INSTITUT POLYTECHNIQUE DE PARIS

NNT : 20XXIPPAXXXX

Thèse de doctorat

Joint Offloading-Scheduling Policies for Future Generation Wireless Networks

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Réseaux, Information et Communications

Thèse présentée et soutenue à Palaiseau, le xxx., par

IBRAHIM DJEMAI

Composition du Jury :

Loutfi Nuaymi Professeur, IMT Atlantique (IRISA)	Rapporteur
Philippe Mary Professeur, INSA Rennes (IETR)	Rapporteur
Luc Vandendorpe Professeur, UCLouvain (ICTEAM)	Examinateur
E. Veronica Belmega Professeure, ESIEE Paris (LIGM)	Examinatrice
Salah El Ayoubi Professeur, CentraleSupélec (CNRS-L2S)	Examinateur
Mireille Sarkiss Maîtresse de conférences, Télécom SudParis (SAMOVAR)	Co-Encadrante de thèse
Philippe Ciblat Professeur, Télécom Paris (LTCI)	Directeur de thèse
Frédéric Lehmann Professeur, Télécom SudParis (SAMOVAR)	Co-Directeur de thèse

Contents

List of Figures

Acronyms

2QL Double Q-Learning 35, 36, 56, 57, 61

5G Fifth Generation 2, 4, 18, 19, 46

AI Artificial Intelligence 2, 5, 7, 12, 26, 28, 72, 96

ANN Artificial Neural Network 16, 27

AWGN Additive White Gaussian Noise 20, 49, 74

BS Base Station 3, 6, 12, 13, 15, 19–22, 41, 46–50, 71–77, 79, 83, 85, 86, 90, 96–98

CDF Cumulative Distribution Function 54

CDMA Code Division Multiple Access 19

CH Cluster Head 71, 73, 74, 79, 83

CNN Convolutional Neural Network 28

CSIT Channel State Information at the Transmitter 49, 74

DL Downlink 20–23, 46, 49–51, 56, 57, 71, 73–75, 77, 80, 85, 90, 97

DNN Deep Neural Network 5

DPP Determinantal Point Process 17

DQN Deep Q-Network 36–38, 56, 57, 60, 61, 64, 65, 68, 70, 72, 96

DRL Deep Reinforcement Learning 5–7, 46–48, 62, 64, 68, 70–72, 83, 89, 96, 97

EH Energy Harvesting 4, 5, 7, 17, 46, 47, 62–64, 68, 73, 74, 90, 96, 97

EMBB Enhanced Mobile Broadband 3, 4, 19, 23

EN Evaluation Network 36, 37

ER Experience Replay 37, 38, 64, 65

FDMA Frequency Division Multiple Access 4, 19, 22, 71, 74, 75, 77, 97

FedRL Federated Reinforcement Learning 6, 70–72, 83, 98

FL Federated Learning 6, 27, 40, 41, 70, 72, 82, 83, 94, 97

GoP Groups-of-Picture 18

ICT Information and Communication Technologies 2, 3, 14

IMM Immediate Scheduler 65, 91

IoT Internet of Things 2–4, 12–14, 18, 23, 41, 46, 47, 72, 78

KL Kullback-Leibler 38, 39

LAN Local Area Network 12

LLM Large Language Model 26

M2M Machine-to-Machine 2, 3

MC Markov Chains 29

MCC Mobile Cloud Computing 3, 12, 13

MDP Markov Decision Process 28–31, 33, 37, 46–48, 52, 54, 56, 63, 70, 73, 81, 83, 96

MEC Mobile Edge Computing 3, 5–7, 13–15, 41, 46–48, 56, 57, 64, 68, 71–74, 79, 83, 84, 90, 94, 96–98

MIMO Multiple-Input Multiple-Output 4, 47

ML Machine Learning 5, 16, 17, 26–28, 40, 98

MMTC Massive Machine Type Communications 19, 23

MRP Markov Reward Process 29

NL Naive Local 57, 65, 91

NN Neural Network 5, 6, 27, 28, 36–38, 40, 46, 47, 56, 57, 61, 64, 68, 89, 90, 96

NO Naive Offload 57, 61, 65, 91

NOMA Non-Orthogonal Multiple Access 4–7, 19–23, 46–48, 50–52, 57, 58, 60, 65, 68, 70–75, 85, 94, 96–98

NR Naive Random 57, 65

OFDMA Orthogonal-Frequency Division Multiple Access 4, 19

OMA Orthogonal Multiple Access 4, 19, 23, 47, 71, 85

PI Policy Iteration 34, 56, 57, 60, 61, 96

PPO Proximal Policy Optimization 7, 38, 39, 47, 62, 64, 65, 68, 71–73, 89–91, 94, 96, 97

