problème de découpage du RAN en tranches comme un problème Mixed-Integer Linear Programming (MILP). Au meilleur de notre connaissance, il s'agit de la première modélisation MILP de la gestion des ressources radio avec découpage du réseau en tenant en compte (i) de la numérologie mixte, (ii) des exigences en matière de latence et de débit (iii) de l'association de plusieurs tranches par UE (iv) de Inter-Numerology Interference (INI). Après avoir montré que la résolution du problème prend un temps exponentiel, nous avons introduit une nouvelle approche pour le résoudre en un temps polynomial, ce qui est très important pour la fonction de l'ordonnancement (scheduling) des ressources radio. La nouvelle approche consiste à formaliser et résoudre ce problème par le biais l'apprentissage par renforcement profond (DRL). Nous évaluons l'utilisation de DRL pour résoudre le problème pour différentes configurations de réseau et comparons sa performance avec la solution optimale obtenue en résolvant le problème MILP.

Dans la deuxième partie de la thèse, nous proposons une solution basée sur le DRL pour permettre un TDD dynamique dans une seule cellule 5G NR. L'algorithme DRL proposé surveille les files de trafic UL et DL au niveau de la gNB afin de dériver l'affectation des resources UL/DL optimal en fonction du trafic. Les résultats de la simulation ont démontré l'efficacité de l'algorithme proposé pour éviter le débordement des files d'attente et garantir la généralité en réagissant aux changements du trafic. La solution a été implémentée dans la plateforme OpenAirInterface (OAI) et testée avec des UEs réels. Nous avons ensuite étendu la solution, en tirant parti de l'algorithme Multi-Agent Deep Reinforcement Learning (MADRL), pour prendre en charge plusieurs cellules en tenant compte de l'interférence radio entre les liaisons transversales entre les cellules. D'après les résultats de la simulation, l'algorithme évite efficacement les débordements des files d'attente ainsi que les interférences entre les liaisons. Aussi, l'algorithme proposé s'adapte aux changements de la dynamique du trafic, ce qui garantit sa polyvalence. Nous avons comparé notre solution à la solution optimale et à diverses configurations TDD statiques. Nous avons constaté que notre solution est plus performante que les configurations TDD statiques. Enfin, nous discutons des limites de l'algorithme en termes de nombre de cellules, de variance du trafic et de probabilité d'interférence entre les liaisons.

Dans la dernière partie de la thèse, nous avons proposé trois solutions pour optimiser le RAN afin de prendre en charge les services URLLC. Tout d'abord, nous avons proposé une solution en deux étapes basées sur l'apprentissage automatique pour prédire les coupures du lien radio ou Radio Link Failure (RLF). Nous avons combiné les mémoires à long terme (LSTM) et la machine é vecteur de support (SVM) pour trouver la corrélation entre les mesures radio et le RLF. Le modèle de prédiction RLF a été entraîné avec des données réelles obtenues à partir d'un banc d'essai 5G. Le processus de validation du modèle a montré une précision de 98% lors de la prédiction de l'état de la connexion. Dans la deuxième contribution, nous avons proposé une solution basée sur le DRL pour réduire la latence UL. Notre solution alloue (prédit) dynamiquement les futurs besoins en ressource radio du UL en apprenant du modèle de trafic. Les résultats de la simulation démontrent clairement l'efficacité de l'algorithme proposé pour réduire la latence UL jusqu'à 1 ms et garantir la généralité en réagissant à different types de modèles de trafic. Dans la dernière contribution, nous introduisons une solution basée sur le DRL afin d'équilibrer la latence et la consommation d'énergie en calculant conjointement les paramètres C-DRX et la configuration BWP. Les résultats de la simulation démontrent l'efficacité de la méthode proposée pour réduire la consommation énergétique (c-à-d., un gain de puissance de 50 a 95%) tout en évitant le dépassement de la latence pour un nombre different d'UEs connectés (c-à-d., de 1 a 20 UE) à la gNB.

Remerciements

Je voudrais exprimer ma gratitude à mon superviseur, Prof. Adlen Ksentini. Sa vision cristalline des problèmes m'a aidé à faire mes premiers pas dans le monde de la recherche, et son soutien constant m'a offert des opportunités dont je suis extrêmement reconnaissant.

Je tiens également à remercier EURECOM et les grands chercheurs avec lesquels j'ai eu la chance de collaborer. En particulier, Raymond Knopp et Sagar Arora du département des systèmes de communication et Bagaa Miloud de l'université UQTR.

Je n'aurais pas pu arriver à ce stade de la soutenance de ma thèse sans un soutien bien huilé qui a été présent dès mes premières années à l'école et, ensuite, tout au long de mes études à l'université (ESI, Alger). Je suis extrêmement reconnaissant pour le soutien constant, l'encouragement, l'amour et la générosité de ma famille : ma mère bien-aimée, ma merveilleuse Saghouche, mon père, mes frères, mes tantes.

Je voudrais exprimer ma sincère reconnaissance à mes amis : Abdelkader, Mohamed, Sofiane, Abdelrahmane, Giulio, etc. Vous avez joué un rôle important dans mon parcours académique et dans la réalisation de cette thèse.

Contents

List of Figures

List of Tables

Acronyms

3GPP 3rd Generation Partnership Project.

5G Fifth wireless Generation.

5GC 5G Core.

6G Sixth wireless Generation.

ACKTR Actor-Critic using Kronecker-Factored Trust Region.

aDS always Dynamic Scheduling.

AI Artificial Intelligence.

AM Acknowledged Mode.

AMF Access and Mobility Management Function.

ANN Artificial Neural Network.

AR Augmented Reality.

BSR Buffer Status Report.

BWP BandWidth Part.

C-DRX Connected-mode Discontinuous Reception.

CAPEX CAPital EXpenditures.

CDF Cumulative Distribution Function.

CN Core Network.

CP Control Plane.

CQI Channel Quality Indicator.

CRC Cyclic Redundancy Check.

CU Centralized Unit.

D2D Device-to-Device.

DCI Downlink Control Indicator.

DDPG Deep Deterministic Policy Gradient.

DG Data Grant.

DL Downlink.

DQN Deep Q-Network.

DRB Data Radio Bearer.

DRL Deep Reinforcement Learning.

DRL-LLS DRL-based Low Latency Scheduler.

DRP DRL-based 5G RAN TDD Pattern.

DS Dynamic Scheduling.

DU Distributed Unit.

eMBB enhanced Mobile BroadBand.

eNB evolved Node B.

ETSI European Telecommunications Standards Institute.

FDD Frequency Division Duplexing.

FR1 Frequency Range 1.

FR2 Frequency Range 2.

GF Grant Free.

gNB next Generation Node B.

GSMA Global System for Mobile Communications Association.

GTP GPRS Tunneling Protocol.

HARQ Hybrid Automatic Repeat Request.

HMI Human-Machine Interface.

IAT Inter-Arrival Time.

IE Information Element.

INI Inter-Numerology Interference.

IoT Internet Of Things.

IP Internet Protocol.

ITU International Telecommunication Union.

JCAS Joint Communication And Sensing.

KPI Key Performances Indicator.

LC Logical Channel.

LCM Life-Cycle Management.

LD Lagrangian Duality.

LP Linear Programming.

LSTM Long Short Term Memory.

LTE Long Term Evaluation.

M2M Machine-to-Machine.

MAC Medium Access Channel.

MADDPG Multi-Agent Deep Deterministic Policy Gradient.

MADRL Multi-Agent Deep Reinforcement Learning.

MADRL Multi-Agent Deep Reinforcement Learning (MADRL)-based 5G RAN TDD Pattern.

MCS Modulation and Coding Scheme.

MDP Markov Decision Process.

MEC Multi-access Edge Computing.

MILP Mixed-Integer Linear Programming.

MIMO Multiple Input and Multiple Output.

ML Machine Learning.

mMTC massive Machine-Type Communication.

mmW millimeter Waves.

MSE Mean Square Error.

NF Network Function.

NGAP Next Generation Application Protocol.

NGMN Next Generation Mobile Networks.

NLP Natural Language Processing.

NN Neural Networks.

NR New Radio.

NSSAI Network Slice Selection Assistance Information.

O-RAN Open RAN.

OAI OpenAirInterface.

OFDM Orthogonal Frequency-Division Multiplexing.

OFDMA Orthogonal Frequency-Division Multiple Access.

OPEX OPerating EXpenses.

PC Power Consumption.

PDCCH Physical Downlink Control Channel.

PDCP Packet Data Convergence Protocol.

PDU Packet Data Unit.

PHR Power HeadRoom.

PHY PHysical Layer.

PLMN Public Land Mobile Network.

PRB Physical Resource Block.

PUCCH Physical Uplink Control Channel.

PUSCH Physical Uplink Shared Channel.

QoE Quality of Experience.

QoS Quality of Service.

RAN Radio Access Network.

RIC RAN Intelligent Controller.

RL Reinforcement Learning.

RLC Radio Link Control.

RLF Radio Link Failure.

RMSE Root Mean Square Error.

RNIS Radio Network Information Service.

RNN Recurrent Neural Network.

RRC Radio Resource Control.

RS Reference Signals.

RSRP Reference Signal Receive Power.

RSRQ Reference Signal Receive Quality.

RT Real Time.

RU Radio Unit.

SBA Service-Based Architecture.

SCS Sub-Carrier Spacing.

SCTP Stream Control Transmission Protocol.

SD Slice Differentiator.

SDAP Service Data Adaptation Protocol.

SDN Software-Defined Network.

SLA Service Level Agreement.

SMF Session Management Function.

SO Slice Orchestrator.

SON Self-Organizing Network.

SPS Semi-Persistent scheduling.

SR Scheduling Request.

SST Slice/Service Type.

SVM Support Victor Machine.

TBS Transport Block Size.

TDD Time Duplex Division.

TTI Transmission Time Interval.

UAV Unmanned Aerial Vehicle.

UE User Equipment.

UL Uplink.

UM Unacknowledged Mode.

UP User Plane.

UPF User Plane Function.

URLLC Ultra-Reliable and Low-Latency Communication.

VM Virtual Machine.

VR Virtual Reality.

XnAP Xn Application Protocol.

XR eXtended Reality.

Chapter 1

General Introduction

1.1 Context

Fifth wireless Generation (5G) networks are designed to support variant network services, mainly organized into three categories: Ultra-Reliable and Low-Latency Communication (URLLC), enhanced Mobile BroadBand (eMBB), and massive Machine-Type Communication (mMTC). URLLC supports use cases that require high network reliability, more than 99.999%, and extremely low latency of approximately 1 millisecond for data transmission. For example, autonomous driving requires a very low latency to make decisions, as there is such high risk involved. eMBB supports high bandwidth applications such as Augmented Reality (AR)/Virtual Reality (VR) and streaming, providing higher Downlink (DL)/Uplink (UL) throughput and improved user experiences. mMTC allows connecting a large number of users while fulfilling their quality of services.

For the year 2030 and beyond, Our daily lives will be augmented by ultra-high speed and ultra-reliable wireless connections, native Artificial Intelligence (AI), and advanced sensing technologies. Even though 5G is still not fully exploited commercially, the research community started to define the basis of Sixth wireless Generation. Many of the features planned for 6G leverage 5G features, such as dynamic Time Duplex Division (TDD), 5G New Radio (NR) features, and 5G power-saving techniques. Five major categories of usage scenarios, depicted in Figure 1.1, have been defined, of which eMBB+, URLLC+, and mMTC+ are extensions and combinations of the usage scenarios defined in 5G, while sensing and AI are two new usage scenarios that will flourish in the era of 6G [3].

eMBB for human-centric communication use cases has evolved into eMBB+. In eXtended Reality (XR) applications, it will enable extremely immersive experiences and multi-sensory interactions. Throughput and connection density requirements for eMBB+ are going to be much higher. Moreover, 6G will accelerate the full-scale digital transformation of vertical industries. URLLC+ is the next step in the evolution of URLLC for essential machine communication in Industry 4.0 and beyond. It also applies to new applications made possible by the widespread use of robotics, Unmanned Aerial Vehicle (UAV), and new Human-Machine Interface (HMI) in manufacturing, public service, autonomous driving, and household management. 6G will continue the journey started by 5G to connect everything, but it will do so with a wider variety of devices, new HMI, and a higher density of connections. mMTC+ is the continuous evolution of mMTC, which is characterized by the massive number of lightly connected devices with sporadic traffic in smart cities, healthcare, buildings, transportation, manufacturing, and agriculture. A

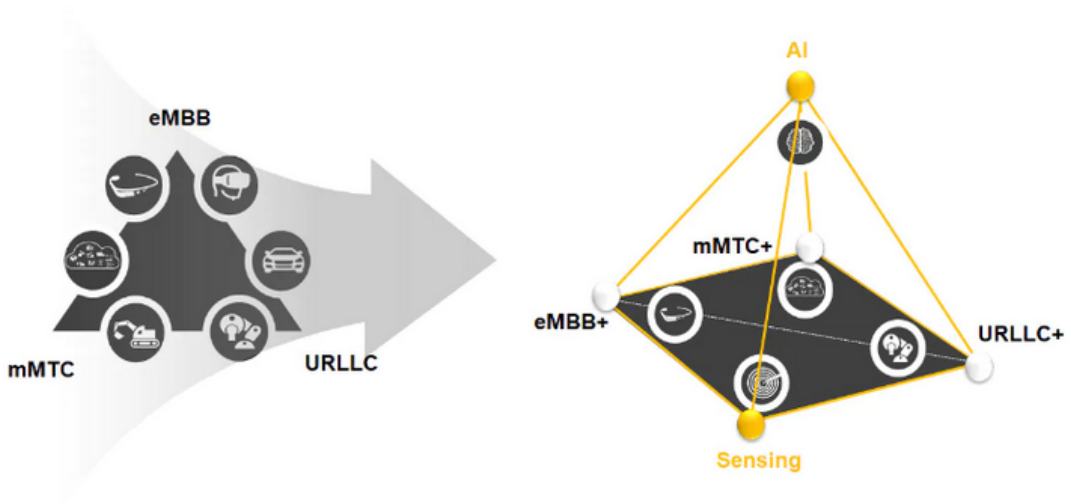


Figure 1.1: 6G new services [3]

key requirement is for sensors to have a long lifetime. Green and sustainable development is the core requirement and ultimate goal of network and terminal designs in 6G. By introducing the green design concept and native AI capability, 6G aims to improve the overall energy efficiency by 100 times across the network while also ensuring optimal service performance and experience. To fulfill these different requirements, the use of AI becomes very important to optimize the network, especially the Radio Access Network (RAN), since it is a very dynamic environment and requires very low execution latency, at the range of milliseconds.

1.2 Motivation

At the present time, AI technologies have matured and are now considered stable, with state-of-the-art techniques being used to solve many types of hard problems. The latter are problems that take an exponential time to be solved optimally. Among AI techniques, Machine Learning (ML) techniques excel in situations where there is inherent randomness and non-determinism. Indeed, ML is able to capture patterns and correlations in available data, sometimes based on very complex inputs. Besides, the execution of AI does not take a huge amount of time compared to solving hard problems using traditional optimization methods such as the Simplex algorithm [4]. In these situations, ML techniques can be trained on available data, which allows them to learn, represent, and predict the inherent behavior in the data. In the case of the unavailability of data, Reinforcement Learning (RL) techniques can be used to train models following the trial and error concept. The idea is to build models that learn by trying different solutions, and according to the response of the environment, the model will be able to take good actions.

In the telecom domain, the increasing relevance of AI in several use cases has been observed for some years now [5]. There are multiple drivers for the adoption of AI-based solutions in the telecom industry. The 3rd Generation Partnership Project (3GPP) specifications of 5G and 5G Advanced adopt AI to improve network performance and enable intelligent network automation. AI applications under consideration are primarily in the RAN, including network energy saving, load balancing, and mobility optimization. Potential use cases in the RAN are

so numerous that a small subset has been selected for study in the upcoming 3GPP Release 18 [6], including Channel State Information (CSI) feedback, beam management, and positioning. It is important to note that 3GPP is not developing AI/ML models. Instead, it seeks to create common frameworks and evaluation methods for AI/ML models being added to different RAN functions. Outside of the 3GPP, Open RAN (O-RAN) ALLIANCE [7], an alliance that gathers telcos together under one vision, "making the RAN open", explores how AI/ML can improve the network. For example, O-RAN ALLIANCE has a unique feature in its architecture called the RAN Intelligent Controller (RIC), designed to host AI/ML optimization applications. The RIC can host xApps, which run in near real-time, and rApps, which run in non-real-time. xApps for improving spectral and energy efficiency and rApps for network orchestration that leverage AI already exist today [8]. More xApps/rApps and applications using AI / ML in the RIC will become available as the O-RAN ecosystem grows and matures.

1.3 Thesis Challenges and Contributions

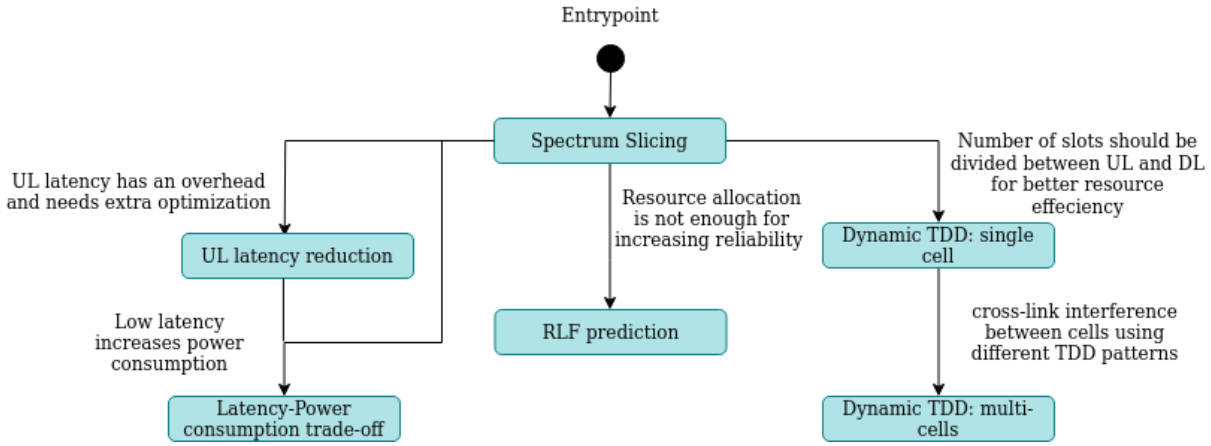


Figure 1.2: Thesis story map

This Thesis is part of the transition from 5G to 6G. It uses the actual 5G New Radio (NR) features to enhance the RAN functions, enabling more efficient, AI-powered beyond 5G networks toward 6G. The thesis started with the problem of spectrum slicing. The latter becomes more challenging with the use of 5G NR features such as numerology and BandWidth Part (BWP). In this context, we proposed a complete problem formulation of resource allocation taking into account a mixed numerology environment and BWPs. The problem is formulated as a Mixed-Integer Linear Programming (MILP) problem. We proved that the problem is NP-hard. Then, we proposed a DRL-based solution to solve it in linear time. Although, smart and efficient allocation of resources fulfills different service requirements in terms of throughput and latency, reliability is a challenging problem when the users are out of coverage or experience a Radio Link Failure (RLF). Consequently, the next station of the thesis is RLF prediction using ML to increase the network reliability. Indeed, once the network predicts an RLF, it takes actions to avoid it either by triggering a handover or sending a notification to the service application that handles and mitigates the RLF at the application level (e.g., changing drones' trajectory in the case of drone control application). The proposed spectrum-slicing solutions work in both UL and DL directions. However, the UL direction has more challenges to tackle, especially with the latency. Indeed, the network is not aware of the quantity of data that the user wants to send

(i.e., in the UL), which makes it difficult to allocate resources efficiently. For this purpose, we tackle the problem of UL latency reduction by predicting the UL grant of users using DRL. But, by reducing the latency, the UEs consume more energy. To reduce the UEs' power consumption, we proposed a DRL-based solution to find a trade-off between power consumption and latency. Coming back to the spectrum slicing solutions, we noticed that even if applied to both UL and DL direction, resources can be better allocated if the number of UL and DL slots are optimized according to the traffic ratio (UL/DL). For this purpose, we enabled dynamic TDD using DRL in a single cell first, then in multi-cells scenario by mitigating cross-link interference between neighboring cells. The proposed solutions in this thesis are used to develop automated control systems for beyond 5G networks, enabling Self-Organizing Network (SON) and AI-powered 6G networks. Figure 1.2 summarizes my PhD story using a mind map.

In this thesis, we propose AI/ML and DRL methods to solve complex problems at the RAN. We tackled the following problems:

1. **Enforcing RAN slicing with mixed numerology and multiple BWPs:**

   (a) *Problem description:* Network slicing is considered as one of the critical enablers of 5G to support a wide range of applications and use cases with different requirements. Although network slicing has been widely studied and solutions have started to appear, particularly at the transport and core network parts, slicing (sharing) the RAN resources is still challenging. Indeed, with the new features introduced by the 5G NR, further investigation is needed to enforce network slicing at the RAN. 5G NR introduces several new features that can be beneficial for slicing the RAN [9]. Among these new features, we may mention the concept of BWPs. The latter aims to enable flexible assignment and configuration of Physical Resource Block (PRB). BWPs are subsets of contiguous PRBs allocated per User Equipment (UE), i.e., the UE expects to use resources only in a specific part of the bandwidth. Besides, 5G NR introduces the concept of numerology, which uses a specific physical layer configuration, mainly Sub-Carrier Spacing (SCS). Unlike 4G, where all the time-slots have the same duration (1 ms), 5G NR, by adapting SCS, allows reducing the slot-time duration down to 125 microseconds, which can considerably reduce the RAN latency. Consequently, each BWP has its own numerology, enabling more efficient sharing of the spectrum among the heterogeneous services in 5G RAN and, hence, among slices.

   (b) *Proposed solutions:* We introduce NRflex, a novel framework that dynamically enforces RAN slices in 5G, relying on the concept of BWPs. NRflex addresses the joint PRBs scheduling and allocation problem with mixed numerology in 5G NR, in order to meet the latency requirement of URLLC services while considering the other type of services (mainly eMBB). NRflex redefines the Life-Cycle Management (LCM) of the RAN slices by leveraging on the O-RAN architecture [7]. NRflex operates under a dedicated bandwidth for each numerology. After that, we address the challenging problem of resource management in 5G NR featuring network slicing by first modeling it as a mixed-integer linear program taking into account *i*) multiple numerology in the same bandwidth while avoiding the Inter-Numerology Interference (INI); *ii*) multiple slices attached per UE; *iii*) different throughput and latency requirements per slice. Then, we prove that the problem is NP-hard; i.e., solving it will take a considerable amount of time, which is not tolerable when scheduling radio resources in real time. Meanwhile, Reinforcement Learning (RL) that can be seen as a learning heuristic search strategy when it is applied to optimization problems. Accordingly,

we formalize this problem in the RL framework and propose a Deep Reinforcement Learning (DRL)-based approach. The approach aims to select the numerology to be used and the number of resources allocated per UE at each time slot during a time window while taking into account the channel quality of the UE. Specifically, we designed the DRL scheduler to be independent of the number of UEs in the system. We have modeled the DRL state to make the solution scalable for larger bandwidths covering both Frequency Range 1 (FR1) and Frequency Range 2 (FR2) frequency bands. The latter with a bandwidth up to 400 MHz, which corresponds to the usage of the millimeter Waves (mmW) bands.

(c) *Publications:*

NRflex: Enforcing network slicing in 5G New Radio
Boutiba, Karim; Ksentini, Adlen; Brik, Bouziane; Challal, Yacine; Balla, Amar
Computer Communications, 13 October 2021, Elsevier

Radio resource management in multi-numerology 5G new radio featuring network slicing
Boutiba, Karim; Bagaa, Miloud; Ksentini, Adlen
ICC 2022, IEEE 1st International Workshop on Semantic Communications, 16-20 May 2022, Seoul, South Korea

Optimal radio resource management in 5G NR featuring network slicing
Boutiba, Karim ; Bagaa, Miloud; Ksentini, Adlen
Computer Networks, Vol. 234, October 2023, Elsevier

2. **Dynamic TDD in a single 5G cell:**

(a) *Problem description:* Emerging 5G and 6G network services are shifting from DL-dominant traffic to more equilibrate DL and UL traffic, even to more UL-dominant traffic [10]. Indeed, emerging network services such as AR and VR applications generate high UL traffic corresponding to offloaded intensive computation to be run at a remote application sitting at the Edge. Another example is the high-quality video streaming captured by drones for building surveillance that requires more UL traffic than DL. One solution to accommodate this new trend in terms of traffic model is TDD. TDD allows using the entire bandwidth by dividing it into time slots where some are assigned to UL and some to DL. Long Term Evaluation (LTE) or 4G proposed 7 different configurations of the TDD frame, which allows configuring the evolved Node B (eNB) according to the traffic patterns. But, one concern with this solution is that the configuration is fixed in time and cannot be adapted to the traffic dynamic, i.e., if at a certain moment of time, more DL or UL need to be accommodated, then there is no possibility to increase the number of UL or DL slots, without rebooting the eNB. To overcome this issue, 5G NR introduces a more flexible solution, where the number of UL and DL slots in the TDD frame can be changed dynamically. This flexibility will allow the next Generation Node B (gNB) to adapt to the frame configuration according to the traffic pattern by selecting the number of slots dedicated to UL and DL. However, the 5G NR specifications only cover the mechanism allowing the gNB to inform the UE about the UL/DL slots pattern in a TDD frame, leaving the algorithm deriving the pattern UL/DL opens.

(b) *Proposed solution:* we fill this gap by proposing a novel algorithm, namely DRL-based 5G RAN TDD Pattern (DRP), which allows deriving the UL/DL pattern of TDD frames dynamically and accommodating cell traffic, whatever it is DL or UL dominant. DRP monitors UL and DL traffic periodically and derives the optimal pattern. DRP uses the Buffer Status Report (BSR) sent by UEs for the UL traffic and the state of the radio bearer channel queues at gNB, which avoids having an exact pattern of the traffic. DRP is particularly efficient for 5G private network deployment, allowing to deploy of gNB in a plug-and-play mode.

(c) *Publication:*

On using deep reinforcement learning to dynamically derive 5G new radio TDD pattern
Bagaa, Miloud; Boutiba, Karim; Ksentini, Adlen
GLOBECOM 2021, IEEE Global Communications Conference, 7-11 December 2021, Madrid, Spain

3. **Dynamic TDD with cross-link interference:**

(a) *Problem description:* In a multi-cell environment, dynamic TDD is more challenging. Indeed, whereas in a single-cell environment, the only challenge was to find the UL/DL ratio without knowing the traffic pattern, in a multi-cell environment, the challenge is twofold: (*i*) Finding the UL/DL ratio without knowing the traffic pattern; (*ii*) Mitigating cross-link interference. The latter is defined as interference that occurs when one gNB transmits while another receives in the same frequency band (i.e., two neighboring gNBs that use a different TDD pattern). This usually occurs within the same operator using the same frequency band for its gNBs.

(b) *Proposed solution:* we extend DRP to solve the dynamic TDD problem in a multi-cell environment and introduce the Multi-Agent Deep Reinforcement Learning (MADRL)-based 5G RAN TDD Pattern (MADRL) framework. The MADRP approach is fully decentralized, with each MADRP agent located close to the gNB serving a particular cell. Compared to a centralized approach, MADRP is executed close to the gNB, which reduces the control latency between the gNB and MADRP. In addition, it reduces the signaling overhead normally generated when gNBs send data to a central entity. Each MADRP agent monitors the gNB's UL and DL buffers and the number of edge users with neighboring cells. It then uses this local observation along with messages from neighboring cells to derive the optimal TDD pattern to accommodate connected users while avoiding cross-link interference with neighboring cells.

(c) *Publication:*

Multi-agent deep reinforcement learning to enable dynamic TDD in a multi-cell environment
Boutiba, Karim; Bagaa, Miloud; Ksentini, Adlen
IEEE Transactions on Mobile Computing, 15 September 2023

4. **Radio link failure prediction:**

(a) *Problem description:* RLF is an important criterion in modern wireless networks, including 5G. RLF is critical for services requiring high reliability, which is one of the key features of 5G, such as those supported by URLLC class[11]. RLF impacts principally network services that involve high-mobile devices, such as vehicles and flying drones. Indeed, combining highly mobile users at high altitude like UAV [12], with the current limited number of deployed 5G cells and their small coverage due to wider frequency bands' usage [13] (which make the radio link between the UE and the base station more susceptible to blockage and degradation leading to sudden interruption of the communication link), may yield to frequent RLF. RLF corresponds to a disconnection from the network, which strongly impacts sensitive services, such as UAV safety, where RLF increases the network latency and, in the worst cases, leads to UAV's losses and collisions. In this context, it is important that 5G tackles the RLF issue to improve reliability by, for instance, predicting when a mobile device has a high probability of seeing a RLF and taking the appropriate action, like anticipating the Handover or updating the trajectory of the UAVs.

(b) *Proposed solution:* RLF can be estimated by measuring radio key indicators such as Reference Signal Receive Power (RSRP), Reference Signal Receive Quality (RSRQ), Channel Quality Indicator (CQI), and Power HeadRoom (PHR). By combining this information, it is possible to know the quality of the radio channel, hence the probability of RLF. However, deriving a closed-form of the RLF distribution probability is very challenging. To fill this gap, we propose to use ML to predict the RLF relying on the measurement of RSRP, RSRQ, CQI, and PHR. More specifically, we used a two-step algorithm based on Long Short Term Memory (LSTM) and Support Victor Machine (SVM). The first layer (i.e., the set of LSTMs) is used to predict the future values of RSRQ, CQI and PHR. Then, the SVM layer is used to classify the measurement values (i.e., the predicted RSRQ, CQI and PHR) into connectivity status (i.e., connected or not). The output of the ML model will be used as input by third-tier applications to improve the reliability of 5G networks. Therefore, we will propose two applications that rely on the RLF prediction model: (1) an application that optimizes the handover in the context of O-RAN [14]; (2) an application that adapts UAVs trajectory to avoid RLF in the context of the European Telecommunications Standards Institute (ETSI) Multi-access Edge Computing (MEC) framework [15].

(c) *Publication:*

Radio link failure prediction in 5G networks
Boutiba, Karim; Bagaa, Miloud; Ksentini, Adlen
GLOBECOM 2021, IEEE Global Communications Conference, 7-11 December 2021, Madrid, Spain

5. **UL latency reduction:**

(a) *Problem description:* Emerging URLLC use cases are shifting from DL-dominant traffic to a more equilibrated DL and UL traffic. Hence, it is crucial to ensure a low latency in UL and DL directions. However, achieving low latency in UL is more challenging since the gNB has to be aware of the UE's queue status to allocate PRBs according to the UE needs, which results in increasing the latency by the signaling overhead. Semi-Persistent scheduling (SPS) [16] and Grant Free (GF) [17] scheduling methods are proposed to overcome the signaling overhead, hence reducing the

UL latency. However, they are less efficient for dynamic traffic patterns (dynamic inter-arrivals and dynamic data sizes) and less adaptable to quick channel condition changes, especially in FR2 frequencies (> 6Ghz) and high mobility users. Furthermore, GF may impact the reliability of the services since resource usage can collide in frequency between users.

(b) *Proposed solution:* To overcome the aforementioned challenges, we introduce a novel DRL-based algorithm that reduces the UL latency by predicting the subsequent arrival of data and its size. We dubbed this solution DRL-based Low Latency Scheduler (DRL-LLS). The latter monitors the BSR sent by UEs and derives the future inter-arrival time and the grant size that should be allocated to the user in the next UL opportunity. DRL-LLS is adapted to changes happening in channel conditions very quickly since it dynamically sends grants to the user before the data arrival. Hence, when the data is ready to be sent, the UE will find the grant and fully transmit all the data to gNB, which decreases the overhead signaling latency while adapting to the channel conditions.

(c) *Publication:*

On using deep reinforcement learning to reduce uplink latency for URLLC services
Boutiba, Karim; Bagaa, Miloud; Ksentini, Adlen
GLOBECOM 2022, IEEE Global Communications Conference, 4-8 December 2022, Rio de Janeiro, Brazil

6. **Power consumption and latency trade-off:**

(a) *Problem description:* 5G NR has introduced several features to increase throughput and decrease latency, mainly numerology, Multiple Input and Multiple Output (MIMO) and higher bandwidths. These features increase Power Consumption (PC) for both the network infrastructure and the UE. The latter, which are typically powered by a limited battery, can suffer from poor Quality of Experience (QoE) [18] due to the rapid discharge of the battery. The communications industry should, therefore develop strategies to optimize the energy efficiency of 5G networks without compromising Quality of Service (QoS) [19]. Leading wireless equipment vendors have begun studying UE power-saving schemes in 5G NR [20]. They have investigated several techniques to reduce PC, such as Connected-mode Discontinuous Reception (C-DRX) with a new set of parameters introduced by 5G NR and BWP adaptation. C-DRX allows UEs to periodically enter a sleep state during which the physical layer functions of UEs become inactive, and thus PC is reduced. However, the latency may increase as UEs may be in the sleep state when the data arrives at the gNB. On the other hand, BWP adaptation consists of reducing the BWP size based on the demand of UEs, which decreases the PC when UEs do not ask for high data rates. The 3GPP study, conducted in [20], shows the impact of C-DRX parameters and BWP adaptation on network performance, mainly on latency and power gain. However, the 5G NR standard does not provide solutions to derive the C-DRX parameters and the BWP configuration dynamically.

(b) *Proposed solution:* To address this shortcoming, we introduce a DRL-based solution, called DRL-based Latency and Power optimizer (DRL-LP), to jointly derive the C-DRX parameters and the BWP configuration. The proposed solution is designed to

be scalable in terms of the number of UEs and traffic patterns. Indeed, The DRL agent observes UEs' history of: (i) the experienced latency; (ii) the buffer status, and (iii) the number of scheduled UEs; during a time window. Then, for each UE, the DRL agent sets the C-DRX parameters and the BWP size for the next time window. To the best of our knowledge, no prior work has combined C-DRX and BWP adaptation to reduce the PC further while ensuring low latency.

(c) *Publication:*

On using deep reinforcement learning to balance power consumption and latency in 5G NR
Boutiba, Karim; Ksentini, Adlen
ICC 2023, IEEE International Conference on Communications, 28 May-1 June 2023, Rome, Italy

1.4 Thesis Structure

This thesis is structured as follows: Chapter 2 is dedicated to the exploration of the state of the art on several concepts crucial to the research work of this thesis, which can be mainly grouped into two topics: 5G NR features and ML/RL techniques. Then, we divided the Thesis into four Parts: In Part 1, we introduce the contributions related to Network Slicing in 5G NR. Part 1 consists of two chapters: In Chapter 3, we introduce NRFlex, while in Chapter 4, we introduce the optimal solution and the DRL-based solution. In Part 2, we present the contributions related to Dynamic TDD in 5G NR. We divided Part 2 into two chapters: In Chapter 5, we introduce our contribution regarding Dynamic TDD in a single cell environment, while in Chapter 6, we introduce our contribution of Dynamic TDD in multi-cells environments. In Part 3, we present the RAN optimizations for URLLC: In Chapter 7, we introduce the RLF prediction contribution while in Chapters 8 and 9, we present the UL latency reduction and the power-latency trade-off solutions, respectively. Finally, Part 4 contains two chapters: Chapter 10 gives a general conclusion while Chapter 11 introduces the Thesis perspectives.

Chapter 2

State of The Art

2.1 Introduction

5G wireless technology is the next major evolution of cellular networks. It offers significant improvements over previous generations in terms of throughput, latency, and connection density. 5G is revolutionizing many industries and applications, including telecommunications, healthcare, transportation, manufacturing, and entertainment. ML is playing a crucial role in communication and networking [2] with the ability to provide a self-configured and self-optimized network which is a key characteristic of 6G networks. Particularly, DRL is a lightweight framework that enables quick decisions and hence takes real-time actions in the network characterized by its dynamicity and needs fast decisions. This chapter provides an outline of 5G Networks and the features of the 5G NR air interface. After that, it introduces the ML/RL techniques used during this thesis to optimize the RAN leveraging the 5G NR features.

2.2 5G Networks

5G networks are designed to provide higher data rates, lower latency, increased connection density, and improved reliability compared to their predecessors. The architecture of the 5G Networks is shown in Figure 2.1. It consists of the Core Network (CN), called the 5G Core (5GC), and the RAN.
The RAN consists of base stations and UEs. The base stations, previously called eNB in 4G LTE, are called gNB in 5G NR. Their primary role is to handle all radio-related functions, including the provision of wireless connectivity to the UEs. The CN is designed with a service-oriented architecture called Service-Based Architecture (SBA). The latter divides the CN network function into multiple logical entities called Network Function (NF). 3GPP defines the interfaces between NFs to support interoperability between different NF vendors. The NFs can be combined to create slice-specific CNs, either being dedicated or shared among slices. The 5G CN also follows the Software-Defined Network (SDN) concept. Indeed, some NFs are dedicated to Control Plane (CP) functions, whereas one NF is dedicated to User Plane (UP). This NF is called the User Plane Function (UPF). It is responsible for data forwarding. It acts as a gateway between the RAN and external data networks, such as the internet or the edge network, and performs packet forwarding and QoS handling. The latter is achieved through one or more Packet Data Unit (PDU) sessions per UE. Each PDU session consists of one or many QoS flows, allowing the overall network to perform differentiated packet treatment. When Internet Protocol (IP) packets arrive to the UPF, the latter marks them according to the QoS flow requirements and forwards them to the RAN. The latter will map the QoS flows to Data Radio Bearer (DRB),

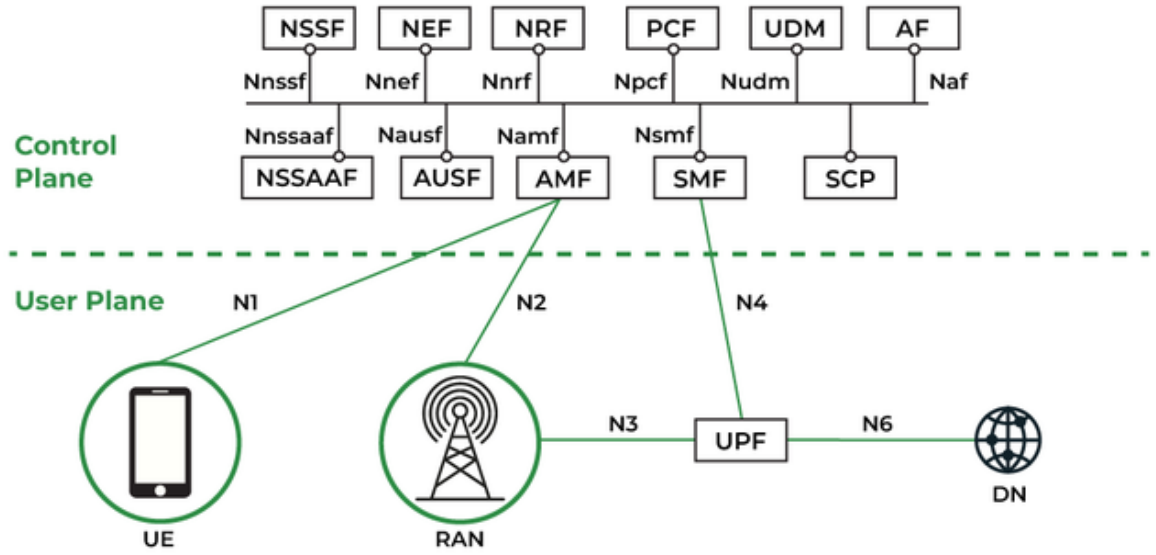


Figure 2.1: 5G Networks architecture [21]

which are logically separated entities at the RAN common between gNB and UE (i.e., both UE and gNB can identify which DRB the data belongs to). The other NFs are dedicated to CP, mainly the Access and Mobility Management Function (AMF), responsible of UE's registration and mobility, and the Session Management Function (SMF), responsible for session management, including IP address assignment. We will not dig into the details of other NFs since it is not the scope of this thesis. The reader may refer to [22] for more details about the 5GC.

2.3 5G New Radio (NR)

5G NR is the air interface or radio access technology that is at the core of the 5G wireless communication system. 5G NR is designed to provide the wireless connectivity required for 5G networks, which offer higher data throughput, lower latency, and improved performance compared to their 4G predecessors. The following sub-chapters will describe the main features of the 5G NR air interface.

2.3.1 NR stack

The 5G NR protocols stack is depicted in Figure 2.2. The protocol stack is the same for gNB and UE. However, the functions executed at each layer can be different depending on whether it is at the UE or the gNB. The 3GPP standards clearly define the functions of UE and gNB at each layer. The functionalities of the different layers can be divided into a CP and a UP. Moreover, The layers are divided into two logical entities: (i) Centralized Unit (CU); (ii) Distributed Unit (DU). The CU is divided into CU-CP and DU-CP. The CU-CP contains the Radio Resource Control (RRC) and the CP function of the Packet Data Convergence Protocol (PDCP) while the CU-UP contains the Service Data Adaptation Protocol (SDAP) and the UP of the PDCP functions (PDCP-U). The RRC layer is responsible for broadcasting system information, management of UE connections, security, mobility, or measurement reporting in support of mobility functions. It also forwards CP messages to the AMF via the Next Generation Application Protocol (NGAP) protocol. The NGAP protocol provides control plane signalling between the RAN

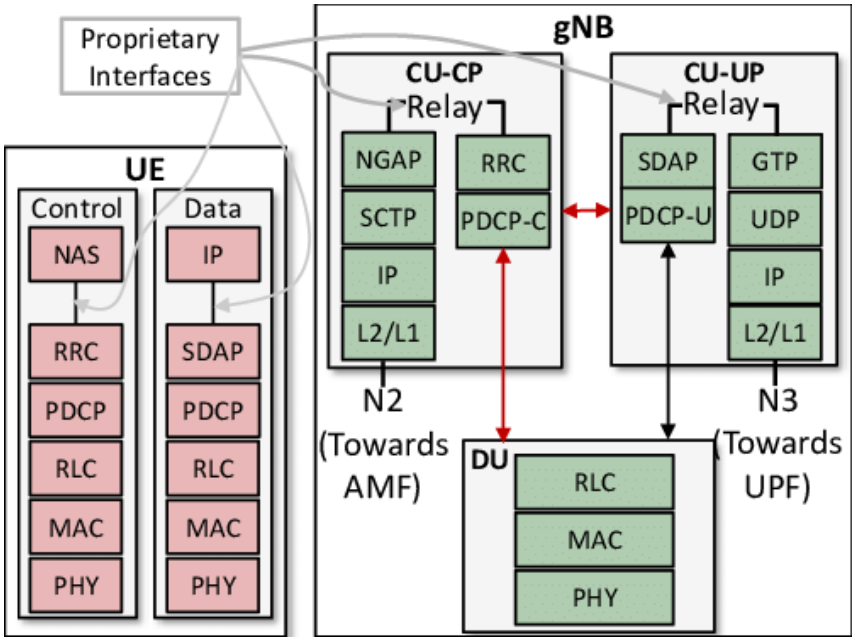


Figure 2.2: 5G NR protocol stack [23]

nodes and the AMF over the Stream Control Transmission Protocol (SCTP) transport protocol. The PDCP-U transfers user data, which relies on sequence numbering for retransmission and reordering purposes, header compression, and ciphering. The PDCP-C notably performs integrity protection and transports control data. The SDAP is responsible for the mapping from QoS flows to radio bearers and the marking of the corresponding packets in UL and DL. In the UE, the mapping might be configured explicitly, or reflective QoS is applied, where the UE applies the same mapping in UL as it appeared in the DL. The GPRS Tunneling Protocol (GTP) layer encapsulates/decapsulates user data by adding GTP headers and forwarding the UL data to UPF and SDAP in case of DL data. The DU contains the Radio Link Control (RLC), MAC and the PHysical Layer (PHY) layers. The RLC layer buffers data received from the PDCP until it can be sent on the wireless link. It furthermore segments data packets that cannot be sent at once, and performs retransmissions if configured in the Acknowledged Mode (AM) as opposed to the Unacknowledged Mode (UM). The MAC layer manages the wireless link by assigning resources to UEs by means of scheduling. This involves priority handling between multiple UEs through dynamic scheduling and priority handling between a UE's radio bearers by prioritization. Through the corresponding scheduling decisions, it multiplexes the data of a UE into transport blocks that are sent over the air. Errors in transmissions to/from a UE are handled by the Hybrid Automatic Repeat Request (HARQ) protocol, which triggers retransmissions of data. Finally, the PHY encodes data to be sent over the antenna, following the commands of the MAC. This involves (de-)coding and (de-)modulation of data, mapping to antennas, etc.

2.3.2 Physical Layer features

Radio resources

At the PHY layer, "radio resources" refer to the part of the spectrum, time, and space that are allocated to support the transmission of data and communication between devices (i.e., UEs and gNBs). Efficient management and allocation of radio resources are crucial to ensure the optimal performance of 5G networks. The unit of allocation of radio resources is the PRB. The

latter exists in both 4G Long Term Evolution LTE and 5G NR air interfaces. A PRB is defined by a specific amount of frequency spectrum (in Hertz) and a duration of time (in milliseconds). The LTE PRB (Physical Resource Block) is made up of 12 consecutive subcarriers (with a fixed SCS of 15 KHz) for a duration of one slot (i.e., 0.5 ms). In contrast, the 5G NR PRB only defines the frequency domain, i.e., 1 PRB = 12 subcarriers. The spacing between the subcarriers and the time slot duration are based on the Numerology, which will be introduced in section 2.3.2. It should be noted that the PRB's shape in 5G NR with numerology 0 is similar to a 4G LTE PRB as depicted in Figure 2.3. The time slot duration is divided into 14 Orthogonal Frequency-Division Multiplexing (OFDM) symbols. To summarize, a PRB is a two-dimensional object that takes 12 subcarriers in frequency and 14 OFDM symbols in time. PRBs are used to carry data in both the UL and DL directions. In the DL, the gNB allocates PRBs to UEs for receiving data. In the UL, UEs transmit data on assigned PRBs. Both 4G LTE and 5G NR use the Orthogonal Frequency-Division Multiple Access (OFDMA) as the modulation and access scheme. OFDMA allows multiple UEs to share the same frequency band and time period by dividing the available spectrum into PRBs, each serving a single UE or group of UEs. This division enhances the efficiency and capacity of the network. A PRB is allocated to a UE via signaling through the Downlink Control Indicator (DCI). The latter provides the UE with the necessary information, such as the resource allocation dedicated to the UE in UL or/and DL. The DCI is sent to the UE on the Physical Downlink Control Channel (PDCCH) channel.

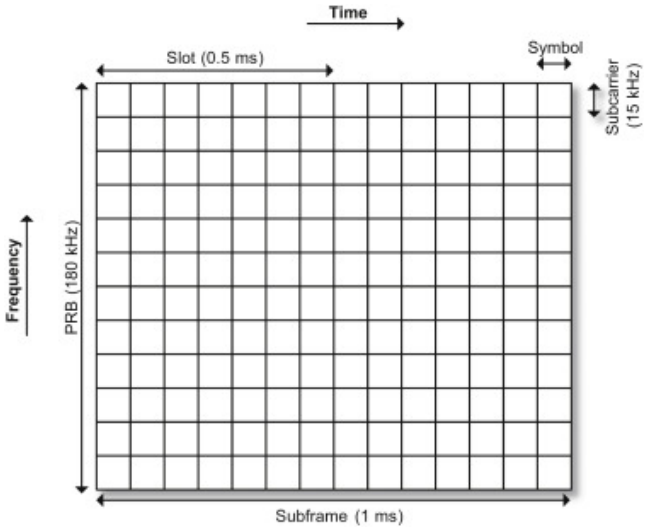


Figure 2.3: Physical Resource Block in numerology 0 (similar to 4G LTE)

The amount of data carried by a PRB is computed according to the Modulation and Coding Scheme (MCS), number of MIMO layers, and the code rate (i.e., the portion of real data without the Cyclic Redundancy Check (CRC) attachment) [24]. Network operators and gNBs dynamically allocate PRBs to UEs based on factors like the UE's QoS requirements, traffic demand, and channel conditions. This allocation is part of the overall radio resource management in cellular networks, performed by the 5G scheduler, which is the main intelligence of the gNB.

Numerology

The Numerology in 5G refers to the SCS that is used to transmit data. The SCS is the distance between two adjacent subcarriers, which are the individual frequencies that are used to transmit data. 5G NR supports a variety of numerologies, which gives it the flexibility to be tailored

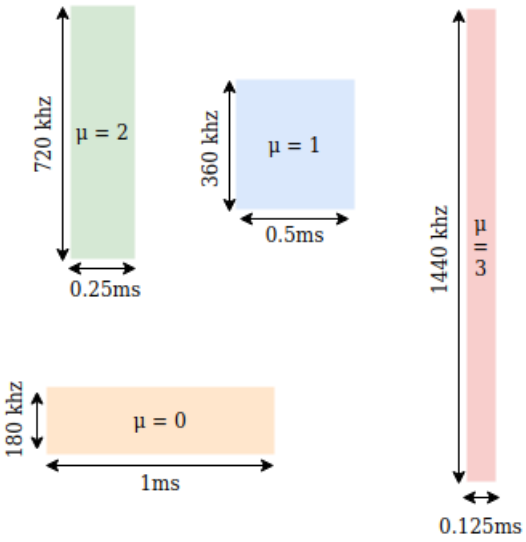


Figure 2.4: Numerology illustration

to different types of devices and applications. For example, lower numerologies can be used to achieve longer range and lower latency, while higher numerologies can be used to achieve higher data rates. 5G NR numerologies come to make radio resource allocation more flexible. Indeed, 5G NR numerologies reshape radio units in time and frequency. It reduces the Transmission Time Interval (TTI) to 2, 4, 8, 16 times smaller than the 4G's 1 ms. Figure 2.4 illustrates 4 PRBs with different numerology.
In 5G NR, each numerology μ is defined by a SCS, and a cyclic prefix [25]. 5G NR Release-15 [26] specifies five main numerologies (μ) and defines a SCS of $15 * 2^{\mu}$ kHz and a slot duration of $1/2^{\mu}$ ms, allowing to reduce the access latency considerably at the RAN. The following table shows the different numerologies that are supported by 5G NR. Numerology is a powerful tool that allows 5G NR to be tailored to the specific needs of different devices and applications. This makes 5G a versatile and adaptable technology that can be used for a wide range of purposes.

BandWidth Part (BWP)

In 5G wireless communication systems, "BWP" stands for Bandwidth Part. A BWP is a concept introduced to optimize and manage the allocation of available spectrum within the 5G network. BWPs are subsets of contiguous PRBs allocated per UE, i.e., the UE expects to use resources only in a specific part of the bandwidth. It is a way to divide and configure the available spectrum to meet specific communication requirements and adapt to various use cases efficiently. BWP allows mobile network operators to tailor their networks to the specific needs of their UEs and applications. For example, an operator can use a small BWP to support a low-power Internet Of Things (IoT) device or a large BWP to support a UE that is downloading a large file. BWP also allows mobile network operators to make more efficient use of spectrum resources. For example, an operator can use a single BWP to support multiple UEs or multiple BWPs to support a single UE. This flexibility allows operators to maximize the number of UEs that can be served on a given amount of spectrum. Moreover, each BWP has its own numerology, enabling more efficient sharing of the spectrum among the heterogeneous services in 5G RAN and hence among slices. A UE can be configured with up to 4 BWPs in UL and DL, but only one can be used at

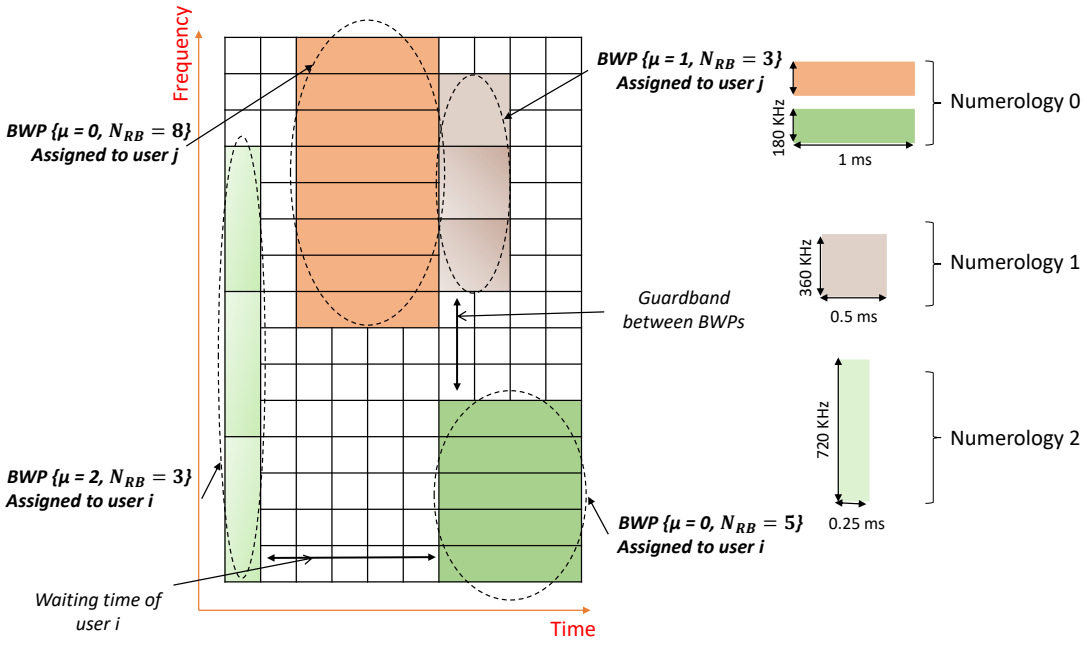


Figure 2.5: BWPs definition in time and frequency domain with mixed numerologies

a given time. Figure 2.5 shows 4 BWP, with different numerologies, defined in the time domain (x-axis) and frequency domain (y-axis). The BWPs concept with mixed numerology enables a dynamic allocation of numerology and PRBs. Hence, a UE can benefit from more than one service with different numerology values. Besides, it will reduce the UE power consumption since the UE will only operate on a part of the bandwidth instead of processing all the bandwidth, and the power saving schemes with UE adaptation to BWP bandwidth introduced in 5G NR standards [27] show 16% - 45% power saving gain. BWP is a key feature of 5G NR that plays an important role in enabling 5G to meet the demands of the future. It is worth noting that 5G NR specifications [24] RRC protocol and DCI. However, deciding the size and the numerology of BWPs and BWPs scheduling decisions are still open problems.

Connected Discontinuous Reception (C-DRX)

Next-generation cellular networks should accommodate the explosive growth in mobile data traffic and support different heterogeneous services to empower industry verticals and enable new business models. To achieve this goal, 5G NR has introduced several features to increase throughput and decrease latency, mainly numerology, massive MIMO, and higher bandwidths. These features increase PC for both the network infrastructure and the UE. The latter, which are typically powered by a limited battery, can suffer from poor QoE [18] due to the rapid discharge of the battery. The communications industry should therefore develop strategies to optimize the energy efficiency of 5G networks without compromising QoS [19]. Leading wireless equipment vendors have begun studying UE power-saving schemes in 5G NR [20]. They have investigated several techniques to reduce PC, such as C-DRX with a new set of parameters introduced by 5G NR and BWP adaptation.
C-DRX is a power-saving technique that is used in 5G NR to reduce the power consumption of UEs. Without C-DRX in 5G NR, a UE stays awake all the time to decode the DL data. This consumes a large amount of the UE's power. C-DRX works by allowing UEs to sleep for certain

periods of time. When a UE is in sleep mode, it consumes less power than when it is active. The gNB configures the UE with a set of C-DRX parameters. These C-DRX parameters are selected based on the type of application to minimize PC. However, they may increase the latency because the UE may be in a C-DRX sleep state when the data arrives at the gNB, and the latter should wait for the UE to become active. When C-DRX is activated, the time is divided into cycles, which can be long or short. Each cycle consists of an ON period and an OFF period. The ON period, defined in terms of milliseconds, is the period during which the UE remains awake and decodes the DL data. We consider the C-DRX parameters in 5G NR, depicted in Figure 2.6, to be (i) the cycle length; (ii) the duration of the ON period; (iii) the offset between the start of the cycle and the start of the ON period. The latter is a new parameter introduced by the 5G NR standard in order to shift the ON period since the arrival of the data is not always aligned with the cycle length, i.e., the data may arrive in the middle or at the end of the cycle. The ON period can be shifted to the middle or end of the cycle instead of leaving the UE active for the entire cycle, thus reducing latency while saving maximum power. C-DRX is a very effective power-saving technique, and it can help to extend the battery life of UEs significantly. However, it is important to note that C-DRX can also increase latency, as the UE may not be able to receive data immediately when it is needed. The 3GPP study, conducted in [20], shows the impact of C-DRX parameters and BWP adaptation on network performance, mainly on latency and power gain. However, the 5G NR standard does not provide solutions to derive the C-DRX parameters and the BWP configuration dynamically.

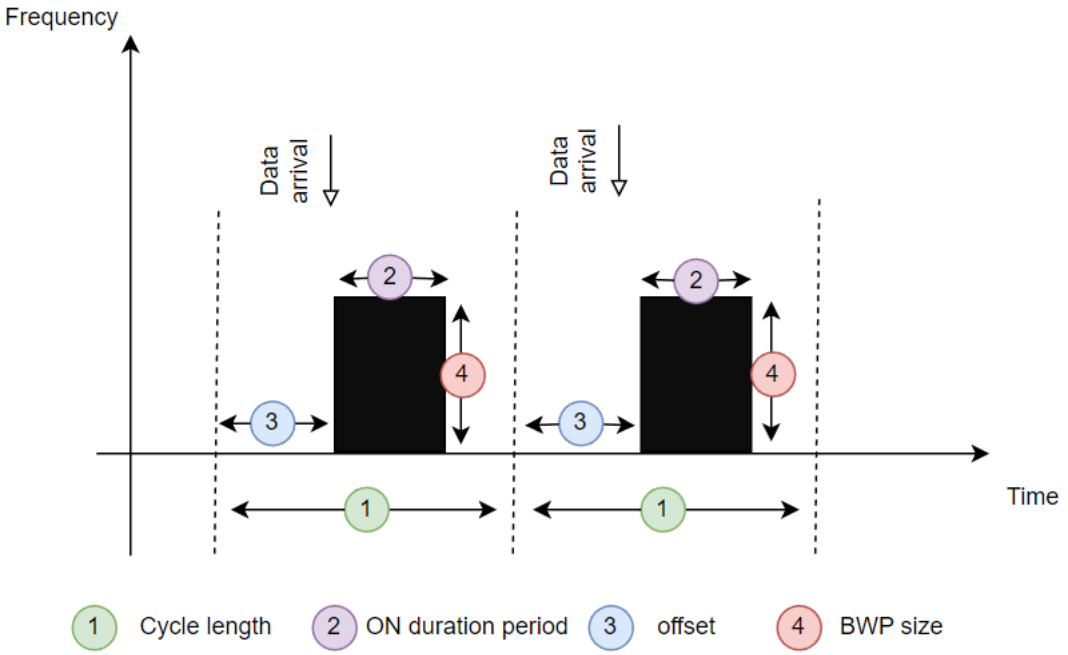


Figure 2.6: C-DRX parameters and BWP

Radio coverage

Radio coverage in 5G is the area in which a 5G radio signal is strong enough to be used by devices. The coverage of a 5G network depends on a number of factors, including the frequency bands that are being used, the type of antennas that are being used, and the environment in

which the network is deployed. RLF is an important criterion in modern wireless networks, including 5G. RLF is critical for services requiring high reliability, which is one of the key features of 5G, such as those supported by URLLC class[11]. RLF impacts principally network services that involve high-mobile devices, such as vehicles and flying drones.
Indeed, combining highly mobile UEs, at high altitude like UAVs [12], with the current limited number of deployed 5G cells and their small coverage due to wider frequency bands' usage [13] (which make the radio link between the UE and the gNB more susceptible to blockage and degradation leading to sudden interruption of the communication link), may yield to frequent RLF. RLF corresponds to a disconnection from the network, which strongly impacts sensitive services, such as UAV safety, where RLF increases the network latency and, in the worst cases, to UAV's losses and collisions. In this context, it is important that 5G tackles the RLF issue to improve reliability by, for instance, predicting when a mobile device has a high probability of seeing a RLF and taking the appropriate action, like anticipating the Handover or updating the trajectory of the UAVs.
RLF can be estimated by measuring radio key indicators such as RSRP, RSRQ, CQI, and PHR. By combining this information, it is possible to know the quality of the radio channel, hence the probability of RLF. However, deriving a closed form of the RLF distribution probability is very challenging. Different metrics are considered to measure the radio channel quality; among these metrics, the most used are:

- **Reference Signal Receive Power (RSRP)**: defined as the linear average received power (in Watts) of the signals that carry cell-specific Reference Signals (RS) within the frequency bandwidth. RSRP provides a cell-specific signal strength metric. This measurement is mainly used for handover decisions. It enables the gNB to rank different candidate cells according to their signal strength.

- **Reference Signal Strength Indicator (RSSI)**: defined as the total received power observed by the UE from all sources, including serving and non-serving cells, adjacent channel interference, and thermal noise within the measurement bandwidth. RSSI is not reported as a measurement in the 3GPP standard since [28], instead, it is used to compute the RSRQ measurement.

- **Reference Signal Receive Quality (RSRQ)**: provides a cell-specific signal quality metric. Similar to RSRP, this metric is used mainly for handover and cell re-selection decisions. RSRQ enables the gNB to rank different candidate cells according to their signal quality. It is used in scenarios where RSRP measurements do not provide sufficient information to perform reliable mobility decisions. RSQR is computed as follows: $RSRQ = \frac{N*RSRP}{RSSI}$
  Where N is the number of PRBs of the bandwidth, RSRP indicates the wanted signal strength, RSRQ enables the combined effect of signal strength and interference to be reported efficiently.

- **Channel Quality Indicator (CQI)**: It is an indicator carrying the information on how good/bad the communication channel quality is. It is reported by UEs to indicate to the gNB the level of modulation and coding the UE could operate (i.e., means choosing the Transport Block Size (TBS)) [29], which in turn can be directly converted into throughput.

- **Power headroom (PHR)**: indicates how much transmission power is left for a UE to use in addition to the power being used by the current transmission. If the PHR value is negative, it indicates that the UE is already transmitting with greater power than it is allowed to use.

2.3.3 Radio resource allocation

The DCI is a signaling mechanism that is used in 5G NR to control the DL transmission of data to UEs and UL transmission of data from the UE to the gNB. DCI is transmitted on the PDCCH. DCI carries a variety of information, including: (i) The UE(s) that are scheduled to receive/send DL/UL data; (ii) The PRBs that are allocated to each UE; (iii) The MCS that is used to transmit the data; (iv) The HARQ process that is used to retransmit lost or corrupted data; (v) the time slot where data is sent/received. The 5G scheduler is responsible of computing the different information sent in DCI (mentioned above). The 5G scheduler monitors the data queues of each UE, then according to the size of the queue, the channel quality, and the scheduler policy (e.g., fair scheduling between the UEs or with priority), the 5G scheduler computes the number of PRBs required to transmit the UEs data. After receiving the DCI from the concerned UE, it will send/receive data on the allocated PRBs using the parameters carried by the DCI (e.g., the MCS). The gNB can also switch the BWP of a UE using DCI. For instance, the UE can be scheduled on multiple BWPs.
However, resource allocation in UL is more challenging since the gNB has to be aware of the UE's queue status to allocate PRBs according to the UE needs, which results in increasing the latency by the signaling overhead. SPS [16] and GF [17] scheduling methods are proposed to overcome the signaling overhead, hence reducing the UL latency. On the one hand, SPS is a resource allocation scheme that allocates radio resources to users for a semi-persistent period of time, typically in the order of milliseconds or seconds. The resource allocation in SPS is static during the whole period. On the other hand, GF is a mechanism that allows UEs to transmit data without explicitly requesting permission from the gNB. Although these techniques are able to reduce the latency, they are less efficient for dynamic traffic patterns (dynamic inter-arrivals and dynamic data sizes) and less adaptable to quick channel condition changes, especially in FR2 frequencies ($>$ 6Ghz) and high mobility UEs. Furthermore, GF may impact the reliability of the services since resource usage can collide in frequency between UEs. The UL Dynamic Scheduling (DS) method is widely used among network operators due to its efficiency to allocate PRBs and high adaptation capability to channel conditions. However, it is inefficient for URLLC services due to its negative impact on the latency. Indeed, the PRBs are allocated to a UE upon the gNB receiving a Scheduling Request (SR) on the Physical Uplink Control Channel (PUCCH) channel. The gNB then assigns PRBs to the UE and sends a DCI to the UE, informing mainly the MCS, the number of PRBs, and an integer, called the k_2 parameter, to be used for that transmission. The UE will send the data after k_2 slots from receiving the DCI. The first DCI contains a minimum grant in terms of the number of PRBs (since SR is just one bit and does not inform the gNB about the current queue status of the UE). If the UE needs to send more data (i.e., the minimum grant is not enough), it piggybacks a BSR with the data to inform the gNB about the UL UE's queues status. Then, the gNB will send a DCI with an additional number of PRBs. Figure 2.7 illustrates the 3GPP-compliant dynamic UL scheduling procedure. Since SR is sent each time before UE sends data, the dynamic scheduler can adapt to changes in channel conditions very quickly. However, this method leads to a high latency which consists of the SR latency (delay between generating the SR and receiving the Physical Uplink Shared Channel (PUSCH) data) and the BSR delay (delay between sending the BSR and receiving the last PUSCH of the same transmission).
Basically, the standard dynamic UL scheduling is unsuitable for URLLC services requiring a latency of less than 1 ms. SPS and GF scheduling methods are proposed to reduce the latency. The gNB allocates PRBs to the UE without waiting for an SR. The PRBs are reserved for the UE periodically. However, these methods give a fixed grant size regarding PRBs and periodic slots, which is unsuitable for dynamic data patterns. Besides, they respond to changing channel

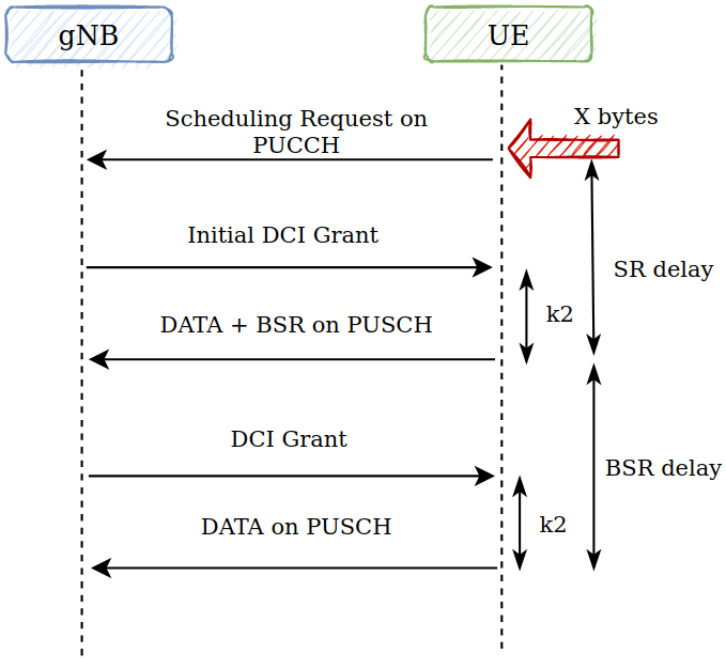


Figure 2.7: 3GPP dynamic UL scheduling procedure

conditions much slower than dynamic scheduling since the MCS is reported only once at the activation step. Hence, it reduces the reliability of transmissions, especially in very dynamic channel conditions like FR2 and high mobility UEs, which is unsuitable for URLLC services.

2.3.4 Network slicing

Network slicing allows the creation of virtual networks on top of the same infrastructure. Each virtual network is considered as a network slice and tailored to satisfy the service or application needs. Network slicing can help to reduce CAPital EXpenditures (CAPEX) and OPerating EXpenses (OPEX) as one physical infrastructure is shared efficiently to fulfill the heterogeneous communication service requirements of emerging network services. Apart from the research perspective, a number of industry consortium and standardization organizations have highlighted the architectural need for slicing, such as International Telecommunication Union (ITU) [30], Next Generation Mobile Networks (NGMN) alliance [31], and Global System for Mobile Communications Association (GSMA) [32].
A slice is signalled within a Public Land Mobile Network (PLMN) through the Network Slice Selection Assistance Information (NSSAI) [22], which is sent by a UE after random access in order to assist the network with slice selection. The NSSAI is further divided into the Slice/Service Type (SST), which can be either the eMBB, URLLC or mMTC and the Slice Differentiator (SD) to differentiate slices with the same SST. The SBA architecture allows customizing the CN for specific slices. The RAN should respect principles of slicing such as the support of QoS and resource isolation, but slicing is otherwise implementation-specific [33]. It is worth mentioning that an application or a network service can rely on more than one network slice. A PDU session is created for each slice per UE, allowing the handling of slices in the core network and appropriate mapping to the radio bearers in the RAN through the SDAP layer. An example is the case of autonomous drones for surveillance, which require a eMBB service to stream a high-quality video stream and a uRLLC service to control the drone. Therefore, from the side

of the UE, it can be part to parallel network slices, up to 8 different network slices according to [34].
On the RAN side, enforcing network slicing is more challenging as the radio resources are very limited, and the new features introduced by 5G NR make the management of radio resources more complex. Particularly, mixed-numerology is one of the main features of 5G NR. Moreover, a UE can not use more than one numerology at a given time slot due to physical layer constraints. Besides, the coexistence of mixed numerology in the same carrier introduces non-orthogonality to the system and causes INI as subcarriers associated with different numerology will no longer be orthogonal to each other. In this context, the 5G NR radio management system should cautiously select the numerology per UE while serving different network slices. While the eMBB slices need more bandwidth to satisfy the high throughput requirements, the URLLC slices require a shorter time slot duration to ensure a lower latency requirement, and mMTC slices need better frequency management to sustain a massive number of connected devices. By adding the network slice dimension, the radio resource management in 5G NR becomes more complex compared to the classical resource management (used in 4G and 5G without network slicing). Indeed, it should consider not only (i) time-domain scheduling, (ii) frequency-domain scheduling but also (iii) numerology selection and (iv) multiple slice management per UE.

2.3.5 Dynamic Time Division Duplex

In wireless communication systems, TDD is a duplexing method that allows devices to use the same frequency for both transmitting and receiving, but not at the same time. Instead, it divides time into alternating time slots for transmitting and receiving data. This is in contrast to Frequency Division Duplexing (FDD), which uses separate frequency bands for UL (transmitting) and DL (receiving) communications. During one time slot, the device can transmit data, and during the next, it can receive data. This switching between transmit and receive modes occurs rapidly, typically in two or three OFDM symbols depending on the hardware capabilities. The time in 5G systems is divided into frames. A frame is a 10-millisecond period that is divided into 10 subframes, each of which is 1 millisecond long. The number of slots in a subframe depends on the numerology (see Section 2.3.2). A frame is divided into periods called TDD period. In each TDD period, a TDD pattern is followed by both gNB and UE. However, unlike LTE, which specifies seven predefined TDD patterns for UL and DL allocation in a radio frame, 5G NR allows defining UL/DL patterns more flexibly. Indeed, it is possible that a slot may not be configured to be fully used for DL or for UL. OFDM symbols in a slot can be classified as "downlink", "flexible", or "uplink". Flexible symbols can be configured either for UL or for DL transmissions. Finally, like LTE, a guard period is necessary for the transceiver to switch from DL to UL and allow timing advance in UL.
The slot configuration, or TDD pattern, is indicated to UE either via Broadcast or RRC (re-)configuration messages. We distinguish between a common configuration that concerns all the slots marked as DL or UL, and a dedicated configuration that covers all slots and symbols noted as Flexible. The DL/UL pattern is repeated periodically according to $dl - UL - TransmissionPeriodicity$, noted δ. The value of δ depends on the NR numerology (μ). For instance, $\delta = 0.625ms$ can be used only with $\mu \in 3, 4$, while $\delta = 2.5ms$ can be used for all numerology expecting $\mu = 0$. Hence, depending on the numerology and δ, the number of slots (noted $\mathcal{T}_\delta$) varies completely. It is derived as follows: $\mathcal{T}_\delta = \delta \times 2^\mu$. For instance, if $\delta = 2.5ms$ and $\mu = 2$, the number of available slots is equal to 10. If $\delta = 5ms$ and $\mu = 4$, the number of available slots equal to 80. Table 2.1 summarizes the number of slots per subframe/frame per numerology.
In addition to δ, the common configuration includes the number of slots for DL (d_{slots}) located at

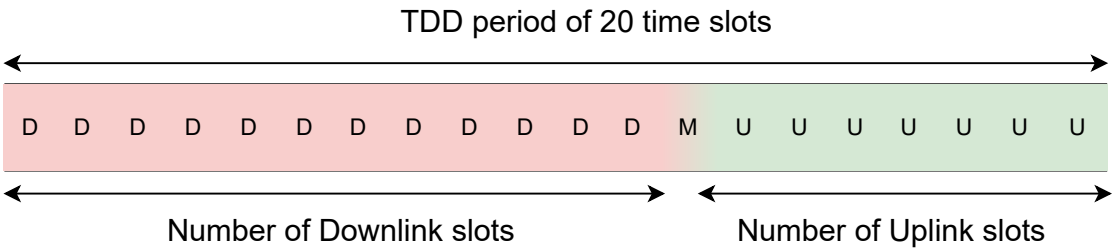


Figure 2.8: 5G NR TDD example pattern

Table 2.1: Number of slots in a subframe/frame for each numerology for normal CP

μ	SCS	No. of slots per subframe	No. of slots per radio frame	Slot duration
0	15khz	1	10	1
1	30khz	2	20	0.5
2	60khz	4	40	0.25
3	120khz	8	80	0.125
4	240khz	16	160	0.0625

the beginning of the transmission period and for UL (u_{slots}) located at the end of the transmission period. d_{sym} symbols within the slot immediately following the last full DL slot and last u_{sym} symbols in the slot preceding the first full UL slot are also indicated in the common configuration. The remaining symbols are considered flexible symbols. These flexible symbols can further be allocated to either DL or UL by making use of dedicated configuration. Figure 2.8 illustrates a UL/DL pattern. For more details on TDD pattern management in 5G NR, readers may refer to [35].

5G networks and beyond are designed to support an extensive range of applications [36][37], including those requiring high-speed data transfer and low-latency communication, such as immersive holographic communication, Internet of Skills, and 4D Interactive mapping [1]. Compared to previous generations of mobile networks, these emerging applications generate high UL traffic, corresponding to offloaded intensive computations that must be executed by a remote application located at the edge of the network. As a result, emerging services in 5G and 6G networks may be DL dominant, UL dominant or balanced between UL and DL. As a result, dynamic TDD has become a key enabler for 5G and beyond, as it allows resources to be allocated to these applications as needed, ensuring optimal performance and QoS. Dynamic TDD allows the gNB to change the TDD scheme dynamically without interrupting the UE's connectivity, i.e., by changing the number of dedicated UL and DL slots based on the UE's traffic patterns. Thus, Dynamic TDD is able to: (*i*) Improve resource utilization efficiency, e.g., when traffic is dominant on the UL or DL, allocating more slots on the UL or DL will avoid wasting resources in the other non-dominant direction; (*ii*) Reduce the latency since dynamic TDD reduces the queue buffer size faster than static TDD [38]; (*iii*) Increase application throughput as more slots can be allocated to the UL or DL according to various traffic patterns. However, the 5G NR specifications only cover the mechanism allowing the gNB to inform the UE about the UL/DL slots pattern in a TDD frame, leaving the algorithm deriving the pattern UL/DL open.

Compared to LTE, a more flexible frame structure in terms of slots ratio (UL/DL) can provide a greater traffic adaptation gain but also lead to more dynamic cross-link interference. Cross-link interference occurs when one gNB transmits while another receives in the same frequency band (i.e., two neighboring gNBs using a different TDD pattern). As shown in Figure 2.9, two

types of cross-link interference are introduced, namely gNB-to-gNB interference and UE-to-UE interference, which can significantly degrade system performance and result in decreased UE's throughput.

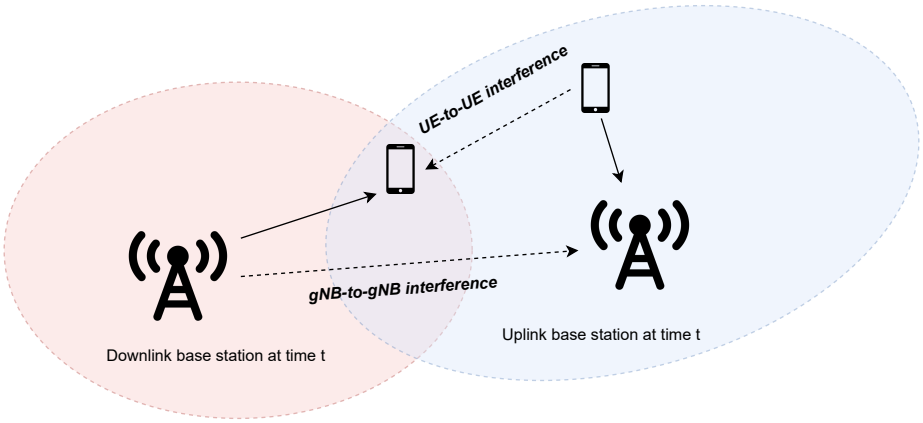


Figure 2.9: Illustration of cross-link interference in dynamic TDD system

The UE-to-UE interference impacts edge UEs, while gNBs are affected by the gNB-to-gNB interference when they are neighbors with high transmission power gNBs. Generally, gNBs use high transmission power to accommodate edge UEs. Thus, edge UEs are considered a main factor for cross-link interference. As a result, we assume that neighboring cells may not experience cross-link interference if there are no edge UEs.

2.4 Machine Learning (ML)

ML is a subfield of AI that focuses on the development of algorithms and statistical models that enable computer systems to improve their performance on a specific task through learning from data. In other words, ML allows computers to learn and make predictions or decisions without being explicitly programmed for each specific task.
ML is applied in various fields, including Natural Language Processing (NLP), image recognition, recommendation systems, autonomous vehicles, finance, healthcare, and more. ML is playing a crucial role in communication and networking [2] with the ability to provide a self-configured and self-optimized network which is a key characteristic of 6G networks.

2.4.1 Classification of ML

ML implementations are classified into four major categories, depending on the nature of the learning "signal" or "response" available to a learning system which are as follows:

1. Supervised learning: is the ML task of learning a function that maps an input to an output based on example input-output pairs. The given data is labeled.

2. Unsupervised learning: is a type of ML algorithm used to draw inferences from datasets consisting of input data without labeled responses.

3. Reinforcement learning: is the problem of getting an agent to act in the world so as to maximize its rewards. An agent is not told what actions to take as in most forms of ML but instead must discover which actions yield the most reward by trying them.

4. Semi-supervised learning: is an approach to ML that combines small labeled data with a large amount of unlabeled data during training. Semi-supervised learning falls between unsupervised learning and supervised learning.

Another categorization of ML tasks arises when one considers the desired output of a machine-learned system:

1. Classification: When inputs are divided into two or more classes, the learner must produce a model that assigns unseen inputs to one or more of these classes. This is typically tackled in a supervised way.

2. Regression: This is also a supervised problem, A case when the outputs are continuous rather than discrete.

3. Clustering: When a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

2.4.2 ML algorithms

ML algorithms are mathematical models that learn from data and can be used to make predictions or decisions. There are many different types of ML algorithms, each with its own strengths and weaknesses.
Some of the most common ML algorithms include:

1. Linear Regression (LR): LR is a statistical method for modeling the relationship between a scalar response variable and one or more explanatory variables (also known as dependent and independent variables). The case of one explanatory variable is called simple LR; for more than one, the process is called multiple LR. LR fits a linear equation to the data, with the slope of the line representing the relationship between the explanatory and response variables. The goal is to find the best-fitting line that minimizes the difference between the predicted and actual values of the response variable. Here is an example of a simple LR model:
   $y = mx + b$
   where: y is the response variable, x is the explanatory variable, m is the slope of the line, b is the bias. The slope of the line represents the change in the response variable for every unit change in the explanatory variable. The bias represents the value of the response variable when the explanatory variable is equal to zero.

2. Support Vector Machine (SVM): It is a supervised ML algorithm that can be used for both classification and regression tasks. SVMs work by finding a hyperplane in the input space that separates the data points into classes. The hyperplane is chosen to maximize the margin between the classes, which is the distance between the hyperplane and the closest data points from each class.

3. Long Short-Term Memory (LSTM): It is a type of Recurrent Neural Network (RNN) that is capable of learning long-term dependencies in sequential data. LSTM networks are particularly well-suited for tasks such as NLP, machine translation, and time series forecasting. LSTM networks work by using a special type of memory cell that can store information for long periods of time. This memory cell allows LSTM networks to learn long-term dependencies in sequential data, which is something that traditional RNNs are unable to do.

2.4.3 Deep Reinforcement Learning (DRL)

DRL, a very fast-moving field, is a combination of RL and Deep Learning. It is also the most trending type of ML because it can solve a wide range of complex decision-making tasks that were previously out of reach for a machine to solve real-world problems with human-like intelligence. Reinforcement learning is a generic framework for representing and solving control tasks, but within this framework, we are free to choose which algorithms we want to apply to a particular control task. Deep learning algorithms are a natural choice as they are able to process complex data efficiently. The deep neural network is trained to predict the expected reward of the agent for taking a given action in a given state. Once the deep neural network is trained, the DRL agent can use it to select actions that are likely to lead to high rewards. The agent can also use the deep neural network to learn from its mistakes and improve its performance over time.

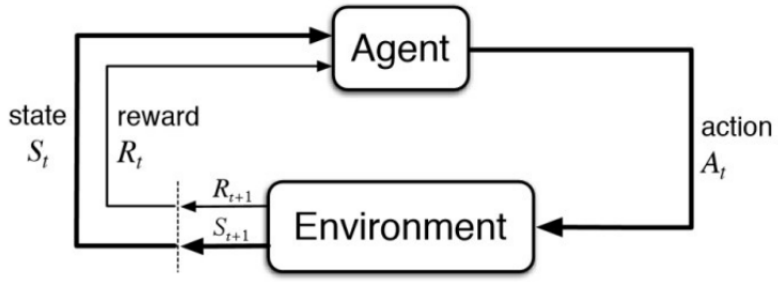


Figure 2.10: DRL framework

2.4.4 DRL algorithms

DRL is playing a crucial role in communication and networking [2] with the ability to provide a self-configured and self-optimized network that easily adapts to network changes. Moreover, DRL is a lightweight framework that enables quick decisions and hence takes real-time actions in the network characterized by its dynamicity and needs fast decisions. DRL techniques are based on the interaction of the DRL Agent with its environment by applying different actions and receiving rewards according to the actions taken. Let $\mathcal{S}$ denotes a set of possible states, and $\mathcal{A}$ denotes a set of actions. The state $s \in \mathcal{S}$ is a tuple of the environment's features relevant to the problem at hand. Also, it describes the agent's relation with its environment. Assuming discrete time steps, the agent observes the state of its environment, $s^t \in \mathcal{S}$ at time step t. It then takes action $a^t \in \mathcal{A}$ according to a certain policy π. Once the agent takes action a^t, its environment moves from the current state s^t to the next state s^{t+1}. As a result of this transition, the agent gets a reward r^{t+1} that characterizes its benefit from taking action a^t at state s^t. This scheme forms an experience at time $t+1$, hereby defined as $e^{t+1} = (s^t, a^t, r^{t+1}, s^{t+1})$, which describes an interaction with the environment. The set of interactions $e^t \in \mathcal{E}$ is called Replay Buffer, which is used to train the DRL agent in order to derive the optimal policy π^*. The latter provides the optimal action a^t, to take in each state s^t, in a way to maximize future cumulative discounted reward G^t defined as follows:

$$G^t \doteq \sum_{k=0}^{T} \gamma^k r^{t+k+1} = r^{t+1} + \gamma G^{t+1} \tag{2.1}$$

With $\gamma \in [0, 1]$ defined as the discount rate that penalizes the future rewards, and T equal to the time horizon, which is finite for episodic problems (i.e., problems that end when the environment

is a final state) and infinite for continuing problems.
DRL algorithms can be broadly categorized into two main types: Markov Decision Process (MDP)-based and Bandits-based. The MDP-based DRL is when the action impacts the environment following predefined MDP transitions, while Bandits-based DRL is when the actions do not move the environment from one state to another state following predefined MDP transitions. In this thesis, we focus on MDP-based DRL methods. The latter are divided into two main types: Model-based and Model-free. Model-based DRL algorithms learn a model of the environment, which is a function that predicts the next state and reward given the current state and action. The agent then uses the model to plan its actions, i.e., to select the action that is most likely to lead to the desired outcome. Model-free DRL algorithms do not learn a model of the environment. Instead, they learn directly from experience, i.e., by trial and error. The agent starts by randomly selecting actions and observing the consequences. Over time, the agent learns to select the actions that are most likely to lead to the desired outcome. In this thesis, we focus on Model-free methods since it is difficult to create a model for the complex RAN environment. The Model-free methods are divided into two categories: Value-based methods and Policy-based methods. Value-based DRL algorithms learn a value function, which estimates the expected value of taking a given action in a given state. The agent then selects the action with the highest expected value. Policy-based DRL algorithms learn a policy that directly maps states to actions. The agent then selects the action specified by the policy.

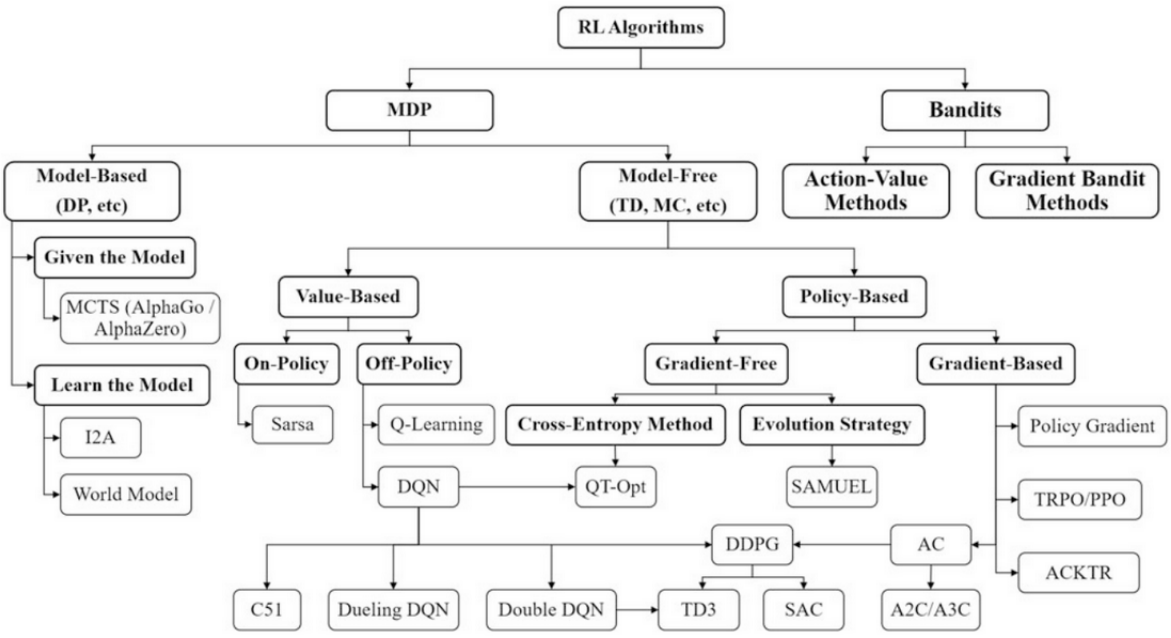


Figure 2.11: DRL algorithms classification

In this thesis, we mainly used two methods: Deep Q-Network (DQN) [39] and Deep Deterministic Policy Gradient (DDPG) [40]. DQN is a Value-based method, and DDPG is a Policy-based method. We also used Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [41], which is the DDPG extension for multi-agents environments. DQNs are trained using a technique called experience replay. Experience replay allows the agent to learn from its past experiences without having to interact with the environment in real-time. This makes DQNs more efficient and less likely to get stuck in local optima. Here is a simplified example of how a DQN works: (i) The agent starts in a state and selects an action based on its current policy; (ii) The agent takes the action and observes the next state and reward; (iii) The agent stores the experience (state,

action, next state, reward) in its memory; (iv) The agent samples a batch of experiences from its memory and updates its policy using a Q-learning algorithm.; (v) The agent repeats steps (i)-(iv) until it reaches the desired level of performance. DDPG is an actor-critic algorithm, which means that it has two separate neural networks: the actor-network and the critic network. The actor-network learns the policy, and the critic network learns the value function. It also uses the Experience replay to learn from its past experiences. Here is a simplified example of how DDPG works: (i) The agent starts in a state and selects an action based on its current policy; (ii) The agent takes an action and observes the next state and reward; (iii) The agent stores the experience (state, action, next state, reward) in its memory; (iv) The agent samples a batch of experiences from its memory and updates its policy and value function using a policy gradient algorithm and a Q-learning algorithm, respectively; (v) The agent repeats steps (i)-(iv) until it reaches the desired level of performance.
Unfortunately, traditional DRL approaches, such as Q-Learning or policy gradient, are poorly suited to multi-agents environments. One issue is that each agent's policy changes as training progresses, and the environment becomes non-stationary from the perspective of any individual agent (in a way that is not explainable by changes in the agent's own policy). This presents learning stability challenges and prevents the straightforward use of past experience replay. Policy gradient methods, on the other hand, usually exhibit very high variance when coordination of multiple agents is required [41]. Besides, the state-action space will grow exponentially when a learning agent keeps track of all agent actions. Hence, MADRL approach was introduced to address the above issues.
MADRL system can be represented by the tuple of $(\{\mathcal{S}_j\}_1^{\mathcal{N}}, \{\mathcal{A}_j\}_1^{\mathcal{N}}, \{\pi_j\}_1^{\mathcal{N}}, \{r_j\}_1^{\mathcal{N}})$. Let $\mathcal{G}$ denotes a set of agents. Each agent $j \in \mathcal{G}$ observes a state $s_j^t \in \mathcal{S}_j$ from the environment and executes an independent action a_j^t from its own set of actions $\mathcal{A}_j$ on the basis of its local policy $\pi_j : S_j \rightarrow S_j$. Agents perform joint action $a^t = a_1^t, a_2^t, \cdots, a_N^t \in \mathcal{A}$, where $A = (\mathcal{A}_1 \times \mathcal{A}_2 \times \cdot \cdot \cdot \times \mathcal{A}_N)$, which leads the environment to move from state $s_j^t \in \mathcal{S}_j$ to a new state $s_j^{t+1} \in \mathcal{S}_j$, then the agent j receives a reward r_j^{t+1}. In a centralized reward setting (i.e., the agents are cooperating), the agents receive a common reward r^{t+1}. The goal of each agent is to learn a local optimal policy π_j^* that forms a central optimal policy $\pi^* = \pi_1^*, \pi_2^*, \cdots, \pi_N^*$.
In general, MADRL leverages DRL methods for each agent. DRL methods are classified into three categories: *i*) *value-based* methods, such as DQN; *ii*) *policy-based* methods, such as REINFORCE (i.e., Monte-Carlo Policy Gradient); *iii*) *actor-critic* methods that combine the two previous methods, such as A3C and DDPG [42]. In the actor-critic approach, we have mainly two families, the stochastic policy approach (e.g., A2C and A3C) and the deterministic approach (e.g., DDPG). In the stochastic policy approach, the actions are selected from the Actor with different probabilities using the Softmax activation function. The agent should pick the action that has a high probability. Unfortunately, the main limitation of the stochastic policy approach is the number of actions that should be limited. In contrast, in the deterministic approach, the actions are generated directly from the actor-network, enabling continuous actions. In this paper, we are interested in specifying the percentage of UL and DL slots. For this reason, we have adopted the MADDPG algorithm [41], which is an improved version of DDPG applied in a multi-agents environment.

2.5 Conclusion

In this chapter, we defined 5G and introduced the key concepts related to 5G NR, the radio access technology of 5G, as well as the main AI techniques used to optimize prediction and control tasks. We have alsoidentified the main challenges of slicing and optimizing network

services at the RAN. In the next chapter, we propose frameworks and solutions to slice the RAN in order to fulfill different service's requirements. The different frameworks leverage 5G NR features such as BWP and numerologies.

Part I

Network Slicing in 5G NR

Chapter 3

NRflex: Radio Resources Allocation in 5G NR in dedicated bandwidths environment

3.1 Introduction

Network slicing is considered as one of the critical enablers of 5G to support a wide range of applications and use cases with different requirements [43]. 5G is assumed to support variant network services that are organized into three main categories: URLLC, eMBB, and mMTC [44]. Although network slicing has been widely studied and solutions have started to appear, particularly at the transport and core network parts, slicing (sharing) the RAN resources is still challenging. Indeed, with the new features introduced by the 5G NR, further investigation is needed to enforce network slicing at the RAN. 5G NR introduces several new features that can be beneficial for slicing the RAN [9]. Among these new features, we may mention the concept of BWPs. The latter aims to enable flexible assignment and configuration of PRBs. BWPs are subsets of contiguous PRBs allocated per UE, i.e., the UE expects to use resources only in a specific part of the bandwidth. Besides, 5G NR introduces the concept of numerology, which uses specific physical layer configuration, mainly SCS. Unlike 4G, where all the slot-time has the same duration (1 ms), 5G NR by adapting SCS allows reducing the slot-time duration down to 125 microseconds, which can considerably reduce the RAN latency. Consequently, each BWP has its own numerology, enabling more efficient sharing of the spectrum among the heterogeneous services in 5G RAN and hence among slices.
In this first contribution, we introduce NRflex, a novel framework that dynamically enforces RAN slices in 5G, relying on the concept of BWPs. NRflex addresses the joint PRBs scheduling and allocation problem with mixed numerology in 5G NR, which allows meeting the latency requirement of URLLC services while considering the other types of services (mainly eMBB). NRflex redefines the life-cycle management (LCM) of the RAN slices by leveraging on the O-RAN architecture [7]. Besides keeping the well-known creation and deletion steps, NRflex introduces five new steps that enable the RAN resources' dynamic sharing of running slices. A preadaptation step runs at the RIC, which dynamically defines the size of a BWP dedicated to a slice according to gNBs' feedback. The four other steps run at the gNBs to periodically allow : (1) deciding which BWP (i.e., slice) a UE has to connect to according to the UE's buffer status and the previous active BWP as UEs cannot use two different BWP in parallel; (2) sharing the BWP of a slice among its UEs according to the UE channel quality and service requirements. The key contributions of this chapter are:

- We introduce a novel framework mapped to the O-RAN architecture.
- We introduce a novel definition of RAN slice LCM.
- We propose an algorithm to run at RIC near-time that dynamically computes the size of BWP dedicated to a RAN slice.
- We propose a multiplexing and scheduling algorithm to run at the gNB that periodically decides for each UE the active BWP to use and the amount of PRB assigned to it.

3.2 Related works

3.2.1 RAN slicing

Given that flexible numerology plays an important role in network service optimization, especially to reduce latency, several works have considered combining flexible numerology with network slicing. The aim is to satisfy URLLC slices' low-latency requirements while ensuring eMBB slices throughput requirements. In [45], the authors review the impact of changing numerology on the throughput and end-to-end latency while taking into account the traffic patterns and the processing delays. Work in [46] explores the potential of optimizing resource allocation with flexible numerology in the frequency domain and variable frame structures in the time domain while considering the presence of services with different types of requirements. The authors used linear programming and Lagrangian duality to design an optimization algorithm to optimize resource allocation in that case. In [47], the authors study the admission control and network slicing design for 5G NR systems in which the total bandwidth is sliced (i.e., shared) to support eMBB and URLLC services. They propose allowing traffic from the eMBB BWP to be overflowed to the URLLC BWP in a controlled manner by using an efficient iterative algorithm. In [48], the authors introduce a novel 5G slice resource allocation approach that combines the utilization of both complete slots (or PRB) and mini-slots with adequate physical layer design and service requirement constraints. They advocate for a probabilistic characterization that allows estimating feasibility and characterizes the behavior of the constraints. In [49], the authors propose a self-adaptive flexible TTI scheduling strategy in the eMBB and URLLC coexistence scenario using ML. They reduce the delay and packet loss rate of the URLLC services while guaranteeing the eMBB requirements by dynamically selecting TTI according to traffic load and service requirements.
However, most of the mentioned solutions did not consider a dynamic assignment of BWP as in NRflex; they use the same numerology throughout the network slice life-time. Besides, they consider that a UE is attached only to one slice, which is not realistic as 3GPP allows UEs to be attached to up to 8 network Slices. In contrast, NRflex uses a multiplexing mechanism to allow UEs to be attached to more than one network slice parallelly.

3.2.2 O-RAN architecture

In parallel with 3GPP groups, a group of network operators as well as big telecommunication manufacturers names has launched a new initiative known as O-RAN alliance [7]. The latter specifies the new architecture of RAN in 5G and beyond. The alliance objective is to redesign the RAN architecture to unlock it from proprietary-locked solutions to an open system allowing the deployment of novel services and applications on top of the RAN. The O-RAN Alliance is committed to evolving radio access networks by offering open and standardized interfaces for every network component. This transformation will reduce network cost, improve network

efficiency, and give the agility to import new network capability [14].
O-RAN vision relies on the programmable RAN concept, a new trend that enforces the SDN concept at the RAN. Programmable RAN introduces the RAN controller's notion that runs different RAN applications, such as mobility management, user scheduling, etc. It enforces the policies issued by these applications on the eNB/gNB under its control via a southbound protocol. An example of a programmable framework is FlexRAN [50] [51], which is composed of a RAN controller and agents deployed on top of OpenAirInterface (OAI) eNB/gNB [52]. Figure

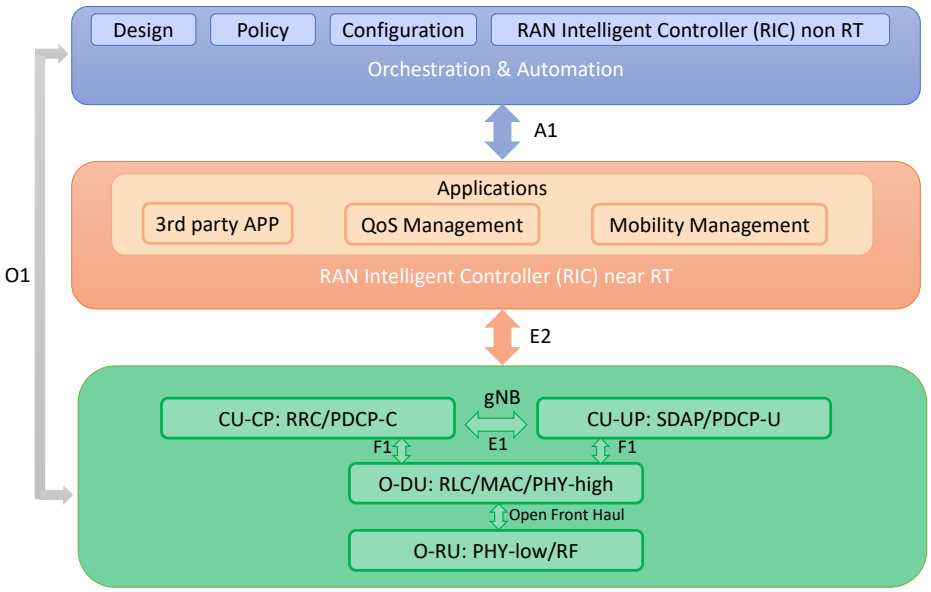


Figure 3.1: The O-RAN network management architecture and open interfaces

3.1 illustrates the O-RAN architecture that considers a fully functional split of RAN functions. Indeed, the new trend in terms of 5G RAN architecture is to split the RAN functions, which were constituting monolithic gNB entities, among the CU-CP and CU-UP, the DU, and the Radio Unit (RU). CU contains the functions related to higher layers of the RAN, i.e., RRC and PDCP CP, and SDAP and PDCP-U, respectively, for CU-CP and CU-UP. In comparison, O-DU (O-RAN DU) hosts the low-layer functions, i.e., RLC, MAC, and Physical high. It should be noted that CU and DU can run in a virtualized environment using VM or container technology. Finally, the O-RU (O-RAN RU) hosts all the functions that cannot be virtualized, such as Physical low and RF. The O-RU will be kept in the field and deployed as hardware. Besides, O-RAN adds two layers to the RAN, namely Orchestration and Automation and RIC Near-Real Time (RT).
Orchestration and Automation: contains the Non-RT (Non-Real-Time) RIC functions, which support intelligent RAN optimization in non-real-time (i.e., greater than one second) relying on data analytics and ML training/inference solutions. It also manages the network functions such as network design, control, policies, inventory, and configuration.
RIC Near-RT: enables near real-time control (i.e., 10ms to 1s) and optimization of O-CU (CU-CP and CU-UP), O-DU, and O-RU nodes. The RIC Near-RT hosts Applications that use the E2 interface to collect near real-time RAN information (as Key Performances Indicator (KPI)) to ease service deployment using these primitives such as QoS and mobility management services, which can also be guided by the policies provided through the A1 interface by the

Non-RT RIC.
In summary, O-RAN represents the future of the RAN architecture; therefore, NRflex is relying on it to build the components needed to enforce 5G RAN slicing.

3.3 NRflex framework

NRflex's main idea is to jointly allocate numerology and radio resources for each slice so that UEs can use multiple slices with different requirements. To achieve this objective, NRflex introduces several components that interact together, namely, Bandwidth manager, Pre-processor, and BWP multiplexer, distributed into two separate entities, i.e., RIC and gNB. In this section, we detail the NRflex framework's components and algorithms. We start by defining a RAN slice in NRflex and introducing the revised life-cycle process of RAN slices. Then, we present the components of NRflex, and for each one, we give its role in the O-RAN architecture as well as its related algorithms.

3.3.1 NRflex slice definition

A network slice is an isolated logical sub-network tailored to fulfill diverse requirements requested by an application. A network slice can be isolated at the RAN level in terms of radio resources (for each slice, a number of PRBs is allocated) and network functions (each slice has personalized network functions). We consider that a slice is defined at three levels: application, MAC, and Physical. At the Application level, a slice is an object that contains: the type of the slice (URLLC or eMBB or mMTC), the requirements of the slice (maximum latency for URLLC slices, and the desired throughput for eMBB slices), the duration in which the slice is active, the UEs associated with that slice, and the region where the slice is active. These objects are managed by a high-level entity, the RAN Controller, which divides the gNB's bandwidth among slices. The RAN Controller increases/decreases the number of PRBs for each slice according to specific KPIs sent by the gNBs. At the MAC level, a slice is a set of Logical Channel (LC) belonging to different UEs. As the gNB can have more than one data LC for a UE, the latter can be associated with many slices, suitable for multi-service applications. The LCs are scheduled in a way to fulfill services' requirements with a minimal number of PRBs. At the Physical level, a slice is considered as a BWP associated with numerology according to the slice type. For instance [49], numerology 0 for eMBB slices, numerology 1 for URLLC slices with Max Latency > 5 ms, and numerology 2 for URLLC slices with Max Latency < 5 ms. The amount of PRBs in a BWP is computed to fulfill the slice requirements (see Section 3.2). For each UE, the BWPs with the same numerology are aggregated to reduce the complexity of the Multiplexing step (see Section 3.3).

3.3.2 Network Slice life-cycle in NRflex

According to 3GPP [53], an end-to-end network slice LCM is composed of four phases (Figure 3.2): Preparation, Activation, Run-time, and Decommissioning. The preparation phase is dedicated to the description of the network slice components and attributes using a blueprint. The activation phase consists of configuring and instantiating the network slice resources, such as instantiating the virtual resources (Virtual Machine (VM) or containers) and reserving physical resources such as radio, network, and compute resources. The run-time phase covers the supervision and the modification of the resources dedicated to the slice, such as increasing the radio resources or computing resources. Finally, the decommissioning phase consists of releasing the

resources that have been reserved to the network slice. In NRflex, we revisit the run-time phase, focusing on the RAN part of the end-to-end network slice, or the RAN slice.

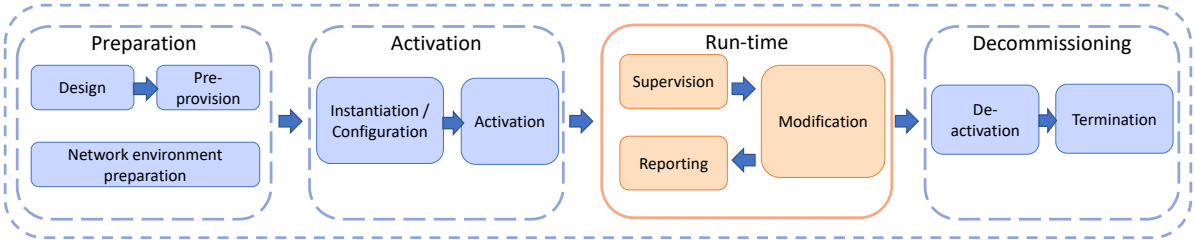


Figure 3.2: Lifecycle phases of a Network Slice Instance [53]

Since gNB traffic load and UEs channel quality can change over time, NRflex adjusts the slice performance by adding or removing PRBs to running slices. Besides, a slice can not be active during all UE's slots. Indeed, a UE can use only one numerology at a given time (for example, a UE belonging to both an URLLC slice and an eMBB slice can not use both of them at the same slot). Therefore, the network slice lifecycle at run time is detailed in NRflex as follows (Figure 3.3):

1. **Pre-processing** in this step, the amount of PRBs needed to ensure the slice requirement is calculated for each slice. At the end of this step, a sorted list of LCs according to the deadline criterion is provided to the Multiplexer (at the MAC layer level); each LC has a pre-allocated amount of PRBs.

2. **Multiplexing** knowing that only one BWP is active for a UE at a given time, and a UE can not be associated with more than one slice type (more than one configured BWP), a multiplexing step is mandatory. This step is needed to select which BWP will be active in the next slot by considering the different slices' requirements to which the UE belongs. At the end of this step, LCs not supposed to transmit in the next slot will be removed from the sorted list, which is transferred to the MAC Scheduler.

3. **Allocation** in this step, the MAC scheduler will allocate PRBs to different LCs by considering the result of the preprocessing and multiplexing steps.