QL Q-Learning 34–36, 56, 57, 60, 96

QoS Quality of Service 2, 13, 23

RAN Radio Access Network 3, 5, 12, 13

RF Radio Frequency 15–17, 62

RFID Radio Frequency IDentification 16

RL Reinforcement Learning 5–7, 27, 28, 30, 31, 34, 38, 46–48, 56–58, 62, 68, 70–72, 81, 83, 96–98

RNN Recurrent Neural Network 28

SIC Successive Interference Cancellation 4–6, 19–23, 47, 51, 71, 73–76, 97

SNR Signal-to-Noise Ratio 74, 85, 86, 88

SVM Support Vector Machines 16

TD Temporal Difference 35, 38

TDMA Time Division Multiple Access 4, 19, 57, 58, 65, 68

TN Target Network 37, 38

TRPO Trust Region Policy Optimization 38, 39

UE User Equipment 12, 13, 20–23, 46, 48–51, 53, 54, 57, 62–64, 68, 70–77, 79–91, 94, 96–98

UL Uplink 20, 21, 23, 46, 49–51, 56, 71, 73–77

URLLC Ultra-Reliable Low Latency Communications 3, 4, 19, 23

VI Value Iteration 33, 56, 57, 60, 61, 96

Introduction

This thesis was carried out thanks to the funding "Bourse d'Excellence" from SAMOVAR laboratory, Télécom SudParis, Institut Polytechnique de Paris. It has been conducted from November 2020 to February 2024, and was jointly supervised by Dr. Mireille SARKISS from Telecom SudParis, Pr. Philippe CIBLAT Telecom Paris, and Pr. Frederic LEHMANN from Telecom SudParis.

Context

The ever-evolving technologies that provide solutions and improvements to wireless networks and devices, from Artificial Intelligence (AI), Fifth Generation (5G), to Internet of Things (IoT), are a testament to the progress, humanity in general and the industry in particular, have made to make our lives more convenient and connected. Information and Communication Technologies (ICT) such as smartphones, WiFi routers and laptops are used worldwide, with an exponential increase in the number of connected devices to the network.

Statistics show that the estimated number of smartphone mobile subscriptions in 2023 have reached 8.46 billion, and is expected to exceed 9.21 billion by the year 2029 [1]. For IoT devices, the numbers grow even more, where 11.740 billion devices were connected in 2023 in the regions of Europe, North America and Greater China, and a projected 21.37 billion for the year 2030 [2]. These numbers show the huge scale of the current network that we are all connected to. We show examples of the projected growth in the number of mobile subscriptions, as well as the number of connected IoT devices, in Figure 1 and Figure 2 . To handle such a large number of devices, network infrastructures need to be

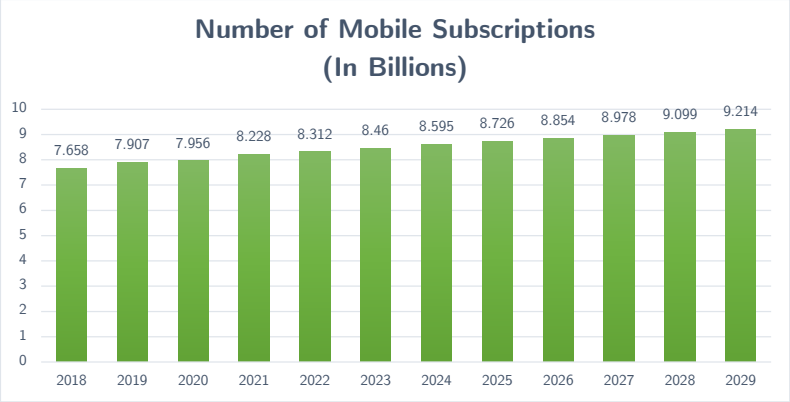

Figure 1: Number of mobile subscriptions worldwide.

at the level of demand, and subsequently provide an adequate Quality of Service (QoS) for each type of device. From streaming, transferring data, to performing calculation, the applications that can be used are various and exploit different aspects of the systems.
5G delivers on the promises of more capable networks that can handle the different requirements of current connected devices and beyond. The areas of focus of 5G include:

- Machine-to-Machine (M2M) communications, that can withstand huge amounts of IoT devices simultaneously that communicate with each other.