4. **Monitoring** in this step, two KPIs are computed. They are used in the life cycle management of a RAN slice, $throughput\_success\_rate$ measures the eMBB slice performance (best value 1, worst value 0), and $deadline\_failure\_rate$ measures the URLLC slice performance (best value 0, worst value 1). They are computed at the gNB level and used by RIC to make decisions to add more resources to a slice. If its value has not reached the targeted

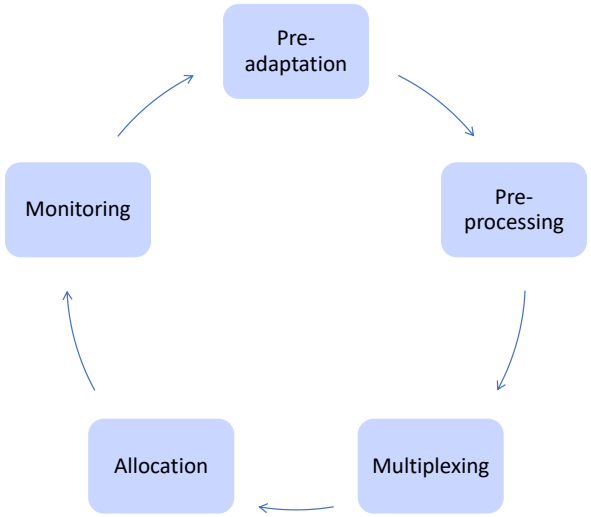


Figure 3.3: Runtime slice lifecycle in NRflex

value, then more resources need to be assigned to the slice. The *throughput_success_rate* takes into account the actual throughput, the desired throughput, and the queue size. Its value reaches 1, if the actual throughput reaches the desired throughput or the data queue is empty. *deadline_failure_rate* monitors the packets that exceed their deadline. A value equal to 0 means that all the slice's packets respect their max latency.

5. **Pre-adaptation** in this step, the amount of PRBs dedicated to each slice is computed by relying on the previous pre-allocation results and the KPIs, aiming at adjusting the slice performance.

This process is executed in an infinite loop until the slice deletion.

3.4 NRflex and the O-RAN architecture

As stated earlier, to support the deployment of NRflex components to ensure 5G NR network slicing, we consider the O-RAN architecture. In figure 3.4, we illustrate the different components' interaction in the O-RAN model. Those added by NRflex are highlighted in orange. This figure is slightly different than Figure 3.1 as we group O-CU, O-DU, and O-RU functions under the same components (i.e., gNB) to ease the figure readability. Also, there is a new entity, Slice Orchestrator (SO), which is not under the scope of O-RAN, but it is an essential element as it manages the LCM of the end-to-end network slices, including the RAN (or RAN slice). In the following, we describe the role of each component highlighted in the figure.

The Slice Orchestrator (SO) This component is the entry of the system. It is not defined by the O-RAN architecture. Its role is to provide the end-to-end slice template to be used by the tenant to define the network slice characteristics and its components and manage the end-to-end slice LCM (Figure 3.2). Although the end-to-end slice includes other components to deploy, such as the Core Network, the applications, and the transport network, we focus only

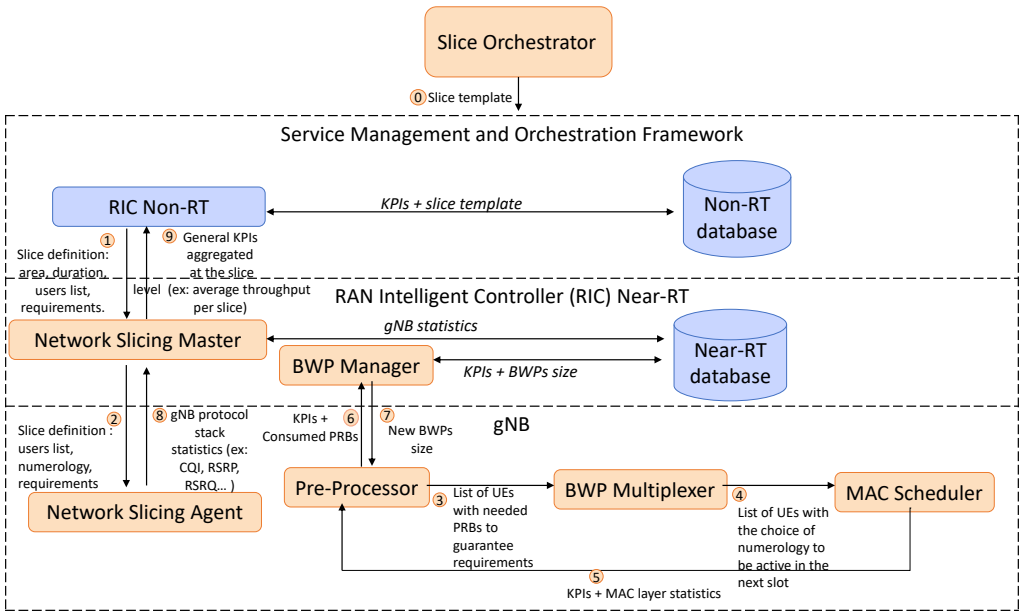


Figure 3.4: NRflex components mapped to the O-RAN architecture

on the RAN part in this work. Therefore, the SO enforces the RAN part of the network slice by communicating with the RIC Non-RT module.

The RIC Non-RT In our system, the RIC Non-RT launches the slice creation and deletion procedures and gathers general performance KPIs from the RAN, such as throughput, latency, bandwidths, etc., via the O-RAN O1 interface.

The RIC Near-RT O-RAN RIC Near-RT allows flexible onboarding of third-party control-applications as QoS enforcement, connectivity management, and handover control. NRflex adds two control applications:

- **The Network slicing master:** it manages the slices among many gNBs. It can be instantiated for a region and checks for UEs association with their slices. It collects gNB statistics used by O-RAN analytic applications.
- **The BWP manager (Algorithm 1):** it uses O-RAN E2 interface to re-adapt the BWP size for each slice according to the schedulers' feedbacks (i.e., the KPIs of the Monitoring step) and the consumed PRBs of each slice. We consider two KPIs:

  $throughput\_success\_rate = \frac{\sum_{i=1}^{slots} Ti \mid Qi}{number\ of\ slots}$ where $T_i = 1$ if the scheduled bytes at slot i exceed the throughput per slot, and $Q_i = 1$ if the traffic queue is empty (all traffic scheduled);

  $deadline\_failure\_rate = \frac{D}{number\ of\ slots}$ where D is the number of times a PDU (Protocol Data Unit) chunk had exceeded its deadline. $N_{embbslices}$ ($N_{urllcslices}$) is the number of eMBB (URLLC) slices, respectively. $NewPRB_j$ represents the number of PRBs to be allocated for slice j (BWP size of slice j) in the next time interval, $consumedPRB_j$ represents the average number of used PRBs by slice j during the past time interval.

Algorithm 1 BWP manager

for $j = 1, 2, \ldots, N_{embbslices}$ **do**
  if $throughput\_success\_rate_j < 1$ **then**
    $NewPRB_j = consumedPRB_j + 2$
  end if
end for
for $i = 1, 2, \ldots, N_{urllcslices}$ **do**
  if $deadline\_failure\_rate_i > 0$ **then**
    $NewPRB_i = consumedPRB_i + 2$
  end if
end for
Send the new BWP sizes to the gNB Pre-processor

The gNB (O-DU mainly) NRflex uses the following components at the gNB:

1. **The Network slicing agent:** it manages slices at the MAC level, receives slices' configuration from the Network slicing master, and stores the information to be used by the pre-processors and the MAC scheduler.

2. **Pre-processor:** it executes for each slice an instance of a pre-processor. We considered two types of pre-processors:

   (a) **The eMBB pre-processor (Algorithm 2):** it takes data from the LCs associated with the slice instance and calculates the effective number of PRBs needed to send the buffered data. It considers the buffer size, the CQI of each UE, and the maximum number of PRBs allocated by the BWP manager that achieves the desired throughput. The α_{UE} is a weight associated with a UE indicating how many eMBB slices of this

Algorithm 2 eMBB pre-processor for slice i

Sort LCs according to the weight α_{UE}
for $LC = 1, 2, \ldots, N_{LCs}$ **do**
  $R \leftarrow bytes\_to\_RBs(CQI_{UE}, \alpha_{UE} * req_{LC})$
  $N \leftarrow bytes\_to\_RBs(CQI_{UE}, queue\_size_{LC})$
  $prealloc\_res_{LC} \leftarrow \min(R, N)$
  Update the available PRBs for slice i
end for

   UE were discriminated in the Multiplexing stage; it is updated in the Multiplexing algorithm. Based on this variable, NRflex controls the required amount of throughput to achieve in the current slot to guarantee the desired throughput in a 1s time interval. *bytes_to_RBs* is a helper function that returns how many PRBs are required to carry the amount of data with the CQI passed in parameters.

   (b) **The URLLC pre-processor (Algorithm 3):** it sorts LCs according to their head PDU deadline, then computes the PRBs amount needed only to transmit data chunks that meet their deadline in the near future.

3. **The BWP multiplexer (Algorithm 4):** it selects which BWP to activate for each UE and generates DCI [54] to be sent to the concerned UEs indicating the decision.

Algorithm 3 URLLC pre-processor for slice i

Sort LCs according to the remaining time of the first pdu
for $LC = 1, 2, \ldots, N_{LCs}$ **do**
    $size \leftarrow 0$
    while $remaining\_time(pdu_j) <= 2TTI_\mu$ **do**
        $size \leftarrow size + pdu\_size_j$
    end while
    $prealloc\_res_{LC} \leftarrow bytes\_to\_RBs(CQI_{UE}, size)$
    Update the available PRBs for slice i
end for

Algorithm 4 BWP multiplexer

for $UE = 1, 2, \ldots, N_{UEs}$ **do**
    for $LC = 1, 2, \ldots, N_{LCs}$ **do**
        if $\mu_{LC} = 2$ and $sched_{LC} > 0$ **then**
            if $\alpha_{UE} < \alpha_{max_1}$ **then**
                activate numerology 2 for UE
                $\alpha_{UE} \leftarrow \alpha_{UE} + 1$
                break
            end if
        else
            if $\mu_{LC} = 1$ and $sched_{LC} > 0$ **then**
                if $\alpha_{UE} < \alpha_{max_2}$ **then**
                    activate numerology 1 for UE
                    $\alpha_{UE} \leftarrow \alpha_{UE} + 1$
                    break
                end if
            else
                if $sched_{LC} > 0$ **then**
                    activate numerology 0 for UE
                end if
                $\alpha_{UE} \leftarrow 1$
                break
            end if
        end if
    end for
end for

This algorithm prioritizes URLLC traffic as it has to respect a deadline. α_{max_1} and α_{max_2} are thresholds to avoid starvation of eMBB traffic. $sched_{LC}$ is the amount of data to schedule in the next slot (the result of the pre-processing step).

4. **The MAC Scheduler:** It allocates PRBs for each UE based on how many PRBs it needs and how many are available for each slice.

3.5 Performance Evaluation

In order to evaluate NRflex framework algorithms, we used a reliable 5G simulator based on Matlab that supports different numerology (Table 3.1) and BWPs with dynamic scheduling. Note that this simulator is an improved version of the one used in [55]; it includes 5G NR features. We validated the simulator using the 3GPP 5G NR simulator [56]. Both of them gave the same throughput with different configurations (Bandwidth, numerology, etc.). Table 3.1 describes the available system bandwidths in the simulation environment. These bandwidths will be divided among different BWPs.
We have simulated 3 scenarios: (1) the evolution of the number of users over time and its impact on the KPIs as well as the PRBs allocated for each slice; (2) the variation of the users' number as well as the traffic load and their impact on the KPIs and the PRBs allocated for each slice; (3) the variation of the traffic load and its impact on the numerology selection. We have run the simulation for 100 iterations. Each value presented in the figures represents the average. We did not include the minimum and maximum values as they are very close to the average, and adding them will reduce the figures' readability. For eMBB slices, we compared our

Table 3.1: 5G NR parameters [57]

Numerology	System available BandWidth (MHZ)	Number of PRBs
0	20	106
1	40	106
2	80	107

algorithm with the one introduced in [55] (called the standard solution for eMBB slices in the rest of the paper), which shares the same idea with other eMBB resource allocation algorithms (as [50, 58]); i.e., they use the slice throughput constraint to compute the required amount of PRBs. To recall, NRflex, in addition to throughput, considers the queue size (see Algorithm 2) to allocate only the needed PRBs for the slice. Thus, we minimize the number of PRBs allocated for eMBB slices while respecting their throughput requirements. For URLLC slices, we compared our algorithm with the Fair Proportional Scheduling algorithm combined with a resource allocation strategy that allocates the amount of PRBs needed to schedule all URLLC traffic first (called the standard solution for URLLC slices in the rest of the paper). NRflex (Algorithm 3) considers PDUs' deadline to allocate only the needed PRBs to schedule PDUs that will exceed their deadline shortly. We have simulated four slices with different requirements (Table 3.2) and a random CQI, which takes values in [12-15] intervals for each UE, indicating a medium to a suitable channel condition.
We specified the arrival rate and the packet size of the traffic associated with each user's slice type in Table 3.3. URLLC slice traffic is characterized by small data chunks with high frequency, while data chunks' sizes are big with low frequency sending for eMBB slice traffic. Moreover, the UEs can join more than one slice. In all scenarios, we are associating each UE to two slices (one eMBB and another one URLLC). However, UEs cannot serve two slices at the same time (i.e. same slot in ms granularity) when the slices use different numerology (5G NR physical layer constraint). We are using the term slot relative to the slice numerology. If numerology n is selected then slot duration is 2^{-n} ms. At t=0s, the system includes 10 UEs among them 5 UEs are connected to each slice tuple ((urllc1,urllc2,embb1), (urllc1,urllc2,embb2)). At t=5s and t=11s, 2 more UEs connect to each slice tuple; i.e., at t=6s, the system includes 14 UEs, and at t=12s, 18 UEs.
Figure 3.5(a) shows the evolution of the $deadline\_failure\_rate$ over time as the number of UE increases. At t=6s and t=12s, the $deadline\_failure\_rate$ increases since 4 more UEs have joined

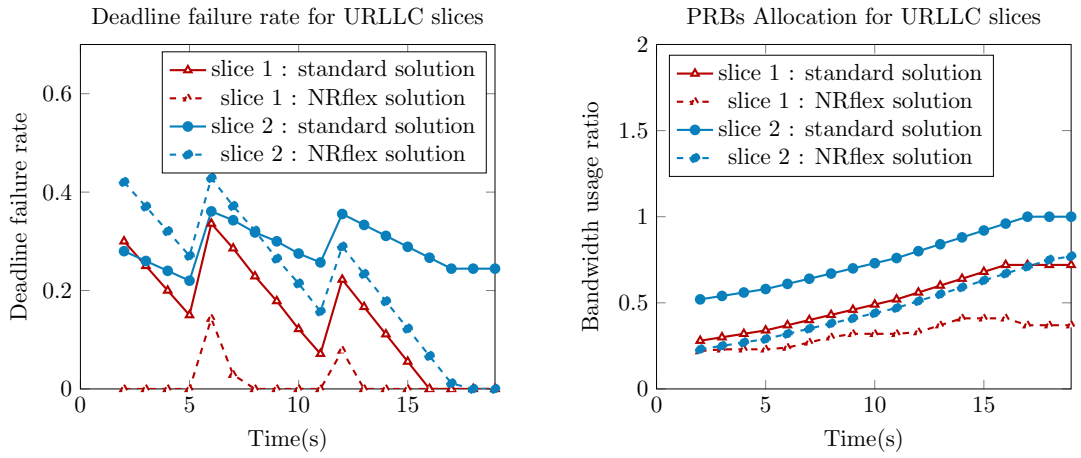
Table 3.2: Slices requirements

slice	Max Latency	desired throughput per UE
embb1	-	0.9 mbps
embb2	-	1.5 mbps
urllc1	5 ms	-
urllc2	1 ms	-

Table 3.3: Traffic simulation parameters

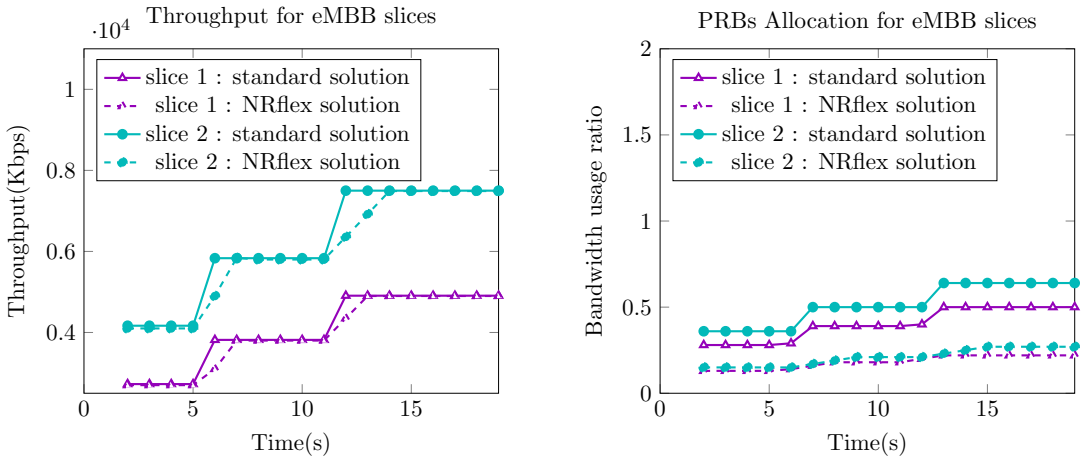
Slice	Inter-arrival time	Packet size
urllc1	4 ms	800 Bytes
urllc2	2 ms	400 Bytes
embb1	70 ms	4596 Bytes
embb2	70 ms	6516 Bytes

the system. However, the preadaptation phase in NRflex decreases the $deadline\_failure\_rate$ in an incremental way. Hence, it is important to recall that the preadaptation phase is realized by RIC, which dynamically computes the size of a BWP dedicated to a slice according to gNB's feedbacks (Algorithm 1).
Figure 3.5(b) shows the evolution of the number of allocated PRBs for each slice. We note that for the first slice, which requires a latency of 5 ms, NRflex ensures a lower $deadline\_failure\_rate$ (Figure 3.5(a)) than the standard solution. Moreover, NRflex adapts itself quickly to the new cell load by adding more PRBs to the slice. We also observe that NRflex meets the slices' required latency deadline with a smaller number of PRBs (Figure 3.5(b)). For the second slice, which requires a latency of 1 ms, we remark that, at t=17s (Figure 3.5(b)), the standard solution can not reduce the $deadline\_failure\_rate$ (Figure 3.5(a)) as it consumed all the bandwidth dedicated for numerology 2 (Table 3.1). In comparison, NRflex is able to reduce the $deadline\_failure\_rate$ to zero after 5s of adaptation with the same bandwidth size. We can also see that our NRflex allocates lesser PRBs (Figure 3.5(b)) to satisfy the slice latency requirement. Thus, we can argue that combining deadline-aware scheduling with our resource allocation strategy can achieve very low latency while using fewer radio resources.
Figure 3.6(a) shows the throughput evolution over time. We remark that both approaches achieve the same throughput, which is different from the desired throughput. We argue this because the practical throughput is calculated on the gNB, while the desired throughput is just a theoretical throughput. In figure 3.6(b), we see that NRflex consumes fewer PRBs to meet the same throughput, compared to the standard solution. As the number of UEs increases, the difference between the two approaches increases in terms of used PRBs. We explain this by the fact that UEs do not require all the PRBs allocated by the standard solution at each slot; hence, the needed PRBs depend on the state of the traffic queues (LCs) of the UEs and their CQI. UEs, which have a good CQI and a small amount of data in their LC, do not need many PRBs even if the desired throughput is significant. That proves the pertinence to calculate the number of needed PRBs from an entity close to the gNB, which is aware of the state of the queues (the CQI of the UEs) and has a global vision to manage the radio resources and distribute them among the RAN slices. Therefore more UEs can be connected, and hence more RAN slices can be created.
It is worth noting that NRflex may introduce overhead due to the exchanged messages (number of messages x message's size) between RIC and gNBs. For each 1s interval, gNB sends a message containing a list of slices with the average of used PRBs and calculated KPIs (i.e., $deadline\_failure\_rate$ and $throughput$), and receives a message containing a list of slices with



(a) (b)

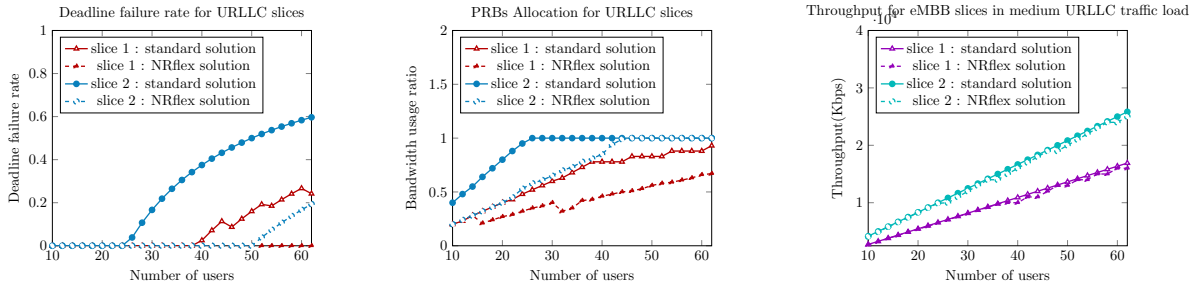
Figure 3.5: URLLC performance and resource allocation over time



(a) (b)

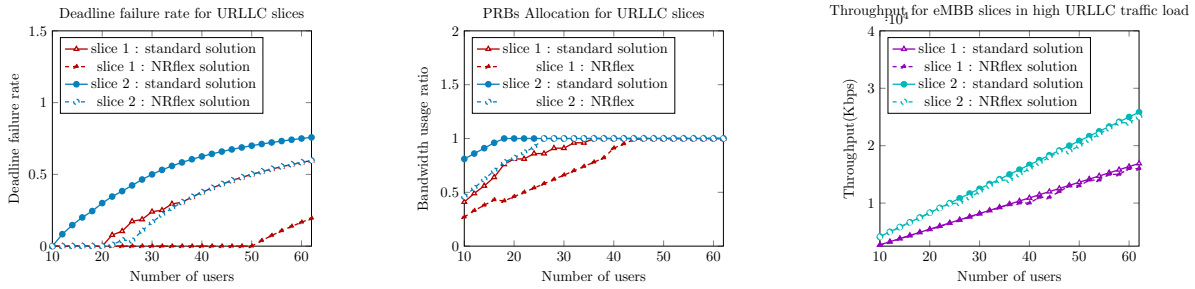
Figure 3.6: eMBB performance and resource allocation over time

their new BWP configuration. However, these messages' impact is very low in terms of needed bandwidth, as their size is very small. Hence, only a few bytes are transmitted periodically. In the second scenario, we varied the URLLC traffic load to see the limit of both approaches. By traffic load, we mean the inter-arrivals time of packets and packet size. Figure 3.7(a) shows $deadline\_failure\_rate$ evolution according to the number of UEs (UEs are equally distributed between the two slices) with a medium URLLC traffic load for each UE. NRflex can handle up to 50 UEs in numerology 2 and up to 60 UEs in numerology 1 with a $deadline\_failure\_rate$=0 (no packet is lost due to deadline exceeded). In contrast, the standard solution can handle up to 25 UEs in numerology 2 and 40 UEs in numerology 1 before the $deadline\_failure\_rate$ increases. For high URLLC traffic load (Figure 3.8(a)), NRflex can handle up to 50 UEs that require a 5 ms latency and 20 UEs that require a 1ms latency, while the other approach can handle only 20 UEs and 10 UEs, respectively. At the same time, NRflex ensures that the eMBB throughput is sustained for both medium and high URLLC traffic loads (Figures 3.7(c), 3.8(c)).



(a) (b) (c)

Figure 3.7: Medium URLLC traffic load



(a) (b) (c)

Figure 3.8: High URLLC traffic load

Table 3.4: URLLC traffic load

Index	average Inter-arrival time (ms)	Average packet size (Bytes)
1	5	200
2	4	280
3	3	360
4	2	440
5	1	520

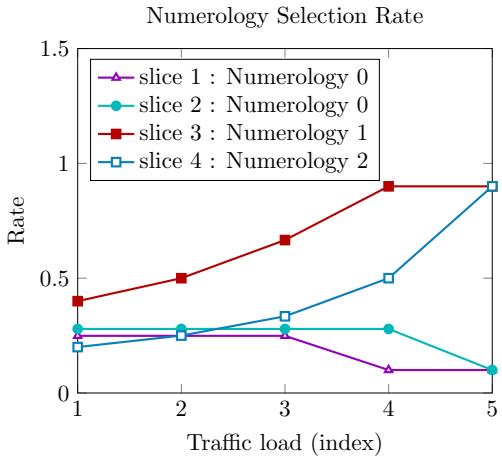


Figure 3.9: Numerology selection over URLLC traffic load

In the third scenario (Figure 3.9, we varied the URLLC traffic load (Table 3.4) to show the impact of URLLC slices on eMBB slices, when URLLC traffic becomes dominant. As the traffic

becomes more intensive, it requires more active slots in the URLLC numerology; hence, fewer slots for the eMBB numerology. (only one numerology active at a time slot). We argue this by the fact that NRflex prioritizes URLLC traffic (Algorithm4). We remark that from index = 3 (see Table 3.4), URLLC traffic starts impacting eMBB traffic. However, the results show that even with the higher loads, eMBB slices are still supported (eMBB numerology is selected).

3.6 Conclusion

In this chapter, we have introduced a 5G NR Network Slicing framework aligned with O-RAN architecture. This framework, namely NRflex, enables UEs to benefit from multi-service applications and leverages 5G NR numerologies to achieve URLLC service latencies while respecting eMBB services throughput. NRflex is divided into two parts: (1) one executed by RIC near-RT to dynamically compute the size of BWP to be dedicated to a slice; (2) one executed by gNBs that periodically decides for each UE the active BWP to be used and the number of PRBs assigned to it. Numerical results show that NRflex succeeds in meeting service requirements, scheduling more UEs, and optimizing PRBs allocation compared to the standard solution. Besides, NRflex architecture offers modularity that allows a flexible modification of the different introduced entities, i.e., schedulers, the BWP manager and the pre-processors.

Chapter 4

DRL-RS: Deep Reinforcement Learning (DRL)-based scheduler in shared bandwidth environment

4.1 Introduction

In this chapter, we address the challenging problem of resource management in 5G NR featuring network slicing by first modeling it as a mixed-integer linear program taking into account: *i*) multiple numerology in the same bandwidth while avoiding the INI; *ii*) multiple slices attached per UE; *iii*) different throughput and latency requirements per slice. Then, we model the problem as a MILP, we analyze its complexity, and we prove that the problem is NP-hard; i.e., solving it will take a considerable amount of time, which is not tolerable when scheduling radio resources in real-time. RL that can be seen as a learning, heuristic search strategy when it is applied to optimization problems. Accordingly, we formalize this problem in the RL framework and propose a DRL-based approach that aims to select the numerology to be used and the number of resources allocated per UE at each time slot during a time window while taking into account the channel quality of the UE. Specifically, we designed the DRL-scheduler to be independent of the number of UEs in the system. We have modeled the DRL state to make the solution scalable for larger bandwidths covering both FR1 and FR2 frequency bands, with a bandwidth up to 400 MHz, which correspond to the usage of the mmWave bands.
The key contributions of this chapter are:

- Model the radio resource management problem in 5G NR featuring network slicing using MILP and prove it is a NP-hard problem. The novelty of the problem formulation compared to the state of the art models, such as [59], [60], [61], [62], consists in selecting both the numerology and number of PRBs dynamically in each time slot to satisfy slices throughput and latency requirements while supporting multiple slices per UE.

- Devise a solution to solve the problem in a polynomial time. The solution is based on Machine Learning (ML), namely DRL. The novelty of the DRL approach is its ability to: (i) solve unseen instances of the problem in reasonable time, (ii) scale with the bandwidth and the number of UEs and (iii) dynamically select the numerology per UE while scheduling the PRBs in time and frequency.

- Extensive simulation campaign to compare the two approaches: (*i*) the LP approach (*ii*) the DRL approach, and evaluate their performances to guarantee the network slice require-

ments. We observed that approach (*i*) (i.e., solving the MILP problem using Gurobi) takes exponential time regarding the number of UEs while approach (*ii*) reduces the execution time down to 1ms. Furthermore, both approaches are able to meet the requirements of the slices in terms of throughput and latency considering different network configurations.

4.2 Related works

The authors in [59] propose a two-level RAN slicing approach based on the O-RAN architecture to allocate the communication and computation RAN resources among uRLLC end-devices. They modeled the resource slicing problem as a single-agent MDP and designed a DRL algorithm to solve it. However, they considered only numerology 0. Further, they did not provide details on the model scalability in terms of bandwidth and number of UEs; up to 16 UEs and a fixed bandwidth that was not defined in the paper. The work in [60] designed a DRL model able to solve the radio resource scheduling problem in 5G networks under different fixed numerology settings. Nevertheless, the proposed solution did not consider using different numerology under the same bandwidth. The authors in [61] formulated a binary non-convex problem that maximizes the aggregate capacity of multiple network slices. The model exploits the channel fading statistic to provide a spectrum allocation that minimizes the INI. The authors leveraged DRL to design a model-free solution computation. However, the presented problem assumed that all UEs belong to one and only one slice and have the same numerology, making the system less flexible. In [62], the authors formulated the resource allocation problem as a non-linear binary program and proved its NP-hardness. They modeled the problem as an MDP. They leveraged the exponential-weight algorithm for exploration and exploitation (EXP3) as well as the Multi-Agent DQN algorithm to solve the single-agent MDP and the multi-agent MDP, respectively. But, they did not consider multiple numerology and the high dimension of the problem (only eight UEs were considered).
The work in [63] optimized resource allocation with flexible numerology in the frequency domain and variable frame structure in the time domain, with different types of requirements. They prove the NP-hardness of the problem and propose a scalable optimization algorithm based on LP and Lagrangian Duality (LD). However, they did not take into account the interference caused by the mixed numerology environment. Besides, the gap between the optimal solution and the proposed solution depends on the required throughput (30% for a 0.5 Mbps rate), making the solution unsuitable for high throughput requirements as expected in 5G. In [64], the authors modeled the resource allocation problem for mixed-numerology systems as a multi-objective optimization problem aiming to increase cell throughput, maintain fairness, and minimize the delay and packet loss. They proposed a heuristic-based solution to perform numerology multiplexing as well as resource allocation taking into account the QoS and channel quality. Nevertheless, the latency requirement was ignored and the comparison between the optimal solution of their model and the solution of the proposed method was not addressed. Work in [65] designed a random forest-based decision algorithm to accomplish the numerology selection for each service. Then, the numerology selection results will be the basis of system resource scheduling and allocation. However, the solution did not consider the frequency efficiency and did not model the problem formally. Also, they did not compare their solution with the optimal solution of their problem. In [66], the authors studied radio resource allocation for mMTC services based on a mixed-numerology system. They followed an efficient heuristic approach to meet diverse QoS requirements of the Machine-to-Machine (M2M) applications while achieving spectral efficiency. However, they ignored the time domain in the problem definition.
All the above works were investigated only on small instances of the problem (small bandwidths

and small number of UEs), and did not take into consideration UEs belonging to multiple slices with different requirements. Furthermore, most of DRL based solutions ignored the mixed numerology environment for the sake of reducing the model complexity. In our work, we consider large instances of the problem with up to 400 MHz of bandwidth, which is the maximum bandwidth in current 5G NR standards. We adapted the state representation of the DRL solution to be scalable regardless of the problem size. In addition, we considered having multiple numerology in the same bandwidth, divided by BWPs, while UEs can switch numerology to satisfy their heterogeneous needs.

4.3 System Model and Problem Formulation

4.3.1 System model

We consider a network that consists of a set of UEs $\mathcal{N}$ that compete to access the radio resources. The UEs use different network slices, each characterized by different objectives, characteristics, and Service Level Agreement (SLA). Each UE can be a member of one or multiple network slices simultaneously. The SLA of a slice consists of a set of KPIs, with each KPI having a target value. In our system, we consider two KPIs: (i) throughput, (ii) latency. For instance, uRLLC slices have a lower target KPI value of latency, while eMBB slices have a higher target KPI value of throughput. The radio resources are divided on the UEs on time window Δ_T and frequency bandwidth Δ_F. Formally, PRBs should be assigned to the UEs within the matrix of shape $\Delta_T \times \Delta_F$ illustrated in Figure 4.1. Let δT and δF be the minimum allocation units in time and frequency domains, respectively. Formally, $\delta T = 2^{-\mu_{max}}$ ms and $\delta F = 12 * 15 * 2^{\mu max}$ Hz, where μ_{max} is the maximum numerology in the system. Let $\mathcal{N}$ denote the set of UEs in the network. Each UE has a set of time slots to transmit both UL and DL traffics. Let $\mathcal{K}$ denote the maximum number of time slots that can be assigned to a UE. To transmit their data, UEs use a set of numerology $\mathcal{M} = \{0, 1, 2, 3, 4\}$. At a given time slot, the UE should use one and only one numerology. However, it can use more than one resource blocks that should be contiguous for the sake of performance.

Table 4.1: Summary of Notations.

Notation	Description
$\mathcal{N}$	The set of UEs existing in the network.
Ω	The maximum number of PRBs assigned to the same UE at the same time slot.
Δ_T	The time slot window of the resource blocks.
Δ_F	The frequency band of the resource blocks.
$\mathcal{M}$	The set of numerology.
μ	A specific numerology. Formally, $\mu \in \mathcal{M}$.
$\mathcal{T}_i^k$	An integer variable that denotes the starting time of the k^{th} time slot of the UE $i \in \mathcal{N}$.
$\mathcal{F}_i^k$	An integer variable that denotes the starting frequency assigned to the UE $i \in \mathcal{N}$ at time slot k.
$\mathcal{X}_{i,\omega}^{k,\mu}$	A decision Boolean variable that shows if a UE i at the time slot k uses ω contiguous PRBs of numerology μ. ω takes values in the range from 1 to Ω.
$\mathcal{Y}_{i,j}^{k,l}$	A decision Boolean variable that shows if the k^{th} and l^{th} time slots of UE i and j overlap.

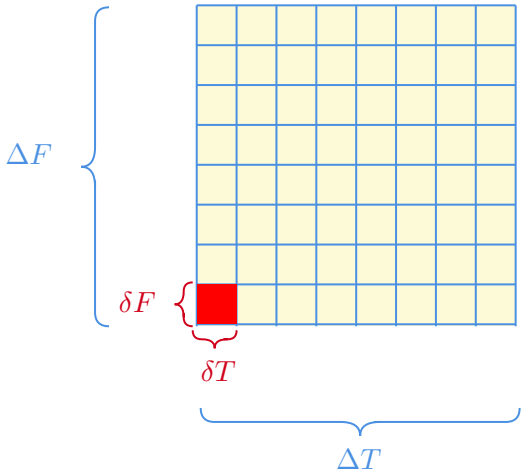


Figure 4.1: Resource matrix illustration for the 5G NR featuring network slicing resource management Model

The size of a resource block in terms of slot duration $\mathcal{E}(\mu)$ (in ms) and frequency $\mathcal{G}(\mu)$ (in Hz) of a numerology μ are defined as follows:

$$\mathcal{E}(\mu) = 2^{-\mu} \tag{4.1}$$

$$\mathcal{G}(\mu) = 12 \times 15 \times 2^{\mu} \tag{4.2}$$

As we have explained previously, a UE at a given time slot can have one or multiple contiguous PRBs ($\omega \times \mathcal{G}(\mu)$) that use the same numerology μ. The PRBs of the same UE or different UEs should never overlap. Formally, a UE i never shares the frequency bound with another UE j at the same time.

4.3.2 Problem formulation and optimal solution design

For the sake of readability, all the symbols and optimization variables are summarized in the table 4.1. Let $\mathcal{T}_i^k$ be an integer variable that denotes the starting time of the k^{th} time slot of the UE $i \in \mathcal{N}$. Note that not all the time slots $\{1, 2, \cdots \mathcal{K}\}$ should be assigned to the UE i. If a time slot k is not assigned to the UE i, then $\mathcal{T}_i^k = -1$. Similarly, let $\mathcal{F}_i^k$ be an integer variable that denotes the starting frequency assigned to the UE $i \in \mathcal{N}$ at time slot k. Each UE can have maximum Ω PRBs at the same time slot. Moreover, all the PRBs of the same UE at the same time slot should use the same numerology. If a frequency $\mathcal{F}_i^k$ is not used by the UE i, then $\mathcal{F}_i^k = -1$.
Formally, we have the following two constraints that ensure the time slot and frequency starting value should be higher or equal to -1:

$$\forall i \in \mathcal{N}, \forall k \in \{1, 2, \cdots \mathcal{K}\} : \mathcal{T}_i^k \geq -1 \tag{4.3}$$

$$\forall i \in \mathcal{N}, \forall k \in \{1, 2, \cdots \mathcal{K}\} : \mathcal{F}_i^k \geq -1 \tag{4.4}$$

Let $\mathcal{X}_{i,\omega}^{k,\mu}$ a decision Boolean variable that shows if a UE i at the time slot k uses ω contiguous PRBs of the same numerology μ. At a given time slot k, only one numerology can be allocated

to UE i. However, many resource blocks can be assigned to the same UE. This constraint is captured thanks to the constraint (4.5).

$$\forall i \in \mathcal{N}, \forall k \in \{1, 2, \cdots \mathcal{K}\} : \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{X}_{i,\omega}^{k,\mu} \leq 1 \tag{4.5}$$

The UE i uses the numerology μ at the time slot k if and only if $\sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{X}_{i,\omega}^{k,\mu} = 1$.
The time slot duration allocated to UE i transmission must not exceed Δ_T. In this constraint, both the k^{th} starting time slot (i.e., $\mathcal{T}_i^k$) and time slot duration $\mathcal{E}(\mu)$ using the numerology μ are considered.

$$\begin{aligned} &\forall i \in \mathcal{N}, \forall k \in \{1, 2, \cdots \mathcal{K}\} : \\ &\quad \mathcal{T}_i^k + \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{E}(\mu) \times \mathcal{X}_{i,\omega}^{k,\mu} \leq \Delta_T \end{aligned} \tag{4.6}$$

From another side, the frequency should not exceed Δ_F. In constraint (4.7), at each time slot $\mathcal{T}_i^k$, both the starting frequency bound $\mathcal{F}_i^k$ and the amount of frequency bound ($\omega \times \mathcal{G}(\mu)$) are considered.

$$\begin{aligned} &\forall i \in \mathcal{N}, \forall k \in \{1, 2, \cdots \mathcal{K}\} : \\ &\quad \mathcal{F}_i^k + \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \omega \times \mathcal{G}(\mu) \times \mathcal{X}_{i,\omega}^{k,\mu} \leq \Delta_F \end{aligned} \tag{4.7}$$

$\mathcal{T}_i^k$ should equal to -1 if time slot k is not allocated to UE i.

$$\begin{aligned} &\forall i \in \mathcal{N}, \forall k \in \{1, 2, \cdots \mathcal{K}\} : \\ &\quad \mathcal{T}_i^k \leq -1 + \Phi \times \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{X}_{i,\omega}^{k,\mu} \end{aligned} \tag{4.8}$$

, such that Φ is a big number $\Phi \approx +\infty$.
Based on (4.3) and (4.8), $\mathcal{T}_i^k$ equals to -1 if no numerology is used by that time slot ($\sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{X}_{i,\omega}^{k,\mu} = 0$).
Similarly, the frequency $\mathcal{F}_i^k$ should equal -1 if it does not use any numerology at the time slot $\mathcal{T}_i^k$.

$$\begin{aligned} &\forall i \in \mathcal{N}, \forall k \in \{1, 2, \cdots \mathcal{K}\} : \\ &\quad \mathcal{F}_i^k \leq -1 + \Phi \times \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{X}_{i,\omega}^{k,\mu} \end{aligned} \tag{4.9}$$

Based on (4.4) and (4.9), $\mathcal{T}_i^k$ equals to -1 if no numerology is used by that time slot.

A UE should not use two numerology at the same time. Also, the time slots should not overlap with each other. In our solution, the time slots of a UE are defined from the smaller to the last. Then, the remaining time slots should have only -1.

$$\begin{aligned}
&\forall i \in \mathcal{N}, \forall k \in \{1, 2, \cdots \mathcal{K}-1\}: \\
&\mathcal{T}_i^k + \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{E}(\mu) \times \mathcal{X}_{i,\omega}^{k,\mu} \leq \\
&\mathcal{T}_i^{k+1} + \Phi \times (1 - \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{X}_{i,\omega}^{k+1,\mu})
\end{aligned} \tag{4.10}$$

Equation 4.11 ensures that only the first slots assigned for a UE i. In our solution, we fill the first slots before moving to the last ones.

$$\forall i \in \mathcal{N}, \forall k \in \{1, 2, \cdots \mathcal{K}-1\}: \sum_{\mu \in \mathcal{M}} \mathcal{X}_i^{k+1,\mu} \leq \sum_{\mu \in \mathcal{M}} \mathcal{X}_i^{k,\mu} \tag{4.11}$$

Equation 4.12 ensures that two UEs i and j should not use the same frequency resources if their time slots overlap.

$$\begin{aligned}
&\forall i, j \in \mathcal{N}, i \neq j, \forall k, l \in \{1, 2, \cdots \mathcal{K}-1\}: \\
&\text{If } \mathcal{T}_j^l \leq \mathcal{T}_i^k \leq \mathcal{T}_j^l + \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{E}(\mu) \times \mathcal{X}_{i,\omega}^{k,\mu}: \\
&\quad \mathcal{F}_j^l + \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \omega \times \mathcal{G}(\mu) \times \mathcal{X}_{j,\omega}^{l,\mu} < \mathcal{F}_i^k \\
&\quad \text{OR} \\
&\quad \mathcal{F}_i^k + \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \omega \times \mathcal{G}(\mu) \times \mathcal{X}_{i,\omega}^{k,\mu} < \mathcal{F}_j^l
\end{aligned} \tag{4.12}$$

Unfortunately, the above inequality is not linear. In order to convert the optimization problem to a linear integer program, we replace the constraint (4.12) by the following constraints and variables:
First, we define a Boolean variable $\mathcal{Y}_{i,j}^{k,l}$ that shows if the k^{th} and l^{th} time slots of UE i and j overlap, respectively. Then, we define the following constraints:

$$\begin{aligned}
&\forall i, j \in \mathcal{N}, i \neq j, \forall k, l \in \{1, 2, \cdots \mathcal{K}\}: \\
&\mathcal{T}_j^l \leq \mathcal{T}_i^k + \Phi \times (1 - \mathcal{Y}_{i,j}^{k,l}) \\
&\mathcal{T}_i^k \leq \mathcal{T}_j^l + \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{E}(\mu) \times \mathcal{X}_{j,\omega}^{l,\mu} + \Phi \times (1 - \mathcal{Y}_{i,j}^{k,l}) \\
&\mathcal{T}_i^k + \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{E}(\mu) \times \mathcal{X}_{i,\omega}^{k,\mu} < \mathcal{T}_j^l + \Phi \times (\mathcal{Y}_{i,j}^{k,l} + \mathcal{Z}_{i,j}^{k,l}) \\
&\mathcal{T}_j^l + \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{E}(\mu) \times \mathcal{X}_{j,\omega}^{l,\mu} < \mathcal{T}_i^k + \Phi \times (\mathcal{Y}_{i,j}^{k,l} + 1 \\
&- \mathcal{Z}_{i,j}^{k,l})
\end{aligned} \tag{4.13}$$

, such that $\mathcal{Z}_{i,j}^{k,l}$ is a decision Boolean variable that should be fixed by the system.
From (4.13), $\mathcal{Y}_{i,j}^{k,l} = 1$ iff $\mathcal{T}_j^l \leq \mathcal{T}_i^k \leq \mathcal{T}_j^l + \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1,\cdots \Omega\}} \mathcal{E}(\mu) \times \mathcal{X}_{j,\omega}^{l,\mu}$. Otherwise, $\mathcal{Y}_{i,j}^{k,l} = 0$, the time slots do not overlap. In case the time slots overlap, we have to ensure that they do not use the same frequency resources.

$$
\begin{aligned}
&\forall i, j \in \mathcal{N}, i \neq j, \forall k, l \in \{1, 2, \cdots \mathcal{K}\} : \\
&\mathcal{F}_j^l + \sum_{\mu \in \mathcal{M}} \mathcal{I}_{i,j}^{k,l,\mu} \times G + \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1,\cdots \Omega\}} \omega \times \mathcal{G}(\mu) \times \mathcal{X}_{j,\omega}^{l,\mu} < \\
&\mathcal{F}_i^k + \Phi \times (1 - \mathcal{Y}_{i,j}^{k,l} + \mathcal{W}_{i,j}^{k,l}) \\
&\mathcal{F}_i^k + \sum_{\mu \in \mathcal{M}} \mathcal{I}_{i,j}^{k,l,\mu} \times G + \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1,\cdots \Omega\}} \omega \times \mathcal{G}(\mu) \times \mathcal{X}_{i,\omega}^{k,\mu} < \\
&\mathcal{F}_j^l + \Phi \times (2 - \mathcal{Y}_{i,j}^{k,l} - \mathcal{W}_{i,j}^{k,l})
\end{aligned}
\tag{4.14}
$$

, such that $\mathcal{W}_{i,j}^{k,l}$ is a Boolean variable that should be fixed by the system to ensure that the frequency i and j do not overlap when their time slots do.
Moreover, UEs sharing contiguous frequencies and having different numerology should be separated by a guard band G to avoid the INI. $\mathcal{I}_{i,j}^{k,l,\mu}$ is a Boolean variable fixed by the system: if $\mathcal{I}_{i,j}^{k,l,\mu} = 1$ a guard band is needed between UE i and j during slots k and l when they are not using the same numerology μ (Equation 4.15).

$$
\begin{aligned}
&\forall i, j \in \mathcal{N}, i \neq j, \forall k, l \in \{1, 2, \cdots \mathcal{K}\}, \forall \mu \in \mathcal{M} : \\
&\mathcal{I}_{i,j}^{k,l,\mu} \geq \sum_{\omega \in \{1,\cdots \Omega\}} \mathcal{X}_{j,\omega}^{l,\mu} - \sum_{\omega \in \{1,\cdots \Omega\}} \mathcal{X}_{i,\omega}^{k,\mu} \\
&\mathcal{I}_{i,j}^{k,l,\mu} \geq \sum_{\omega \in \{1,\cdots \Omega\}} \mathcal{X}_{i,\omega}^{k,\mu} - \sum_{\omega \in \{1,\cdots \Omega\}} \mathcal{X}_{j,\omega}^{l,\mu}
\end{aligned}
\tag{4.15}
$$

Each UE i has a maximum latency Δ_i.

$$
\begin{aligned}
&\forall i \in \mathcal{N}, \forall k \in \{1, 2, \cdots \mathcal{K}\} : \\
&\qquad \mathcal{T}_i^k + \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1,\cdots \Omega\}} \mathcal{E}(\mu) \times \mathcal{X}_{i,\omega}^{k,\mu} < \Delta_i
\end{aligned}
\tag{4.16}
$$

Each UE i has a minimum throughput, which is translated to the number of required resource blocks N_PRB_i taking the MCS as input for each UE i during the time window ΔT. It should be noted that a way to derive the number of PRBs using MCS and throughput is presented in [24]. The system that solves our model is responsible for providing this information as input, and can be dynamic between different time windows.

$$
\forall i \in \mathcal{N} : \sum_{k \in \{1,\cdots \mathcal{K}\}} \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1,\cdots \Omega\}} \omega \times \mathcal{X}_{i,\omega}^{k,\mu} \geq N\_PRB\_i \tag{4.17}
$$

In our solution, we consider network slicing, such as one UE can have multiple network slices. To do so, a (physical) UE is divided into multiple logical UEs. Each logical UE represents a

slice belonging to a UE. Therefore, a UE with multiple slices will have as many logical UEs as slices. For simplicity, we consider logical UEs like UEs with additional constraints (4.18 and 4.19), with logical UEs belonging to the same UE forming a group. Formally, a UE $i \in \mathcal{N}$ is a logical representation of a slice belonging to UE $k \in \mathcal{N}_{ph}$ where $\mathcal{N}_{ph}$ denotes the set of physical UEs. For instance, two logical presentations of a UE $i, j \in \mathcal{N}$ and $i \neq j$ can belong to the same physical UE $k \in \mathcal{N}_{ph}$. To prevent a UE $i \in \mathcal{N}$ from using different numerology simultaneously, we have considered G, whereby the same logical presentations of the same physical UE create a group $g \in G$. Constraints 4.18 and 4.19 jointly ensure that the logical presentations of the same physical UE do not use different numerology when their time slots overlap.

$$\begin{aligned} &\forall g \in \mathcal{G}, \forall i, j \in g, i \neq j, \forall k, l \in \{1, 2, \cdots \mathcal{K}\}, \forall \mu \in \mathcal{M} : \\ &\sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{X}_{j,\omega}^{l,\mu} - \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{X}_{i,\omega}^{k,\mu} \leq 1 - \mathcal{Y}_{i,j}^{k,l} \\ &\mathcal{Y}_{i,j}^{k,l} - 1 \leq \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{X}_{j,\omega}^{l,\mu} - \sum_{\omega \in \{1, \cdots \Omega\}} \mathcal{X}_{i,\omega}^{k,\mu} \end{aligned} \tag{4.18}$$

$$\begin{aligned} &\forall g \in \mathcal{G}, \forall i, j \in g, i \neq j, \forall k, l \in \{1, 2, \cdots \mathcal{K}\} : \\ &\mathcal{T}_i^k - \mathcal{T}_j^l \leq \Phi \times (1 - \mathcal{Y}_{i,j}^{k,l}) \\ &\Phi \times (\mathcal{Y}_{i,j}^{k,l} - 1) \leq \mathcal{T}_i^k - \mathcal{T}_j^l \end{aligned} \tag{4.19}$$

Finally, the objective is to maximize the bandwidth utilization. In the optimization model, we do not consider priorities between UEs. However, the priority can be easily considered by a slight update of the equation 4.20.

$$\textbf{max} \sum_{i=1}^{i=\mathcal{N}} \sum_{k=1}^{k=\mathcal{K}} \sum_{\mu \in \mathcal{M}} \sum_{\omega \in \{1, \cdots \Omega\}} \omega \times \mathcal{X}_{i,\omega}^{k,\mu} \tag{4.20}$$

Now, we are able to formulate the final optimization problem as follows:
Eq. (4.20)
S.t,
Eq.: (4.3), (4.4), (4.5), (4.6), (4.7), (4.8), (4.9), (4.10), (4.11), (4.13), (4.14), (4.15), (4.16), (4.17), (4.18), (4.19).

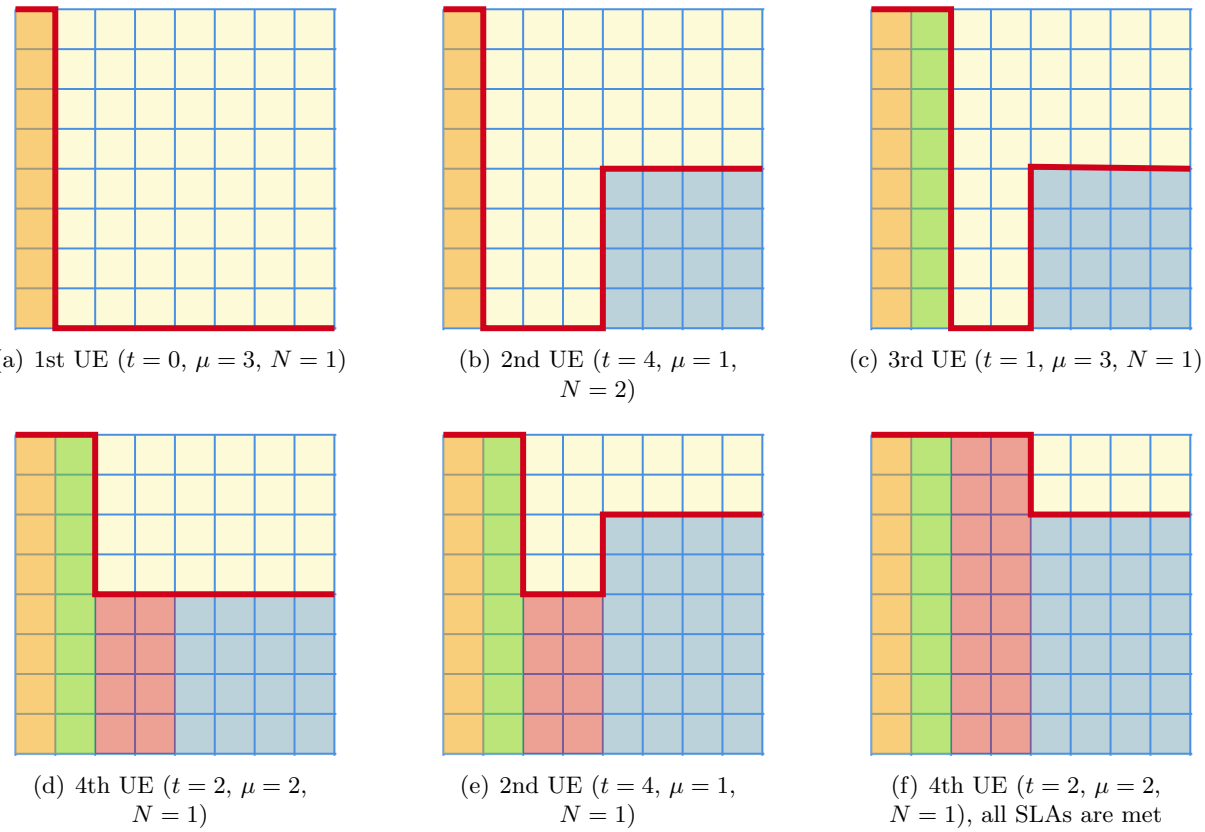
Theorem 1. *Radio resource allocation problem in 5G NR is NP-hard problem.*

Proof. Let P1 denotes the problem of filling the radio resource matrix with BWPs $i \in \{1..n\}$. Let w_i denotes the frequency occupied by each BWP i. Each BWP i is dedicated to a UE j and has a start time t. For the sake of simplicity and without loss of generality, we assume that only one numerology can be used, and the BWPs that exceed UE j latency SLA can be omitted. Let v_i denotes the size of the BWP i. Meanwhile, let P2 denotes the knapsack problem, which is defined as follows: Given a set of objects, each of which has a weight and value, the problem consists in determining the number of each item that should be put in the knapsack so that the total weight does not exceed a specific weight and the total value is maximized. The optimization of knapsack problem is well known in the literature that it is NP-hard problem. To prove that P1 is NP-hard, it is sufficient to proof that the knapsack problem P2 would be reduced to P1 in a polynomial time. If P2 is reduced to P1 in a polynomial time and P1 is

not NP-hard, then P2 is also not NP-hard, which is a contradiction. P2 can be reduced in a polynomial time to P1 by formulating P2 as follows: The total weight that can be handled by the knapsack is ΔF and ΔT (two dimensions) and the set of items that should be put in knapsack is the set of BWPs i, such that the sum of v_i is maximized. □

4.4 DRL-RS general overview

Despite seeing great improvement throughout the years in the ability to find optimal solutions for bigger and more complex problems, the time and the computational power needed to find a solution remain an important bottleneck, making this type of algorithms impractical to solve real-world problems. RL intends to learn the skills required to solve the problem more holistically rather than trying to find an optimal solution for one specific configuration of the problem. Where the other combinatorial optimization solving methods will need to solve the problem from scratch every time, the RL approach leverages what it has learned before and the skills it has acquired to quickly provide a solution to a new instance of the problem. Radio resource management problem in 5G NR can be seen as a sequential decision-making problem where an agent, namely DRL-RS, allocates PRBs and selects numerology for each UE and time slot during ΔT time window. DRL-RS considers the environment as a $\Delta F \times \Delta T$ matrix and aims to fill it by allocating PRBs to a dynamic number of UEs in the system. DRL-RS takes the MCS to use during ΔT as input to ensure the link adaptation for dynamic channels.



(a) 1st UE ($t = 0$, $\mu = 3$, $N = 1$)

(b) 2nd UE ($t = 4$, $\mu = 1$, $N = 2$)

(c) 3rd UE ($t = 1$, $\mu = 3$, $N = 1$)

(d) 4th UE ($t = 2$, $\mu = 2$, $N = 1$)

(e) 2nd UE ($t = 4$, $\mu = 1$, $N = 1$)

(f) 4th UE ($t = 2$, $\mu = 2$, $N = 1$), all SLAs are met

Figure 4.2: Example of resource allocation using DRL-RS

As each UE can belong to multiple slices, DRL-RS should satisfy the throughput and the latency of the SLA for each slice. The throughput SLA is the minimum throughput that needs to be

achieved by a UE for that slice, while the latency SLA is the maximum latency that a UE can not exceed when serving that slice. To allow a UE to belong to multiple slices, we introduce the concept of virtual UEs. Formally, each UE consists of a set of virtual UEs that belong to only one slice. Virtual UEs belonging to the same UE are organized in groups. The time slots of members of the same group should neither overlap in time nor use different numerology. From this point, we mention by UE a real or virtual UE indifferently. In this scope, DRL-RS loops over the active UEs (i.e., UEs having data in their transmission queues and their SLA is not satisfied yet) until the resource matrix is filled or all the UE's SLAs are met. In each iteration, the scheduler chooses a time slot t, a numerology μ, and a number of resources N for the current UE. The three parameters form a rectangle of shape $2^{\mu_{max}-\mu} * (N * 2^{\mu})$ which will be stacked in the resource matrix at time slot t.
We illustrate how DRL-RS works through a simple example illustrated in Figure 4.2. We consider a $1.44MHz \times 1ms$ matrix shared between 4 UEs. Each UE has a latency SLA of 0.125ms, 1ms, 0.25ms, 0.5ms, and a throughput SLA that can be met with 1, 3, 1, 2 PRBs, respectively. The x-axis represents time slots where the unit is related to the maximum numerology in the system (i.e., 3 in this example) and the y-axis represents frequency PRBs where the unit is related to the minimum numerology in the system (i.e., 0 in this example). In the first iteration, for the first UE, DRL-RS allocates 1 PRBs with numerology 3 at t = 0. The selection of numerology 3 allows the first UE to respect its latency SLA. Then, DRL-RS allocates 2 PRBs for the 2^{nd} UE at t = 4 with numerology 1. This choice will allow the third and fourth UEs to respect their latency SLA, which shows that DRL-RS gives the current UE an insight into the state of other UEs. After that, the third UE takes 1 PRB using numerology 3 at $t = 1$, while the 4^{th} UE takes 1 PRB at $t = 2$ using numerology 2. The stop condition is not met yet; hence, DRL-RS continues looping over the active UEs that did not meet their SLAs, i.e., the 2^{nd} and 4^{th} UEs. The 2^{nd} UE takes one more PRBs at $t = 4$ with numerology 1. Finally, the 4^{th} UE takes one PRB at $t = 2$ using numerology 2. At this step, all UEs' SLAs are met, and thus DRL-RS finished the resource allocation process and successfully fulfilled all the slices' SLAs.

4.5 DRL-RS design

In the balance of this section, we model the radio resource management problem in the RL framework by designing the state, the reward and the action of the DRL-RS agent.

4.5.1 State

The state S^i observed by the model when serving UE i is composed of 4 vectors, which are F, T_i, O_i and E_i, respectively. Vectors F and T_i contain the frequency boundary (the red line in Figure 4.2) and the numerology used by UE i for each time slot t, respectively. If a UE is not using any numerology at time t, $T_{i,t}$ is set to -1. Meanwhile, vectors O_i and E_i contain the information about SLA of the current UE and the other UEs, respectively. O_i contains two values, O_i^{thg} measuring the throughput SLA and O_i^{lat} measuring the latency SLA. O_i^{thg} indicates the rate of the achieved throughput over the throughput SLA. The bigger O_i^{thg} is, the better performances of UE i becomes in terms of throughput. Since a UE can achieve a throughput higher than its throughput SLA, O_i^{thg} can have a value bigger than 1. For instance, to decrease the state space size, we can limit O_i^{thg} with a maximum value O_{max}^{thg}. On the other hand, O_i^{lat} indicates the rate of the PRBs used before the latency SLA over all the allocated PRBs for UE i. The bigger O_i^{lat} is, the better performance of the UE i becomes in terms of latency. Note

that a UE will respect the latency requirements if all the allocated PRBs are scheduled before the latency SLA, which is equivalent to $O_i^{lat} = 1$. Formally,

$O_i^{thg} = \min\{\frac{achieved\ throughput}{throughput\ SLA}, O_{max}^{thg}\}$ with $O_{max}^{thg} > 1$
$O_i^{lat} = \frac{Number\ of\ allocated\ PRBs\ before\ latency\ SLA}{Total\ number\ of\ allocated\ PRBs}$

E_i contains three values: N_i^{thg}, N_i^{lat} and Min_i^{thg}. N_i^{thg} and N_i^{lat} count the number of UEs excluding UE i that have met their throughput and latency SLAs, respectively. Min_i^{thg} is the smallest throughput SLA achieved by other UEs.
The state's design considers the solution's scalability regarding the number of UEs, the SLA requirements, and the bandwidth size.

4.5.2 Action

The agent take an action parameterized by the tuple (t, μ, N), where t is a time slot, μ is the numerology to use at the given time slot t, and N is the number of resources to allocate at t. Accordingly, the agent allocates a rectangular shape of ($2^{\mu_{max}-\mu}$,($N * 2^{\mu}$)) at time slot t and put it on the top of the frequency boundary line. Since the agent can output unfeasible actions, we added a pre-processing step in order to compute an action space A that contains only the possible actions at the current state S_t^i.

4.5.3 Reward

We have adopted an episodic approach; i.e., an episode is over when max T steps are reached, the resource matrix is full, or all UEs SLAs are met.
The agent gets the reward r defined as follow:

$$r^t = \begin{cases} \alpha * (O_{i,t}^{thg} - O_{i,t-1}^{thg}) + (1-\alpha) * \mathcal{S} & \text{if not done} \\ \mathcal{K} & \text{if done and SLAs are met} \\ \mathcal{P} & \text{otherwise} \end{cases}$$

Indeed, while the episode is still in progress and for each step, the agent takes a reward equivalent to the improvement made by the current action at step t since the previous step $t-1$. We formulate this improvement by the term $(O_{i,t}^{thg} - O_{i,t-1}^{thg})$. Moreover, to minimize the number of steps needed to finish an episode, we added a penalty $\mathcal{S}$ for each step. Also, we added weights α and $(1-\alpha)$ to the two previous terms in order to control their contribution in the reward term. Once the episode is done, the agent takes a high positive reward $\mathcal{K}$ if the SLAs are met for all UEs, or a penalty $\mathcal{P}$ else.

4.6 DRL-RS detailed description

Among the most efficient off-policy DRL algorithms for continuous state space and discrete actions, we may cite: DQN [39] and A3C [67]. The DQN-based method has a replay memory that is sampled and used to optimize the model at every time step while A3C is only optimized when the episode finishes. This makes DQN learning faster in a single actor environment. For this reason, DRL-RS leverages the DQN algorithm.
DRL-RS executes two steps: decision making and updating the Q-Networks. In DQN, two networks are used: a local Q-Network and a target Q-Network. The latter is the same as the

local network except that its parameters are updated every τ steps. They are combined to help the convergence and stabilization of the learning.

- **Decision making:** The DRL-RS agent observes a state S_t^i for UE i and feeds it to the local Q-Network. In DQN, the Q-Network outputs the Q value of each couple (S_t^i ,a) where a is an integer that identifies the action. In order to evaluate the action, the agent needs to derive the three parameters (t, μ, N) from a. To do so, we partitioned the integer value a, using the method "the partitioning of an integer into different parts" introduced in [68], as follows:

  $t = a \div (\mu_{max} * N_{max})$ where N_{max} is the maximum number of resources that can be allocated to one UE and μ_{max} is the maximum numerology in the system. Note that $\div$ is the division of integer division and mod is the mod of integer division.
  $\mu = a \mod (\mu_{max} * N_{max}) \div N_{max}$
  $N = (a \mod (\mu_{max} * N_{max}) \mod N_{max}) + 1.$

  Then, the agent removes the unfeasible actions from the action space (e.g., allocating resources that overlap with other existing resources) by setting the Q value of each unfeasible tuple (t, μ, N), indexed by a, to a negative value. The DQN algorithm will not explore the unfeasible actions since DQN explores actions having high Q values.

  After that, we apply an ϵ-greedy approach to choose an action. This means that the agent will choose a random action over the feasable actions with ϵ probability and the best action over the action distribution with a 1- ϵ probability. The value of ϵ decays over time during the learning phase. It allows pushing the agent to explore the environment at the beginning of the training phase and better exploit the learned decisions over time.

- **Updating the Q-Networks:** At each step, the current state, the action, the next state, and the reward are stored in a buffer known as the replay buffer. The local Q-Network is updated using a random sample from the replay buffer, which reduces the correlation between the agent's experiences and increases the stability of the learning. Using Mean Square Error (MSE) and ADAM optimizer [69], the parameters of the local Q-Network are optimized at every step by considering the local and target values, while the parameters of the target Q-Network are smoothly updated at each step using a parameter τ.

4.7 Performance Evaluation

In this section, we will introduce the simulation environment and parameters used for O-RS and DRL-RS. We note O-RS the exact solver of the MILP model. Then, we illustrate an example of the resource matrix filled with different approaches on a small instance of the problem for the sake of illustration clarity. After that, we solve bigger instances of the problem with different network configurations. We compare the different approaches in terms of efficiency, scalability, and execution time.

4.7.1 Simulation parameters

To evaluate different contributions in a 5G environment, we used the mixed-numerology 5G Simulator developed in [70] that relies on 6.1.4.2 of TS 38.214 [24] specifications to compute the TBS. The simulator takes the scheduling policy regardless of the source of the policy. For instance, we can plug O-RS and DRL-RS into the simulator and compare their results. All tests

Table 4.2: Illustration example parameters

Parameter	Value
Bandwidth	5 MHz
Maximum numerology	3
Time window	3 ms
Number of UEs	3
Number of slice per UE	2
1st slice throughput requirement	4 Mbps
1st slice latency requirement	3 ms
2nd slice throughput requirement	1 Mbps
2nd slice latency requirement	0.5 ms
MCS	26

are performed on a machine with 64 CPUs, an Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz (2.7 GHz with Turbo Boost technology), and 128 GBs of RAM.

O-RS: The MILP problem is solved using Gurobi version 9.1.2. The absolute optimality gap is set to 10^{-8}, and the time limit to 50000 seconds. We recall that the absolute gap is the absolute difference between the best possible objective value and the objective value of the best feasible solution.

DRL-RS: The simulation environment is implemented using Python. We used PyTorch to implement the DRL-RS agent. We have trained the DRL-RS agent using 500 independent episodes. We have fixed the maximum number of steps at each episode T by 100. We have varied the reward parameters and chosen the values that stabilize the convergence of the model. The different considered parameters are presented in Table 4.3. The model converges after 200 episodes.

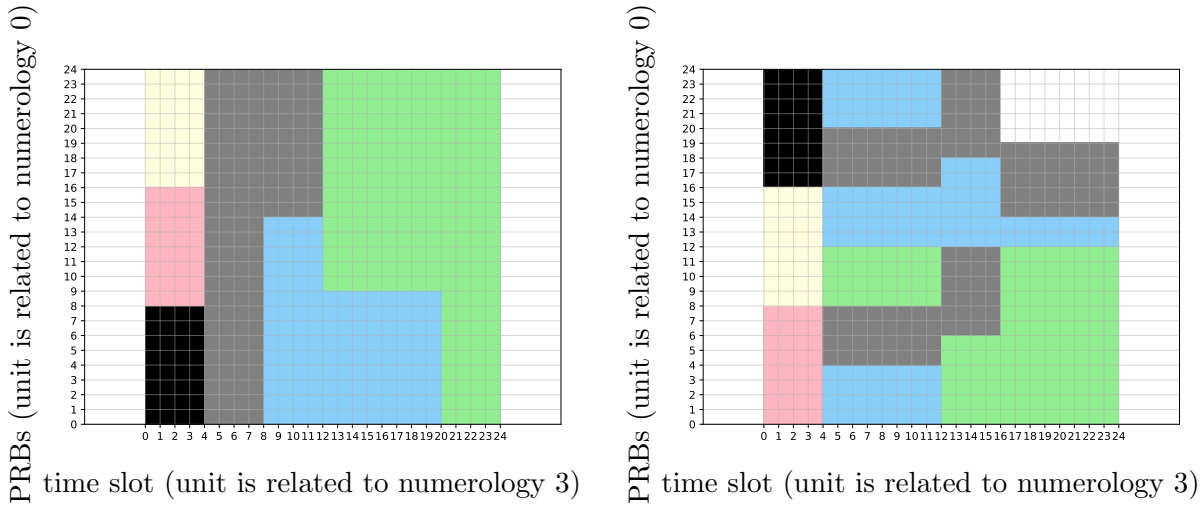
Table 4.3: DRL-RS parameters

Parameter	Value
$\mathcal{P}$	-0.02
$\mathcal{S}$	-0.01
$\mathcal{K}$	5000
α	0.5
O_{max}^{thg}	2
Number of hidden layers	2
Hidden layer size	64 nodes
discount factor γ	0.99
Batch size	128
Learning rate	$5 * 10^{-4}$
Replay buffer size	10^9
Soft update coefficient τ	0.001
Optimizer	ADAM [69]
ϵ-start	1
ϵ-decay	0.99
ϵ-end	0.01

For each bandwidth size and ΔT, we trained an instance of the model using MCS 16, 3 UEs.

Each UE is attached to one eMBB and one uRLLC slice with a fixed SLA. The sum of throughput SLA of UEs nearly equals the maximum achievable throughput in the corresponding bandwidth.

4.7.2 Resource grid illustration



(a) O-RS solution

(b) DRL-RS solution

Figure 4.3: Resource matrix illustration

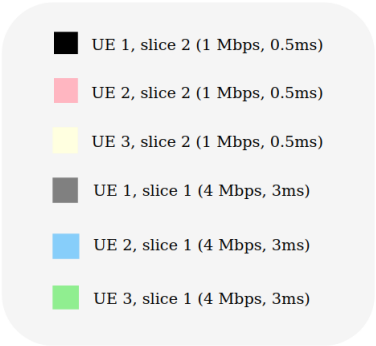


Figure 4.4: Virtual UEs scheduled in the resource matrix illustration example (Figure 4.3)

We solve the problem instance described in Table 4.2 using O-RS and DRL-RS. We selected a small matrix for the sake of illustration clarity. Figure 4.3 compares the resource allocation policies generated by each approach. Figures 4.3(a), 4.3(b) illustrate the radio resource allocation matrix filled by O-RS and DRL-RS, respectively. Each virtual UE has a color, for instance we have 6 virtual UEs. We recall that each slice associated with a UE is a virtual UE. The mapping between colors and virtual UEs is illustrated in Figure 4.4. We observe that, for the two approaches, the black, pink and yellow UEs respect the 0.5 ms latency (resources are allocated before 0.5 ms thanks to the use of numerology 1 by different approaches) and each one uses 4 PRBs (carry 2984 bits of data per time window with MCS 26, we compute the estimated amount of data sent by second: 2984*1000/3 = 0.99 Mbps) required by the 2nd slice in Table 4.2. Gray, blue and green UEs are the virtual UEs belonging to the 1st slice, each one respects the 3 ms latency and the 4 Mbps throughput (each UE uses more than 16 PRBs, which carry 11832 with

Table 4.4: 5G NR Configurations

Index	Bandwidth (MHz)	ΔT (ms)	Number of UEs	slices per UE	Slices SLA list (Mbps, ms)	MCS
1	20	3	3	2	(9,10),(0.2,1)	16
2	20	3	10	2	(5,10),(0.2,1)	26
3	40	1	10	2	(5.5,10),(0.2,0.5)	16
4	40	1	10	2	(13,10),(0.7,0.25)	26
5	100	3	3	3	(40,10),(1,2),(0.2,1)	16
6	100	3	6	3	(40,10),(2,2),(0.7,1)	26
7	400	1	5	2	(110,10),(1,0.125)	16
8	400	1	20	1	(50,10)	26

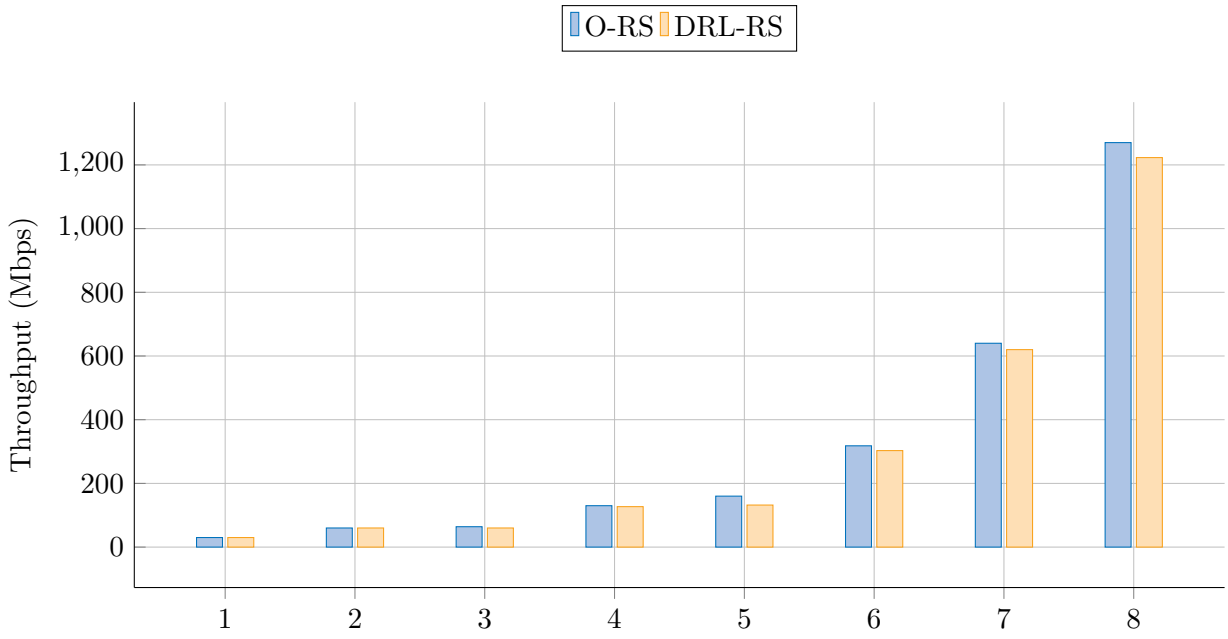


Figure 4.5: Aggregated throughput comparison between O-RS and DRL-RS

MCS 26 per time window, and hence, the throughput per second is estimated to be more than 11832*1000/3 = 3.94 Mbps). We conclude that O-RS and DRL-RS generate different allocation policies and all these policies fulfill the slices SLA in terms of throughput and latency.
We observe that O-RS and DRL-RS dynamically selected the numerology 0 and 1 to fill the matrix. These decisions are due to the fact that DRL-RS and O-RS are designed to be able to select a numerology at each timestamp taking into consideration different constraints. For instance, DRL-RS and O-RS selected numerology 1 to satisfy the 0.5 ms latency for the second slice, and numerology 0 to satisfy the first slice which does not require higher numerology to satisfy its latency SLA.

4.7.3 Comparison with the State of The Art

In the balance of this subsection, we compare DRL-RS results with the results of [63]. We recall that [63] proposed a scalable optimization algorithm based on LP and LD (noted as LP+LD). We adapted our simulation parameters to reproduce the same experiment using a scheduling window of 2 MHz of bandwidth and 2 ms of duration. We assumed a maximum numerology of 3. We considered 10 UEs with a target throughput of 450 kbps (this is due to the limitation of LP+LD). We varied the target latency. Figure 4.6 depicts the achieved throughput using

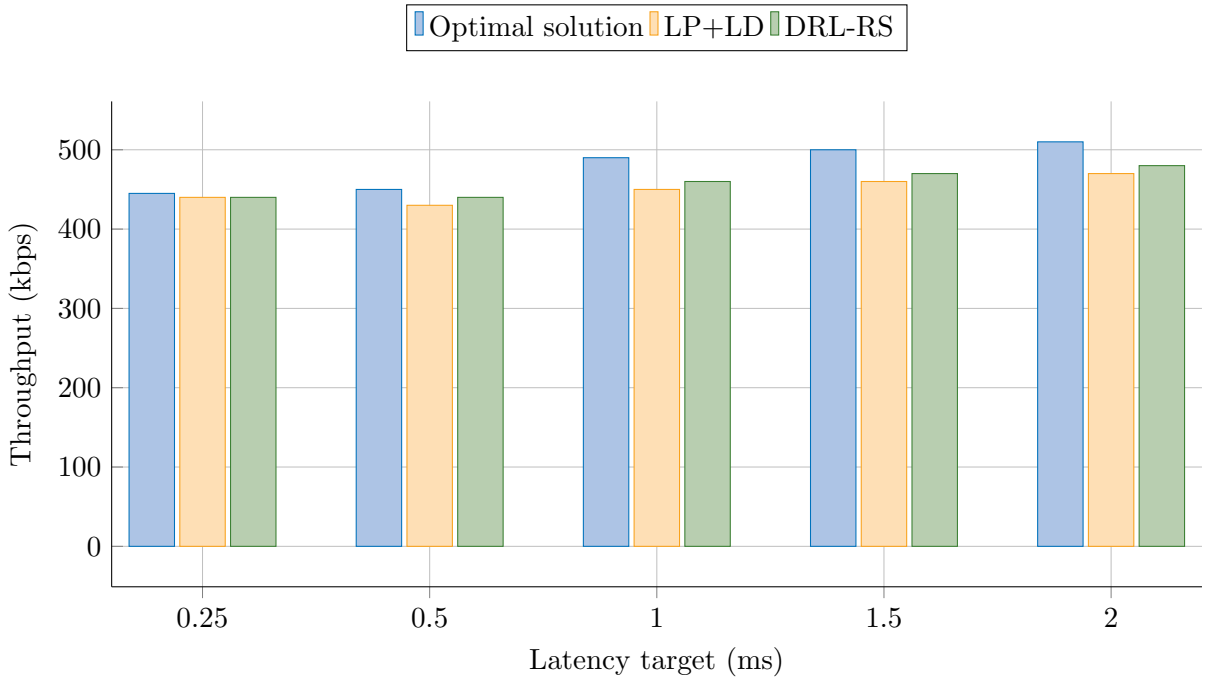


Figure 4.6: Average throughput per user comparison between DRL-RS and the state of The Art

different latency targets. The x-axis represents the latency targets while the y-axis represents the achieved throughput in kbps. We notice that DRL-RS is able to achieve slightly higher throughput than LP+LD solution. We are not able to compare with bigger instances of the problem because [63] did not consider it. In contrast, DRL-RS considers bigger instances, which is one of the critical contributions of our work compared to state of the art solutions.

4.7.4 Efficiency and Scalability Evaluation

In this subsection, we will evaluate the effeciency and the scalability of DRL-RS. Figure 4.5 compares the aggregated UE throughput achieved in each configuration by O-RS and DRL-RS. We notice that the achieved throughput by DRL-RS is close to the optimal achievable throughput obtained by O-RS, which means that DRL-RS is able to fill the resource matrix efficiently while satisfying the throughput and latency SLA for all the slices (Figures 4.7). We observe a small gap between the performance of DRL-RS and O-RS in higher bandwidths (greater than 100 MHz). This gap is caused by the wasted resources that DRL-RS may generate due to the nature of DRL algorithms, which approximate continuous state using neural networks. Indeed, these resources cannot be used by any UE due to shape and numerology constraints.
Figure 4.7 depicts the achieved SLA by DRL-RS for each configuration listed in Table 4.4. The y-axis represents the metrics used for measuring the SLA: $\max_i O_i^{thg}$, $\min_i O_i^{thg}$, $\text{average}_i O_i^{thg}$ for the throughput SLA (blue and red in the figure) and $\max_i O_i^{lat}$, $\min_i O_i^{lat}$, $\text{average}_i O_i^{lat}$ for the latency SLA (green and grey in the figure). We remind that these values are described in Section 4.5.1. The SLA is assumed respected when the value of these metrics is higher than 1 with a tolerance of 5% due to the QNetworks estimation errors. The x-axis represents the configuration indexes summarized in Table 4.4.
We adopted different bandwidth sizes from 20 MHz to 400 MHz, representing the maximum bandwidth defined by 5G NR specifications and corresponding to the usage of the mmWave bands. Moreover, in order to support the dynamic nature of the RAN environment, we varied

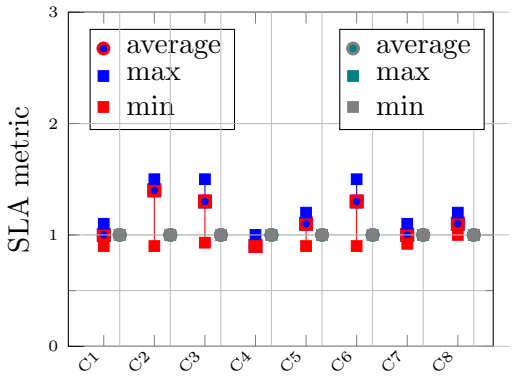
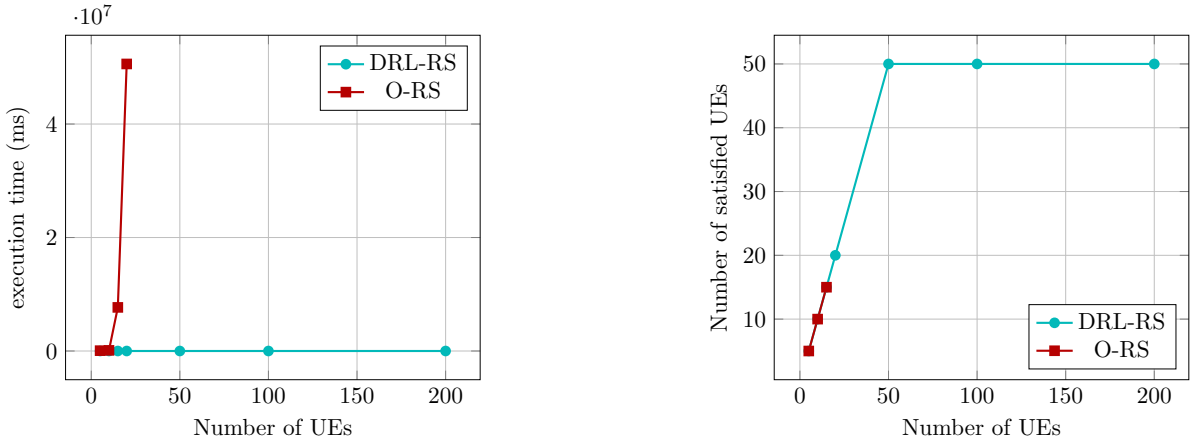


Figure 4.7: Throughput and Latency SLA fulfillment by DRL-RS

the number of UEs, the number of slices per UE (1, 2, or 3 slices per UE), slices SLA and the MCS assigned to each UE (16 and 26 for medium and high channel quality, respectively). Figure 4.7 reveals that the slice' SLAs are met for each configuration. We notice that the maximum and the minimum achieved SLA for throughput are greater than 1 ($\max_i O_i^{thg} > 1$) and 0.94 ($\min_i O_i^{thg} > 0.94$), respectively. This means that all the UEs have achieved their throughput SLA targets. We also notice that the latency SLAs are respected for all the configurations (i.e., $\min_i O_i^{lat} = 1$).

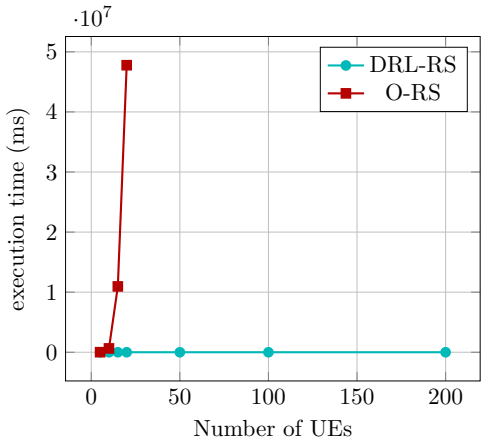


(a) The execution time

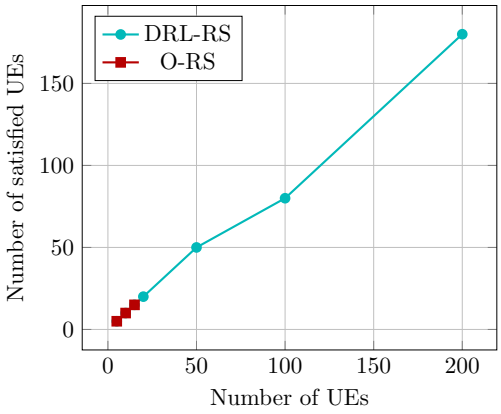
(b) Performance Evaluation

Figure 4.8: 20 MHz matrix

Figures 4.8(a), 4.9(a), 4.10(a), 4.11(a), 4.12(a) compare the execution time needed by O-RS, DRL-RS to find a feasible solution for a bandwidth of 20MHz (configuration 1), 40 MHz (configuration 3), 100MHz (configuration 5) and 400MHz (configuration 7 and 8), respectively. The x-axis represents the number of UEs to be scheduled and the y-axis represents the execution time in milliseconds. The number of UEs is defined by β while the throughput SLA for each UE will be $Initial\ SLA \div \beta$, where $\beta = 5 * k$, with k a positive integer; and $Initial\ SLA$ is the maximum possible throughput SLA in the given configuration. We observe that O-RS's execution time is exponential by report to the number of UEs while DRL-RS take much less time to be executed. We notice that O-RS is not able to solve the problem instances with more than 30 UEs before the fixed time limit (50000 s). Figure 4.13 zoom out the execution time for Figure 4.12(a) to visualize the difference in execution time more clearly for smaller instances (number of UEs less than 30 UEs for O-RS plot). We notice that DRL-RS takes much less time to find a feasible

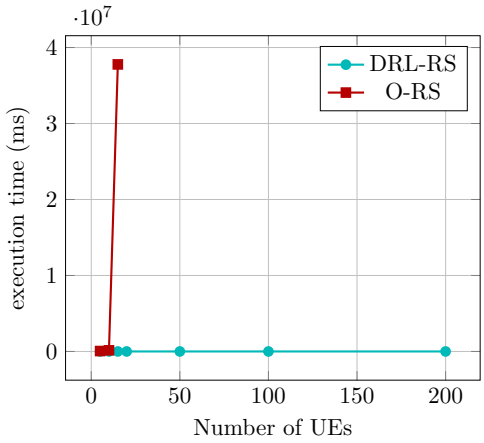


(a) The execution time

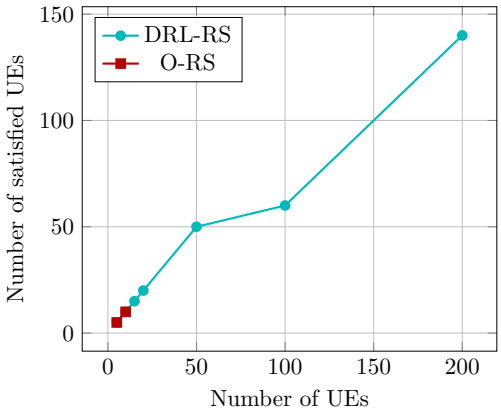


(b) Performance Evaluation

Figure 4.9: 40 MHz matrix



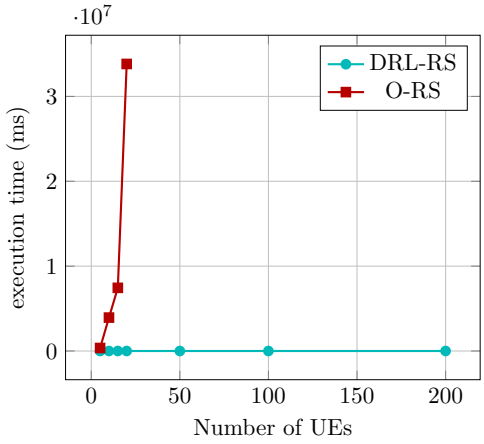
(a) Execution time comparison



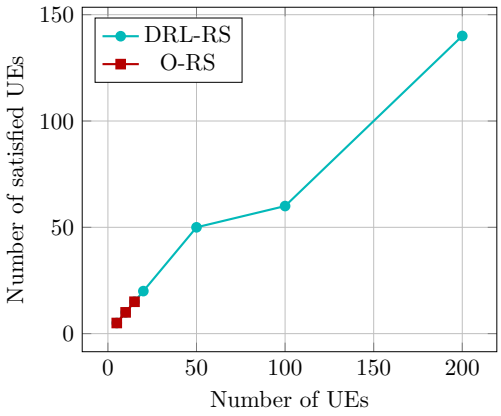
(b) Performance Evaluation

Figure 4.10: 100 MHz matrix

solution (less than 1ms). Hence, we can say that DRL-RS is suitable for real-time scheduling.



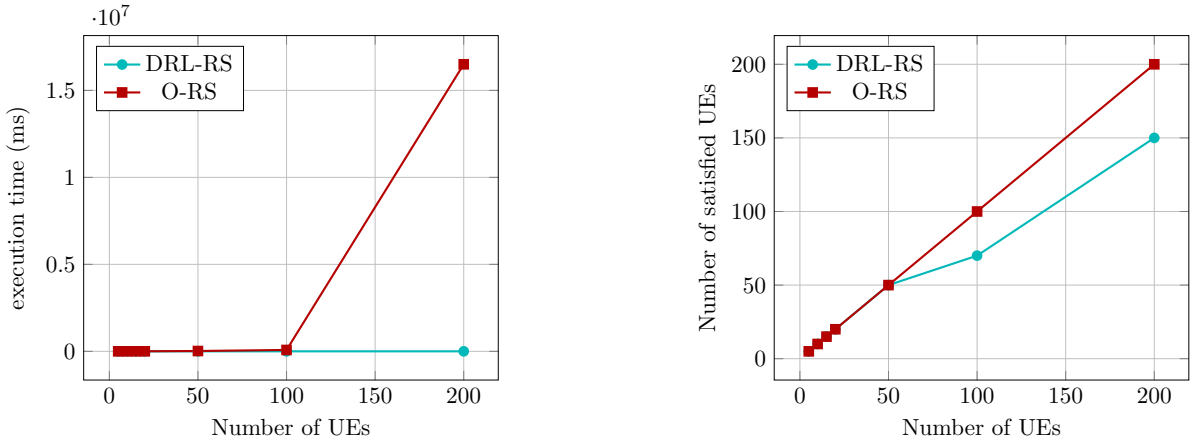
(a) Execution time comparison



(b) Performance Evaluation

Figure 4.11: 400 MHz matrix

Figures 4.8(b), 4.9(b), 4.10(b), 4.11(b), 4.12(b) compare the number of satisfied UEs when applying O-RS and DRL-RS on a bandwidth of 20MHz (configuration 1), 40MHz (configuration 3), 100MHz (configuration 5) and 400MHz (configuration 7 and 8), respectively. The x-axis represents the number of UEs to be scheduled, and the y-axis represents the number of satisfied UEs in terms of both throughput and latency. We observe that DRL-RS is able to fulfill the throughput and latency requirements of up to 50, 180, 130, 140, and 150 UEs out of 200 UEs in configuration 1, 3, 5, 7, and 8, respectively (we recall that configuration requirements are different, see Table 4.4).



(a) Execution time comparison

(b) Performance Evaluation

Figure 4.12: 400 MHz matrix (1 slice attach case)

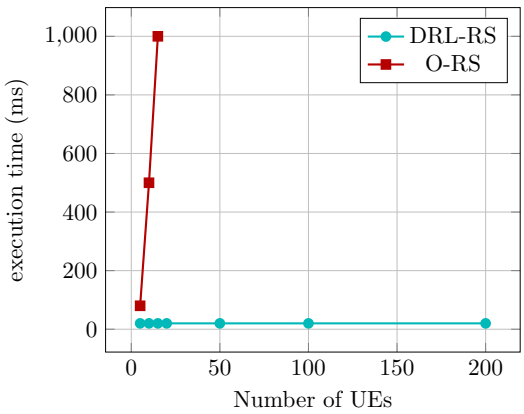


Figure 4.13: Execution time for the 400 MHz matrix (single slice attach case)

To summarize, O-RS is able to derive the optimal solution for the proposed optimization problem. However, we observed that it takes exponential time regarding the number of UEs. DRL-RS is an approach to reducing the execution time. DRL-RS takes less than 1ms to find a feasible solution in different network configurations. Hence, DRL-RS is more suitable for real-time scheduling. Moreover, the DRL solution was able to select the numerology dynamically, preventing from using higher numerology when it is not needed (since the DRL-RS design allows the selection of numerology for each time slot).

4.8 Conclusion

In this chapter, we modeled the radio resource allocation problem in 5G NR featuring network slicing and considering a mixed-numerology environment as a Mixed Integer Linear Program. We proved that the problem is NP-hard and proposed a DRL-based approach to solve it for big instances of the problem (i.e., when the number of UEs is higher than 50 and bandwidth size bigger than 40 MHz). Simulation results demonstrated the proposed approach' efficiency and scalability in meeting the desired quality of service requirements. Indeed, DRL-RS can find feasible solutions for the radio resource management problem in a reasonable amount of time which makes it suitable for real-time scheduling. Besides, DRL-RS is energy efficient since it dynamically selects high numerology only when it is required in the context of multiple slices per UE.

Part II

Dynamic TDD optimization in 5G NR

Chapter 5

Dynamic TDD for a single-cell 5G environment

5.1 Introduction

Emerging 5G and 6G network services are shifting from DL-dominant traffic to more equilibrate DL and UL traffic, even to more UL-dominant traffic [10]. Indeed, emerging network services such as AR and VR applications generate high UL traffic corresponding to offloaded intensive computation to be run at a remote application sitting at the Edge. Another example is the high-quality video streaming captured by drones for building surveillance that requires more UL traffic than DL. One solution to accommodate this new trend in terms of traffic model is TDD. TDD allows using the entire bandwidth by dividing it into time slots where some are assigned to UL and some to DL.
4G proposes 7 different configurations of the TDD frame, which allows configuring the eNB according to the traffic patterns. But, one concern with this solution is that the configuration is fixed in time and cannot be adapted to the traffic dynamic, i.e., if in a certain moment of time more DL or UL need to be accommodated, then there is no possibility to increase the number of UL or DL slots, without rebooting the eNB. To overcome this issue, 5G NR introduces a more flexible solution, where the number of UL and DL slots in the TDD frame can be changed dynamically. This flexibility will allow the gNB to adapt to the frame configuration according to the traffic pattern by selecting the number of slots dedicated to UL and DL. However, the 5G NR specifications only cover the mechanism allowing the gNB to inform the UE about the UL/DL slots pattern in a TDD frame, leaving the algorithm deriving the pattern UL/DL opens. In this contribution, we propose a novel algorithm, namely DRL-based 5G RAN TDD Pattern (DRP), which allows deriving the UL/DL pattern of TDD frames dynamically and accommodating cell traffic whatever it is DL or UL dominant. DRP monitors UL and DL traffic periodically and derives the optimal pattern. DRP uses the BSR sent by UEs for the UL traffic and the state of the radio bearer channel queues at gNB, which avoids having an exact pattern of the traffic. DRP is run by gNB before the MAC scheduling process. DRP is particularly efficient for 5G private network deployment, allowing to run gNB in a plug-and-play mode.

5.2 Related works

Many dynamic TDD resource allocation algorithms for optimizing both power efficiency and transmission performance in LTE were investigated [71, 72, 73]. T. Ding et al. analyze the

performance of employing dynamic TDD for dense small cell networks towards 5G [74]. In [75], authors considered a cluster-based dynamic UL/DL re-configuration considering a centralized-RAN scenario in dense deployments. The central control unit uses a low complexity trellis exploration algorithm to reconfigure the UL/DL ratio aiming at maximizing the overall RAN throughput. In [76], the authors presented a game-theoretic approach where each eNB re-configures its TDD frame aiming at minimizing the UL/DL delay while considering cross-slot interference. As mentioned earlier, all these works consider the LTE TDD configuration, ignoring the new flexibility of 5G NR. In [77], the authors explored employing deep reinforcement learning to adaptively allocate TDD UL/DL resources in the high mobility 5G HetNet. As high mobility of users leads to the high dynamic network traffic and unpredicted link state change, a new method to predict the dynamic traffic and channel condition and schedule the TDD configuration in real-time is proposed. However, this work does not consider the 5G NR specificities and requires additional information that is not available at the base station. In [78], the authors proposed a service-oriented soft spectrum slicing for 5G TDD. The objective is to use the flexibility of TDD to adjust the UL/DL dynamically using forecasted traffic and user mobility. The problem has been modeled using weighted optimisation. Although the paper tackles 5G, it uses the TDD LTE configuration (i.e., fixed patterns). In addition, the optimization problem needs to be solved periodically to define the new pattern, which is not realistic in a real deployment.

Table 5.1: Summary of Notations & Variables.

Notation / Variable	**Description**
$\mathcal{B}$	the gNB.
Γ	the set of UEs in the network.
γ	A UE $\gamma \in \Gamma$.
δ	TDD period.
μ	A numerology used by gNB.
$\mathcal{T}_\delta$	The number of slots during a period δ. $\mathcal{T}_\delta = \{1, 2, 3 \cdots 16\}$.
$\lambda_\gamma^{\mathcal{U}}$	The UL traffic generated by UE γ.
$\lambda_\gamma^{\mathcal{D}}$	The DL traffic of UE γ.
$\lambda_\Gamma^{\mathcal{U}}$	The UL traffic generated by all UEs Γ. Formally, $\lambda_\Gamma^{\mathcal{U}} = \sum_{\gamma \in \Gamma} \lambda_\gamma^{\mathcal{U}}$
$\lambda_\Gamma^{\mathcal{D}}$	The DL traffic of all UEs Γ. Formally, $\lambda_\Gamma^{\mathcal{D}} = \sum_{\gamma \in \Gamma} \lambda_\gamma^{\mathcal{D}}$
$\psi_\gamma^{\mathcal{U}}$	The UL buffer of UE γ.
$\psi_\gamma^{\mathcal{D}}$	The DL buffer of UE γ.
$\Psi_\Gamma^{\mathcal{U}}$	The UL buffer of all UEs Γ. Formally, $\Psi_\Gamma^{\mathcal{U}} = \sum_{\gamma \in \Gamma} \psi_\gamma^{\mathcal{U}}$.
$\Psi_\Gamma^{\mathcal{D}}$	The DL buffer of all UEs Γ. Formally, $\Psi_\Gamma^{\mathcal{D}} = \sum_{\gamma \in \Gamma} \psi_\gamma^{\mathcal{D}}$.
$\mathcal{S}$	OFDM symbol.
$\mu_{\mathcal{S}}$	The amount of traffic in bytes transmitted by $\mathcal{S}$.
α	A constant that specifies the priority between the UL and DL traffics.
$\Phi_\gamma^{\mathcal{U}}$	The initial amount of stored data in bytes in the UL buffer $\psi_\gamma^{\mathcal{U}}$.
$\Phi_\gamma^{\mathcal{D}}$	The initial amount of stored data in bytes in the DL buffer $\psi_\gamma^{\mathcal{D}}$.
$\mathcal{X}_\gamma$	A real variable that denotes the percentage of UL slots reserved for the UE $\gamma \in \Gamma$.
$\mathcal{Y}_\gamma$	A real variable that denotes the percentage of DL slots reserved for the UE $\gamma \in \Gamma$.

5.3 Network model and problem formulation

5.3.1 Network model

In the system, we focus only on one gNB $\mathcal{B}$, whereby a set of UEs are connected. Let Γ denote the set of UEs in the system. UEs $\gamma \in \Gamma$ are attached to the same gNB. We assume that gNB is using TDD operations. Each TDD period δ is fixed by gNB. For the sake of paper readability, the used notations are summarized in Table 5.1.Each UE $\gamma \in \Gamma$ has UL and DL traffics that can vary from an industrial vertical to another. While UL traffic is generated and transmitted from the UE γ to the gNB $\mathcal{B}$, the DL traffic is sent from the gNB $\mathcal{B}$ to the UE γ. Let $\lambda_\gamma^{\mathcal{U}}$ and $\lambda_\gamma^{\mathcal{D}}$ denote the amount of UL and DL traffic in bytes of UE γ, respectively.
In contrast to LTE, where the DL traffic $\lambda_\gamma^{\mathcal{D}}$ is more critical than the UL one $\lambda_\gamma^{\mathcal{U}}$, in the $5G$ system and beyond the amount of traffic in UL and DL is application dependent. For instance, in a network slice that manages autonomous driving or UAV, high UL traffic $\lambda_\gamma^{\mathcal{U}}$ is expected. On another side, in some verticals, such as live streaming, the DL traffic $\lambda_\gamma^{\mathcal{D}}$ is more important. Other industrial verticals and network slices, such as immersive applications (VR, AR, and holograph communication), require high data rates in both directions. The players in these applications are characterized by colossal collaborative interactions, tremendous precision, and high data synchronization. Furthermore, the same UE γ can be subscribed in multiple network slice, which makes hard to predict the UL $\lambda_\gamma^{\mathcal{U}}$ and DL $\lambda_\gamma^{\mathcal{D}}$ traffic of a UE γ.
For the sake of simplicity and without loss of generality, we assume that each UE γ has limited UL $\Psi_\gamma^{\mathcal{U}}$ and DL $\Psi_\gamma^{\mathcal{D}}$ buffers, respectively. Let $|\Psi_\gamma^{\mathcal{U}}|$ and $|\Psi_\gamma^{\mathcal{D}}|$ denote the size of UL and DL buffers in bytes, respectively. While $\Psi_\gamma^{\mathcal{U}}$ is located at the UE γ, $\Psi_\gamma^{\mathcal{D}}$ is located at the gNB $\mathcal{B}$. The UE γ periodically keeps informing the gNB $\mathcal{B}$ about the state of $\Psi_\gamma^{\mathcal{U}}$. In 5G NR, this operation corresponds to the BSR sent by UE when requesting UL resources. Meanwhile, the DL buffers are monitored by gNB, as it corresponds to the radio bearer data channels maintained by gNB for each UE. Let $\Psi_\Gamma^{\mathcal{U}}$ and $\Psi_\Gamma^{\mathcal{D}}$ denote the UL and DL buffer of all the UEs. Formally, $\Psi_\Gamma^{\mathcal{U}} = \sum_{\gamma \in \Gamma} \psi_\gamma^{\mathcal{U}}$, and $\Psi_\Gamma^{\mathcal{D}} = \sum_{\gamma \in \Gamma} \psi_\gamma^{\mathcal{D}}$.

5.3.2 Problem formulation

In this work, we assume that the number of slots $\mathcal{T}_\Delta$ is fixed in each frame Δ. However, their distribution on UL and DL traffic is unknown, and it is the target of this contribution. The main research question is how to distribute the slots among the UL and DL traffic, such that the SLA is preserved and the UL $\Psi_\Gamma^{\mathcal{U}}$ and DL $\Psi_\Gamma^{\mathcal{D}}$ buffers do not overrun their boundary. The main challenge faces the work is both UL $\lambda_\gamma^{\mathcal{U}}$ and DL $\lambda_\gamma^{\mathcal{D}}$ traffics are unknown and hard to predict.
As aforementioned, each slot has 14 OFDM symbol $\mathcal{S}$, each of which can transmit $\mu_\mathcal{S}$ bytes. Thus, each slot $t \in \mathcal{T}_\delta$ transmits $14 \times \mu_\mathcal{S}$. Let $\mathcal{X}_\gamma$ a real variable that denotes the percentage of UL slots reserved for the UE $\gamma \in \Gamma$. Similarly, let $\mathcal{Y}_\gamma$ a real variable that denotes the percentage of reserved slots for DL traffic. Formally, the following statements should hold (5.2), (5.3) and (5.4).
If we denote by $\mathcal{X}_\Gamma$ a real variable that denotes the percentage of UL slots reserved for all the UEs Γ, then $\mathcal{X}_\Gamma = \frac{1}{|\Gamma|} \sum_\gamma \mathcal{X}_\gamma$. Similarly, if $\mathcal{Y}_\Gamma$ is a real variable that denotes the percentage of DL slots reserved for all the UEs Γ, then $\mathcal{Y}_\Gamma = \frac{1}{|\Gamma|} \sum_\gamma \mathcal{Y}_\gamma$. Also, the following statement holds:

$$\mathcal{X}_\Gamma + \mathcal{Y}_\Gamma = 1$$

Let α a given constant ($0 \leq \alpha \leq 1$) that defines the priority between the UL and DL traffics. This parameter can be defined by the customer to specify the priorities between the two traffics.

If $\alpha = 1$, then we are interested only to optimize the UL traffic, whereas, if $\alpha = 0$, then we are interested only to optimize the DL traffic. The proposed solution should be periodically applied to specify $\mathcal{X}_\gamma$ and $\mathcal{Y}_\gamma$ in order to prevent the overflow of UL buffer $\Psi_\gamma^{\mathcal{U}}$ and DL buffer $\Psi_\gamma^{\mathcal{D}}$. Let $\Phi_\gamma^{\mathcal{U}}$ and $\Phi_\gamma^{\mathcal{D}}$ denote the initial stored data of the UL and DL buffers. Both $\Phi_\gamma^{\mathcal{U}}$ and $\Phi_\gamma^{\mathcal{D}}$ are initialized by zero. At each iteration, we aim to optimize the following linear integer programming:

$$\begin{aligned} \textbf{min } & \frac{\alpha}{|\Psi_\Gamma^{\mathcal{U}}|} \times \sum_{\gamma \in \Gamma} \left(\Phi_\gamma^{\mathcal{U}} + \lambda_\gamma^{\mathcal{U}} - 14 \times \mu_{\mathcal{S}} \times \mathcal{X}_\gamma \times \mathcal{T}_\delta \right) \\ & \frac{1-\alpha}{|\Psi_\Gamma^{\mathcal{D}}|} \times \sum_{\gamma \in \Gamma} \left(\Phi_\gamma^{\mathcal{D}} + \lambda_\gamma^{\mathcal{D}} - 14 \times \mu_{\mathcal{S}} \times \mathcal{Y}_\gamma \times \mathcal{T}_\delta \right) \end{aligned} \tag{5.1}$$

S.t,

$$0 \leq \mathcal{X}_\gamma \leq 1 \tag{5.2}$$

$$0 \leq \mathcal{Y}_\gamma \leq 1 \tag{5.3}$$

$$\mathcal{X}_\gamma + \mathcal{Y}_\gamma = 1 \tag{5.4}$$

$$\forall \gamma \in \Gamma : \Phi_\gamma^{\mathcal{U}} + \lambda_\gamma^{\mathcal{U}} - 14 \times \mu_{\mathcal{S}} \times \mathcal{X}_\gamma \times \mathcal{T}_\delta \leq |\Psi_\gamma^{\mathcal{U}}| \tag{5.5}$$

$$\forall \gamma \in \Gamma : \Phi_\gamma^{\mathcal{D}} + \lambda_\gamma^{\mathcal{D}} - 14 \times \mu_{\mathcal{S}} \times \mathcal{Y}_\gamma \times \mathcal{T}_\delta \leq |\Psi_\Gamma^{\mathcal{D}}| \tag{5.6}$$

$$\forall \gamma \in \Gamma : \mathcal{X}_\gamma \times \mathcal{T}_\delta = \mathcal{A}_\gamma \tag{5.7}$$

$$\forall \gamma \in \Gamma : \mathcal{Y}_\gamma \times \mathcal{T}_\delta = \mathcal{B}_\gamma \tag{5.8}$$

$$\forall \gamma \in \Gamma : (\mathcal{A}_\gamma, \mathcal{B}_\gamma) \in \mathcal{N}^2 \tag{5.9}$$

The objective function (5.1) aims to minimize the amount of stored data in the UL and DL buffers to prevent their overflow. While $\lfloor \sum_{\gamma \in \Gamma} \mathcal{X}_\gamma \times \mathcal{T}_\delta \rfloor$ denotes the number of slots reserved for the UL traffic, $\lceil \sum_{\gamma \in \Gamma} \mathcal{X}_\gamma \times \mathcal{T}_\delta \rceil$ is the number of slots reserved for the DL traffic. We have used weighted normalized sum method to prevent an objective (i.e., buffer) dominant the other. Meanwhile, constraints, (5.2), (5.3) and (5.4), ensure that the variables $\mathcal{X}_\gamma$ and $\mathcal{Y}_\gamma$ are rates of slots distribution for UL and DL. Meanwhile, constraints 5.5 and 5.6 ensure that the UL and DL buffer of $\gamma \in \Gamma$ is not overflow, respectively. Meanwhile, constraints, (5.7), (5.8) and (5.9), ensure that the number of UL and DL solts distributed on each UE $\gamma \in \Gamma$ is an integer. Note that $\mathcal{A}_\gamma$ and $\mathcal{B}_\gamma$ are two integer variables should be fixed by the system. While $\mathcal{A}_\gamma$ denotes the number of UL slots of γ, $\mathcal{A}_\gamma$ denotes the DL slots.
Unfortunately, we cannot use the optimization problem mentioned above for distributing the slots mainly due to two reasons: *i*) Solving the optimization problem is time-consuming while we should take decisions within few milliseconds as the solution is online and acting at the RAN level; *ii*) The amount of UL $\lambda_\gamma^{\mathcal{U}}$ and DL $\lambda_\gamma^{\mathcal{D}}$ traffics are unknown and it is hard to predict them a priory.

5.4 DRP System Overview

As aforementioned, it is hard to distribute OFDM slots using optimization efficiently and without prior knowledge of the traffic generation patterns. For this reason, we have proposed the DRP

system that leverages DRL, more precisely the DDPG Algorithm, to define the 5G NR TDD pattern dynamically. The DRL hides the complexity and stochastic of the environment and helps the DRP framework to make efficient and quick decisions that adapt according to the traffic patterns. Moreover, the DRP framework gains the ability to learn with time and adapts to different and unseen situations. In the balance of this section, we will present the DRP system overview and DRL background, more precisely the DDPG Algorithm, and a detailed description of the DRP system.
The ability of DRL-based solutions to deal with unknown and unseen environments makes them the best fit for addressing the 5G TDD pattern configuration problem. The DRL Algorithm mainly consists of two steps: *i*) The learning (exploration) step; *ii*) The exploitation step. In order to overcome the need of model (i.e, transition probability), during the learning step, the agent interacts with the environment by following stochastic policy (i.e., random actions) to explore and build the knowledge about the environment. While, in the exploitation step, the agent exploits the acquired knowledge by following the optimal policy π_* that provides the optimal action a_t, to take in each state s_t, in a way that maximizes future cumulative discounted reward G_t defined as follows:

$$G_t \doteq \sum_{k=0}^{T} \gamma^k r_{t+k+1} = r_{t+1} + \gamma G_{t+1} \tag{5.10}$$

With $\gamma \in [0, 1]$ defined as the discount rate that penalises the future rewards, and T equal to the time horizon which is finite for episodic problems (i.e., problems that ends when the environment is a final state) and infinite for continuing problems.
In general, DRL methods are classified into three categories : *i*) *value-based* methods, such as Monte Carlo, SARSA, Q-Learning, and DQN; *ii*) *policy-based* methods, such as REINFORCE (i.e., Monte-Carlo Policy Gradient) and REINFORCE with baseline; *iii*) *actor-critic* methods that combine the two previous methods, such as A2C, A3C, DDPG, and PPO [42]. In actor-critic approach, we have mainly two families, the stochastic policy approach (e.g., A2C and A3C) and the deterministic approach (e.g., DDPG). In the stochastic policy approach, the actions are selected from the Actor with different probability using the softmax activation function. The agent should pick the action that has a high probability. Unfortunately, the main limitation of the stochastic policy approach is the number of actions should be limited. In contrast, in the deterministic approach, the actions are generated directly from the actor-network, enabling continuous actions. In this contribution, we are interested in specifying the percentage of UL and DL slots as depicted in Figure 5.1. For this reason, we have adapted DDPG Algorithm. We will explain further the DDPG Algorithm when explaining our DRP approach.

5.5 DRP Detailed Description

We have designed the DRP agent to be lightweight to ensure fast interaction with the environment. Also, we have designed the DRP agent to ensure generality and then work in an unseen environment. The DRP agent has been designed to work independently from the number of slots and the size of the buffers. Moreover, it considers the variation and correlation in the buffer states to predict the traffic patterns. In what follows, we define the elements of the DRP agent, including the state, the reward, and the action.
i) State: Let $\xi_t^{\mathcal{U}}$ and $\xi_t^{\mathcal{D}}$ denote the amount of traffic in the UL $\Psi_\Gamma^{\mathcal{U}}$ and the DL $\Psi_\Gamma^{\mathcal{D}}$ at the step t, respectively. To ensure the generalization, we define the observation $\mathcal{O}_t^{\mathcal{U}}$ and $\mathcal{O}_t^{\mathcal{D}}$ of the UL and DL buffers as normalized values (Figure 5.1: 1). Formally, $\mathcal{O}_t^{\mathcal{U}} = \frac{\xi_t^{\mathcal{U}}}{\Psi_\Gamma^{\mathcal{U}}}$ and $\mathcal{O}_t^{\mathcal{D}} = \frac{\xi_t^{\mathcal{D}}}{\Psi_\Gamma^{\mathcal{D}}}$, respectively.

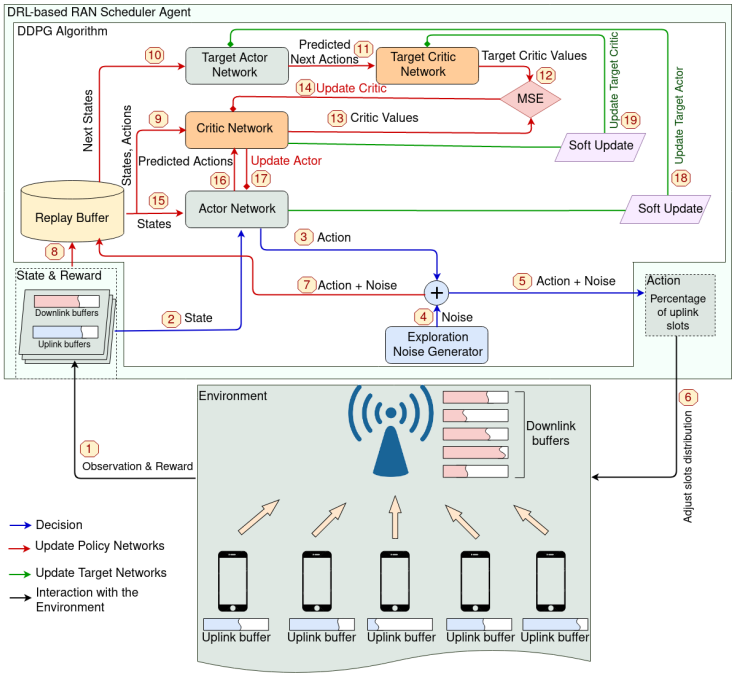


Figure 5.1: Envisioned DRL-based 5G RAN TDD Pattern (DRP) System

The benefits of the normalization are twofold: *i*) It ensures the generality by enabling the DRP agent to be agnostic to the scenario scale. It works similarly in different buffers with different sizes. The most important is to catch the buffer fullness ratio of $\Psi_{\Gamma}^{\mathcal{U}}$ and $\Psi_{\Gamma}^{\mathcal{D}}$; *ii*) It is well known that the activation functions in the neural network work well for small values, which positively impacts DRP's convergence. Moreover, to capture the traffic patterns, we define the state s_t as follow:

$$s_t = \left(\bigcup_{i=0}^{\mathcal{K}} \mathcal{O}_{t-i}^{\mathcal{U}}, \bigcup_{i=0}^{\mathcal{K}} \mathcal{O}_{t-i}^{\mathcal{D}} \right) \tag{5.11}$$

In the state s_t, the DRP agent, besides the current observation, considers $\mathcal{K}$ previous observations before taking any action. This enables to capture the behavior of both buffers and traffics before making any action.

ii) Action: We have only one continuous action a_t that presents the percentage of slots should be reserved for the UL traffic at the step t. The DRP agent enforces the taken decisions as depicted in Figure 5.1:6. Accordingly, the $\lfloor a_t \times \mathcal{T}_\delta \rfloor$ slots are reserved for the UL traffic, and $\lceil (1 - a_t) \times \mathcal{T}_\delta \rceil$ slots are reserved for the DL traffic. It is worth noting that the action is independent on the number of slots $\mathcal{T}_\delta$, which ensures the generality and enables the DRP agent to be agnostic to the scenario scale.

iii) Reward: We have adapted an episodic approach, whereby each episode runs for max T steps before ends. The reward r_t has been defined as follow:

$$r_t = \begin{cases} \alpha \times (1 - \mathcal{O}_t^{\mathcal{U}}) + (1 - \alpha) \times (1 - \mathcal{O}_t^{\mathcal{D}}) & \text{If} |\mathcal{O}_t^{\mathcal{U}}| < 1 \wedge |\mathcal{O}_t^{\mathcal{D}}| < 1 \\ -\mathcal{M} & \text{Otherwise} \end{cases} \tag{5.12}$$

Where, α is the priority between the UL and DL traffics. The DRP agent receives a positive reward for each step succeeds to keep the buffers $\Psi_{\Gamma}^{\mathcal{U}}$ and $\Psi_{\Gamma}^{\mathcal{D}}$ do not exceed their threshold. Moreover, the emptiest the buffers are, the highest reward becomes. When one of the buffers exceeds its capacity, then the agent receives a penalty $-\mathcal{M}$, such that $\mathcal{M}$ is a significant number.

This strategy will enforce the DRP agent to keep both buffers empty as much as possible and prevent their overflow, which has a positive impact on the QoS.
As depicted in Figure 5.1, DRP system leverages DDPG algorithm and executed on three different steps: *i*) Decision making (Figure 5.1: $1-6$) presented with blue color; *ii*) Updating policy networks (Figure 5.1: $7-17$) presented with red color; *iii*) Updating target networks (Figure 5.1: $18-19$) presented by green color. In DDPG, we have two networks: *a*) Policy networks that consist of the Actor and the Critic neural networks. These networks are used to predict the deterministic action a_t. While the actor network has as input the state s_t and it is used to predict the action a_t, the critic network has as inputs s_t and a_t and returns the Q value that is used for criticizing the taken action; *b*) The target network that consists of target actor and target critics. These two networks are frozen and used to help the convergence of policy networks and stabilize their learning. In deep learning, the optimizer (i.e., ADAM) should update the neural network parameters of the policy networks closer to the labels, which are the fixed target neural network values. Moreover, to stabilize the learning, a replay buffer is used. The training is performed using a random sample from the replay buffer, which reduces the correlation between the agent's experiences.
Decision making: At the reception of the observation $(\mathcal{O}_t^{\mathcal{U}}, \mathcal{O}_t^{\mathcal{D}})$, the DRP agent generates the state s_t using the equation (5.11) (Figure 5.1: 2). The received state is used by the actor network to predict the deterministic action a_t (5.11) (Figure 5.1: 3). In order to enable the DRP agent to explore the environment, a noise $\mathcal{N}$ is added to the action (5.11) (Figure 5.1: 4). Accordingly, the UL and DL slots are reserved (Figure 5.1: $5-6$).
Updating policy network: In order to take optimal actions, the policy network should be updated (Figure 5.1: $7-17$). The taken action with noise should be stored in the replay buffer (Figure 5.1: 7), as well as their corresponding state and reward (Figure 5.1: 8). First the critic network is updated by leveraging a random batch sample $(s_t, a_t + \mathcal{N}, r_t)$ from the replay buffer (Figure 5.1: $9-14$). Using the MSE and ADAM optimizer, the parameters of the critic network are optimized by considering the critic values and target critic values. Then, the actor network is also optimized by leveraging the gradient generated against the critic network.
Updating policy network: The target networks (actor and critic) should be updated slowly and periodically towards the policy networks using soft update (Figure 5.1: $18-19$). This strategy helps the Algorithm for providing optimal deterministic action.

5.6 Performance evaluation

We have implemented our simulation environment using Python, and Pytorch . We have used a physical machine with 8 Core i7-8665U 1.90GHz and 16 GB of memory using Centos 7 as an operating system. We have employed two fully connected hidden layers of 400 and 300 nodes for both policy and target networks. We have also used layer normalization between the hidden layers to enable smoother gradients, faster training, and better generalization accuracy. While Rectified Linear Unit (ReLU) activation function has been used in the two hidden layers, Hyperbolic Tangent (*tanh*) activation function has been used in the output layer. We have employed a discount factor γ of 0.99, batch size of 1024, and the learning rates of the actor and critic networks are set to 10^{-5} and 10^{-3}, respectively. We have used the soft update with coefficient τ 0.001. Also, ADAM optimizer has been leveraged in both actor and critic networks.

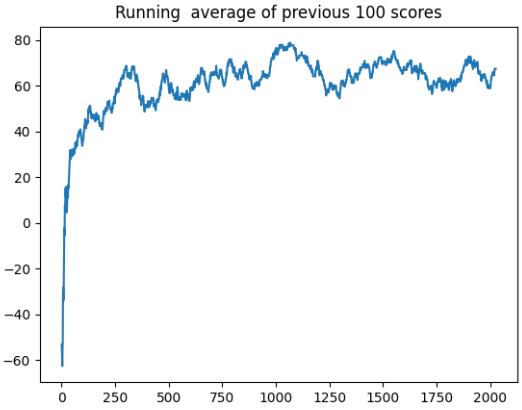
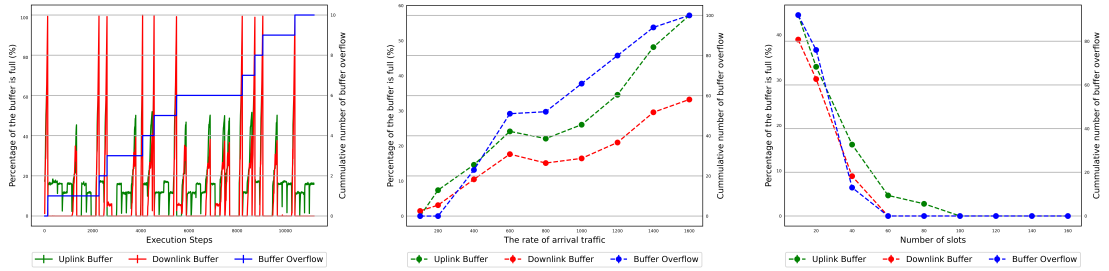


Figure 5.2: Convergence evaluation of DRP agent during the training mode



(a) Impact of time (b) Impact of arrival traffic rate λ (c) Impact of number of slots $\mathcal{T}_\delta$

Figure 5.3: Performance evaluation of DRP system during the inference mode

5.6.1 DRP Training mode

As depicted in Figure 5.2, we have trained our DRP agent using 2000 independent episodes. We have fixed the maximum number of steps at each episode T by 100. We have set the penalty reward $\mathcal{M}$ to -100, and the number of slots $\mathcal{T}_\delta$ to 40. We have considered three successive observations (i.e., $\mathcal{K} = 3$). From the figure, we observe that the DRP agent converges at 500 episodes.

5.6.2 DRP Inference mode

We have evaluated the DRP agent in terms of: *i*) The percentage of the fullness of the UL and DL buffers; *ii*) The number of buffer overflows. In the figure 5.3, the left Y-axis represents the percentage of the fullness of the UL and DL buffers, whereas the right Y-axis represents the number of overflows. In all the experiments, we have fixed the maximum number of steps T by 200. We have conducted three sets of experiments: *i*) We have varied the number of episodes while fixing the number of slots $\mathcal{T}_\delta$ by 40 and the amount of UL $\lambda_\gamma^{\mathcal{U}}$ and DL $\lambda_\gamma^{\mathcal{D}}$ traffic by 400; *ii*) We have varied $\mathcal{T}_\delta$ while fixing $\lambda_\gamma^{\mathcal{U}}$ and $\lambda_\gamma^{\mathcal{D}}$ by 400 and number of episodes by 100; *iii*) We have varied $\lambda_\gamma^{\mathcal{U}}$ and $\lambda_\gamma^{\mathcal{D}}$ while fixing $\mathcal{T}_\delta$ by 40 and number of episodes by 100. Figure 5.3(a) depicts the impact of the execution steps on the DRP agent. The first observation that we can draw from this figure is the average amount of traffics in the UL, and DL buffers are 20%. Also, we observe that the fullness of the buffer did not exceed 45% in the 12000 execution steps. We also

witness that the number of times a buffer overflow happens did not exceed 10 times, which is less than 0.083%. This figure also confirms the ability of the DRP agent for generalizing. While it is trained with $T = 100$, it has been tested with $T = 200$ without feeling any difference.
Figure 5.3(b) depicts the impact of $\lambda_{\gamma}^{\mathcal{U}}$ and $\lambda_{\gamma}^{\mathcal{D}}$ on the DRP agent. Each plotted point presents the average of 100 episodes, each of which with 200 steps. The first observation that we can draw from this figure is the amount of traffic hurts the DRP agent. The more data traffic generation rate is, the higher likelihood to get buffer overflow becomes. We observe that whatever the amount of traffic, the average fullness of both buffers does not exceed 60%. The number of buffer overflows did not exceed 100 cases, which is 0.5% in the worst-case scenarios. The obtained results demonstrate the DRP agent's ability to make the generality and perform well in unseen environments.
Figure 5.3(c) shows the impact of the number of slots on the DRP agent. We observe that the number of slots has a positive impact on the fullness of the buffers and the number of buffer overflows. The more number of slots is, the higher capacity for transmitting the packets becomes. We observe that DRP agent succeeded to get empty buffers when more 60 slots are used. These results demonstrate the efficiency of the proposed solution for making the generality and achieving its main design goals. We also observe that the percentage of buffer fullness does not exceed 45% in the worst-case scenarios. Also, the number of buffer overflows did not exceed 90, which is less than 0.45%.

5.7 Conclusion

In this chapter, we introduced DRP, a DRL-based solution that allows to derive and adjust the TDD pattern in 5G NR. DRP is used by 5G base station before the UE scheduling process to compute the number of slots dedicated to UL and DL in a TDD frame. Without knowing the UEs traffic model, DRP can derive the optimal TDD pattern aiming at reducing both UL and DL buffers. DRP uses available information at the gNB, i.e., BSR, and DL buffer size, to deduce the optimal TDD pattern. Simulation results clearly showed that DRP could avoid buffer overflow and dynamically adapt to the cell traffic.

Chapter 6

Dynamic TDD for a multi-cell 5G environment

6.1 Introduction

In this contribution, we extend DRP to solve the dynamic TDD problem in a multi-cell environment and introduce the MADRL-based 5G RAN TDD Pattern (MADRP) framework. The MADRP approach is fully decentralized. Each MADRP agent is located close to the gNB, serving a particular cell. Compared to a centralized approach, MADRP is executed close to the gNB, which reduces the control latency between the gNB and MADRP. Moreover, MADRP reduces the signaling overhead normally generated when gNBs send data to a central entity. Each MADRP agent monitors the gNB's UL and DL buffers and the number of edge users with neighboring cells. Note that edge users of a cell correspond to UEs attached to that cell and are physically located at the cell's boundaries where the signal strength from neighboring cells is significant. Then, each agent uses this local observation along with messages from neighboring cells to derive the optimal TDD pattern to accommodate connected users while avoiding cross-link interference with neighboring cells.
The key contributions of this chapter are:

- Model the dynamic TDD problem in 5G NR considering a multi-cell environment. The proposed model takes into account dynamic traffic patterns and cross-link interference.
- Design a solution to solving the problem without knowing the traffic pattern. The suggested approach is designed as a fully distributed solution, enabling it to operate near the gNB and minimize control latency consequently.
- The proposed solution is designed to guarantee generality in terms of (i) number of cells (ii) radio configuration in terms of the numerology, the TDD period, and the buffer capacity.
- The proposed solution takes into account the propagation effect of cross-link interference between cells, i.e., the indirect cross-link interference that occurs between two cells without direct cross-link interference due to a third cell interfering with the two cells.
- Implementation of the proposed solution using the MADDPG algorithm [41].
- Simulation results with different numbers of cells under different traffic patterns and cross-link interference probability. We compared MADRP with the optimal solution and different static TDD configurations with different traffic UL/DL proportions. The results show

that MADRP outperforms the static TDD configurations even in high cross-link interference scenario and the gap between the optimal solution and the MADRP solution is small in low-interference scenario.

Table 6.1: Summary of Notations & Variables.

Notation / Variable	**Description**
$\mathcal{C}$	set of radio cells
Γ_c	the set of UEs connected to cell c.
γ	A UE $\gamma \in \Gamma_c$.
δ	TDD period.
ν_c	A numerology used by a cell c.
$\mathcal{T}_\delta^c$	The number of slots in cell c during a period δ . $\mathcal{T}_\delta^c = \{2, \cdots, 16\}$.
$\lambda_\gamma^{\mathcal{U}}$	The UL traffic generated by $\gamma \in \Gamma_c$.
$\lambda_\gamma^{\mathcal{D}}$	The DL traffic from a cell towards the UE $\gamma \in \Gamma_c$.
$\psi_\gamma^{\mathcal{U}}$	The UL buffer of the UE $\gamma \in \Gamma_c$.
$\psi_\gamma^{\mathcal{D}}$	The DL buffer of the UE $\gamma \in \Gamma_c$.
$\lambda_c^{\mathcal{U}}$	The UL traffic generated by Γ_c.
$\lambda_c^{\mathcal{D}}$	The DL traffic from cell c towards the UEs $\gamma \in \Gamma_c$.
$\Psi_c^{\mathcal{U}}$	The UL buffer of cell c.
$\Psi_c^{\mathcal{D}}$	The DL buffer of cell c.
$\mu_c^{\mathbf{D}}$	The amount of traffic in bytes transmitted by cell c per slot.
$\mu_c^{\mathbf{U}}$	The amount of traffic in bytes received by cell c per slot.
α	A constant that specifies the priority between the UL and DL traffics.
$\Phi_c^{\mathcal{U}}$	The initial amount of stored data in bytes in the UL buffer $\psi_c^{\mathcal{U}}$.
$\Phi_c^{\mathcal{D}}$	The initial amount of stored data in bytes in the DL buffer $\psi_c^{\mathcal{D}}$.
$\mathcal{F}_{c_1,c_2}$	A Boolean constant that denotes if there is interference between cells c_1 and c_2.
$\mathcal{X}_c$	A real variable that denotes the percentage of allocated UL slots in cell c.
$\mathcal{Y}_c$	A real variable that denotes the percentage of allocated DL slots in cell c.

6.2 Related works

The problem introduced in this chapter can be decomposed into two sub-problems: (i) The distribution of time slots between the UL and the DL (ii) the mitigation of the cross-link interference between neighboring cells. In the literature, most solutions focus on solving only one of the two sub-problems.

6.2.1 Time slots distribution

In [77], the authors explored employing deep reinforcement learning to adaptively allocate TDD UL/DL resources in 5G networks considering high mobility UEs. However, this work requires additional information that is not available at the gNB, such as the ideal channel capacity and real-time channel capacity. In [78], the authors proposed a service-oriented soft spectrum slicing

for 5G TDD. The objective is to use the flexibility of TDD to adjust the UL/DL dynamically using forecasted traffic and user mobility. The problem has been modeled using a weighted optimization whose objective is to maximize the allocated slice bandwidth for each corresponding load. The weights describe the normalized slice size suitable for each service. Although the paper addresses 5G, it uses the LTE TDD configuration (i.e., fixed TDD patterns) and the proposed solution requires predicting the traffic demands for each service, which can lead to over-allocation or under-allocation of resources when the predicted traffic is not accurate.

6.2.2 Cross-link interference mitigation

The authors of [79] investigated interference management schemes for 5G dynamic TDD and proposed new interference suppression schemes using advanced receivers. They derived the analytic expressions of the receivers for dynamic TDD interference suppression by theoretical analysis. In [80], the authors designed a distributed algorithm to be used by all transmitters to compute their power allocations in real time and thus avoid gNB-to-gNB interference. However, the cost of these solutions is high because they require updating the transmitters and receivers at all the gNBs and UEs. In [81], the authors overview the academic research and standardization efforts undertaken to solve this cross-link interference problem and make the Dynamic TDD system a reality. However, they did not provide any solution.

6.2.3 Joint time slots distribution and cross-link interference mitigation

In [82], the authors proposed a Q-learning approach to reconfigure the TDD pattern of gNBs in order to maximize users' QoE while mitigating cross-link interference. However, they relied on a centralized architecture that does not guarantee generalization, i.e., if they add/remove another gNB, they have to re-train the model from scratch. In addition, they only considered fixed configuration patterns, ignoring the flexibility of 5G NR. In [83], authors proposed a dual RL/ML approach for online pattern optimization in 5G NR TDD deployments. However, they considered a centralized approach with the limitations mentioned above. Also, they require knowledge of the UL latency, which is impossible to obtain in real 5G deployments. In [84], the authors proposed a dynamic resource allocation scheme in TDD. They have designed a clustering algorithm to group the radio units into different sets. Then, they adopted coordinated multipoint technology to eliminate interference in each set. However, they used a centralized approach and considered fixed configuration patterns without taking advantage of the flexibility of 5G NR. In [85], authors modeled the dynamic TDD configuration problem as a dynamic programming problem. Then, they designed a fully decentralized solution with distributed MADRL technology. Each agent in MADRL makes decisions only based on local observations. However, their models lack generality because they assume a fixed number of TDD patterns with a fixed number of slots, so a change in numerology involves retraining all MADRL agents. In addition, they only considered 4 types of traffic, i.e., high UL, high DL, low UL, and low DL, which implies the need to know the traffic type and pattern in advance.
Overall, current solutions rely on a centralized approach that increases control latency and signaling overhead. Furthermore, they do not consider the flexibility of 5G NR in terms of numerology and dynamic number of slots per TDD period which results in a lack of generality and the necessity to retrain the models for each numerology. Furthermore, they ignore the propagation effect of cross-link interference, i.e., indirect cross-link interference that occurs between two cells with no direct interference due to the transitivity relation between cells. Figure 6.1 depicts three cells with a direct cross-link interference between cell 1 and cell 3, cell 2 and cell 1. By transitivity, cell 2 and cell 3 are experiencing indirect cross-link interference. As a result,

the three cells must align their TDD period even without direct cross-link interference between cell 2 and cell 3.

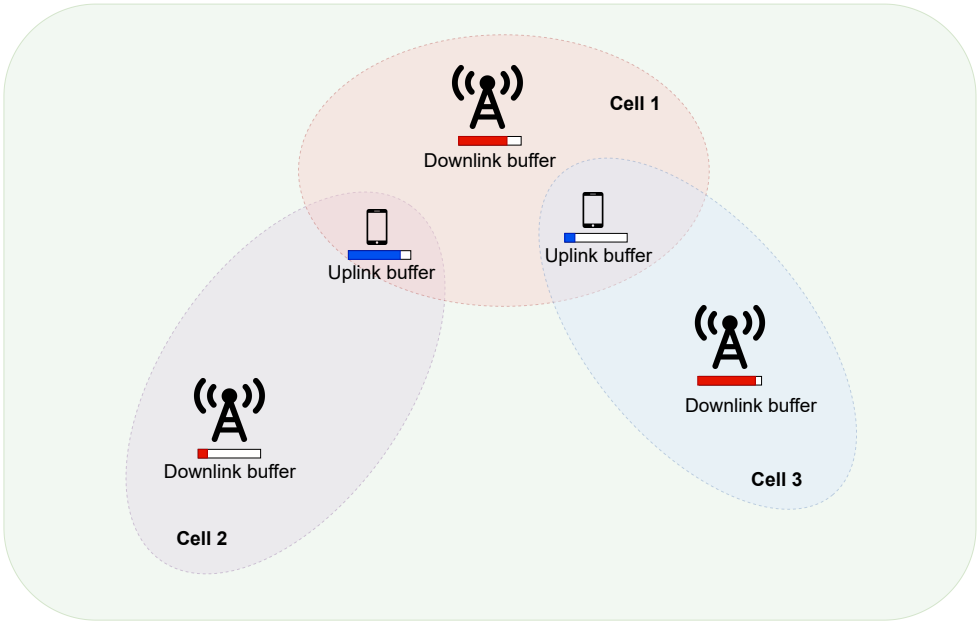


Figure 6.1: Multi-cell scenario with 3 Radio Units (RU) and 2 User Equipment(UE) with different traffic characteristics experiencing indirect interference

6.3 Network model and problem formulation

6.3.1 Network model

In the system, we consider a set of RAN cells, denoted $\mathcal{C}$, whereby a set of UEs, denoted Γ_c, are connected to each cell $c \in \mathcal{C}$. We assume that cells are covered by radio units operating under TDD and using a fixed TDD period δ and the same numerology. For the sake of paper readability, the used notations are summarized in Table 6.1. Each UE $\gamma \in \Gamma_c$ has UL and DL traffic that can vary from one application to another. The UL traffic is generated and transmitted from the UE γ to the cell c radio unit, while the DL traffic is sent from the radio unit of cell c to the UE γ. Let $\lambda_\gamma^{\mathcal{U}}$ and $\lambda_\gamma^{\mathcal{D}}$ denote the amount of UL and DL traffic in bytes of UE γ, respectively. In contrast to LTE, where the DL traffic $\lambda_\gamma^{\mathcal{D}}$ is more critical than the UL traffic $\lambda_\gamma^{\mathcal{U}}$, in $5G$ networks and beyond the amount of traffic in UL and DL is application dependent. For instance, in AR applications that offload the AR processing to the edge, high UL traffic $\lambda_\gamma^{\mathcal{U}}$ is expected. On another side, in some applications, such as video streaming, the DL traffic $\lambda_\gamma^{\mathcal{D}}$ is more important. Other applications, such as immersive applications (e.g., the Metaverse), require high data rates in both directions. The users in new generation applications are characterized by colossal collaborative interactions, tremendous precision, and high data synchronization. Furthermore, the same UE γ can use multiple applications, which makes it hard to predict the UL $\lambda_\gamma^{\mathcal{U}}$ and DL $\lambda_\gamma^{\mathcal{D}}$ traffic of a UE γ. Let $\lambda_c^{\mathcal{U}}$ and $\lambda_c^{\mathcal{D}}$ denote the traffic of the cell c in UL and DL, respectively. Formally, $\lambda_c^{\mathcal{U}} = \sum_{\gamma \in \Gamma_c} \lambda_\gamma^{\mathcal{U}}$, and $\lambda_c^{\mathcal{D}} = \sum_{\gamma \in \Gamma_c} \lambda_\gamma^{\mathcal{D}}$.

Let $\psi_\gamma^{\mathcal{U}}$ and $\psi_\gamma^{\mathcal{D}}$ denote the UL and DL buffer size of UE γ, respectively. While $\psi_\gamma^{\mathcal{U}}$ is located at the UE γ, $\psi_\gamma^{\mathcal{D}}$ is located at the gNB connected to the radio unit serving cell $c \in \mathcal{C}$. The

UE γ periodically keeps informing the gNB about the state of $\psi_{\gamma}^{\mathcal{U}}$. In 5G NR, this operation corresponds to the BSR sent by UE when requesting UL resources. Meanwhile, the DL buffers are monitored by gNB, corresponding to the radio bearer data channels maintained by gNB for each UE. Let $\Psi_c^{\mathcal{U}}$ and $\Psi_c^{\mathcal{D}}$ denote the UL and DL buffer of all the UEs connected to cell $c \in \mathcal{C}$. Formally, $\Psi_c^{\mathcal{U}} = \sum_{\gamma \in \Gamma_c} \psi_{\gamma}^{\mathcal{U}}$, and $\Psi_c^{\mathcal{D}} = \sum_{\gamma \in \Gamma_c} \psi_{\gamma}^{\mathcal{D}}$. For the sake of simplicity and without loss of generality, we assume that each cell c has a maximum UL $\Psi_c^{\mathcal{U}}$ and DL $\Psi_c^{\mathcal{D}}$ buffers size, respectively. Let $|\Psi_c^{\mathcal{U}}|$ and $|\Psi_c^{\mathcal{D}}|$ denote the maximum size of UL and DL buffers in bytes, respectively.

The UEs at the edge of the cells experience cross-link interference when two or more neighbor cells are using the same frequency and different slot directions (i.e., UL and DL) at a given time. Figure 6.2 illustrates a scenario of 3 neighboring cells and 7 UEs with different traffic patterns. Each UE is connected to a cell (the connection is illustrated with a pointed line). The oval shape around each radio unit represents the cell coverage. In this example, the intersection between cells' coverage represents the cross-link interference region wherein UEs can experience cross-link interference.

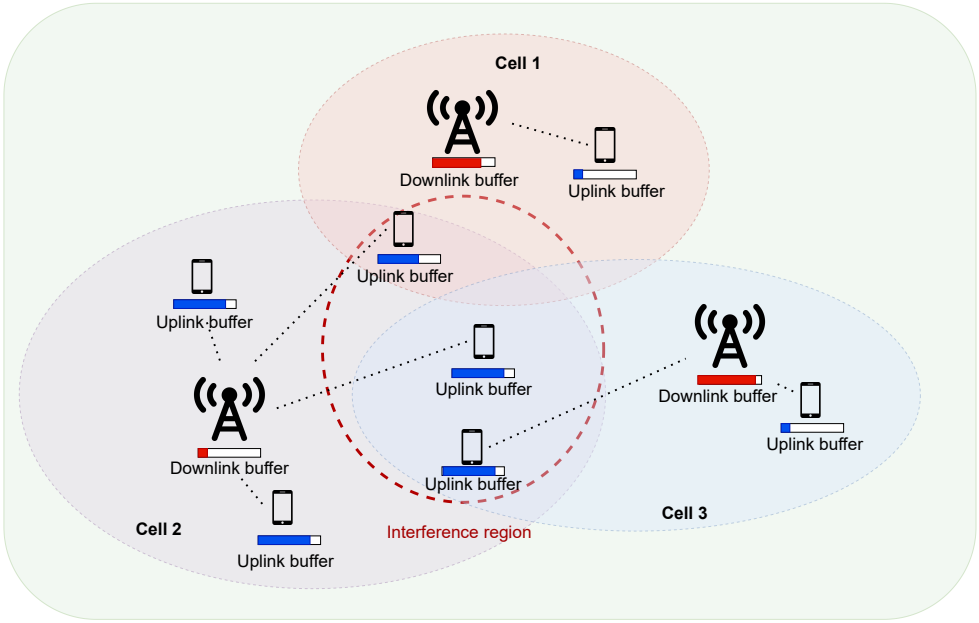


Figure 6.2: Multi-cell scenario with 3 Radio Units (RU) and 7 User Equipment(UE) with different traffic characteristics showing the interference region

6.3.2 Problem formulation

In this work, we assume that all the cells in $\mathcal{C}$ are using the same TDD period δ and are synchronized in time (i.e., slots boundaries of all the cells are aligned). However, the distribution of the UL and DL traffic is unknown. The main research question targeted by this paper is how to distribute the TDD slots among the UL and DL traffic to guarantee that: (i) UL and DL buffers are not overflowed; (ii) edge users do not experience cross-link interference. The main challenge addressed in this paper is that both UL $\lambda_c^{\mathcal{U}}$ and DL $\lambda_c^{\mathcal{D}}$ traffics are unknown and hard to predict. Let $\mu_c^{\mathcal{D}}$ and $\mu_c^{\mathcal{U}}$ denote the transmission capacity of cell c per slot in DL and UL, respectively. We assume that the cell capacity is static during the period δ. Let $\mathcal{X}_c$ be a real variable that denotes the percentage of UL slots available at cell $c \in \mathcal{C}$. Similarly, let $\mathcal{Y}_c$ be a

real variable that denotes the percentage of reserved slots for the DL. Formally, the following statements should hold (6.2), (6.3) and (6.4). All the UEs $\gamma \in \Gamma_c$ will share the UL and DL slots. The number of UL and DL slots in which a UE γ is scheduled is proportional to the UE's UL and DL buffers, respectively. Further, the percentage of UL and DL slots in which a UE $\gamma \in \Gamma_c$ is scheduled is defined as follows $\S_\gamma = \frac{\psi_\gamma^{\mathcal{U}}}{\Psi_c^{\mathcal{U}}} \times \mathcal{X}_c$ and $\dagger_\gamma = \frac{\psi_\gamma^{\mathcal{D}}}{\Psi_c^{\mathcal{D}}} \times \mathcal{Y}_c$, respectively.
Let α be a given constant ($0 \leq \alpha \leq 1$) that defines the priority between the UL and DL traffics. This parameter can be defined by the solution designer to specify the priorities between the two traffics. If $\alpha = 1$, then we are interested only in optimizing the UL traffic. Otherwise, if $\alpha = 0$, we are only interested in optimizing the DL traffic.
The proposed solution should be periodically applied to specify $\mathcal{X}_c$ and $\mathcal{Y}_c$ aiming at preventing the overflow of UL buffers $\Psi_c^{\mathcal{U}}$ and DL buffers $\Psi_c^{\mathcal{D}}$. Let $\mathcal{F}_{c_1,c_2}$ be a Boolean constant (i.e., fixed by the system for one iteration) that denotes if there is an interference between the two neighboring cells c_1 and c_2. $\mathcal{F}_{c_1,c_2}$ equals to 0 when there are no edge users between c_1 and c_2. $\mathcal{F}_{c_1,c_2}$ equals to 1 otherwise. If $\mathcal{F}_{c_1,c_2}$ equals to 1, cells c_1 and c_2 must use the same TDD pattern (i.e., the same percentage of UL and DL slots in the TDD period δ). This helps to avoid the cross-link interference at the edge UEs. Otherwise, cells c_1 and c_2 can use different TDD patterns since there is no edge users that will be impacted by the cross-link interference. Let $\Phi_c^{\mathcal{U}}$ and $\Phi_c^{\mathcal{D}}$ denote the initially stored data of the UL and DL buffers. Both $\Phi_c^{\mathcal{U}}$ and $\Phi_c^{\mathcal{D}}$ are initialized by zero. At each iteration, we aim to optimize the following linear integer programming:

$$\begin{aligned} \min \sum_{c \in \mathcal{C}} \Bigg(& \tfrac{\alpha}{|\Psi_{\mathcal{C}}^{\mathcal{U}}|} \times \left(\Phi_c^{\mathcal{U}} + \lambda_c^{\mathcal{U}} - \mu_c^{\mathcal{U}} \times \mathcal{X}_c \times \mathcal{T}_\delta^c \right) + \\ & \tfrac{1-\alpha}{|\Psi_{\mathcal{C}}^{\mathcal{D}}|} \times \left(\Phi_c^{\mathcal{D}} + \lambda_c^{\mathcal{D}} - \mu_c^{\mathcal{D}} \times \mathcal{Y}_c \times \mathcal{T}_\delta^c \right) \Bigg) \end{aligned} \tag{6.1}$$

S.t,

$$\forall c \in \mathcal{C} : 0 \leq \mathcal{X}_c \leq 1 \tag{6.2}$$

$$\forall c \in \mathcal{C} : 0 \leq \mathcal{Y}_c \leq 1 \tag{6.3}$$

$$\forall c \in \mathcal{C} : \mathcal{X}_c + \mathcal{Y}_c = 1 \tag{6.4}$$

$$\forall (c_1, c_2) \in \mathcal{C}^2,\ c_1 \neq c_2 : \mathcal{F}_{c_1,c_2} \times \ (\mathcal{X}_{c_1} - \mathcal{X}_{c_2}) = 0 \tag{6.5}$$

$$\forall c \in \mathcal{C} : \Phi_c^{\mathcal{U}} + \lambda_c^{\mathcal{U}} - \mu_c^{\mathcal{U}} \times \mathcal{X}_c \times \mathcal{T}_\delta^c \leq |\Psi_{\mathcal{C}}^{\mathcal{U}}| \tag{6.6}$$

$$\forall c \in \mathcal{C} : \Phi_c^{\mathcal{D}} + \lambda_c^{\mathcal{D}} - \mu_c^{\mathcal{D}} \times \mathcal{Y}_c \times \mathcal{T}_\delta^c \leq |\Psi_{\mathcal{C}}^{\mathcal{D}}| \tag{6.7}$$

$$\forall c \in \mathcal{C} : \mathcal{X}_c \times \mathcal{T}_\delta^c = \mathcal{A}_c \tag{6.8}$$

$$\forall c \in \mathcal{C} : \mathcal{Y}_c \times \mathcal{T}_\delta^c = \mathcal{B}_c \tag{6.9}$$

$$\forall c \in \mathcal{C} : (\mathcal{A}_c, \mathcal{B}_c) \in \mathcal{N}^2 \tag{6.10}$$

The objective function (6.1) aims to minimize the amount of stored data in the UL and DL buffers in all the cells to prevent their buffers overflow. While $\lfloor \mathcal{X}_c \times \mathcal{T}_\delta^c \rfloor$ denotes the number of slots reserved for the UL traffic, $\lceil \mathcal{Y}_c \times \mathcal{T}_\delta^c \rceil$ corresponds to the number of slots reserved for the DL traffic in each cell $c \in \mathcal{C}$ during the period δ. We have used the weighted normalized sum method to prevent an objective (i.e., buffer) from dominating the other. Meanwhile, constraints (6.2), (6.3) and (6.4) ensure that the variables $\mathcal{X}_c$ and $\mathcal{Y}_c$ are rates of slots distribution for UL and DL. Meanwhile, constraint (6.5) ensures that two cells experiencing cross-link interference can not use different TDD patterns (i.e., different percentages of UL and DL slots in the TDD period δ), aiming at avoiding the impact of the cross-link interference on the edge UEs. Meanwhile, constraints (6.6) and (6.7) ensure that the UL and DL buffers of $c \in \mathcal{C}$ do not overflow, respectively. Note that $\mathcal{A}_c$ and $\mathcal{B}_c$ are two integer variables that denote the number of UL slots and DL slots of cell c, respectively. Constraints (6.8) and (6.9) transform the percentages of UL and DL slots to integers given the number of slots available in the period δ. Constraint (6.10) ensures that the number of UL and DL slots allocated in each cell $c \in \mathcal{C}$ is an integer.
The proposed model can be reformulated by replacing $\mathcal{Y}_c$ by $1 - \mathcal{X}_c$ in equations (6.1), (6.3), (6.5), (6.7) and (6.9) and removing the constants from the Objective function (6.1). Therefore, we obtain a boxed-constrained linear minimization problem as follows:

$$\min \sum_{c \in \mathcal{C}} \left(\left(\frac{1-\alpha}{|\Psi_{\mathcal{C}}^{\mathcal{D}}|} \times \mu_c^{\mathcal{D}} \times T_\delta^c - \frac{\alpha}{|\Psi_{\mathcal{C}}^{\mathcal{U}}|} \times \mu_c^{\mathcal{U}} \times T_\delta^c \right) \times \mathcal{X}_c \right) \tag{6.11}$$

S.t,

$$\forall c \in \mathcal{C} : 0 \leq \mathcal{X}_c \leq 1 \tag{6.12}$$

$$\forall (c_1, c_2) \in \mathcal{C}^2,\ c_1 \neq c_2 : \mathcal{F}_{c_1,c_2} \times \ (\mathcal{X}_{c_1} \ - \ \mathcal{X}_{c_2}) \ = \ 0 \tag{6.13}$$

$$\forall c \in \mathcal{C} : \frac{\Phi_c^{\mathcal{U}} + \lambda_c^{\mathcal{U}} - |\Psi_{\mathcal{C}}^{\mathcal{U}}|}{\mu_c^{\mathcal{U}} \times T_\delta^c} \ \leq \mathcal{X}_c \tag{6.14}$$

$$\forall c \in \mathcal{C} : \mathcal{X}_c \ \leq \ \frac{-\Phi_c^{\mathcal{D}} - \lambda_c^{\mathcal{D}} + |\Psi_{\mathcal{C}}^{\mathcal{D}}| \ + \mu_c^{\mathcal{D}} \times T_\delta^c}{\mu_c^{\mathcal{D}} \times T_\delta^c} \tag{6.15}$$

If we assume a static environment where we know the traffic distribution, it requires an exponential time to solve the problem optimally in the worst case using the simplex method as shown in [86]. However, we should recall that to solve the optimization problem, there is a need to know in advance the exact traffic model, which is not possible in reality.
Unfortunately, we cannot use the optimization problem mentioned above for distributing the slots in a dynamic environment. This is mainly due to the fact that the amount of UL $\lambda_c^{\mathcal{U}}$ and DL $\lambda_c^{\mathcal{D}}$ traffics are unknown and hard to be predicted a priory.

6.4 MADRP System Overview

As stated earlier, it is hard to efficiently distribute TDD slots by solving the optimization model without prior knowledge of traffic generation patterns. Besides, traffic patterns between cells may differ, and cross-link interference between neighboring cells may exist. For these reasons, using a multi-agent framework where each agent controls a cell configuration is essential. Each agent will adapt the TDD pattern to accommodate its cell traffic while agents of neighboring cells collaborate with each other to align their TDD pattern to avoid cross-link interference

when there are edge UEs between neighboring cells. The multi-agent framework has several motivations: (*i*) Decreases the control latency since the agents are executed near the gNB; (*ii*) Reduces the control overhead compared to the centralized solution since the agents observe local information to make a decision instead of sending the observation to a central entity to take the decision; (*iii*) Decreases the action space since each agent will take one action compared to a single agent solution where the agent will take a tuple of actions. Decreasing the action space makes the training faster and more stable. In this context, we propose the MADRP system that leverages MADRL, more precisely, the MADDPG algorithm, to dynamically define the 5G NR TDD pattern. The MADRL hides the complexity and stochastically of the environment and helps the MADRP framework to make efficient and quick decisions that adapt according to traffic patterns. Besides, MADRL allows different cells to make distributed decisions using local information, enabling communication between cells to collaborate and avoid cross-link interference. Moreover, the MADRP framework gains the ability to learn with time and adapt to different and unseen situations.

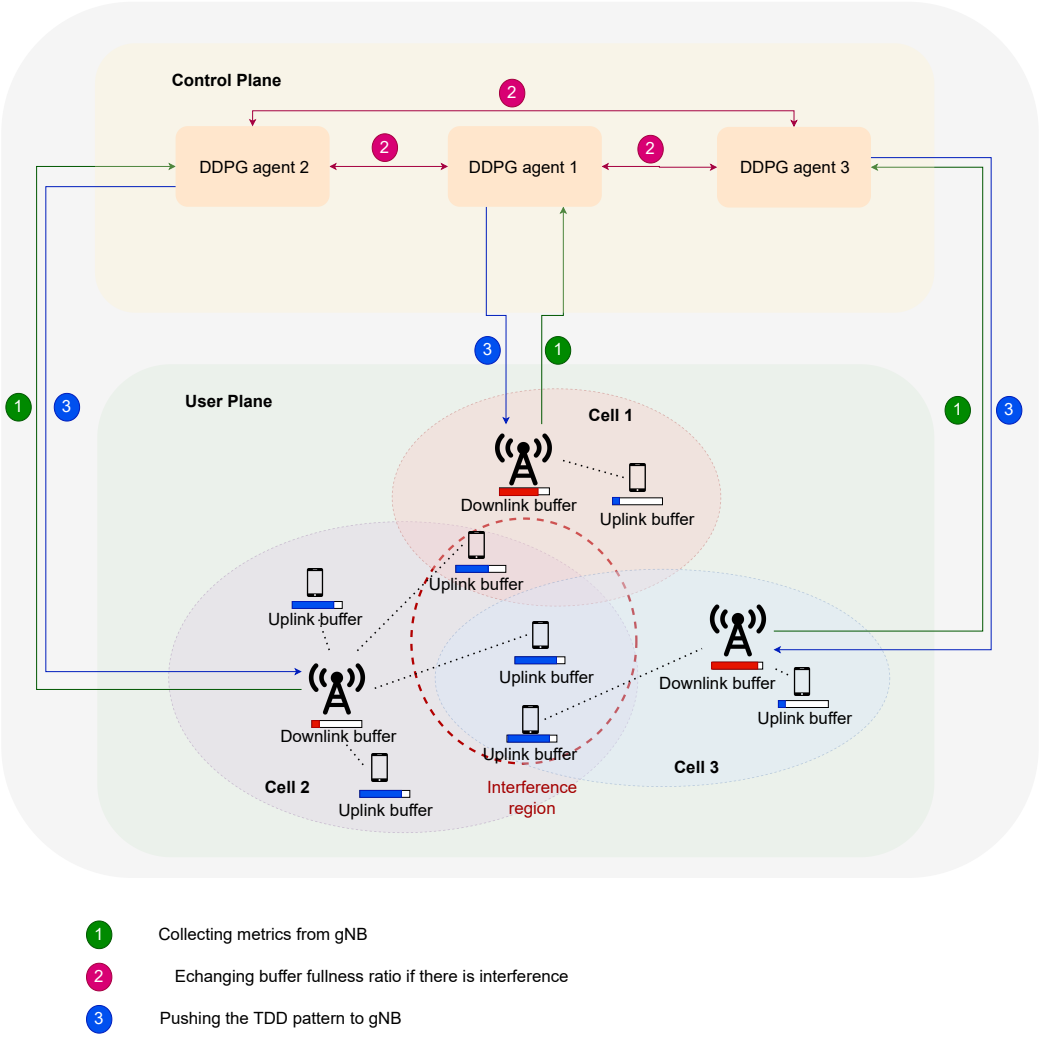


Figure 6.3: MADRP Architecture

As depicted in Figure 6.3, we propose a MADRL learning scheme with one agent for each cell. The set of agents is part of the RAN control plane, while the RAN functions executing dynamic TDD at the gNB level are part of the RAN user plane. Each agent observes the state of

the cell's buffers and exchanges information with neighboring agents (i.e., agents controlling the neighboring cells), thus observing the state of the buffers associated with neighboring cells in case of cross-link interference. Communication between agents can be based on the Xn Application Protocol (XnAP) specification to comply with 3GPP [87]. Specifically, an agent will send an XnAP message containing the Information Element (IE) "Intended TDD DL-UL Configuration NR" to inform the destination gNB of the source gNB's current TDD configuration in order to mitigate cross-link interference.
We have adopted a centralized training and decentralized execution approach. Thus, we allow the policies to use extra information to ease the training step, so long as this information is not used at test time. It is unnatural to do this with Q-learning, as the Q function generally cannot contain different information at training and test time. Thus, MADDPG extends the actor-critic policy gradient methods, whereby the critic is augmented with extra information about other agents' policies.
More concretely, let consider $\mathcal{N}$ agents with policies parameterized by $\theta = (\theta_1, ..., \theta_{\mathcal{N}})$, and let $\pi = (\pi_1, ..., \pi_{\mathcal{N}})$ be the set of all agent policies. Let $[R_i]$ be the cumulative discounted reward of agent i. As we have seen in the background section, each agent aims to maximize R_i. And, since MADDPG is a policy gradient method, it samples the actions directly from the policy π. Let $\mu = (\mu_1,..., \mu_{\mathcal{N}})$ be a continuous and deterministic policy function. Then we can write the gradient of the expected cumulative reward for agent i, $J(\theta_i) = \mathbb{E}[R_i]$ as:

$$\nabla_{\theta_i} J\left(\boldsymbol{\mu}_i\right)=\mathbb{E}_{s, a \sim \mathcal{D}}\left[\nabla_{\theta_i} \boldsymbol{\mu}_i\left(a_i \mid o_i\right) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}\left(s, a_1, \ldots, a_N\right)|_{a_i=\boldsymbol{\mu}_i\left(o_i\right)}\right] \tag{6.16}$$

Where Q_i^{μ} $(s,\ a_1,\ ...,\ a_{\mathcal{N}}\ )$ is a centralized action-value function that takes as input the actions of all agents, $a_1,\ ...,\ a_{\mathcal{N}}$ as well as some state information s and outputs the Q-value for agent i. In the simplest case, s could consist of the observations of all agents, $s = (o_1,\ ...,\ o_{\mathcal{N}}\ )$. However, we may also include additional state information if available. Since each Q_i^{μ} is separately learned, agents can have arbitrary reward structures, including conflicting rewards in a competitive setting. Here the experience replay buffer $\mathcal{D}$ contains the tuples $(s^t,\ s^{t+1},\ a_1^t,\ ...,\ a_{\mathcal{N}}^t,\ r_1^{t+1},\ ...,\ r_{\mathcal{N}}^{t+1}\ )$, recording experiences of all agents.

6.5 MADRP detailed description

We have designed the MADRP to be lightweight to ensure fast interaction with the environment. Also, we have designed the MADRP to ensure generality and then work in an unseen environment. Besides, MADRP has been designed to work independently from the number of slots, the size of the buffers, and the number of cells. Moreover, it considers the variation and correlation in the buffer states to predict the traffic patterns. In what follows, we define the elements of the MADRP, including the state, the reward, and the action.
i) State: Let $\xi_{\mathcal{U},c}^t$ and $\xi_{\mathcal{D},c}^t$ denote the amount of traffic in the UL $\Psi_c^{\mathcal{U}}$ and the DL $\Psi_c^{\mathcal{D}}$ at the step t at the cell c, respectively. To ensure the generalization, we define the observation $\mathcal{O}_{\mathcal{U},c}^t$ and $\mathcal{O}_{\mathcal{D},c}^t$ of the UL and DL buffers as normalized values (Figure 6.4: 1). Formally, $\mathcal{O}_{\mathcal{U},c}^t = \frac{\xi_{\mathcal{U},c}^t}{|\Psi_c^{\mathcal{U}}|}$ and $\mathcal{O}_{\mathcal{D},c}^t = \frac{\xi_{\mathcal{D},c}^t}{|\Psi_c^{\mathcal{D}}|}$, respectively. The benefits of the normalization are twofold: *i*) It ensures generality by enabling the MADRP agent to be agnostic to the scenario scale in terms of the buffer capacity. It works similarly in different buffers with different sizes. The most important is to catch the buffer fullness ratio of $\Psi_c^{\mathcal{U}}$ and $\Psi_c^{\mathcal{D}}$; *ii*) It is well known that the activation functions in the neural network work well for small values, which positively impacts MADRP's convergence. Moreover, to capture the traffic patterns, we define the state s_j^t of the agent j at time t as follows:

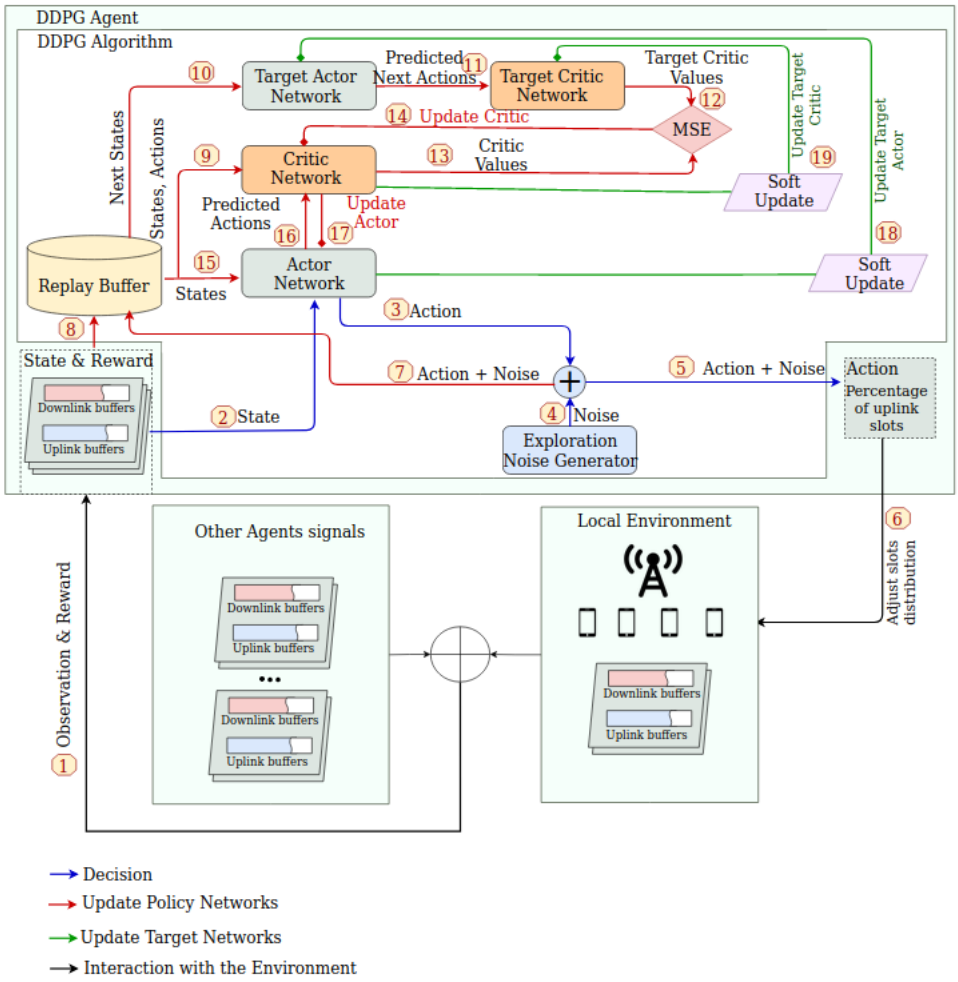


Figure 6.4: MADDPG architecture and workflow

$$s_j^t = \bigcup_{k=0}^{\mathcal{N}} \hat{s}_{j,k}^t \tag{6.17}$$

Where $\hat{s}_{j,k}^t$ is defined as follows:

$$\hat{s}_{j,k}^t = \begin{cases} \left(\bigcup_{i=0}^{\mathcal{K}-1} \mathcal{O}_{\mathcal{U},\,j}^{t-i}, \bigcup_{i=0}^{\mathcal{K}-1} \mathcal{O}_{\mathcal{D},\,j}^{t-i} \right) & if\ k\ =\ j\ or\ \mathcal{F}(k,j) = 1 \\ \left(\bigcup_{i=0}^{\mathcal{K}-1} 0, \bigcup_{i=0}^{\mathcal{K}-1} 0 \right) & else \end{cases} \tag{6.18}$$

, whereby $\mathcal{F}(k,j) = 1$ when there is an interference between cells k and j (i.e., direct cross-link interference) or $\exists k' \in \mathcal{C}$ such that $\mathcal{F}(k,k') = 1$ or $\mathcal{F}(k',j) = 1$ (i.e., indirect cross-link interference)
In the state s_j^t, the MADRP agents, besides the current observation, consider $\mathcal{K}$ previous observations before taking any action. This enables capturing the behavior of both buffers and traffic before taking any action.
Besides, MADRP agents capture the behavior of the function $\mathcal{F}$ by communicating with each other. An agent j sends a message containing $\left(\bigcup_{i=0}^{\mathcal{K}-1} \mathcal{O}_{\mathcal{U},\,j}^{t-i}, \bigcup_{i=0}^{\mathcal{K}-1} \mathcal{O}_{\mathcal{D},\,j}^{t-i} \right)$ to agent k when agent

j observes an edge user with the cell associated to agent k. This message is considered an interference alert. The operator can define a policy to trigger the interference alert, for example, when the number of edge users exceeds a predefined threshold. Each agent stores a list of agents with which it has interference, and each time it receives an interference alert, it broadcasts the received alert to its interference list. This mechanism allows MADRP to capture indirect interference (i.e., interference between two cells without edge users between each other but $\exists k \in \mathcal{C}$ such that there is interference between cell k and both cells).

ii) Action: We have only one continuous action a_j^t per agent j that presents the percentage of slots that should be reserved for the UL traffic at step t. Each MADRP agent $j \in \mathcal{G}$ enforces the taken decisions as depicted in Figure 6.4:6. Accordingly, the $\lfloor a_j^t \times \mathcal{T}_\delta^j \rfloor$ slots are reserved for the UL traffic, and $\lceil (1 - a_j^t) \times \mathcal{T}_\delta^j \rceil$ slots are reserved for the DL traffic. It is worth noting that the action is independent of the number of slots $\mathcal{T}_\delta^j$, which ensures the generality and enables the MADRP agent to be agnostic to the scenario scale.

iii) Reward: We have adapted an episodic approach, whereby each episode runs for max T steps before it ends.

Let $\widehat{r_j^t}$ denotes the buffer reward of each agent j, defined in (6.19) as follows:

$$\widehat{r_j^t} = \begin{cases} \alpha \times (1 - \mathcal{O}_{\mathcal{U},\ j}^t) + (1 - \alpha) \times (1 - \mathcal{O}_{\mathcal{D},\ j}^t) & if\ |\mathcal{O}_{\mathcal{U},\ j}^t| < 1 \wedge |\mathcal{O}_{\mathcal{D},\ j}^t| < 1 \\ -\mathcal{M} & else \end{cases} \tag{6.19}$$

Equation (6.19) allows MADRP agents to get a positive reward for each step, which succeeds in keeping the buffers $\Psi_c^{\mathcal{U}}$ and $\Psi_c^{\mathcal{D}}$ do not exceed their threshold. Moreover, the emptiest the buffers are, the highest reward becomes. When one of the buffers exceeds its capacity, then the agent receives a penalty $-\mathcal{M}$, such that $\mathcal{M}$ is a significant number. This strategy will force the MADRP agents to keep both buffers empty as much as possible and prevent their overflow, which positively impacts the QoS. α is the priority between the UL and DL traffic.

Let $\overline{r_{j,\ k}^t}$ denotes the cross-link interference reward, defined in (6.20) as follows:

$$\overline{r_{j,\ k}^t} = \begin{cases} 0 & if\ \mathcal{F}(j,k)\ = 0\ \vee\ \mathcal{F}(j,k)\ = 1\ \wedge\ a_j^{t-1} = a_k^{t-1} \\ -\mathcal{M} & else \end{cases} \tag{6.20}$$

Equation (6.20) gives a penalty when two agents experience direct or indirect interference and do not choose the same TDD pattern (i.e., the same percentage of UL slots).

The agent reward r_j^t is the sum of the buffer and the cross-link interference reward, defined in (6.21) as follows:

$$r_j^t = \widehat{r_j^t} + \sum_{k\ \in\ \mathcal{C}} \left(\overline{r_{j,\ k}^t} \right) \tag{6.21}$$

As depicted in Figure 6.4, the MADRP system leverages the MADDPG algorithm and is executed on three different steps: *i*) Decision making (Figure 6.4: $1-6$) presented with blue color; *ii*) Updating policy networks (Figure 6.4: $7-17$) presented with red color; *iii*) Updating target networks (Figure 6.4: $18-19$) presented by green color. Each MADDPG agent has two networks: *a*) Policy networks that consist of the Actor and the Critic neural networks. These networks are used to predict the deterministic actions a_j^t. While the actor-network has as input the state s_j^t and it is used to predict the action a_j^t, the critic-network has as inputs $s^t = s_1^t, ..., s_{\mathcal{N}}^t$ and $a^t = a_1^t, ..., a_{\mathcal{N}}^t$ and returns the Q value that is used for criticizing the taken action; *b*) The target networks that consist of target actor-networks and target critic-network. These two networks are frozen and used to help the convergence of policy networks and stabilize their learning. In deep learning, the optimizer (e.g., ADAM) should update the neural network parameters of the policy networks closer to the labels, which are the fixed target neural network values. Moreover, to stabilize the learning, a replay buffer is used. The training is performed

using a random replay buffer sample, reducing the correlation between the agents' experiences.

Algorithm 5 Multi-Agent Deep Deterministic Policy Gradient for $\mathcal{N}$ agents

1: **for** episode = 1 to $\mathcal{M}$ **do**
2: Initialize a random process $\mathcal{J}_i$ for action exploration for each agent i
3: Receive initial state s
4: **for** t = 1 to max-episode-length **do**
5: for each agent i, select action $a_i = \boldsymbol{\mu}_{\theta_i}(o_i) + \mathcal{J}_i^t$ w.r.t. the current policy and exploration
6: Execute actions $a = (a_1, \ldots, a_N)$ and observe reward r and new state s'
7: Store (s, a, r, s') in replay buffer $\mathcal{D}$
8: $s \leftarrow s'$
9: **for** agent i = 1 to N **do**
10: Sample a random minibatch of S samples $\left(s^j, a^j, r^j, s'^j\right)$ from $\mathcal{D}$
11: Set $y^j = r_i^j + \gamma Q_i^{\boldsymbol{\mu}'}\left(s'^j, a'_1, \ldots, a'_N\right)\Big|_{a'_k = \boldsymbol{\mu}'_k(o_k^j)}$
12: Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\mu}\left(s^j, a_1^j, \ldots, a_N^j\right)\right)^2$
13: Update actor using the sampled policy gradient: $\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\boldsymbol{\mu}_i\left(o_i^j\right)\nabla_{a_i} Q_i^{\boldsymbol{\mu}}\left(s^j, a_1^j, \ldots, a_i, \ldots, a_N^j\right)\Big|_{a_i = \boldsymbol{\mu}_i(o_i^j)}$
14: **end for**
15: Update target network parameters for each agent i: $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$
16: **end for**
17: **end for**

Decision making: At the reception of the observation $(\mathcal{O}^t_{\mathcal{U},c}, \mathcal{O}^t_{\mathcal{D},c}, \bigcup_{i=0}^{\mathcal{N}} n_j^t)$, with n_j^t is the number of users at the edge between the current agent' cell c and agent j' cell. n_j^t is used to set the variable $\mathcal{F}(c,j)$, i.e., $\mathcal{F}(c,j) = 1$ if $n_j^t > 0$ or a message is received from agent j, $\mathcal{F}(c,j) = 0$ otherwise. Each MADRP agent generates the state s_j^t using the equation (6.18) (Figure 6.4: 2). The received state is used by the actor-network to predict the deterministic action a_j^t. In order to enable the MADRP agents to explore the environment, a noise $\mathcal{J}_j^t$ is added to the action. Accordingly, the UL and DL slots are reserved (Figure 6.4: $5-6$).

Updating policy network: In order to take optimal actions, the policy network should be updated (Figure 6.4: $7-17$). The action taken by all the agents should be stored in the replay buffer (Figure 6.4: 7), as well as their corresponding state and reward (Figure 6.4: 8). First the critic-network is updated by leveraging a random batch sample (s^t, s^{t+1}, $a_1^t + \mathcal{J}_1^t$, ..., $a_{\mathcal{N}}^t + \mathcal{J}_n^t$, r_1^{t+1}, ..., $r_{\mathcal{N}}^{t+1}$) from the replay buffer (Figure 6.4: $9-14$). Using mean square error (MSE) and ADAM optimizer, the parameters of the critic-network are optimized by considering the critic values and target critic values. Then, the actor-network is also optimized by leveraging the gradient generated against the critic-network.

Updating target networks: The target networks (actor and critic) should be updated slowly and periodically towards the policy networks using soft update (Figure 6.4: $18-19$). This strategy helps the Algorithm for providing optimal deterministic action. We utilize a distributed training procedure, where only the critic-networks are identical among the agents while the actor-

networks are specific for each agent. Hence, the execution phase is distributed, with each agent making its own decision at each interval relying only on the specific observations it receives from the environment and its neighbors.
Overall, the MADDPG algorithm is summarized in Algorithm 5 where $\mu' = (\mu'_1, ..., \mu'_{\mathcal{N}})$ is the set of target policies with delayed parameters θ'_i.

6.6 Performance Evaluation

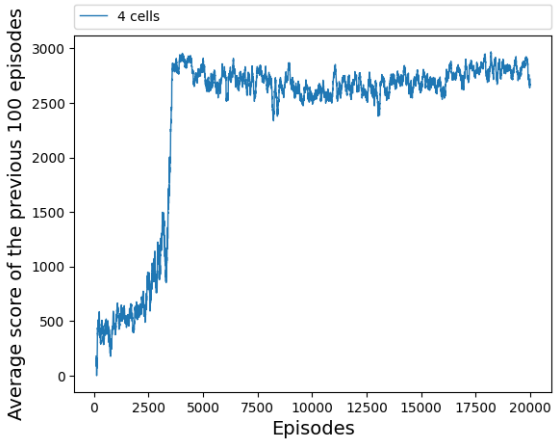
We implemented our simulation environment using Python and Pytorch. We leveraged the simulator built in [40] and the open-source MADDPG implementation [41]. We used a physical machine with Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz with 64 GB of memory and NVIDIA GP102 GPU using Ubuntu 20 as an operating system. Following an empirical approach, we conducted several experiments with different neural network parameters and activation functions. We selected those which gave us the best performance. For all the agents, we employed two fully connected hidden layers of 400 and 300 nodes for both policy and target networks. We also used layer normalization between the hidden layers to enable smoother gradients, faster training, and better generalization accuracy. While Rectified Linear Unit (ReLU) activation function has been used in the two hidden layers, Hyperbolic Tangent ($tanh$) activation function has been used in the output layer. We employed a discount factor γ of 0.99, batch size of 1024, and the learning rates of the actor and critic-networks are set to 10^{-5} and 10^{-3}, respectively. We used the soft update with coefficient τ 0.001. Also, ADAM optimizer has been leveraged in both actor and critic-networks. Finally, the optimization problem is solved using Gurobi version 9.1.2. The absolute optimality gap is set to 10^{-8}.

MADRP Training mode

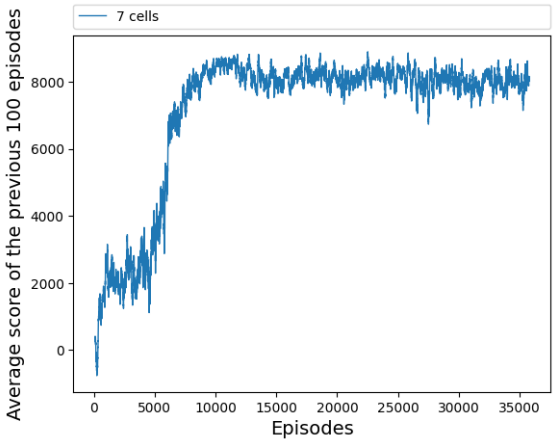
As shown in Figure 6.5, we considered three different configurations: (i) 4 neighboring cells; (ii) 7 neighboring cells; (iii) 14 neighboring cells. For each configuration, we trained 4, 7, and 14 agents, respectively. We trained the MADRP agents using 20000, 35000, and 35000 independent episodes for configuration (i), (ii) and (iii), respectively. We set the maximum number of steps in each episode T to 200. We set the penalty $\mathcal{M}$ to -100, and the number of slots $\mathcal{T}_\delta^c$ to 40. We considered three successive observations (i.e., $\mathcal{K} = 3$). We used a Poisson distribution to generate the traffic in each cell for the UL and DL. The arrival rates for UL and DL are randomly chosen at each iteration $\lambda \in \{50, 100, 200, 300\}$ with unit u per δ. We assumed a fixed serving rate μ for each cell (i.e., the amount of data scheduled at each slot equals 14 times the unit u; 14 being the number of symbols per slot in 5G NR). Figures 6.5(a), 6.5(b), and 6.5(c) depict the evolution of the averaged 100 sum rewards of all the agents over time. We observe that MADRP converges at 5000, 15000, and 30000 episodes for configuration (i), (ii), and (iii), respectively. Thus, we observe that the more agents we add, the more episodes the agents need to converge.

MADRP Inference mode

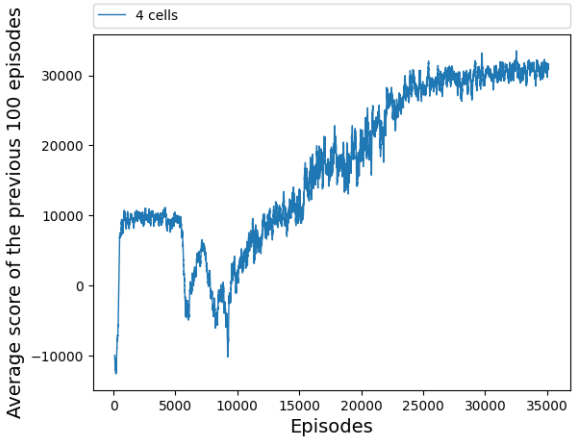
For the three configurations and under different interference probabilities, we have evaluated the MADRP agents in terms of i) The percentage of the fullness of the UL and DL buffers; ii) The number of conflicts successfully mitigated. Let p be the probability that agent i has a direct cross-link interference with agent j (i.e., cell i has edges users with cell j). We recall that indirect cross-link interference can happen by transitivity even if there are no edge users between two cells. The terms dTDD and sTDD in the figures' legend stand for dynamic TDD and static TDD, respectively. It should be noted that we considered different static TDD configurations:



(a) 4 cells



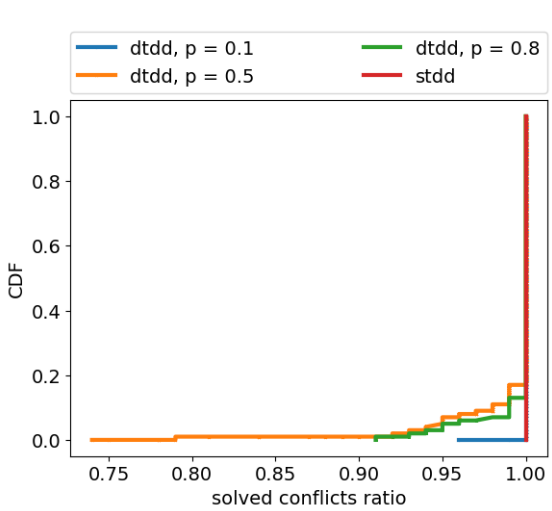
(b) 7 cells



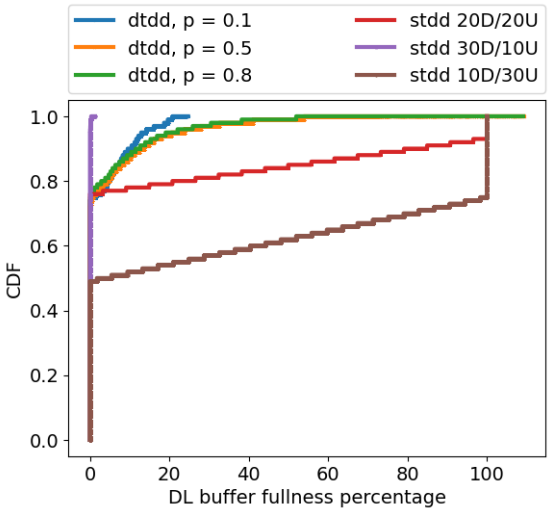
(c) 14 cells

Figure 6.5: Convergence evaluation of MADRP agent during the training mode

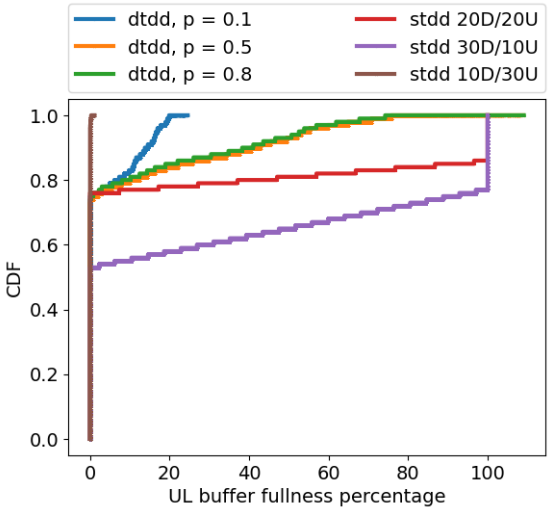
(*i*) stdd 20D/20U: half of the TDD period slots (i.e., 20 slots) are dedicated to DL, while the other half are dedicated to UL; (*ii*) stdd 30D/10U: 75% of the slots are dedicated to DL (*iii*) stdd 10D/30U: 75% of the slots are dedicated to UL.
In Figure 6.6, we evaluated the performance of 4 agents during 1000 independent episodes, each of which with 200 iterations. We considered three different interference probabilities p. Figure 6.6(a) depicts the Cumulative Distribution Function (CDF) of the ratio of solved interference conflicts among all the interference cases. A solved interference conflict means two agents aligned their TDD pattern while there is direct or indirect interference between them. The x-axis represents the solved conflicts ratio, while y-axis represents its CDF. For $p = 0.1$ (i.e., agent i' cell has a probability of 0.1 to have an edge user with agent j' cell), we notice that MADRP is able to solve more than 96% of the conflicts. While for $p = 0.5$ and $p = 0.8$, MADRP is able to solve more than 80% and 92% of the conflicts, respectively. We noticed that with $p = 0.8$, there is always interference (i.e., direct + indirect interference). While in static TDD, no interference is present since all the cells share the same TDD pattern. For instance, we consider all the conflicts are solved since there is no unsolved conflict. We can see that MADRP outperforms all static TDD solutions. For the 20D/20D configuration, MADRP is better in UL and DL, while for the 30D/10U configuration, MADRP is better in UL, and for the 10D/30U configuration, MADRP



(a) Conflicts mitigation



(b) DL buffer

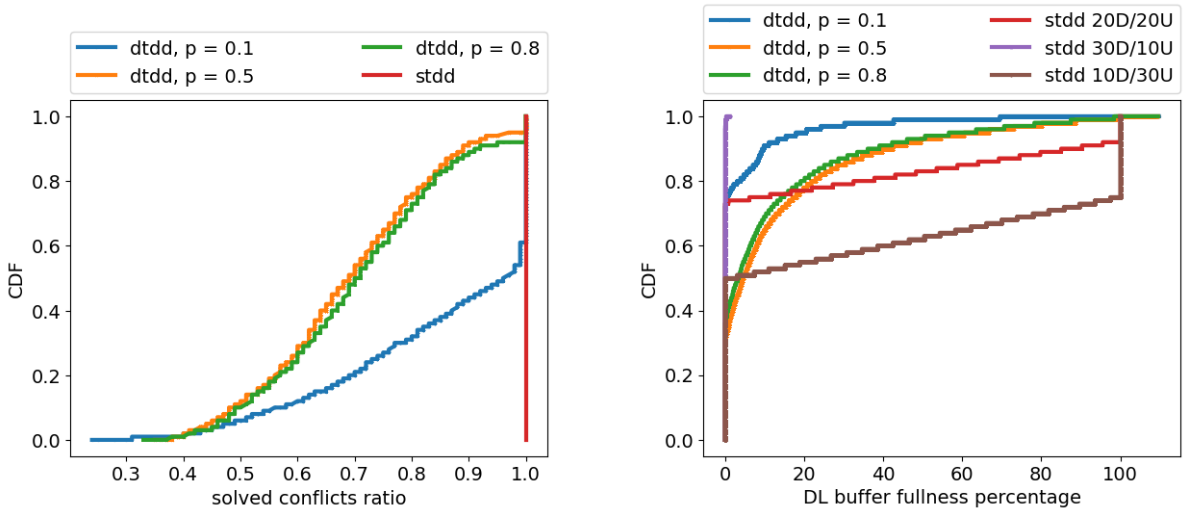


(c) UL buffer

Figure 6.6: Performance evaluation of MADRP of 4 agents during the inference mode

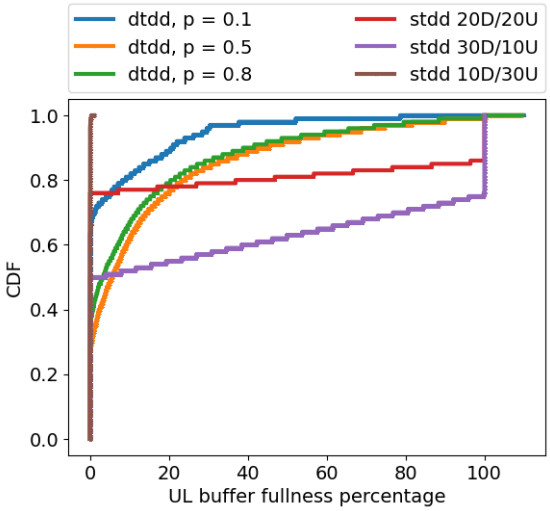
is better in DL. Indeed, over 60% of the samples represent a buffer overflow for 30D/10U and 10D/30U static TDD in DL and UL, respectively. As a result, MADRP is able to balance DL and UL traffic dynamically.

In Figure 6.7, we evaluated the performance of 7 neighboring agents during 1000 independent episodes, each of which with 200 iterations. Figure 6.7(a) depicts the CDF of the ratio of solved interference conflicts among all the interference cases. For $p = 0.1$, MADRP solved more than 90% of the conflicts during 60% of the samples and more than 70% of the conflicts among the 80% of the samples. While for $p = 0.5$ and $p = 0.8$, we notice that MADRP solves more than 70% of the conflicts among 50% of the samples. Figures 6.7(c) and 6.7(b) depict the CDF of the UL and the DL buffers' fullness percentage, respectively. For $p = 0.1$, the buffer size is lower than 10% in 80% of the samples and higher than 20% in less than 2% of the samples. While for $p = 0.5$ and $p = 0.8$, the buffer size is lower than 20% in 70% of the samples and lower than 40% in 80% of the samples. In all cases, MADRP outperforms all static TDD solutions. In the



(a) Conflicts mitigation

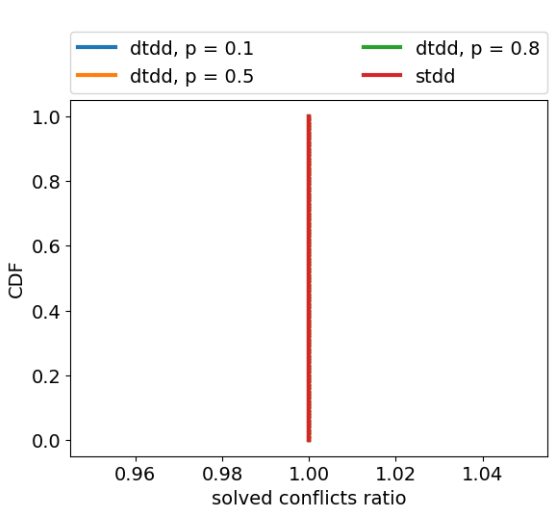
(b) DL buffer



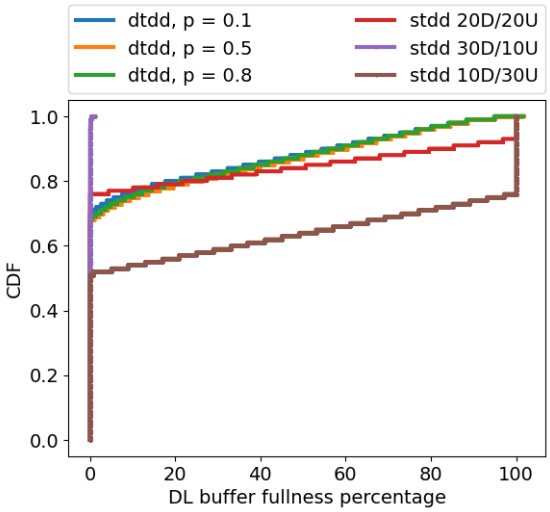
(c) UL buffer

Figure 6.7: Performance evaluation of MADRP of 7 agents during the inference mode

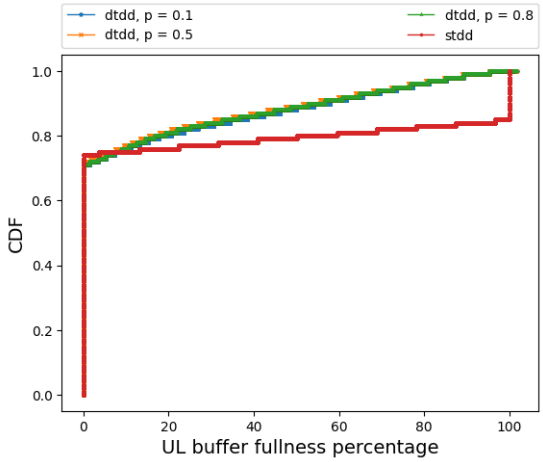
case of static TDD 20D/20U, we observe that around 20% of samples represent a buffer overflow (i.e., the buffer is full) in both UL and DL. In contrast, in the 30D/10U and 10D/30U static TDD solutions, around 70% of samples represent buffer overflow in UL and DL, respectively.
In Figure 6.8, we evaluated the performance of 14 neighboring agents during 1000 independent episodes, each of which with 200 iterations. Figure 6.8(a) depicts the CDF of the ratio of solved interference conflicts among all the interference cases. We observe that all the conflicts are solved for all the conflict rates. This is because MADRP found that the configuration (50% DL slots, 50% UL slots) is the best due to the strong dynamics of the environment. Since the configuration is fixed among all the agents, all the interference cases are mitigated. Figures 6.8(c) and 6.8(b) depict the CDF of the UL and the DL buffer's fullness percentage, respectively. We observe that the buffer size is lower than 40% in 80% of the samples and lower than 80% in 90% of the samples. In all the cases, MADRP outperforms all the static TDD solutions. We observe that, in static TDD 20D/20U, around 20% of the samples represents buffer overflow (i.e., the buffer



(a) Conflicts mitigation



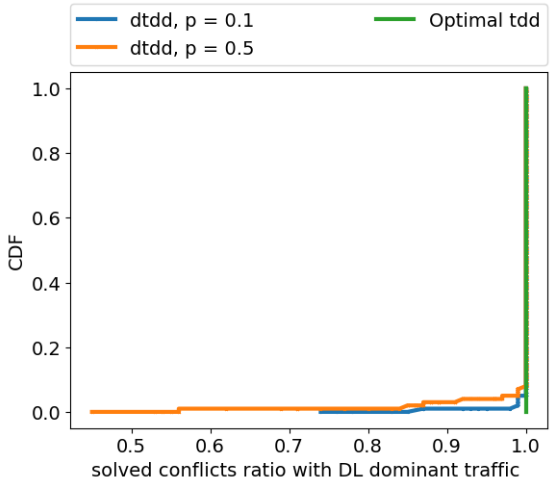
(b) DL buffer



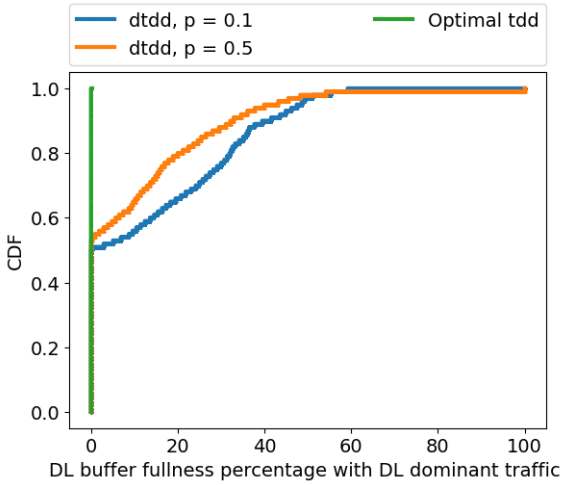
(c) UL buffer

Figure 6.8: Performance evaluation of MADRP of 14 agents during the inference mode

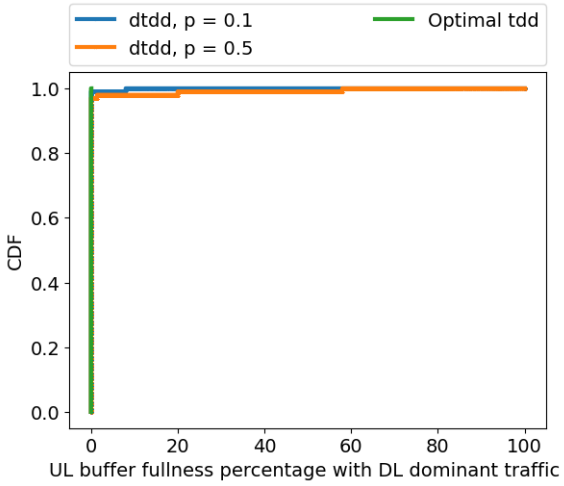
is full) in both UL and DL. While in static TDD 30D/10U and 10D/30U, around 70% of the samples represents buffer overflow in UL and DL, respectively.
Figures 6.9 and 6.11 illustrate the performance evaluation of MADRP when traffic is DL-dominant. That is, traffic arrives with higher arrival rates in the DL direction, while traffic arrives with lower arrival rates in the UL direction. We compared the MADRP solution with the optimization model solution.
In Figure 6.9(b), we see that the optimal solution empties the buffer continuously, whereas the MADRP solution keeps the buffer below 60% in 99% of the time. The gap between the optimal solution and the MADRP solution is 1%, representing the percentage of cases where the DL buffer was full. We recall that the aim of MADRP is to avoid buffer overflow without the knowledge of the arriving traffic model.
In Figure 6.11(b), we see that the optimal solution empties the buffer permanently, whereas the MADRP solution keeps the buffer below 20% in 90% of cases when the probability of interference is low, while the buffer overflows in 60% of cases when the probability of interference is high.



(a) Conflicts mitigation



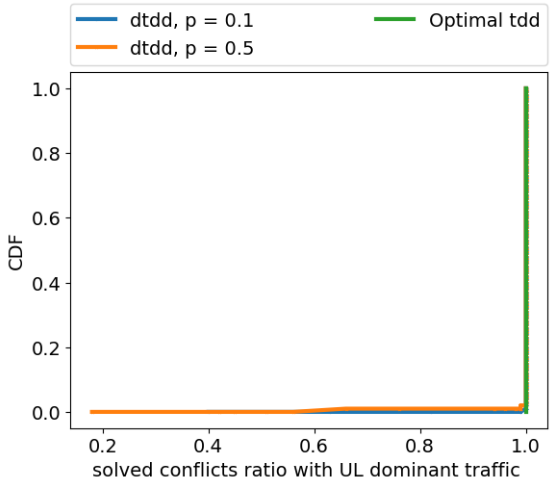
(b) DL buffer



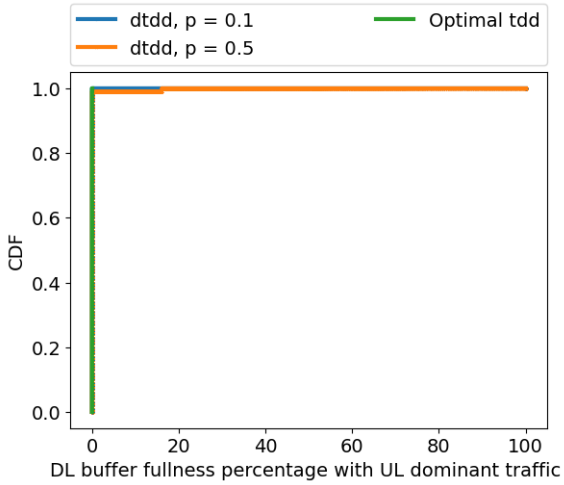
(c) UL buffer

Figure 6.9: Performance evaluation of MADRP of 4 agents during the inference mode with DL dominant traffic

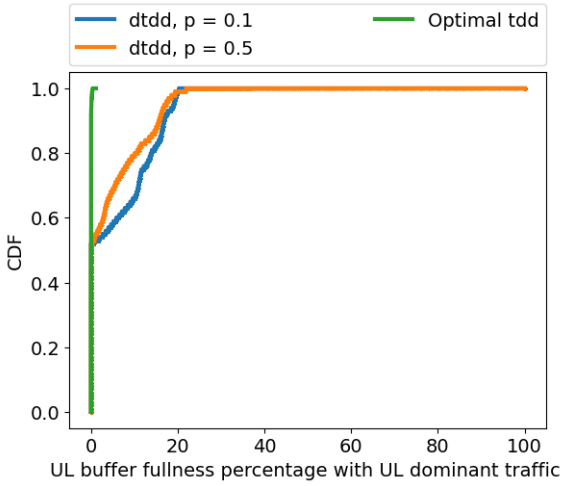
Figures 6.10 and 6.12 depict the performance evaluation of MADRP when the traffic is UL-dominant. We have compared the MADRP solution with the optimization model solution.
In Figure 6.10(c), we note that the optimal solution always empties the buffer, whereas the MADRP solution keeps the buffer below 20% in 99% of the time. In figure 6.12(c), the MADRP solution keeps the buffer empty in 40% of samples and below 40% in 80% of the time in the UL buffers when the probability of interference is low. We note that the buffer overflows in 60% of cases when the probability of interference is high.
Figures 6.13 and 6.14 depict the impact of arrival rates on the MADRP agents for configurations with 4 and 7 cells, respectively. Each plotted point represents the average of 1000 episodes, each of which with 200 steps. The first observation we can draw from this figure is the amount and the variance of the traffic handled by the MADRP agents. Figures 6.13(a) and 6.14(a) depict the average solved interference ratio variation over the arrival rates for different interference probabilities. The x-axis represents the arrival rate of the traffic, while the y-axis represents the average solved conflicts. We notice that while λ increases, the average ratio of solved conflicts



(a) Conflicts mitigation



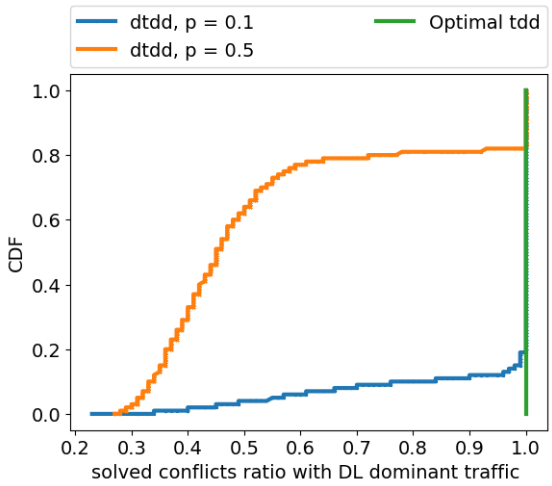
(b) DL buffer



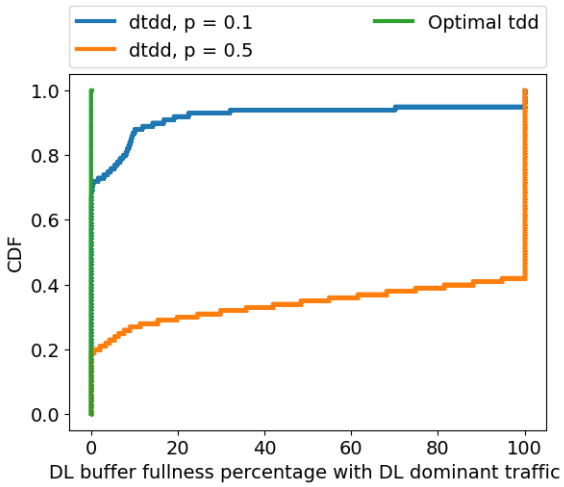
(c) UL buffer

Figure 6.10: Performance evaluation of MADRP of 4 agents during the inference mode with UL dominant traffic

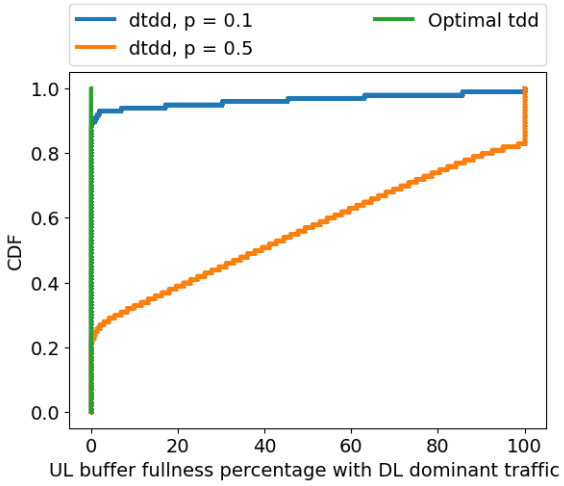
decreases. We recall that the serving cell capacity is $\mathcal{T}_{\delta}^{c} \times \mu$ (i.e., 560). Hence, we notice that when $\lambda \geq \mu$, MADRP is able to solve around 30% and 60% of the interference. When $\frac{\mu}{2} \geq \lambda \leq \mu$ in the 7 agents scenario, MADRP is able to solve between 60% and 95% for $p = 0.1$ and between 50% and 60% of the interference for $p = 0.5$. While in the 4 agents scenario, MADRP is able to solve all the interferences. We conclude that when increasing λ, the traffic variance increases (i.e., due to Poisson distribution), which introduces more aggressive conflicts in the agents' observations. This will results in difficulties in aligning the TDD pattern between interfered neighbors while serving each cell's traffic.
Figures 6.13(b) and 6.14(b) depict the average DL buffer fullness percentage in the 4 cells and 7 cells environment, respectively. While figures 6.13(c) and 6.14(c) depict the average UL buffer fullness percentage in the 4 cells and 7 cells environment, respectively. We notice that while λ increases, the average buffer fullness ratio increases. When λ reaches 800 units, the average buffer size is 70% for $p = 0.1$ and 85% when $p = 0.5$. We conclude that even when the arrival rate is bigger than the serving rate, MADRP is able to balance the traffic between UL and DL.



(a) Conflicts mitigation



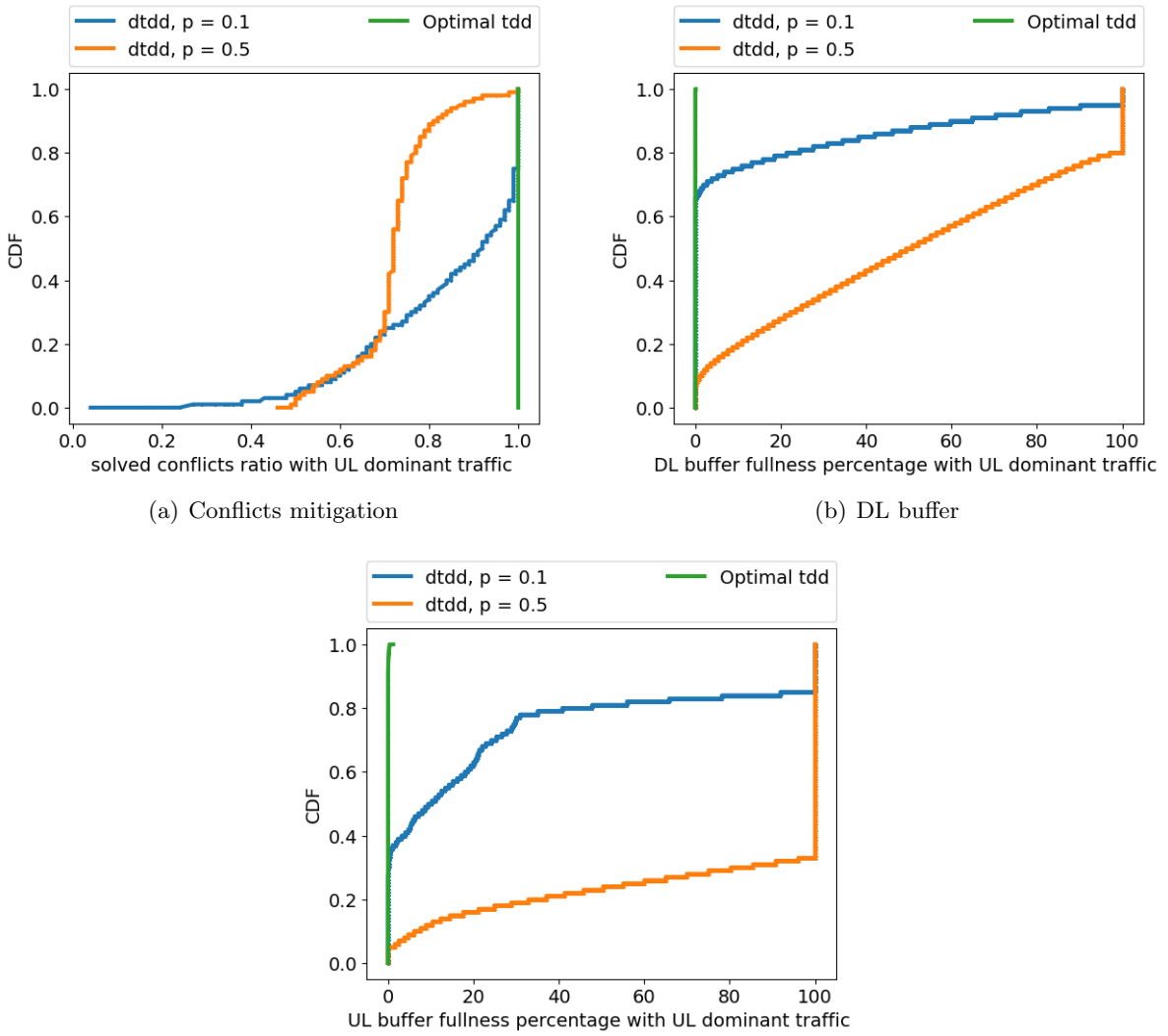
(b) DL buffer



(c) UL buffer

Figure 6.11: Performance evaluation of MADRP of 7 agents during the inference mode with DL dominant traffic

Overall, MADRP is able to change the TDD pattern dynamically by allocating UL and DL slots to different cells while avoiding direct and indirect cross-link interference between cells. When there is a small number of neighbors (i.e., 4 cells), small direct interference probability (i.e., $p = 0.1$), and an arrival rate lower than a serving rate with a traffic variance lower than half of the arrival rate, MADRP is able to avoid more than 96% of the interference cases while serving different UL and DL traffic rates. Indeed, MADRP allocates the number of UL/DL slots needed for each cell in each iteration according to the UL/DL buffer size, and hence it avoids buffer overflow while avoiding cross-link interference. However, when we increase the number of cells, the probability of direct interference, or the variance of the traffic, MADRP starts finding difficulties in satisfying the traffic load in UL/DL while solving the interference issues. In all the cases, MADRP outperforms the static TDD solution by reducing the probability of buffer overflow in both UL and DL.
In conclusion, we propose to create small subsets of neighboring cells (e.g., 7 cells per subset) that use the same frequency band in order to make dynamic TDD more efficient.



(a) Conflicts mitigation

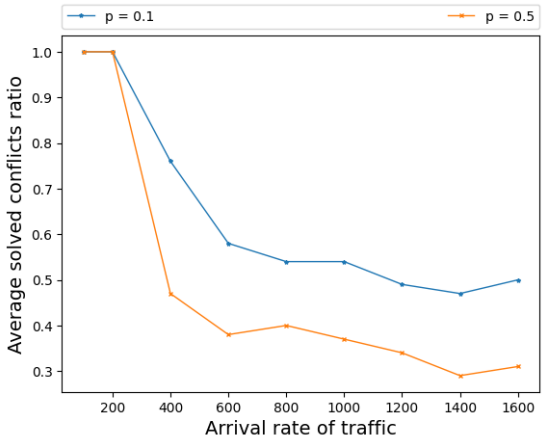
(b) DL buffer

(c) UL buffer

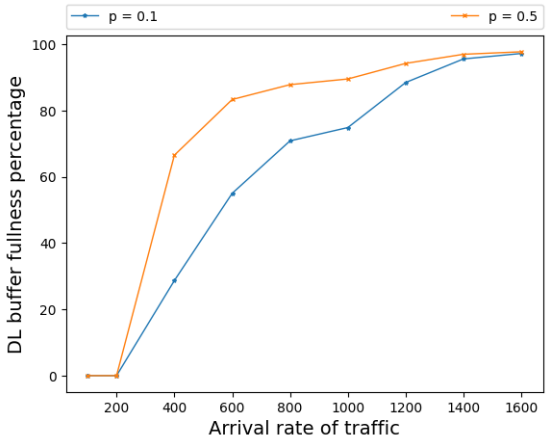
Figure 6.12: Performance evaluation of MADRP of 7 agents during the inference mode with UL dominant traffic

6.7 Conclusion

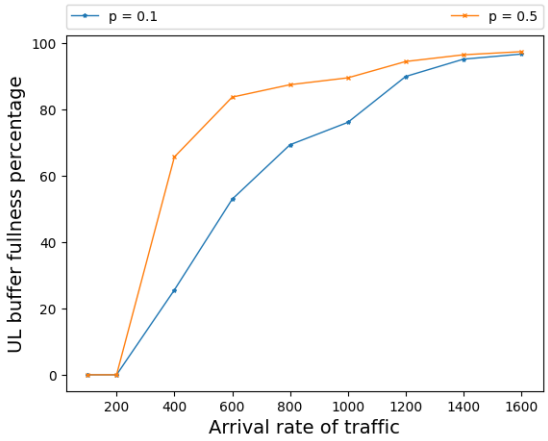
In this chapter, we introduced MADRP, a MADRL-based solution that permits deriving and adjusting the TDD pattern in 5G NR while mitigating cross-link interference. MADRP approach consists in deploying a MADRP agent at each gNB serving a cell. Without prior knowledge of the UEs traffic model, each MADRP agent computes the number of slots dedicated to UL and DL in a TDD frame aiming at reducing both UL and DL buffers while avoiding cross-link interference with the neighboring cells. Simulation results clearly showed that MADRP could avoid buffer overflow and dynamically adapt to the cell traffic while avoiding cross-link interference. Further, the results showed that MADRP clearly outperforms the different static TDD configurations with different UL/DL proportions and the gap between the optimal solution and the MADRP solution is small in low-interference scenarios



(a) Conflicts mitigation

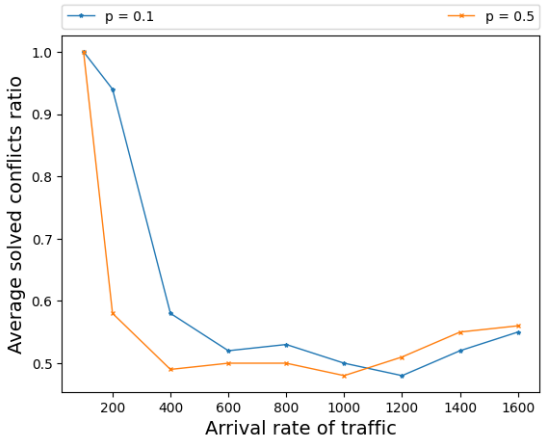


(b) DL buffer

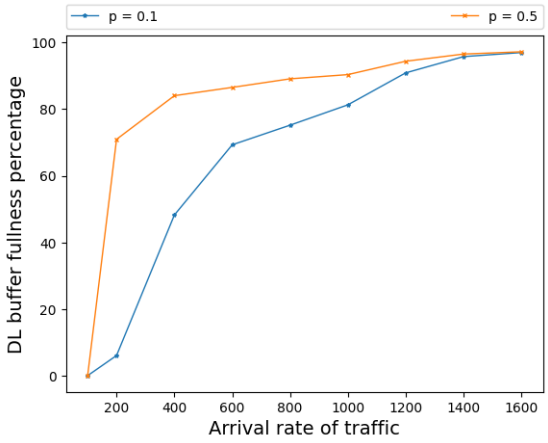


(c) UL buffer

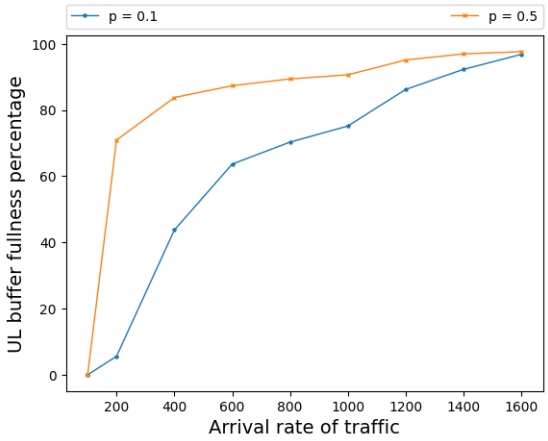
Figure 6.13: Performance evaluation of MADRP of 4 agents during the inference mode



(a) Conflicts mitigation



(b) DL buffer



(c) UL buffer

Figure 6.14: Performance evaluation of MADRP of 7 agents during the inference mode

Part III

RAN optimizations for uRLLC

Chapter 7

Radio Link Failure prediction in 5G NR

7.1 Introduction

In this chapter, we introduce a novel framework that predicts future RLFs using ML techniques and pushes notifications to third-party applications whenever radio connectivity between the targeted UE and the RAN might be lost. The framework is designed to work in micro-services environment, any third-party component can register to the framework in order to listen to radio connectivity loss events for chosen UEs. These third-party apps can be a RIC that triggers the handover process at the base station level, or a UAV controller application that modifies the drone's trajectory when our framework sends the predicted radio connectivity loss notification. The key contributions of this chapter are:

- We propose a ML model based on LSTM to predict the future trend of radio channel measurements, taking into account the evolution of past measurements.
- We propose a novel method based on SVM to classify the connectivity status (connectivity/ no connectivity) of UEs according to radio channel measurements and hence predict RLF.
- We trained and tested the framework on a real dataset using a 5G testbed based on OAI platform [52].
- We present two use cases where the RLF prediction can be used. The first one is handover enforcement, and the second one is UAV trajectory modification. These use cases are aligned with O-RAN and MEC architecture, respectively.

7.2 Related works

In [88], the authors used deep learning to reduce RLF when performing the handover process. They relied on the measured RSRP of the serving cell and the strongest neighbor cell to predict if the future handover (1-2 sec ahead) will succeed or fail. The work in [89] presented a clustering method to group cells with similar handovers' behavior to trigger early handovers without requesting the measurements of each UE. However, these works are based only on computer simulation, limiting the usability of the proposed solutions when they are deployed in real networks.

The authors in [90] proposed a deep learning approach to predict RLF according to RSRP measurements. But, the authors considered only RLF that may occur when a UE switches between a 4G and a 5G cell; hence RLF is not used to optimize network operations.
Besides, these works are focused only on handover, while other 5G use-cases may need information about RLF. Indeed, we can mention the UAV control use case, as the UAV controller needs to be notified whenever the drone may lose connectivity. This will allow for updating the drone trajectory to avoid RLF. Another use case is the smart video streaming use case, where gNB may schedule more resources for UEs to anticipate RLF by buffering more data.

7.3 Proposed framework

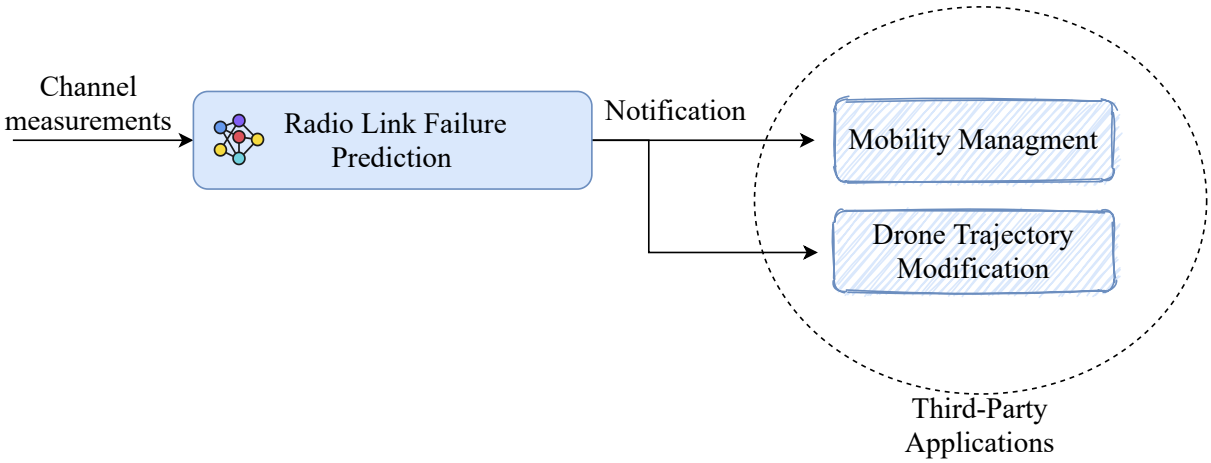


Figure 7.1: Framework overview

As indicated earlier, our main objective in this work is the early detection of radio connectivity loss and anticipation of RLF. To achieve this objective, we first conducted a campaign to measure RSRQ, RSRP, PHR, and CQI using a 5G testbed based on OAI. Second, we analyzed radio measurements and found a correlation between some measurements (RSRQ, PHR, and CQI) and connectivity status (i.e., whether there is connectivity or not). RSRP did not contribute explicitly to the correlation since it is implicitly used to compute the RSRQ metric (section 2.3.2). Nevertheless, applying this correlation directly to the reported measurements leads to deriving the connectivity status only when the measurements are collected. Therefore, there is a need to predict how radio measurements will evaluate in the near future. Based on this prediction, we used a classifier to find the correlation with the connectivity status (i.e., RLF). Once RLF is predicted, third-party components can consume this information (i.e., radio connectivity loss events) for targeted UEs aiming at applying policies to avoid RLF, such as:

- Mobility Management application that takes handover decisions before link failure.
- UAV Trajectory Modification application that updates the pre-flight drone trajectory whenever the radio connectivity might be lost.

In the following sections, we will start by describing the RLF prediction model (Figure 7.1). We will show how this model can be used to optimize handover and update UAVs' trajectories.

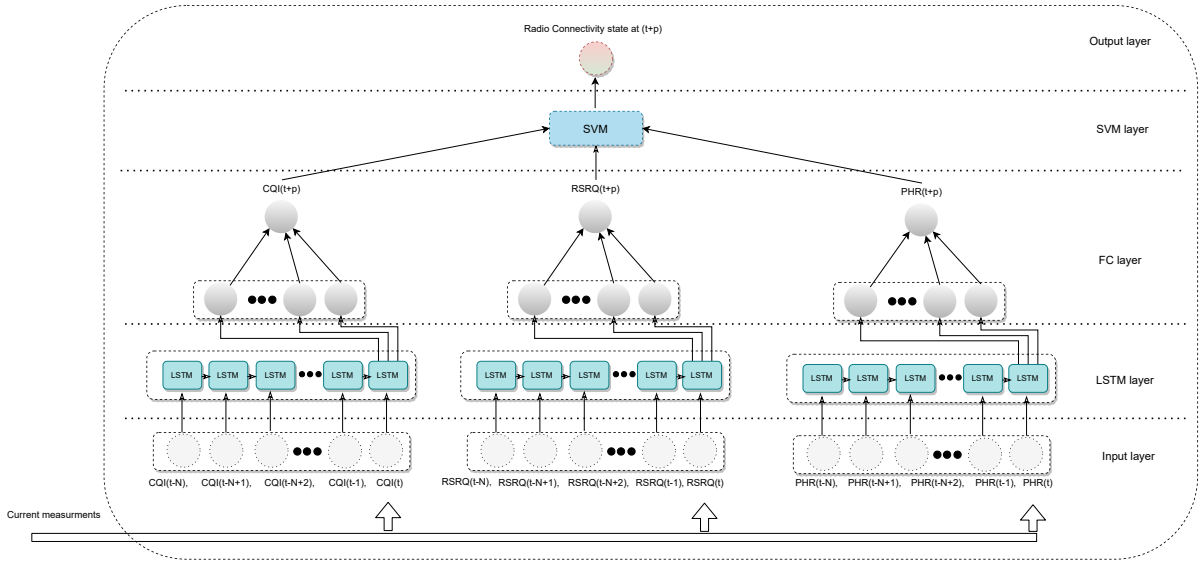


Figure 7.2: RLF Prediction Model design

7.4 Radio Link Failure Prediction model

7.4.1 Design

The RLF prediction model is structured into five stacked layers (Figure 7.2):

- **Input layer**: Composed of three-time windows; a window for each radio information (i.e., CQI, RSRQ, PHR). A time window is a vector that keeps the N last radio information collected from the RAN at the current time-step t.
- **LSTM layer**: LSTM is an improved version of RNN, designed to forecast time series data. The advantage of LSTM compared to RNN is that the former deals with vanishing and exploding gradient problems. In RNNs, the gradient problem becomes smaller due to long data sequences. Hence, the network parameter updates become insignificant, which means no real learning is achieved. Besides the existing hidden state of RNNs, LSTM introduces the cell state to make the gradient computation more stable (avoid exploding or vanishing).

  LSTM can learn the order dependence between items in a sequence and the context required to make predictions in time series forecasting problems.

  The key function of LSTM is its ability to use a local memory (i.e., learn when it needs to memorize or forget the information) to extract temporal information from time sequences. Then, it uses this information to predict the future value at time $t + p$, $p > 0$.
- **Fully Connected layer**: This layer will sum the vector from the output of the LSTM layer to produce one predicted value.
- **SVM layer**: This layer takes the predicted values of (CQI, RSRQ, PHR) and classifies them into two classes (connectivity/ no connectivity).
- **Output layer**: This layer represents the final connectivity state at time $t + p$.

7.4.2 Data Collection

Although many recent works used deep learning algorithms to predict the channel quality, they mainly rely on simulation to generate data for the learning step. However, it is well-accepted that simulation has limitations to reproduce the behavior of real networks. In our work, we trained the RLF prediction model using data collected from a testbed based on OAI. We collected (CQI, PHR, and RSRQ) values when both the connectivity is available and non-available (i.e., RLF). We labeled the data (i.e., the tuple CQI, PHR, and RSRQ) by 1 if there is connectivity, 0 otherwise.

7.4.3 Data Pre-processing

After collecting the data and creating the data set, we cleaned the latter by removing the static values as we were more interested in the data trend over time (up/down). This manipulation made the prediction more accurate. We also needed to format the data before feeding the model. The LSTM's input data is structured into a 3D vector (samples, time-steps, features), while the SVM's input data is structured into a 2D vector (samples, features). This step is mandatory to keep the integrity between different layers and improve the overall model accuracy.

7.4.4 Training process

During this step, we trained the LSTMs and the SVM models separately using the Keras framework [91]. We used the Adam optimizer [69] to have an adaptive learning rate. It is important to note that for this kind of forecasting model, two main problems arise. The first one is overfitting. It means that the model will learn too much about the particularities of the training data and will not be able to generalize the model to new data. In this case, the predicted signal matches exactly the real signal in the same data set, even if the latter is very complex. The second problem is that the model generates a signal that follows the real signal instead of predicting it. To overcome the first problem, we followed many techniques. First, we used two separate data sets, one for training and one for testing. Moreover, we used a Dropout layer after the LSTM hidden layer. In neural networks, some units may fix up the mistakes of other units leading to complex coordination, which, in turn, yields to overfitting as these coordinations do not generalize to unseen data.
Meanwhile, the Dropout layer randomly ignores neurons during a particular forward or backward pass. Hence, it will break up situations where network layers coordinate to correct mistakes from prior layers, making the model more robust and reducing the overfitting effect. We also used L2 regularization to push the model weights to 0 to have a less complex model and hence less over-fitting.
To overcome the second-mentioned problem, we used the Teacher Forcing [92] method. It is a method for training RNNs that uses the output from a previous time step as an input. When the RNN is trained, it can generate a sequence using the previous output as current input, contrary to the standard method that uses real data as an input during the training phase.

7.4.5 Examples of application using the RLF prediction model

In this section, we describe how the RLF prediction model can be used via two representative applications or services: Improving the handover using O-RAN architecture and optimizing UAVs trajectory based on ETSI MEC architecture.

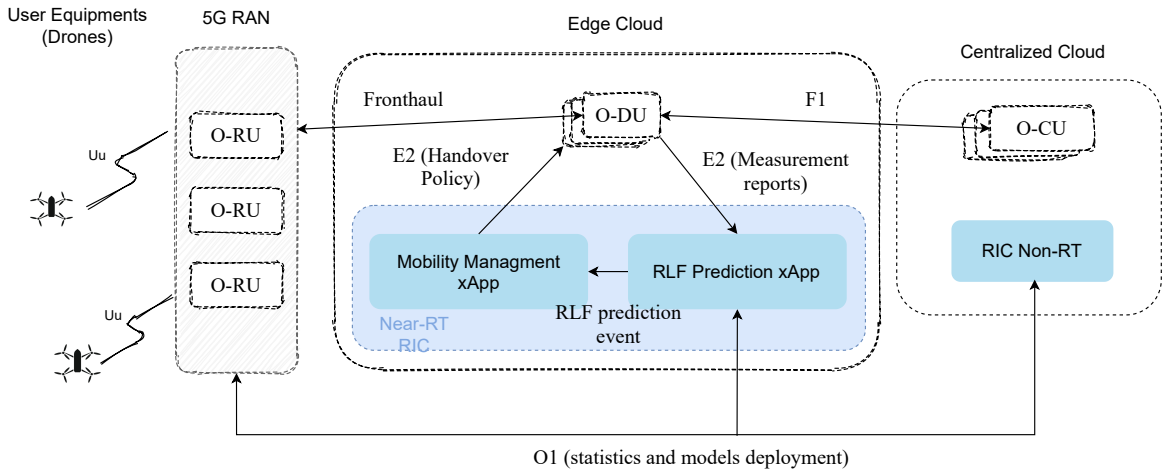


Figure 7.3: RLF Prediction in O-RAN scenario

RLF prediction as an O-RAN xApp According to O-RAN terminology, we assume that the RLF prediction modules and mobility management applications are run as xAPP. The mobility management xApp uses the input of the RLF xApp to optimize handover, which means, whenever an RLF is predicted, the network will trigger an early handover to avoid disconnection from the network.
The considered use-case is described in Figure 7.3. The RLF prediction is located at O-RAN Near Near-RT RIC. It fetches the latest radio information (CQI, RSRQ, and PHR) available for a specific/set of UEs from the E2 interface. It keeps a history of past information organized as a time window for each UE and predicts connectivity loss. On the other hand, the Mobility Management service is located at O-RAN Near-RT as a xApp. It consumes RLF prediction events from the RLF prediction xApp to trigger the handover procedure whenever the connectivity is lost. It should be noted that in O-RAN, RIC Non-RT is responsible for training ML models and pushing them to RIC Near-RT modules. Therefore, we assume that RLF prediction xApp receives the trained models from the RIC Non-RT and updates the local models significantly when the inference error increases.

RLF prediction as a MEC application In this use case scenario, we assume that the RLF prediction module is deployed as a MEC service available to other MEC applications. In this scenario, the MEC application is a UAV Trajectory Modification Application. The latter is deployed at the edge as it has to send commands to update the trajectory of UAVs requiring low-latency communications. Figure 7.4 illustrates the relationship between all the components. The RLF prediction module collects the CQI, RSRQ, and PHR via the MEC Radio Network Information Service (RNIS) [93] API. This API is provided by the MEC platform through MP2 interface. To recall, the RNIS enables third-party applications to fetch radio measurements for a specific/set of UEs to accomplish data analytic or optimization jobs. The RLF prediction module uses the measurements from the RNIS and predicts whenever the connectivity will be lost for a set of UEs. If a connectivity loss is predicted, the RLF prediction application sends a notification to the UAV Trajectory modification application via the MP1 interface, informing the connectivity status of the near future of the concerned list of UEs. The UAV Trajectory Modification Application uses the prediction events from the RLF Prediction Application and modifies the UAV trajectory in order to avoid link failure.

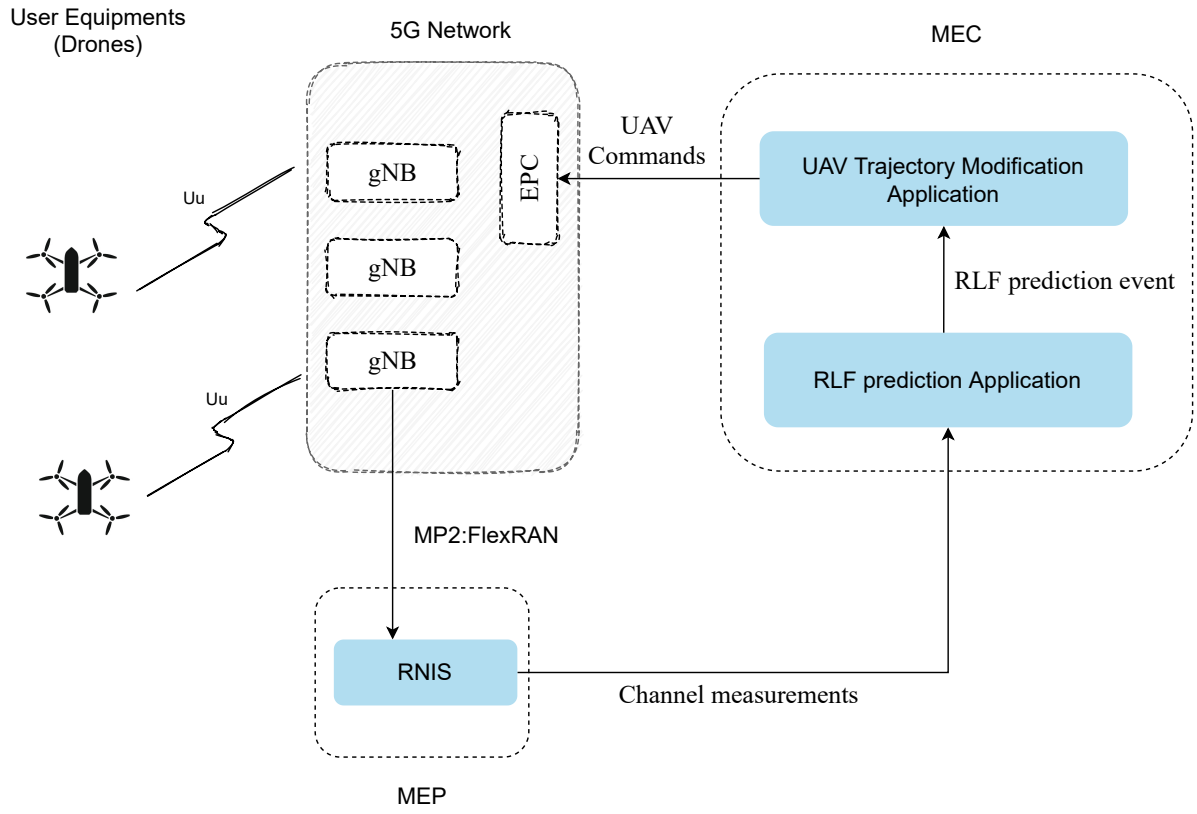


Figure 7.4: RLF Prediction in MEC scenario

7.5 Performance Evaluation

This section is divided into two parts. The first one is dedicated to the LSTM model evaluation of CQI, RSRQ, and PHR measurements, while the second part focuses on the classification model. We compared two algorithms: Neural Networks (NN) and SVM. We tested the classifier within two cases; the first is when the algorithm gets its input directly from the test data-set, while the second one gets its input from the LSTM models.
We used a different data-set for testing to avoid over-fitting and ensure that our results are not dependant on a particular data set. Two main metrics are used to evaluate the LSTM model:
Accuracy: the ratio of the number of correct predictions to the total number of input samples.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ prediction\ made}$$

Precision: the number of correct positive results divided by the number of positive results predicted by the classifier.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Recall: the number of correct positive results divided by the number of all relevant samples (i.e., all samples that should have been identified as positive).

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

F1 score: the balance between precision and recall.

$$F1\ Score = 2 * \frac{1}{\frac{1}{Recall} + \frac{1}{Precision}}$$

It tells how precise the classifier is (how many instances it classifies correctly), as well as how robust it is.

Root Mean Square Error (RMSE): the distance between the predictions and the real observations.

$$RMSE = \sqrt{\frac{1}{T} * \sum_{t=0}^{T-1} (real_t - predicted_t)^2}$$

LSTM models Evaluation We varied the LSTM parameters and tuned them to achieve the best performances: 1 LSTM hidden layer with 20 perceptrons, trained for 100 epochs with a batch size of 50 samples. We used a dropout layer with a rate of 0.2.

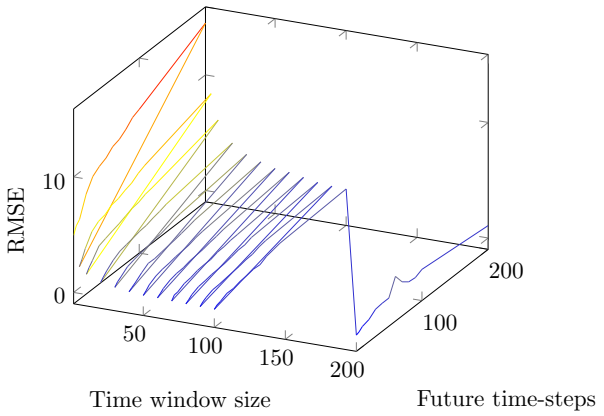


Figure 7.5: RMSE variation over past and future timesteps

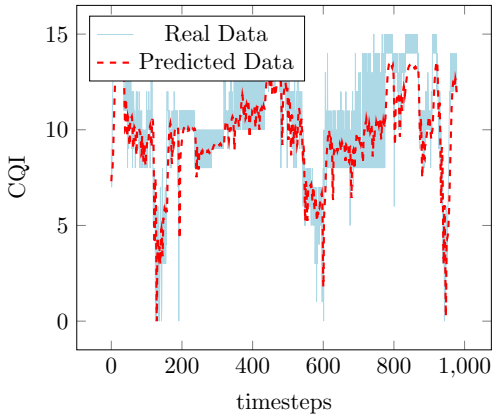


Figure 7.6: CQI prediction over time

Figure 7.5 illustrates the impact of changing the time window size (how many past samples the LSTM takes as input) and the future time-step (the prediction's time-step) on the RMSE. We remark that if the LSTM does not consider enough past samples, the error is high. The model is not able to predict the data. However, when more than 20 samples are used, we observe that the

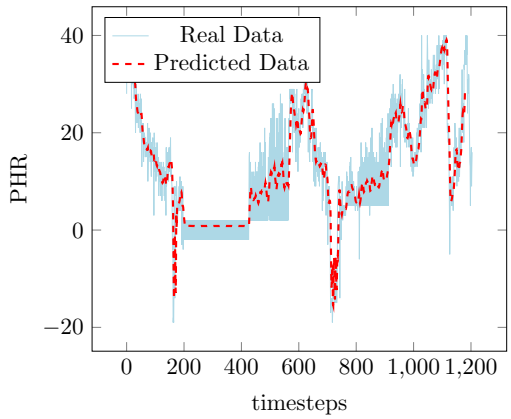


Figure 7.7: PHR prediction over time

error starts to converge to a smaller value, approximately 0.1. It should be noted that LSTM is able to forget information; then, even if we continue to increase the window size, the model keeps using only the needed number of past samples to decrease the prediction error. Besides, we noticed that when the future time-step increases, the error increases. This indicates that the model is not able to predict the measurements correctly for the long-term future.

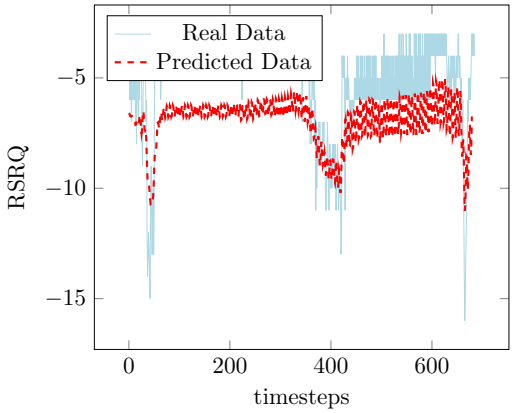


Figure 7.8: RSRQ prediction over time

Figures 7.6, 7.7, 7.8 illustrate, respectively, a part of the data-set plotted with the predicted values of CQI, PHR, and RSRQ. The test data-set is shifted in time to compare the real values and the predicted ones easier. We used 50 samples as input, and we predicted five-time steps in the future. The generated results show the accuracy of our LSTM model to estimate the measurement, i.e., CQI, PHR, and RSRQ.

Classification model Evaluation For the sake of comparison, we tested two algorithms: SVM

Table 7.1: Classifier Model metrics

metric	SVM (%)	NN (%)
Accuracy	98,44	99,25
Precision	97,76	96,14
Recall	96,56	99,59
F1 score	97,15	97,78

and a Neuron Network (NN) classifier. Table 7.1 compares the performance of the algorithms according to performance metrics. We notice that NN performs slightly better than SVM when

the real measurements are used as input directly from the dataset. However, when we consider the LSTM's prediction output as input, we can see that SVM is slightly better for the same metrics (Table 7.2).

Table 7.2: Overall Model metrics

metric	SVM (%)	NN (%)
Accuracy	98,03	97,95
Precision	91,42	91,21
Recall	97,81	97,53
F1 score	94,32	94,08

7.6 Conclusion

In this chapter, we addressed the challenge of predicting RLF in 5G networks. To predict RLF, we used a ML model that combines both LSTM and SVM using the radio measurement, composed of RSRQ, CQI, and PHR. To train the RLF prediction model, we generated a data set that matches the tuple RSRQ, CQI, and PHR with connectivity state (available or not) using a 5G testbed based on OAI. The RLF prediction model was able to predict 98,03 % of connectivity status using the collected data set. Besides, we presented two use-case scenarios that exploit the RLF prediction module to optimize UE mobility (i.e., Handover Optimization) and UAV trajectory. Both scenarios were aligned with two well-accepted architectures in 5G, O-RAN, and MEC.

Chapter 8

Uplink grant prediction to reduce uplink latency

8.1 Introduction

In this contribution, we introduce a novel DRL-based algorithm that reduces the UL latency by predicting the subsequent arrival of data and its size. We dubbed this solution DRL-based Low Latency Scheduler (DRL-LLS). The latter monitors the BSR sent by UEs and derives the future inter-arrival time and the grant size that should be allocated to the user in the next UL opportunity. DRL-LLS is adapted to change happening in channel conditions very quickly since it dynamically sends grants to the user before the data arrival. Hence, when the data is ready to be sent, the UE will find the grant and fully transmit all the data to gNB, which decreases the overhead signaling latency while adapting to the channel conditions.
The key contributions of this chapter are:

- We introduce a DRL-based framework (DRL-LLS) to reduce the UL latency of URLLC services.
- We introduce a DRL-based agent to predict the future UL slot for a UE by leveraging its traffic inter-packets arrival history.
- We introduce a DRL-based agent to compute the grant size for the next UL slot according to the BSR history.
- We combine the above models to enhance the UL MAC scheduler. The new methodology dramatically reduces the signaling overhead. For instance, the UL latency will decrease since the UE will not send a SR nor send BSR to get more grants for the same transmission.
- We validate DRL-LLS on different traffic models [94] used in 5G new use cases like XR and near real-time video streaming.

8.2 Related works

Authors of [95] have used deep learning to predict future traffic. They show that deep learning outperforms linear statistical learning when the number of past observations is significant enough to learn the traffic pattern. Meanwhile, authors in [96] have studied the impact of NR scheduling timings (mainly the k_2 parameter) on the UL traffic latency. They have proposed a heuristic

approach to derive k_2 value from the inter-packet arrival time considering only periodic traffic with fixed packet size. Authors in [97] have proposed an enhanced dynamic scheduling method to reduce VoIP traffic UL latency. However, they support only periodic traffic with fixed packet inter-arrival times. They have reduced the latency by removing the SR delay. However, the BSR delay was not reduced when the packet size is significant and can not be handled with the minimum grant allocated to the UE in the first grant. Authors in [98] have presented a novel framework for traffic prediction of IoT devices activated by binary Markovian events modeled by an On-Off Markov process with known transition probabilities. However, this work considered only ON-OFF traffic distribution and did not consider packet size. Authors of [99] leveraged Artificial Neural Network (ANN) to predict the ON-OFF period of burst traffic and estimate the bandwidth to be allocated to reduce the latency. However, they only considered IoT traffic following a burst pattern and over-estimated bandwidths compared to the real traffic. In fact, the bandwidth is estimated using the data rate of the network. Authors in [16] have suggested a predictive SPS scheme to predict data size during the SPS period. However, they considered only haptic data patterns, and the inter-packet arrival time is deemed static during the SPS period. Besides, the SR latency is not reduced. Authors of [100] developed an LSTM architecture to predict future traffic and minimize radio latency. However, they did not consider the size of traffic in their prediction.
All the above solutions considered a specific traffic pattern (either periodic, following Exponential distribution, or ON-OFF distribution), and most of them did not consider joint arrival prediction with data size prediction.

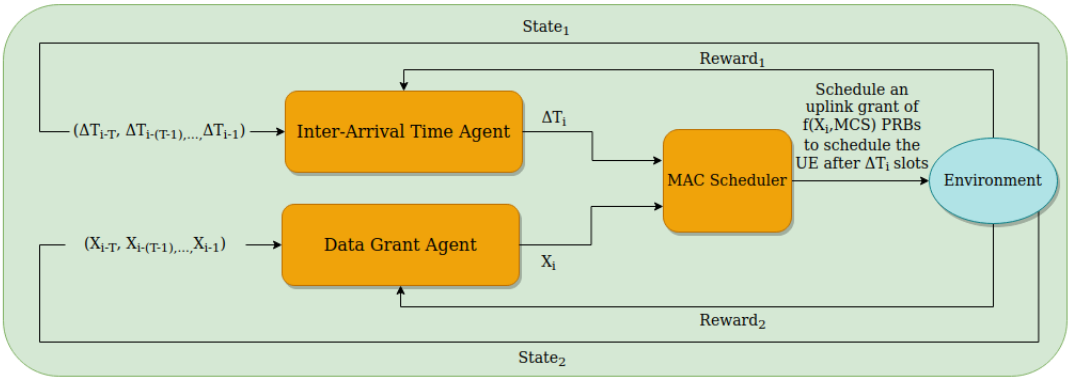


Figure 8.1: DRL-LLS design

8.3 DRL-LLS design

DRL-LLS provides a high dynamic scheduler that has the ability to avoid the SR and BSR latency and adapt according to channel conditions. To support URLLC, gNB may anticipate UL traffic and then dynamically send DCI grants (to avoid the SR latency) with the adapted MCS (to adapt to channel conditions) and with the right PRBs grant (to avoid the BSR latency). To achieve this, we propose to predict the future traffic arrival time ΔT_{i+1} and the future traffic size X_{i+1}. Once the gNB knows ΔT_{i+1}, it will send a DCI grant, with f(X_{i+1}, MCS) PRBs grant, k_2 slots before $t + \Delta T_i$, where t is last arrival time slot and $f(x, y)$ is the 3GPP compliant function [24] that returns the number of PRBs needed to transport x bytes under MCS y. The k_2 parameter is computed by $k_2 = t + \Delta T_i - t_c$, where t_c is the time of sending the DCI. Figure 8.2 illustrates the dynamic UL scheduling featuring low latency.

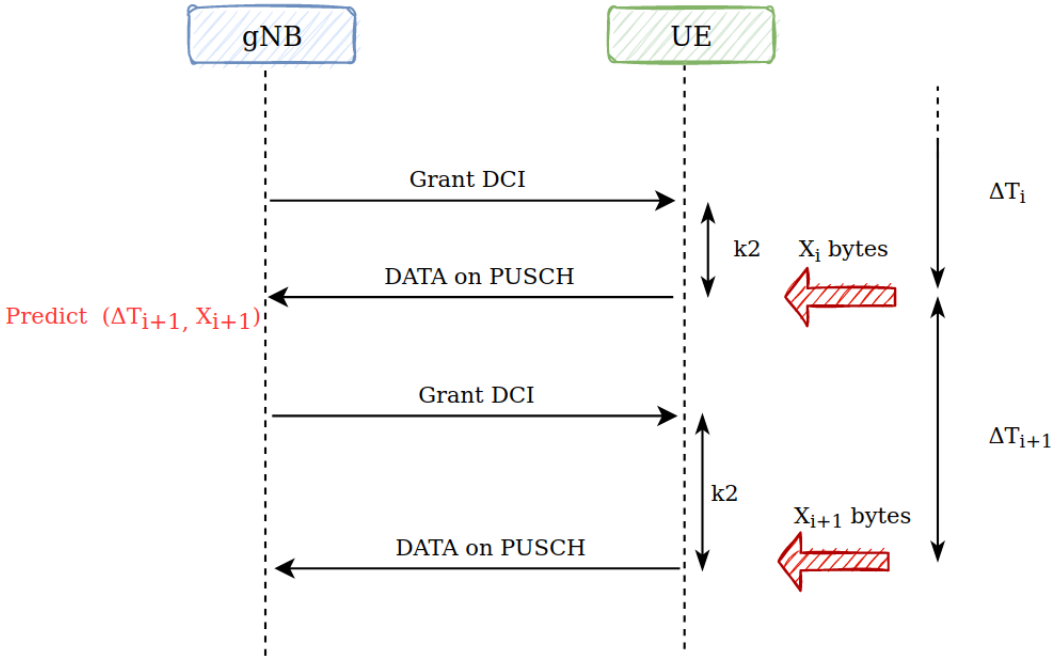


Figure 8.2: 3GPP dynamic UL scheduling procedure featuring low latency

DRL-LLS aims to reduce the UL latency for a set of UEs, whereby each UE can serve more than one service. DRL-LLS predicts the next UL slot for each UE u given its traffic history and schedules it before the actual data arrival. The environment is considered as a sliding window $W^i_{u,j}$ of size T containing information about the inter-arrival time intervals and BSR information history for each couple (UE u, service j) at the i^{th} data arrival. Formally:
$W^i_{u,j} = \{I^i_{u,j}, D^i_{u,j}\}$;
$I^i_{u,j} = (\Delta T_{i-T}, \Delta T_{i-(T-1)}, ..., \Delta T_i)$;
$D^i_{u,j} = (X_{i-T}, X_{i-(T-1)}, ..., X_i)$
where ΔT_i and X_i are the i^{th} inter-arrival time interval (in slot granularity) and BSR information (in bytes) of UE u, service j, respectively. We differentiate the BSR belonging to service j by the LC Identifier (LCID) available at the MAC header, considering that each service has a separate LC in the context of network slicing at the RAN [101]. For instance, by giving $W^i_{u,j}$ as input, DRL-LLS schedules an UL slot for UE u following service j traffic pattern. That gives the generalization ability to our agent and makes our DRL-LLS independent from the number of UEs and the number of services per UE.
We recall that the UL latency at the RAN is composed of two parts: the SR and the BSR latency (Figure 2.7). In order to reduce the SR latency, DRL-LLS predicts the next arrival of UE's u data. Whereas, to reduce the BSR latency, DRL-LLS predicts the size of the future data arrival of UE u. To achieve the two goals, DRL-LLS includes two agents: *i*) Inter-Arrival Time (IAT) Agent; *ii*) Data Grant (DG) Agent. Since traffic size and traffic inter-arrivals are independent [94], the two agents can be independent if their states and rewards are well designed. Figure 8.1 illustrates the architecture of DRL-LLS.
IAT Agent takes $(\Delta T_{i-T}, \Delta T_{i-(T-1)}, ..., \Delta T_i)$ as input and generates the next arrival interval ΔT_{i+1}.
DG Agent takes $(X_{i-T}, X_{i-(T-1)}, ..., X_i)$ as input, and then generates the next arrival size

X_{i+1}.
DRL-LLS combines the values of ΔT_{i+1} and X_{i+1}, and to schedule the future UL slot after $t + \Delta T_{i+1}$ slot with f(X_{i+1}, MCS) PRBs grant, where t is the current slot (last data arrival) and $f(x, y)$ is the 3GPP compliant function [24] that returns the number of PRBs needed to transport x bytes under MCS y.

8.3.1 IAT Agent

- **State:** The observation of IAT agent at time t as normalized value is $\frac{I^t_{u,j}}{M}$ for UE u, service j. M is the maximum time interval in the system in slots.
- **Actions:** The action is ΔT_{t+1}, which takes values from 1 to M.
- **Reward:** $r_1 = \frac{L_{max}-l}{L_{max}}$, where L_{max} is the maximum latency and l is the observed latency. In order to make the IAT agent independent from the DG agent, we ignored the data size in l computation. Formally;

$$l = \begin{cases} p - c + 1 + S_{max} * \Delta f & \text{if } p > c \\ (S_{max} - c + 1) - p + S_{max} * \Delta f, & \text{otherwise} \end{cases}$$

  Where p is the predicted slot in the TDD pattern, $p = (t + \Delta T_{i+1} \mod S_{max})$, c is the current slot, S_{max} is the number of slots in the frame and Δf is the difference between the current frame and the predicted frame $\Delta f = \mid \lceil \frac{(c+\Delta T_{i+1})}{S_{max}} \rceil - f' \mid$, f' is the frame offset of the actual data arrival.

  The reward is negative when the predicted slot is far from the actual data arrival. It increases when the latency approaches L_{max} and becomes positive when the latency is smaller than L_{max}.

  Reward and States are normalized since it is well-known that the activation functions in the neural network work well for small values, which positively impacts the model convergence.

8.3.2 DG Agent

- **State:** The observation of IAT agent at time t as normalized value is $\frac{D^t_{u,j}}{N}$ for UE u, service j. N is the maximum data size in the system in bytes.
- **Actions:** The action is X_{t+1}, which takes values from 1 to N. The unit is kbytes instead of bytes to reduce the action space.
- **Reward:** $r_2 = -\alpha * \mid \lceil \frac{X_{t+1}-d_r}{N} \rceil \mid + (1-\alpha) * l'$, whereby, d_r is the real $(t+1)^{th}$ data arrival size. In order to make the DG agent independent from the IAT agent, we ignored the time slot prediction in l' computation, Formally,

$$l' = \begin{cases} 0 & \text{if } X_{t+1} < d_r \\ 1 & \text{otherwise} \end{cases}$$

  The reward increases when the predicted size is slightly bigger than the actual size. It decreases when the predicted size is smaller than the actual size or the predicted size is much bigger than the actual size (over allocated PRBs). α controls the contribution of each term of the reward. For instance, it influences how much we tolerate the gap between actual and predicted sizes.

8.4 DRL-LLS detailed description

For both agents, DRL-LLS leverages the DQN algorithm [39], which is one of the most efficient DRL algorithms for continuous state space and discrete actions. DRL-LLS executes two steps: decision making and updating the Q-Networks. In DQN, two networks are used: a local Q-Network and a target Q-Network. The latter is the same as the local network except that its parameters are updated every τ^{-1} step. They are combined to help the convergence and stabilization of the learning.

8.4.1 Decision making

IAT agent observes a state $I^t_{u,j} \div M$ and feeds it to the local QNetwork to get the discrete action distribution of ΔT_{t+1}. While DG agent observes a state $D^t_{u,j} \div N$ and feeds it to the local QNetwork to get the discrete action distribution of X_{t+1}. Then, we apply an ϵ-greedy approach to choose an action from each distribution, which means IAT and DG agents will choose a random action over the possible actions with ϵ probability and the best action over the action distribution with a 1-ϵ probability. ϵ will decrease over time during the learning pushing the agent to explore the environment at the beginning of the training and driving it to exploitation over time.

8.4.2 Updating the Q-Networks

At each step, the current state, the action, the next state, and the reward are stored in a buffer known as the replay buffer. The local Q-Network is updated using a random sample from the replay buffer, which reduces the correlation between the agent's experiences and increases the stability of the learning. Using MSE and ADAM optimizer [69], the parameters of the local Q-Network are optimized at every step by considering the local and target values. In contrast, the parameters of the target Q-Network are updated every τ^{-1} step to stabilize the algorithm's convergence.

8.5 Performance Evaluation

In this section, we will introduce the simulation environment and parameters used for training DRL-LLS agents. Then, we will evaluate the trained agents in a 5G simulated environment using different 5G traffic models from [94].

Table 8.1: Traffic patterns parameters

Traffic model index	packet inter-arrival	packet size
0	Exponential distribution ($\lambda = 1/20$)	Exponential distribution ($\lambda = 1/10k$)
1	Exponential distribution ($\lambda = 1/5$)	Static (20k)
2	Static (10)	Static (10k bits)
3	On-Off distribution ($\lambda = 1/10$, $\beta_1 = 1/20$, $\beta_2 = 1/20$)	Static (15k bits)
4	On-Off distribution ($\lambda = 1/10$, $\beta_1 = 1/20$, $\beta_2 = 1/10$)	Exponential distribution ($\lambda = 1/20k$)
5	Truncated Pareto distribution ($\alpha = 1.1$, $m = 5$, $max = 30$)	Truncated Pareto distribution ($\alpha = 1.2$, $m = 10k$ bits, $max = 30k$ bits)
6	Truncated Pareto distribution ($\alpha = 1.2$, $m = 10$, $max = 40$)	Static (50k bits)

8.5.1 Simulation parameters and training phase

We have trained both IAT and DG agents using 5000 independent episodes. For each episode, a different traffic inter -arrival and data size patterns are used in a circular way from the list:

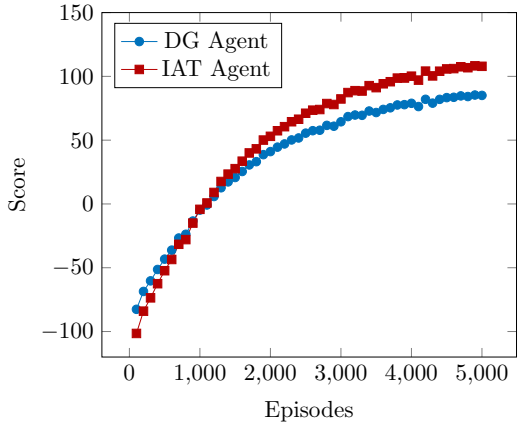


Figure 8.3: Convergence evaluation of DRL-LLS agents during the training mode

Exponential distribution with a random λ; $0 < \lambda < \frac{1}{M}$ for IAT agent and $0 < \lambda < \frac{1}{N}$ for DG agent, Static distribution with a random value m ($0 < m < M$ for IAT agent and $0 < m < N$ for DG agent), On-Off distribution (For IAT agent only, λ, β_1 and β_2 are the parameters of the exponential distributions for traffic inter-arrival during the ON period, the intervals of ON period and the intervals of OFF period, respectively) and the truncated Pareto distribution (random α, $1 < \alpha < 3$, $0 < m < M$, $m < max < M$ for IAT agent and $0 < m < N$, $m < max < N$ for DG agent). The value of M was fixed by 100 slots and the value of N by 10^3 kbytes. We have fixed the maximum number of steps at each episode by 200 and the window size T by 1000. Before starting the learning process, the window is filled by arrivals generated by the current episode distribution. The different considered parameters for both agents are presented in Table 8.2

Table 8.2: DRL-LLS parameters

Parameter	Value
α	0.2
L_{max}	4 slots (1 ms)
Number of hidden layers	3
Hidden layer size	64 nodes
discount factor γ	0
Batch size	128
Learning rate	$5 * 10^{-4}$
Replay buffer size	10^9
Soft update coefficient τ	0.001
Optimizer	ADAM [69]
ϵ-start	1
ϵ-decay	0.9991
ϵ-end	0.01
Number of training episodes	5000

To evaluate DRL-LLS in a 5G environment, we extended the 5G Simulator developed in [70] to support UL scheduling, and we fixed the numerology by 2 (SCS = 60 khz) which is available in both FR1 and FR2. We used a TDD period of 1.25 ms, which is the smallest TDD period for numerology 2. We used a random MCS for each episode in the range of 20..26, which

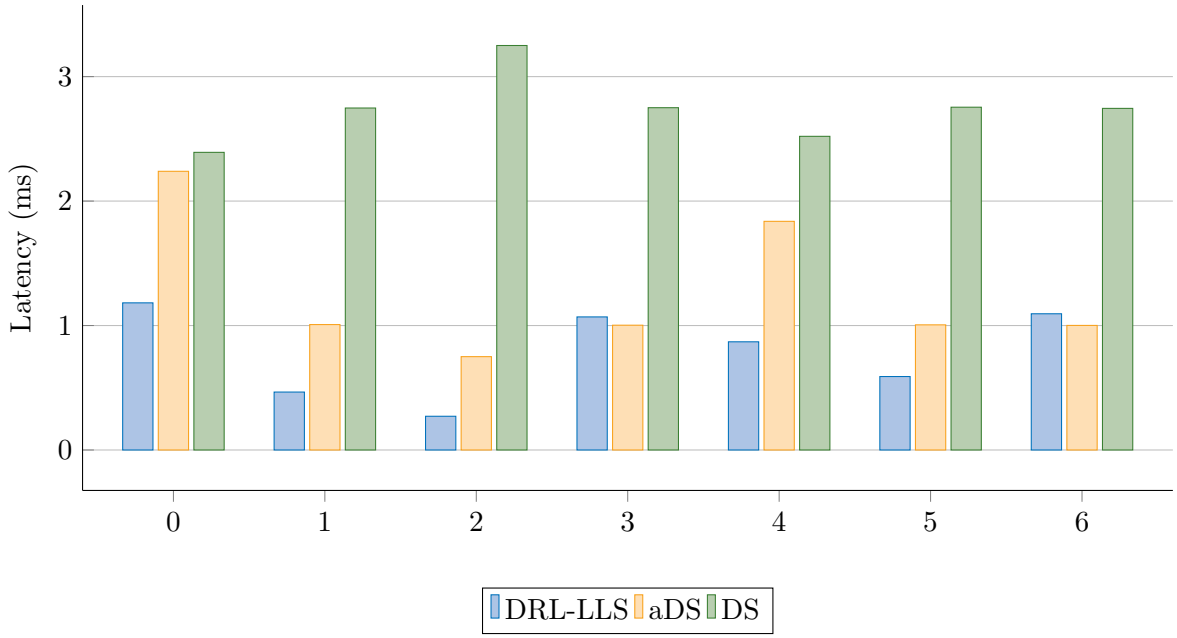


Figure 8.4: Average UL latency comparison between DRL-LLS, aDS and DS

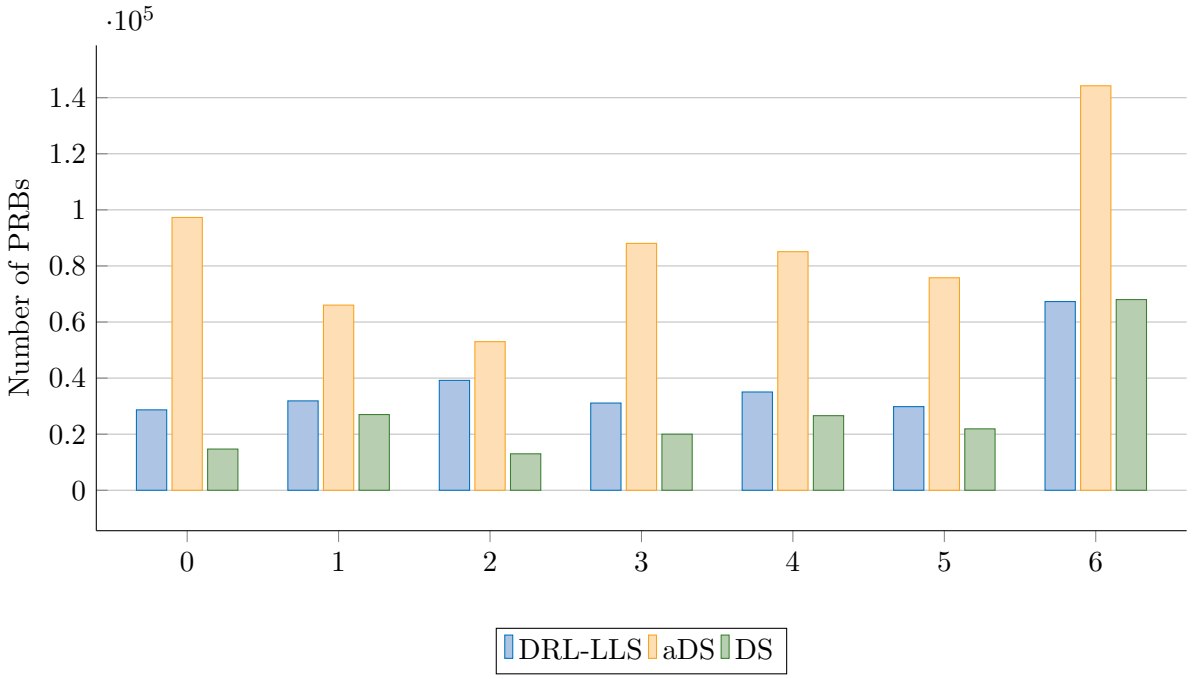


Figure 8.5: Number of the allocated PRBs comparison between DRL-LLS, aDS and DS

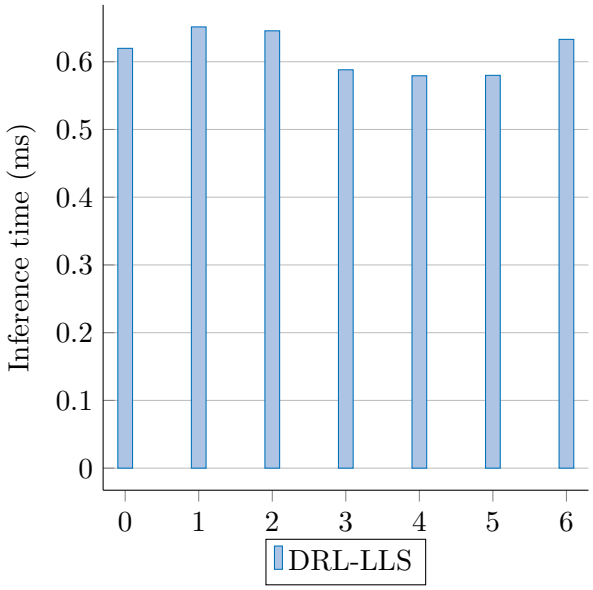


Figure 8.6: Average inference execution time of DRL-LLS

maps to medium or good channel condition. Since DRL-LLS is independent of the number of UEs and services, we focused on training and testing the framework on a single UE with different traffic patterns to show the ability of DRL-LLS to reduce the UL latency. We have implemented our simulation environment using Python and Pytorch library. We have used a machine with 32 CPUs, an Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz (2.7 GHz with Turbo Boost technology), and 128 GBs of RAM.
Figure 8.3 depicts the convergence evaluation of DRL-LLS score (sum of rewards during an episode) averaged every 100 episodes. We observe that the DRL-LLS agents converge after 4800 episodes since the curve tangents tend toward 0.

8.5.2 Inference phase

We have evaluated the DRL-LLS framework in terms of: *i*) The average UL latency between data chunk arrival and the transmission of the whole chunk; *ii*) The number of PRBs allocated during the test; *iii*) The average inference time of both IAT and DG agents executed sequentially (one after the other).
In Figure 8.4, The y-axis represents the UL latency (in ms), which is measured by $t_t^c - t_a^c$, where t_a^c is the arrival time of a data chunk c and t_a^c the transmission time of the last part of the data chunk c. The x-axis represents the traffic pattern index introduced in Table 8.1. Each pattern has a packet inter-arrival time and data size distribution. We used different distributions with different parameters to show that DRL-LLS is able to predict the pattern of different distributions with different parameters. We recall that these distributions correspond to 5G use case patterns [94]. We compared DRL-LLS with two methods: *a*) DS, where the UE sends an SR to gNB when it has data to transmit; *b*) always Dynamic Scheduling (aDS), where gNB schedules a minimal grant in all the UL slots to avoid the SR procedure. The minimum grant for both DS and aDS is 5 PRBs. We should note that the time unit in packet-inter arrival is slots. For instance, $\lambda = \frac{1}{20}$ for an Exponential distribution means that the mean of the inter-arrival intervals is 20 slots. We run the test over 1000 packet arrival for each pattern configuration. We observe that DRL-LLS offers a lower latency compared to aDS and DS. Indeed, DS latency includes both SR and BSR latency, while aDS only consists of the BSR latency. However, DRL-LLS removes both SR and

BSR latency since it predicts the subsequent arrival and the size of the next arrival. Hence, the UE sends the data directly without sending the SR and does not need a BSR to extend the transmission. The smallest achieved latency by DRL-LLS is 0.25 ms (1 slot in numerology 2) in the case of periodic traffic (traffic pattern index 2). This is due to the simplicity of the pattern, which is predictable ideally by DRL-LLS. The highest latency achieved by DRL-LLS is 1.1 ms, which makes the framework suitable for URLLC services with a latency requirement of 1 ms.
Figure 8.5 depicts the number of allocated PRBs (for UL) during the 1000 packet arrivals for each traffic pattern (Table 8.1). We notice that aDS consume much more PRBs compared to DRL-LLS and DS. This is since aDS schedules 5 PRBs in every UL slot even if there is no transmission. We should shed light that DS is optimal in terms of resource allocation since it allocates only the needed amount of PRBs to transmit the data. We also observe a smaller difference between DRL-LLS and DS in terms of resource allocation while achieving a much better latency by DRL-LLS (Figure 8.4).
Finally, Figure 8.6 shows the average execution time of DRL-LLS inference over the 1000 data arrival for each traffic pattern (Table 8.1). We notice that the execution time does not exceed 0.62 ms, which is suitable for real-time scheduling and hence, better adaptation with channel conditions and better latency.

8.6 Conclusion

This chapter introduced DRL-LLS, a DRL-based solution that reduces the UL latency in 5G NR. DRL-LLS will be leveraged by the 5G base station before the UE scheduling process to derive *i*) the next UL slot to schedule and *ii*) the number of PRBs to allocate to the user. Without knowing the UEs traffic model, DRL-LLS can learn the traffic inter-arrival and data size patterns. DRL-LLS uses the available BSR information history to derive the following UL grant. Simulation results clearly showed that DRL-LLS is able to reduce the UL latency down to 0.25 ms.

Chapter 9

Balance between latency and power consumption

9.1 Introduction

In this contribution, we introduce a DRL-based solution, called DRL-based Latency and Power optimizer (DRL-LP), to jointly derive the C-DRX parameters and the BWP configuration. The proposed solution is designed to be scalable in terms of the number of UEs and traffic patterns. Indeed, The DRL agent observes UEs' history of: (i) the experienced latency; (ii) the buffer status; and (iii) the number of scheduled UEs; during a time window. Then, for each UE, the DRL agent sets the C-DRX parameters and the BWP size for the next time window. To the best of our knowledge, no prior work has combined C-DRX and BWP adaptation to reduce the PC further while ensuring low latency.
The key contributions of this chapter are:

- We model the problem and the objective function to minimize the PC and the latency considering the C-DRX parameters and BWP adaptation.
- We introduce a DRL-based solution to jointly derive the C-DRX and BWP configuration and find a compromise between low PC and low latency. The C-DRX configuration includes new parameters introduced by 5G NR that help decrease the latency such as the C-DRX slot offset.
- The proposed DRL solution is designed to support multiple UEs, and different traffic patterns, i.e., the DRL agent is trained only once and then deployed regardless of the number of UEs and their traffic patterns.
- We evaluate the solution on periodic and aperiodic traffic for different numbers of UEs. The periodic traffic is characterized by random inter-arrival periods, and the aperiodic traffic follows the Poisson distribution with random parameters.

9.2 Related works

In [102], a logic controller-based C-DRX system was proposed to adaptively adjust the C-DRX parameters by learning the information of historical delay time and the packet arrival rate. However, the authors only considered the C-DRX cycle length parameter and did not consider multiple UE scenarios that can impact the latency. Authors of [103] proposed using

a DRL-based actor-critic algorithm to choose the C-DRX cycles based on traffic statistics and to use symmetric sampling to accelerate online learning. However, they only (i) considered C-DRX cycle length parameter; (ii) evaluated two scenarios (i.e. the traffic follows two Poisson distributions with fixed mean arrival rates). Authors of [104] presented a novel Contextual Bandit-based approach to optimize the C-DRX configuration of UEs in 5G NR. However, they did not consider the latency factor, which is strongly impacted by C-DRX. In [105], the authors leveraged channel capacity predictions to minimize the energy usage of UEs and create longer sleep opportunities while preventing video interruptions. However, authors only considered the C-DRX cycle length parameter without optimization for low-latency services. Authors of [106] designed a control policy to adjust the ON duration period parameter in order to satisfy XR requirements and minimize PC. Authors of [107] developed a model to evaluate the impact of the BWP adaptation on power gains. However, they did not consider multiple UE scenarios since the scheduling opportunities in a BWP are limited per slot.
To the best of our knowledge, no prior work has combined C-DRX and BWP adaptation to reduce PC further. The choice of this combination (i.e., combining C-DRX and BWP) is motivated by the possibility of reducing PC by using the same narrow BWP for more UEs by shifting C-DRX cycles (i.e., making the UEs ON duration period not cross) and thus reduce their PC. In addition, most of the work on C-DRX has considered only one parameter to be optimized to make the problem easy to solve.

9.3 Network Model and Problem formulation

We consider a network consisting of a set of UEs denoted $\mathcal{K}$ and sharing bandwidth of size $\mathcal{W}$. Let Δ, Φ, Γ and Ω be the sets of cycle length, ON periods, offsets and BWP sizes, respectively. Each UE $i \in \mathcal{K}$ may use a different C-DRX configuration and a different BWP during a set of $\mathcal{S}$ time slots. Let Δ_i^j, Φ_i^j, γ_i^j, and ω_i^j be Boolean decision variables that indicate whether the UE i uses a cycle length of $f(j)$, an ON period of $g(j)$ an offset of $h(j)$, and a BWP of size $l(j)$, respectively; $f(x)$, $g(x)$, $h(x)$, and $l(x)$ are functions that return the x^{th} element of Δ, Φ, Γ, Ω, respectively. Let $\mathcal{P}_{i,t}$ be a variable that measures the PC of UE i at time slot $t \in \mathcal{S}$. P^{max} is a constant that defines the maximum PC of a given UE in all time slots of $\mathcal{S}$. Let $\mathcal{L}_{i,t}$ a variable that measures the delay of the data arrived at slot t for UE i. If no data has arrived for UE i at slot t, then $\mathcal{L}_{i,t} = -1$. $\mathcal{L}_i^{max}$ is a constant that defines the maximum latency allowed for UE i. Note that the latency is the waiting time between data arrival and transmission. Hence, $\mathcal{L}_{i,t}$ is influenced by the C-DRX parameters (i.e., data arrives during the sleep period and thus latency increases), the BWP size (i.e., the BWP size does not allow all data to be sent in one transmission and thus latency increases), and the packet arrival slot.
We can formulate the objective function as follows:

$$\min \sum_{i=1}^{|\mathcal{K}|} \sum_{t=1}^{|\mathcal{S}|} \alpha \times \frac{\mathcal{L}_{i,t}}{\mathcal{L}_i^{max}} + (1-\alpha) \times \frac{\beta_i \times \mathcal{P}_{i,t}}{\mathcal{P}^{max}} \tag{9.1}$$

S.t.

$$\forall i \in \mathcal{K}, \forall t \in \mathcal{S} : \mathcal{L}_{i,t} \geq -1, \mathcal{P}_{i,t} \geq 0 \tag{9.2}$$

Where α is a given constant ($0 \leq \alpha \leq 1$) that defines the priority between latency and PC. If $\alpha = 1$, then we are only interested in optimizing latency, whereas, if $\alpha = 0$, then we are only interested in optimizing PC. According to the energy model introduced by the 3GPP specification in [20], the relationship between the PC and the BWP size is linear and the scaling factor β_i is given by equation 9.3.

$$\forall i \in \mathcal{K} : \beta_i = 0.4 + 0.6 \times \frac{\left(\sum_{j=1}^{|\Omega|} \omega_i^j \times l(j) - 20\right)}{80} \tag{9.3}$$

Equation 9.4 ensures that each UE i has only one C-DRX configuration and one BWP during $\mathcal{S}$.

$$\forall i \in \mathcal{K}, \sum_{j=1}^{|\Omega|} \omega_i^j \leq 1, \sum_{j=1}^{|\Delta|} \Delta_i^j \leq 1, \sum_{j=1}^{|\Phi|} \Phi_i^j \leq 1, \sum_{j=1}^{|\Gamma|} \gamma_i^j \leq 1 \tag{9.4}$$

Equation 9.5 ensures that the sum of the BWP sizes does not exceed the available bandwidth $\mathcal{W}$ when the UEs ON periods cross. Let $\mathcal{B}_{i,t}^j$ be a Boolean system variable that indicates whether UE i is active (i.e., during the ON period) and uses a BWP of size $l(j)$ at time slot t.

$$\forall t \in \mathcal{S}, \sum_{i=1}^{|\mathcal{K}|} \sum_{j=1}^{|\Omega|} \mathcal{B}_{i,t}^j \times l(j) \leq \mathcal{W} \tag{9.5}$$

Each BWP has a limited amount of scheduling opportunities (i.e. DCIs) per slot. Let $\mathcal{K}_j^{max}$ a constant that defines the maximum number of UEs allowed per slot for the BWP j. Equation 9.6 ensures that for each BWP, the number of UEs scheduled per slot does not exceed the maximum number of DCIs per slot, in each BWP.

$$\forall t \in \mathcal{S}, \forall j \in \Omega \sum_{i=1}^{|\mathcal{K}|} \mathcal{B}_{i,t}^j \leq \mathcal{K}_j^{max} \tag{9.6}$$

Equation 9.7 ensures that $\mathcal{P}_{i,t} = 0$ when UE i is in sleep mode at slot t. Otherwise $\mathcal{P}_{i,t}$ takes a positive value, since we minimize the sum of $\mathcal{P}_{i,t}$. It is sufficient to set $\mathcal{P}_{i,t} = 1$ when UE i is awake at slot t. In addition, the equation 9.7 allows us to define the system Boolean variable $\mathcal{B}_{i,t}^j$ used in equations 9.5 and 9.6. Let $\mathcal{N}$ denotes a subset of integers in range 0 to the maximum number of cycles in the system (i.e. $|\mathcal{S}| \div \min_j f(j)$).

$$\forall i \in \mathcal{K}, \forall t \in \mathcal{S}, \forall k \in \mathcal{N} :$$
$$if\ k \times \sum_{j=1}^{|\Delta|} \Delta_i^j \times f(j) + \sum_{j=1}^{|\Phi|} \Phi_i^j \times g(j) \leq t$$
$$And$$
$$t \leq k \times \sum_{j=1}^{|\Delta|} \Delta_i^j \times f(j) + \sum_{j=1}^{|\Phi|} \Phi_i^j \times g(j) + \sum_{j=1}^{|\Gamma|} \gamma_i^j \times h(j)$$
$$then$$
$$\mathcal{P}_{i,t} = 1,\ \mathcal{B}_{i,t}^j = \begin{cases} 1 & \text{if } \gamma_i^j = 1\ \forall j \in (1..|\Omega|) \\ 0 & \text{otherwise} \end{cases}$$
$$else$$
$$\mathcal{P}_{i,t} = 0,\ \mathcal{B}_{i,t}^j = 0\ \forall j \in (1..|\Omega|)$$

(9.7)

Unfortunately, we cannot use the optimization problem mentioned above, mainly because the arrival of traffic is unknown and it is difficult to predict it and hence the variable $\mathcal{L}_{i,t}$ is difficult, if not impossible, to compute. Here the problem can be transformed into a linear problem, but since we can not solve it, it is not worthy doingso.

9.4 DRL-LP Overview

As aforementioned, it is hard to solve the optimization problem efficiently and without prior knowledge of the traffic patterns. For this reason, we propose the DRL-LP framework that leverages DRL. The DRL hides the complexity and the stochastic nature of the environment. It also helps the DRL-LP framework to make efficient and quick decisions that adapt according to the traffic patterns. Moreover, DRL-LP gains the ability to learn with time and adapts to different and unseen situations. In the balance of this section, we will present DRL and DRL-LP overview followed by a detailed description of DRL-LP.
DRL-LP periodically loops over the UEs in $\mathcal{K}$. For each UE i, DRL-LP captures an observation, then applies a configuration (i.e., C-DRX parameters and BWP size) and finally gets a reward after applying the configuration during $\mathcal{S}$ slots. We have designed the DRL-LP agent to ensure generality and then work in an unseen environment. The DRL-LP agent has been designed to work independently from the number of UEs and the traffic pattern. In what follows, we define the elements of the DRL-LP agent, including the state, the reward, and the action.
i) State: The DRL-LP agent captures an observation, per UE i, composed of four parts: $\{\mathcal{L}_i, \mathcal{B}_i, \mathcal{C}_i, \mathcal{U}\}$. The first three parts are specific for UE i, while the fourth part is common between all UEs. $\mathcal{L}_i$ is the history of latency $\mathcal{L}_{i,t}$ experienced by UE i during the past $|\mathcal{S}|$ slots. For each slot t, if arrived data $\mathcal{L}_{i,t}$ represents the time delay between t and the data transmission, else if no data arrived in slot t then $\mathcal{L}_{i,t} = -1$. $\mathcal{B}_i$ is the history of buffer status of UE i during the past $\mathcal{S}$ slots. $\mathcal{C}_i$ summarizes the action applied to the environment (i.e., both the C-DRX parameters and the BWP size). Each element $\mathcal{C}_{i,t}$ of $\mathcal{C}_i$ equals to the BWP size if UE i is awake at slot t, else it equals to 0. $\mathcal{U}$ is an array of $\mathcal{U}_t$ that represents the number of scheduled UEs for each slot t.
ii) Action: The DRL-LP agent has 4 discrete actions: (δ_i, ϕ_i, γ_i, ω_i). For each UE i, δ_i is the C-DRX cycle length, ϕ_i is the ON period, γ_i is the offset between the start of the ON period and the start of the cycle, and ω_i is the size of the BWP to configure. Accordingly, The C-DRX configuration (δ_i, ϕ_i, γ_i) is applied for UE i and BWP of size ω_i is configured for UE i.
iii) Reward: We have adapted an episodic approach, whereby each episode runs for fixed number of steps. For each episode, a randomly selected traffic pattern is applied. The reward r_t has been defined as follows:

$$r_t = \alpha \times \left(1 - \frac{\max_t \mathcal{L}_{i,t}}{\mathcal{L}^{max}}\right) + (1-\alpha) \times \left(1 - \frac{\left(0.4 \ + \ 0.6 \ \times \frac{(\omega_i \ - \ 20)}{80}\right) \times \mathcal{P}_i}{|\mathcal{S}|}\right)$$

Where $\max_t \mathcal{L}_{i,t}$ is the maximum latency for UE i during the past time window and $\mathcal{P}_i$ is the number of slots in which UE i was awake. $\mathcal{L}^{max}$ is the maximum latency in the system. α is a

given constant ($0 \leq \alpha \leq 1$) that defines the priority between latency and PC. The agent gets a higher reward whenever the maximum latency gets smaller or the sleep period is larger while using a smaller BWP.

9.5 DRL-LP detailed description

DRL-LP leverages the DQN algorithm with local and target networks, which is one of the most efficient DRL algorithms for continuous state space and discrete actions. We have tried the A2C [67] and the Actor-Critic using Kronecker-Factored Trust Region (ACKTR) [108] algorithms but the exploration phase was not efficient for DRL-LP environment as the agent was not able to converge. DRL-LP executes two steps: decision-making and updating the Q-Networks. We used two networks: a local Q-Network and a target Q-Network. The target network is the same as the local network, except that its parameters are updated every τ^{-1} step. They are combined to help the convergence and stabilization of the learning.

9.5.1 Decision making

DRL-LP agent observes a state and feeds it to the local Q-Network to get a discrete action distribution. Then, an ϵ-greedy approach is applied to choose an action from each distribution, which means DRL-LP agent will choose a random action over the possible actions with ϵ probability and the best action over the action distribution with a 1-ϵ probability. ϵ will decrease over time during the learning pushing the agent to explore the environment at the beginning of the training and driving it to exploitation over time.

9.5.2 Updating the Q-Networks

At each step, the current state, the action, the next state, and the reward are stored in a buffer known as the replay buffer. The local Q-Network is updated using a random sample from the replay buffer, which reduces the correlation between the agent's experiences and increases the stability of the learning. Using MSE and ADAM optimizer [69], the parameters of the local Q-Network are optimized at every step by considering the local and target values. In contrast, the parameters of the target Q-Network are updated every τ^{-1} step to stabilize the algorithm's convergence.

9.6 Performance Evaluation

In the balance of this section, we will introduce the simulation environment and parameters used for training DRL-LP agent. Then, we will evaluate the trained agent in a 5G simulated environment.

9.6.1 Simulation parameters and training phase

We have trained the DRL-LP agent using 3000 independent episodes. In each episode, the traffic pattern is selected randomly among two traffic pattern categories with different parameters: (i) periodic arrival rate with a period $\lambda_p \in \{10, 20, 50, 100, 200\}$ (ii) aperiodic traffic that follows a Poisson distribution with the mean arrival rate $\lambda_a \in \{1/10, 1/20, 1/50, 1/100\}$. For the periodic traffic, we add an initial offset o_{init} selected randomly, such as o_{init}, a positive integer smaller than the period. The goal behind o_{init} is avoiding data arrivals aligned with C-DRX cycles, which makes the simulation more realistic. The data size distribution is selected randomly

among (i) a fixed data size; (ii) a Poisson distribution data size. For both distributions, the mean data size is selected randomly among $\in \{10^3, 10^6\}$ Bytes in each episode. The number of steps in each episode is equal to 100 steps. In each step, the simulation runs for 100 slots. We used numerology 0 wherein 1 slot is 1 ms. We have trained the agent using 12 UEs. The set of cycle lengths is $\{10, 20, 50, 100\}$. The set of ON periods is $\{3, 5, 10\}$. The set of offsets is $\{0, 0.5 \times T_c\}$, such as T_c is the cycle length. The set of BWPs is $\{20, 50\}$. Each BWP has a limited amount of scheduling opportunities (i.e., DCI). We assume an aggregation level of 2, meaning that the maximum number of scheduled UEs per slot is 4 and 10 for the 20 MHz and 50 MHz BWP, respectively.

Table 9.1: DRL-LP parameters

Parameter	Value
α	0.5
$\mathcal{L}^{max}$	3 ms
Number of hidden layers	2
Hidden layer size	128 nodes
Discount factor γ	0.99
Batch size	256
Learning rate	$5 * 10^{-4}$
Replay buffer size	10^9
Soft update coefficient τ	0.001
Optimizer	ADAM [69]
ϵ-start	1
ϵ-decay	0.998
ϵ-end	0.01
Number of training episodes	3000

The considered parameters of the DRL-LP agent are presented in Table 9.1. To evaluate DRL-LP in a 5G environment, we extended the 5G system level simulator developed in [70] to support C-DRX operations. We have implemented our simulation environment using Python and Pytorch library. We have used a machine with 32 CPUs, an Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz (2.7 GHz with Turbo Boost technology), and 128 GB of RAM.
Figure 9.1 depicts the convergence evaluation of the DRL-LP agent during training. The x-axis represents the episodes, while y-axis represents the score (sum of rewards during an episode) averaged every 100 episodes. We observe that the DRL-LP agent converges after 2000 episodes since the curve tangents tend toward 0.

9.6.2 Inference phase

We evaluated the DRL-LP framework in terms of: *i*) Latency; *ii*) PC; *iii*) Number of UEs. We ran the simulation for 4000 slots. We compared DRL-LP with static C-DRX configurations and without C-DRX. We used the same traffic pattern selection mechanism as the training phase. Configuration A and configuration B denote the static configuration (cycle length: 10ms, ON period: 5ms, offset: 0ms and BWP: 50 MHz) and (cycle length: 100ms, ON period: 10ms, offset: 0ms and BWP: 20 MHz) respectively.
In Figure 9.2, the x-axis represents the latency (in ms), which is measured by $t_t^c - t_a^c$, where t_a^c is the arrival time of a c data block and t_t^c is the transmission time of the last part of the c

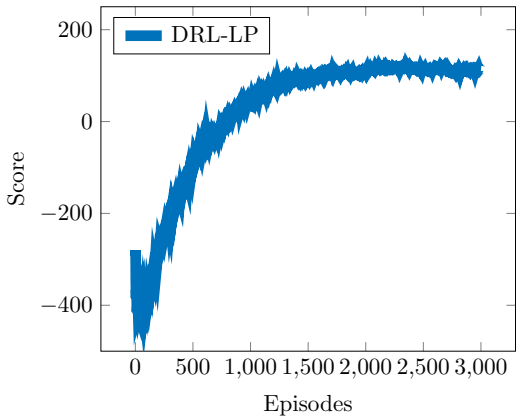


Figure 9.1: Convergence evaluation of DRL-LP agent during the training mode

data block. The y-axis represents the CDF of the latency collected by running the simulations for 4000 ms. We observe that without C-DRX, the latency is less than 2 ms because the UE is always on, and the data is scheduled directly on arrival. In configuration A, the latency is less than 5 ms because the maximum sleep time of a UE is 5 ms, and in the 50 MHz BWP up to 10 UE can be scheduled per slot. While for configuration B, more than 50% of the samples have latency greater than 50 ms, and 10% have latency greater than 150 ms because the sleep time is higher (up to 90 ms) and the BWP does not allow more than 4 UEs to be scheduled per slot. We note that for DRL-LP, 78% of the samples have latency less than 5 ms, which is better than configuration A, and 90% less than 25 ms, which is better than configuration B. We conclude that DRL-LP is able to avoid latency overflow while dynamically changing the configuration of C-DRX and BWP.

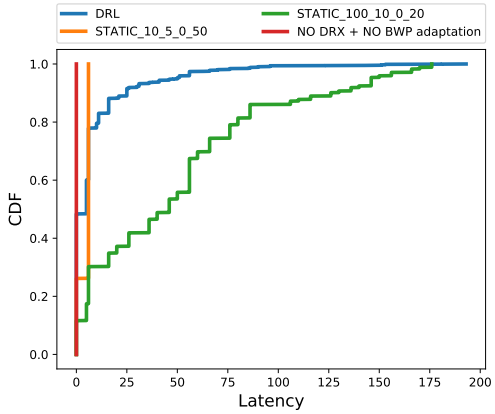


Figure 9.2: CDF of the Latency during inference mode

In figure 9.3, the x-axis represents the PC of a UE, calculated using the energy model in [20]. The y-axis represents the CDF of the PC collected at the same time as the latency in Figure 9.2. We observe that without C-DRX, the PC is the highest because the UEs are awake all the time. While in configuration A, we observe that all UEs achieve a gain of about 50% because all UEs are sleeping for half of the cycle. In Configuration B, the power gain is higher (90%) because the UEs sleep for 90% of the time. We note that 20% of the DRL-LP samples have a lower PC than Configuration B, which means a power gain of over 90%. In addition, all DRL-LP samples achieve a lower PC than configuration A (i.e., a power gain of more than 50%). We conclude that DRL-LP achieves a good balance between PC and latency (i.e., it achieves more than 50%

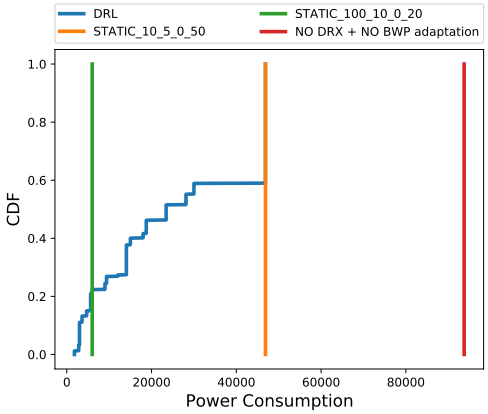


Figure 9.3: CDF of the PC during inference mode

power gain while maintaining less than 5 ms latency).

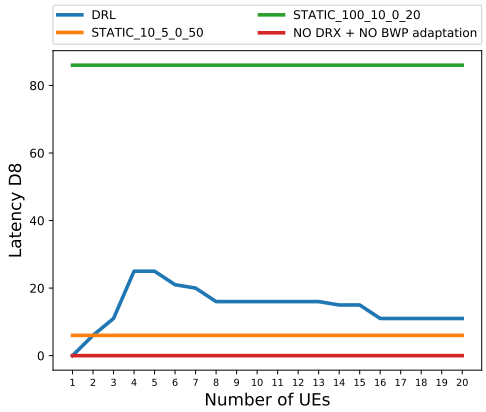


Figure 9.4: The 8th Decile of Latency during inference mode

In Figure 9.4, the x-axis represents the number of UEs and the y-axis represents the 8th decile (i.e., 80% of the samples have a value less than this) of the samples collected during the 4000 slots. We observe that DRL-LP is able to keep the latency below 25ms. We note that DRL-LP achieves better latency (i.e., less than 20ms) when run on more than 8 UEs. We justify this by the nature of DRL, which consists of approximating continuous states using neural networks, making it more biased to the observed states, and by the fact that the agent was trained on 12 UEs, which makes it more biased for larger number of UEs.

9.7 Conclusion

This chapter introduced DRL-LP, a DRL-based solution to balance PC and latency in 5G NR. DRL-LP will be used by the 5G base station to derive the C-DRX and BWP configuration per UE. The simulation results clearly showed that DRL-LP is able to find a trade-off between latency (i.e., achieve latency less than 5ms) and PC (i.e., achieve power gain more than 50%).

Part IV

Conclusions and Perspectives

Chapter 10

Conclusion

In conclusion, this thesis has delved deeply into the multifaceted realm of RAN slicing, a pivotal facet of advanced wireless communication systems. As the telecommunications landscape evolves, transitioning from 5G to 6G networks, the concept of RAN slicing becomes increasingly indispensable for catering to diverse service requirements.

Our research started with the challenge of spectrum slicing, made more intricate by the incorporation of 5G NR features such as numerology and BWPs. We undertook the complete problem formulation for resource allocation, considering mixed numerology environments and BWPs, and we mathematically demonstrated its NP-hard nature. Subsequently, a pioneering solution was proposed, hinging on RL, to expedite resource allocation with remarkable efficiency. Yet, the scope of our work extended beyond merely optimizing spectral resources. Network reliability, particularly in scenarios involving users on the fringe of coverage or experiencing RLFs, was another critical focus. We employed ML for RLF prediction, a vital stride in bolstering network stability. When an impending RLF is anticipated, proactive measures are initiated, be it through handovers or notifications to application services that mitigate the RLF's impact.

The spectrum-slicing solutions we developed are versatile, functioning seamlessly in both UL and DL directions. Nonetheless, the UL direction presented unique challenges, notably latency concerns due to the network's limited insight into user data volume for UL transmissions. Here, we harnessed RL to predict UL grants, striking a balance between latency reduction and the consequential rise in energy consumption.

Recognizing that resource allocation could be further optimized based on the UL/DL traffic ratio, we introduced dynamic TDD management using RL. This began in a single-cell and was successfully extended to multi-cell scenarios, skillfully mitigating cross-link interference patterns.

The insights gained from this thesis have far-reaching implications, serving as a cornerstone for the development of autonomous control systems for beyond 5G networks, thereby facilitating the evolution of SONs and the advent of AI-powered 6G networks. RAN slicing, as explored in this thesis, plays a central role in realizing the vision of dynamic, adaptable, and efficient wireless networks.

Chapter 11

Future Perspectives

11.1 Distributed Multi-Agents with different objectives: Distributed tracking and conflict mitigation

In future beyond 5G networks, we imagine the RAN as a set of independent smart agents working together to control RAN functions. These agents may come from different vendors and impact different RAN functions. RL is one of the key enablers of this ecosystem. RL offers a framework for understanding how agents should make effective decisions within an environment to maximize a cumulative reward defined based on the problem at hand. Traditional RL techniques are effective for tasks with small, discrete state and action spaces. However, many real-world applications involve more complex tasks with larger state spaces and continuous action spaces. Classical RL methods struggle to handle high-dimensional input. With the advancement in computing and storage capabilities, DRL, which combines deep learning with RL principles, has demonstrated remarkable achievements, reaching or surpassing human-level performance in challenging games. In our DRL-based contributions (i.e., DRL-RS, DRL-LLS, DRL-LP, and DRP), we used single-agent DRL to solve their respective problems. However, the most complex problems in telecommunications require multi-agent designs. The latter are more robust than single monolithic designs since agent failures can be compensated, which makes the system more robust and scalable. MADRL applies the idea and algorithm of DRL to the learning of multi-agent systems, which is an important Actuators method to develop swarm intelligence and has received increasing attention in recent years. In this thesis, we used MADRL to design and build MADRP, a multi-agent solution to tackle the dynamic TDD problem in a multi-cells environment where each agent controls a cell. While MADRL is a growing field, relatively little research has been carried out on distributed tracking and conflict mitigation problems in the field. In the state of the art, each agent observes all the other agents, and the training process of multiple agents is centralized. In practical tracking processes, each agent could hardly obtain the observation of all the other agents. Besides, the right reward function should be given to each agent to satisfy the goal of the overall system. In addition, the centralized training mode is not suitable for distributed tracking of multi-agent learning where the objectives of agents are different. Moreover, agents with different objectives may have conflicting actions, which is hard to mitigate with partial observations. Future research works should focus on enabling the co-existence of multiple agents with different goals and rewards while preserving the coherence of different RAN functions.

11.2 RAN slicing in FR2 systems

The 5G NR design has a notable characteristic, which is its capability to operate across two distinct frequency ranges: sub-6 GHz and mmW. With the sub-6 GHz spectrum becoming increasingly scarce, mmW frequencies, known for their wider bandwidths, are becoming more prevalent. mmW operates at high frequencies above 24 GHz, providing greater capacity. However, the characteristics, including propagation loss, signal blockage, and fading effects, differ significantly between mmW and the sub-6 GHz band. These differences introduce new challenges for system design and have an impact on end-to-end throughput and the quality of the user experience. To tackle these challenges, the 3GPP 5G NR standards have introduced novel features at the PHY and MAC layers to support directional communications. One of these critical features is beam management, a technique used to acquire and maintain beams. A beam is essentially a focused signal produced by a substantial antenna array. The 5G scheduler assigns beams to users in each time slot according to the user's positions and the radio environment. In this thesis, we enhanced RAN slicing using mixed numerology and BWPs in both time and frequency dimensions. In FR2, beam management will play an essential role in enhancing RAN slicing by isolating different beams in the frequency domain and hence reusing the spectrum. Future research works should focus on joint beam management and RAN slicing to enhance user experience.

11.3 RAN slicing in Device-to-Device (D2D) communication systems

Direct Device-to-Device (D2D) communication, which involves direct interaction between devices, such as users, without the need for data traffic to traverse through any infrastructure node, has been widely recognized as a crucial element in enhancing system performance and enabling new services in advanced systems beyond 5G. In general, the advantages of employing D2D communication encompass significantly higher spectral efficiency, improved data rates for typical users, increased capacity per unit area, expanded coverage, reduced latency, and enhanced cost and power efficiency. These benefits stem from several factors. Firstly, the close proximity of users utilizing D2D communication results in a "proximity gain". Secondly, there's an increased spatial reuse of time and frequency resources, known as the "reuse gain". Lastly, D2D communication utilizes a single link instead of both an UL and a DL resource when communicating through a base station in cellular mode, leading to a "hop gain". These combined gains contribute to the overall advantages of D2D operation. Sometimes, D2D communication becomes mandatory when the users are out of coverage or during a disaster when base stations are not available. D2D communication has two modes: (i) Mode 1, where the resources are controlled by the base station; (ii) Mode 2, where the resources are managed directly by the devices. In order to share the spectrum efficiently between different services and fulfill their requirements, RAN slicing should also consider D2D communication. Future work should focus on enabling RAN slicing in D2D systems leveraging distributed resource allocation solutions such as MADRL.

11.4 RAN slicing in Joint Communication and Sensing (JCAS) systems

Integrating sensing and communication functionalities into one Joint Communication And Sensing (JCAS) system exploits the same signal for sensing and communication. This integration substantially boosts the efficient use of scarce wireless resources by utilizing the same spectrum band simultaneously for the two functionalities. Furthermore, such integration remarkably reduces hardware power consumption, cost, and size by using the same hardware for sensing and communication. However, dedicating more resources to sensing negatively impacts the user experience in terms of throughput and latency. Hence, RAN slicing should consider resources dedicated to communication and resources dedicated to sensing. Indeed, a new metric is introduced to measure the sensing performance and sensing accuracy. The latter should be used to decide how many resources the system should allocate to sensing. Future works should focus on enhancing JCAS systems using RAN slicing to share the same infrastructure among different services, which may have different communication requirements (i.e., throughput, latency, and reliability) and sensing requirements (i.e., sensing accuracy).

Bibliography

[1] NGMN Alliance. "6G Use Cases and Analysis". In: (2022).

[2] N. C. Luong et al. "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey". In: *IEEE Communications Surveys Tutorials* (2019).

[3] *6G: The Next Horizon: From Connected People and Things to Connected Intelligence*. Cambridge University Press, 2021. DOI: 10.1017/9781108989817.

[4] John Fearnley and Rahul Savani. "The Complexity of the Simplex Method". In: *CoRR* abs/1404.0605 (2014). arXiv: 1404.0605. URL: http://arxiv.org/abs/1404.0605.

[5] Ericsson. *Defining AI native: A key enabler for advanced intelligent telecom networks*. 2023. URL: https://www.ericsson.com/en/reports-and-papers/white-papers/ai-native. (accessed: 6.10.2023).

[6] 3GPP. "Study on Artificial Intelligence (AI)/Machine Learning (ML) for NR air interface". In: *3GPP TR 38.843 version 1.0.0* (2023).

[7] ORAN Alliance. *Operator Defined Next Generation RAN Architecture and Interfaces*. 2020. URL: https://www.o-ran.org/. (accessed: 02.06.2020).

[8] Keysight. *Solving 5G and 6G Challenges with Artificial Intelligence (AI)*. 2023. URL: https://www.keysight.com/blogs/keys/thought-leadership/2023/06/15/solving-5g-6g-challenges-with-artificial-intelligence-ai. (accessed: 6.10.2023).

[9] Stefan Parkvall et al. "NR: The New 5G Radio Access Technology". In: *IEEE Communications Standards Magazine* (2017).

[10] Ahmed Amokrane et al. "Congestion control for machine type communications". In: *Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 2012*. IEEE, 2012, pp. 778–782.

[11] Adlen Ksentini et al. "Providing Low Latency Guarantees for Slicing-Ready 5G Systems via Two-Level MAC Scheduling". In: *IEEE Netw.* 32.6 (2018), pp. 116–123.

[12] Bouziane Brik, Adlen Ksentini, and Maha Bouaziz. "Federated Learning for UAVs-Enabled Wireless Networks: Use Cases, Challenges, and Open Problems". In: *IEEE Access* 8 (2020), pp. 53841–53849.

[13] Seiamak Vahid, Rahim Tafazolli, and Marcin Filo. "Small Cells for 5G Mobile Networks". In: May 2015, pp. 63–104.

[14] ORAN Alliance. "O-RAN Use Cases and Deployment Scenarios: Towards Open and Smart RAN". In: *White Paper, February 2020* (2020).

[15] Adlen Ksentini and Pantelis A. Frangoudis. "Toward Slicing-Enabled Multi-Access Edge Computing in 5G". In: *IEEE Netw.* 34.2 (2020), pp. 99–105.

[16] Ye Feng, Ampalavanapillai Nirmalathas, and Elaine Wong. "A Predictive Semi-Persistent Scheduling Scheme for Low-Latency Applications in LTE and NR Networks". In: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. 2019.

[17] Nurul Huda Mahmood and al. "Uplink Grant-Free Access Solutions for URLLC services in 5G New Radio". In: *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*. 2019.

[18] Tarik Taleb and Adlen Ksentini. "QoS/QoE predictions-based admission control for femto communications". In: *Proc. of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15*. IEEE, 2012.

[19] Pantelis A. Frangoudis et al. "An architecture for on-demand service deployment over a telco CDN". In: *Proc. of ICC 2016, Kuala Lumpur, Malaysia, May 22-27*. IEEE, 2016.

[20] 3GPP. "Study on User Equipment (UE) power saving in NR". In: *TR 38.840* Release 16 (2019).

[21] Geeksforgeeks. *5G Network Architecture*. 2023. URL: https://www.geeksforgeeks.org/5g-network-architecture/. (accessed: 18.10.2023).

[22] 3GPP. "System architecture for the 5g system (5gs)". In: *3GPP, TS 23.501 V15.12.0* (2021).

[23] Meghna Khaturia, Pranav Jha, and Abhay Karandikar. "5G-Flow: Flexible and Efficient 5G RAN Architecture Using OpenFlow". In: Oct. 2020.

[24] 3GPP. "5G NR; Physical layer procedures for data". In: *TS 38.214* Release 15 (2018).

[25] 3rd Generation Partnership Project (3GPP) TR 38.804. "5G; Study on New Radio Access Technology; Radio Interface Protocol Aspects". In: *3Gpp Tr 38.804* Release 14 (14.0.0 2017).

[26] 3rd Generation Partnership Project (3GPP). "Physical channels and modulation". In: *3GPP TS 38.211 version 15.3.0 Release 15* (2018).

[27] 3rd Generation Partnership Project (3GPP). "Study on User Equipment (UE) power saving in NR". In: *3GPP TR 38.840 V16.0.0 Release 16* (2019).

[28] ETSI. "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer - Measurements". In: *3GPP TS 36.214 version 8.4.0* (2010).

[29] Ahmed Amokrane et al. "Congestion control for machine type communications". In: *Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 2012*. IEEE, 2012, pp. 778–782.

[30] ITU-R. "framework and overall objectives of the future development of imt for 2020 and beyond". In: *ITU-R, Tech. Rep* (2015).

[31] P. Hedman. "Description of network slicing concept". In: *NGMN Alliance, white paper* (2016).

[32] GSMA. "Network slicing use case requirements". In: *GSMA, Tech. Rep* (2018).

[33] 3GPP. "NNr and ng-ran overall description". In: *3GPP, TS 38.300 V15.12.0* (2021).

[34] 3GPP. "System architecture for the 5G System (5GS)". In: *TS 23.501* Release 15 (2018).

[35] *5G; NR; Radio Resource Control (RRC); Protocol specification (3GPP TS 38.331 version 15.3.0 Release 15)*. Oct. 2018.

[36] Shihao Shen et al. "EdgeMatrix: A Resource-Redefined Scheduling Framework for SLA-Guaranteed Multi-Tier Edge-Cloud Computing Systems". In: *IEEE Journal on Selected Areas in Communications* 41.3 (2023), pp. 820–834. DOI: 10.1109/JSAC.2022.3229444.

[37] Shaoyuan Huang et al. "Fine-Grained Spatio-Temporal Distribution Prediction of Mobile Content Delivery in 5G Ultra-Dense Networks". In: *IEEE Transactions on Mobile Computing* (2022), pp. 1–14. DOI: 10.1109/TMC.2022.3226448.

[38] Karim Boutiba, Miloud Bagaa, and Adlen Ksentini. "On enabling 5G Dynamic TDD by leveraging Deep Reinforcement Learning and O-RAN". In: *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*. 2023, pp. 1–3. DOI: 10.1109/NOMS56928.2023.10154404.

[39] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: (2013).

[40] Miloud Bagaa, Karim Boutiba, and Adlen Ksentini. "On using Deep Reinforcement Learning to dynamically derive 5G New Radio TDD pattern". In: *2021 IEEE Global Communications Conference (GLOBECOM)*. 2021, pp. 1–6. DOI: 10.1109/GLOBECOM46510.2021.9685820.

[41] Ryan Lowe et al. "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6382–6393. ISBN: 9781510860964.

[42] K. Arulkumaran et al. "Deep Reinforcement Learning: A Brief Survey". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38.

[43] 5G Slicing Association. "Categories and Service Levels of Network Slicing White Paper". In: *White Paper, March 2020* (2020).

[44] ITU-R. ""Framework and overall objectives of the future development of IMT for 2020 and beyond M.2083". In: (2015).

[45] Natale Patriciello et al. "5G New Radio Numerologies and their Impact on the End-To-End Latency". In: *IEEE CAMAD* (2018).

[46] Lei You et al. "Resource Optimization with Flexible Numerology and Frame Structure for Heterogeneous Services". In: *IEEE Communications Letters* (2018).

[47] Vu Nguyen Ha et al. "Admission Control and Network Slicing for Multi-Numerology 5G Wireless Networks". In: *IEEE Communications Letters* (2019).

[48] J. J. Escudero-Garzas, Carlos Bousono-Calzon, and Alfredo Garcia. "On the Feasibility of 5G Slice Resource Allocation with Spectral Efficiency: A Probabilistic Characterization". In: *IEEE Access* (2019).

[49] Jingxuan Zhang et al. "Machine Learning Based Flexible Transmission Time Interval Scheduling for eMBB and uRLLC Coexistence Scenario". In: *IEEE Access* (2019).

[50] X. Foukas, N. Nikaein, and M. M Kassem. "FlexRAN: A flexible and programmable platform for software-defined radio access networks". In: *CONEXT 2016, 12th International on Conference on Emerging Networking Experiments and Technologiess* (2016).

[51] Adlen Ksentini and Navid Nikaein. "Toward Enforcing Network Slicing on RAN: Flexibility and Resources Abstraction". In: *IEEE Commun. Mag.* 55.6 (2017), pp. 102–108.

[52] Florian Kaltenberger et al. "The OpenAirInterface 5G new radio implementation: Current status and roadmap". In: *WSA 2019, 23rd ITG Workshop on Smart Antennas, Demo Session, 24-26 April 2019, Vienna, Austria*. 2019.

[53] 3rd Generation Partnership Project (3GPP). "Study on management and orchestration of network slicing for next generation network (Release 15)". In: *3GPP TS 28.801 version 15.1.0 Release 15* (2018).

[54] sharetechnote. *DCI*. 2020. URL: `https://www.sharetechnote.com/html/DCI.html`. (accessed: 16.10.2020).

[55] Sihem Bakri, Pantelis Frangoudis, and Adlen Ksentini. "Dynamic slicing of RAN resources for heterogeneous coexisting 5G services". In: *GLOBECOM 2019, IEEE Global Communications Conference* (2019).

[56] 3GPP 5G tools. *5G NR Throughput calculator*. 2021. URL: `https://5g-tools.com/5g-nr-throughput-calculator/`. (accessed: 04.01.2021).

[57] 3rd Generation Partnership Project (3GPP). "5G NR User Equipment (UE) radio transmission and reception; Part 1: Range 1 Standalone". In: *3GPP TS 138 101-1 V15.5.0 Release 15* (2019).

[58] X. Foukas, M. Marina, and K. Kontovasilis. "Orion: RAN Slicing for a Flexible and Cost-Effective Multi-Service Mobile Network Architecture". In: *The 23rd Annual International Conference on Mobile Computing and Networking (MobiCom '17)* (2017).

[59] Abderrahime Filali et al. *Communication and Computation O-RAN Resource Slicing for URLLC Services Using Deep Reinforcement Learning*. 2022. arXiv: `2202.06439 [cs.NI]`.

[60] Faroq Al-Tam, Noélia Correia, and Jonathan Rodriguez. "Learn to Schedule (LEASCH): A Deep Reinforcement Learning Approach for Radio Resource Scheduling in the 5G MAC Layer". In: *IEEE Access* (2020).

[61] Marco Zambianco and Giacomo Verticale. "Spectrum Allocation for Network Slices with Inter-Numerology Interference using Deep Reinforcement Learning". In: *PIMRC*. 2020.

[62] Abderrahime Filali et al. *Dynamic SDN-based Radio Access Network Slicing with Deep Reinforcement Learning for URLLC and eMBB Services*. 2022. arXiv: 2202.06435 [cs.NI].

[63] Lei You et al. "Resource Optimization With Flexible Numerology and Frame Structure for Heterogeneous Services". In: *IEEE Communications Letters* (2018).

[64] Anique Akhtar and Hüseyin Arslan. "Downlink resource allocation and packet scheduling in multi-numerology wireless systems". In: *WCNCW*. 2018.

[65] Jingxuan Zhang et al. "Machine Learning Based Flexible Transmission Time Interval Scheduling for eMBB and uRLLC Coexistence Scenario". In: *IEEE Access* (2019).

[66] Yalcin Sadi, Serhat Erkucuk, and Erdal Panayirci. "Flexible Physical Layer based Resource Allocation for Machine Type Communications Towards 6G". In: *2nd 6G SUMMIT*. 2020.

[67] Volodymyr Mnih et al. "Asynchronous methods for deep reinforcement learning". In: *International conference on machine learning*. PMLR. 2016, pp. 1928–1937.

[68] Donald E. Knuth. *The Art of Computer Programming: Fascicle 3: Generating All Combinations and Partitions*. Vol. 4. 2005.

[69] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[70] Karim Boutiba et al. "NRflex: Enforcing network slicing in 5G New Radio". In: *Computer Communications* (2021).

[71] Ming Ding et al. "Dynamic TDD transmissions in homogeneous small cell networks". In: *IEEE International Conference on Communications, ICC 2014, Sydney, Australia, June 10-14, 2014, Workshops Proceedings*. IEEE, 2014, pp. 616–621.

[72] Zukang Shen et al. "Dynamic uplink-downlink configuration and interference management in TD-LTE". In: *IEEE Commun. Mag.* 50.11 (2012), pp. 51–59. URL: https://doi.org/10.1109/MCOM.2012.6353682.

[73] Alexey Khoryaev et al. "Performance analysis of dynamic adjustment of TDD uplink-downlink configurations in outdoor picocell LTE networks". In: *4th International Congress on Ultra Modern Telecommunications and Control Systems, ICUMT 2012, St. Petersburg, Russia, October 3-5, 2012*. IEEE, 2012, pp. 914–921.

[74] Tian Ding et al. "Performance Analysis of Dense Small Cell Networks With Dynamic TDD". In: *IEEE Trans. Veh. Technol.* 67.10 (2018), pp. 9816–9830.

[75] Dalin Zhu and Ming Lei. "Cluster-based dynamic DL/UL reconfiguration method in centralized RAN TDD with trellis exploration algorithm". In: *2013 IEEE Wireless Communications and Networking Conference (WCNC), Shanghai, Shanghai, China, April 7-10, 2013*. IEEE, 2013, pp. 3758–3763.

[76] Mohammed Saad ElBamby et al. "Dynamic uplink-downlink optimization in TDD-based small cell networks". In: *11th International Symposium on Wireless Communications Systems, ISWCS 2014, Barcelona, Spain, August 26-29, 2014*. IEEE, 2014, pp. 939–944.

[77] Fengxiao Tang, Yibo Zhou, and Nei Kato. "Deep Reinforcement Learning for Dynamic Uplink/Downlink Resource Allocation in High Mobility 5G HetNet". In: *IEEE J. Sel. Areas Commun.* 38.12 (2020), pp. 2773–2782.

[78] Rudraksh Shrivastava, Konstantinos Samdanis, and Vincenzo Sciancalepore. "Towards service-oriented soft spectrum slicing for 5G TDD networks". In: *J. Netw. Comput. Appl.* 137 (2019), pp. 78–90.

[79] Shaozhen Guo, Xiaolin Hou, and Hanning Wang. "Dynamic TDD and interference management towards 5G". In: Apr. 2018, pp. 1–6. DOI: 10.1109/WCNC.2018.8377314.

[80] Yasar Sinan Nasir and Dongning Guo. "Multi-Agent Deep Reinforcement Learning for Dynamic Power Allocation in Wireless Networks". In: *IEEE Journal on Selected Areas in Communications* 37.10 (2019), pp. 2239–2250. DOI: 10.1109/JSAC.2019.2933973.

[81] Hyejin Kim, Jintae Kim, and Daesik Hong. "Dynamic TDD Systems for 5G and Beyond: A Survey of Cross-Link Interference Mitigation". In: *IEEE Communications Surveys and Tutorials* 22.4 (2020), pp. 2315–2348. DOI: 10.1109/COMST.2020.3008765.

[82] Cho-Hsin Tsai et al. "QoE-aware Q-learning based approach to dynamic TDD uplink-downlink reconfiguration in indoor small cell networks". In: *Wireless Networks* 25 (Aug. 2019). DOI: 10.1007/s11276-019-01941-8.

[83] Ali A. Esswie, Klaus I. Pedersen, and Preben E. Mogensen. "Online Radio Pattern Optimization Based on Dual Reinforcement-Learning Approach for 5G URLLC Networks". In: *IEEE Access* 8 (2020), pp. 132922–132936. DOI: 10.1109/ACCESS.2020.3011026.

[84] Zhi Yu et al. "Dynamic resource allocation in TDD-based heterogeneous cloud radio access networks". In: *China Communications* 13.6 (2016), pp. 1–11. DOI: 10.1109/CC.2016.7513198.

[85] Xiangyu Chen, Gang Chuai, and Weidong Gao. "Multi-Agent Reinforcement Learning Based Fully Decentralized Dynamic Time Division Configuration for 5G and B5G Network". In: *Sensors* 22.5 (2022). ISSN: 1424-8220. DOI: 10.3390/s22051746. URL: https://www.mdpi.com/1424-8220/22/5/1746.

[86] John Fearnley and Rahul Savani. "The Complexity of the Simplex Method". In: *CoRR* abs/1404.0605 (2014). arXiv: 1404.0605. URL: http://arxiv.org/abs/1404.0605.

[87] 3GPP. *5G; NGRAN; Xn Application Protocol (XnAP)*. Technical Specification (TS) 38.423. Version 16.5.0. 3rd Generation Partnership Project (3GPP), Apr. 2021.

[88] Shubham Khunteta and Ashok Kumar Reddy Chavva. "Deep Learning Based Link Failure Mitigation". In: *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2017).

[89] Luong Vy Le, Li-Ping Tung, and Bao-Shuh Lin. "Big data and machine learning driven handover management and forecasting". In: *2017 IEEE Conference on Standards for Communications and Networking (CSCN)* (2017).

[90] Syed Muhammad Asad Zaidi et al. "AI-Assisted RLF Avoidance for Smart EN-DC Activation". In: *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. 2020.

[91] François Chollet. *Keras: the Python deep learning API*. 2021. URL: https://keras.io/. (accessed: 04.05.2021).

[92] Alex Lamb et al. "Professor Forcing: A New Algorithm for Training Recurrent Networks". In: *NIPS 2016* (2016).

[93] Sagar Arora, Pantelis A. Frangoudis, and Adlen Ksentini. "Exposing radio network information in a MEC-in-NFV environment: the RNISaaS concept". In: *5th IEEE Conference on Network Softwarization, NetSoft 2019, Paris, France, June 24-28, 2019*. IEEE, 2019, pp. 306–310.

[94] Jorge Navarro-Ortiz et al. "A Survey on 5G Usage Scenarios and Traffic Models". In: *IEEE Communications Surveys Tutorials* (2020).

[95] Amin Azari et al. "User Traffic Prediction for Proactive Resource Management: Learning-Powered Approaches". In: *2019 IEEE Global Communications Conference (GLOBECOM)*. 2019.

[96] Natale Patriciello et al. "The Impact of NR Scheduling Timings on End-to-End Delay for Uplink Traffic". In: *2019 IEEE Global Communications Conference (GLOBECOM)*. 2019.

[97] Ahmet Gizik, Ozgun Alkin Sensoy, and Engin Masazade. "Enhanced Dynamic Scheduling for Uplink Latency Reduction in Broadband VoLTE Systems". In: *2021 55th Asilomar Conference on Signals, Systems, and Computers*. 2021.

[98] Mohammad Shehab and al. "Traffic Prediction Based Fast Uplink Grant for Massive IoT". In: *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*. 2020.

[99] Lihua Ruan, Maluge Pubuduni Imali Dias, and Elaine Wong. "Machine Learning-Based Bandwidth Prediction for Low-Latency H2M Applications". In: *IEEE Internet of Things Journal* (2019).

[100] Eslam Eldeeb, Mohammad Shehab, and Hirley Alves. "A Learning-Based Fast Uplink Grant for Massive IoT via Support Vector Machines and Long Short-Term Memory". In: *IEEE Internet of Things Journal* (2022).

[101] Karim Boutiba, Miloud Bagaa, and Adlen Ksentini. "Radio resource management in multi-numerology 5G new radio featuring network slicing". In: *ICC 2022*. Ed. by IEEE. Seoul, 2022.

[102] Ziyang Zhang and Xin Zhang. "Logic Controller-Based Discontinuous Reception (DRX) System for NR". In: *Journal of Physics: Conference Series* (2021).

[103] JianHong Zhou et al. "Actor-Critic Algorithm Based Discontinuous Reception (DRX) for Machine-Type Communications". In: *2018 IEEE Global Communications Conference (GLOBECOM)*. 2018.

[104] Philipp Bruhn and German Bassi. "Machine Learning Based C-DRX Configuration Optimization for 5G". In: *Mobile Communication - Technologies and Applications; 25th ITG-Symposium*. 2021.

[105] Farnaz Moradi et al. "Flexible DRX Optimization for LTE and 5G". In: *IEEE Transactions on Vehicular Technology* (2020).

[106] Stefano Paris, Klaus Pedersen, and Qiyang Zhao. "Adaptive Discontinuous Reception in 5G Advanced for Extended Reality Applications". In: *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*. 2022.

[107] Venkatesh Ramaswamy, Jeffrey T. Correia, and Darcy Swain-Walsh. "Analytical Evaluation of Bandwidth Part Adaptation in 5G New Radio". In: *2021 IEEE PIMRC*. 2021.

[108] Yuhuai Wu et al. "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation". In: *Advances in neural information processing systems* (2017).