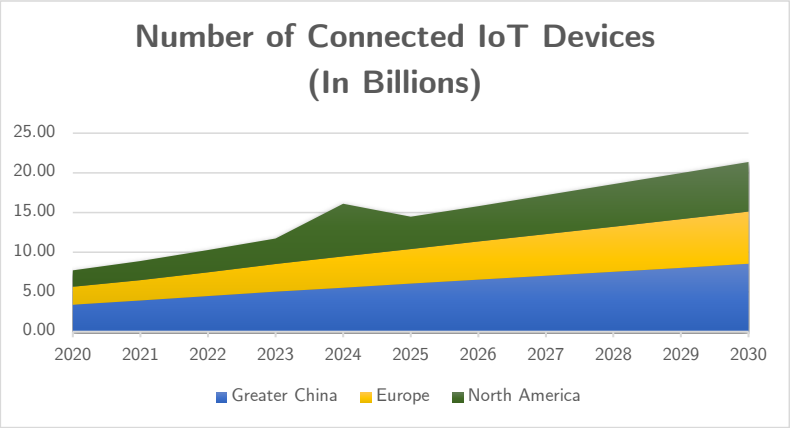

Figure 2: Number of connected IoT devices worldwide.

- Ultra-Reliable Low Latency Communications (URLLC), with faster delay times than the previous Fourth Generation for delay-sensitive applications, e.g., autonomous and smart vehicles.

- Enhanced Mobile Broadband (EMBB), which can offer higher data rates for applications that require a lot of data to be transmitted, e.g., high resolution streaming.

In massive M2M communications, IoT devices often require performing heavy computations that can be very slow to run on the small and limited internal processors. In addition to which, computations may endure high energy costs, depleting thus the battery rapidly. Mobile Cloud Computing (MCC) was used to overcome these computational challenges by offloading the resource-hungry tasks to a remote server in the internet, carry away the execution, and send back the result to the IoT devices once finished. However, transmission, processing and reception times are delay concerns that are raised when offloading to a remote cloud server, especially for delay-sensitive tasks. Recently, Mobile Edge Computing (MEC) appeared as a promising solution to make the cloud server closer to the devices at the edge of the network (i.e., at the Base Station (BS) level in a Radio Access Network (RAN)). In this way, the MEC server ensures that IoT devices can deliver faster execution times than they can perform local processing, while enduring a smaller delay penalty compared to MCC, hence it can achieve M2M on a large scale [3].

Furthermore, energy consumption is a very important aspect when considering IoT devices in M2M communications. An IoT device can be located in difficult-to-attain locations for better sensing or communication, therefore changing its battery becomes a hassle, especially if it utilizes its battery quite frequently, without being able to connect to the electrical grid. In addition, global emissions of CO_2 represent a major concern for the environment and the temperature of the planet, and ICT is one of the contributing factors to this issue. It is reported that network infrastructures and connected devices represent 4% of the global CO_2 emissions [5], which makes working towards 'green' communications an essential objective.

Research on optimizing network resources has attracted much attention to minimize the energy consumption and reduce carbon footprint. Sustainable and green energy sources

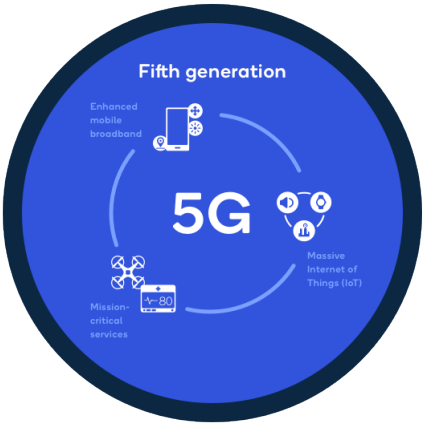

Figure 3: 5G areas of focus. Source: [4].

have been also the focus of research in many areas, including communications, to further reduce carbon emissions and provide an alternative and a durable solution to energy consumption issues. In particular, Energy Harvesting (EH) has emerged as the process of collecting energy from external and ambient sources, e.g., from solar radiation using photovoltaic panels, wind energy using turbines, and Radio-Frequency ambient energy coming from transmission signals. This type of energy can be captured using antennas and specialized sensors, and then stored in batteries. EH capabilities at IoT devices allow them to be free from frequent human intervention, and from the need for fossil-fuel and nuclear-based energy, while relying on sustainable energy resources.

In a different manner, achieving higher bit-rates during communications is a constant area of researchers focus when developing technologies for every new generation of wireless networks, and 5G constitutes the latest practical illustration of this goal. As an example, this new generation of cellular networks requires the deployment of more antennas in the system and thus increasing the diversity and the number of users that can access simultaneously, i.e., adopting Multiple-Input Multiple-Output (MIMO) technologies in a massive way, i.e., Massive MIMO, in addition to the use of *mm*Wave bandwidth to enhance the communication speeds and use an unexploited range of frequencies that can accommodate the growing needs, and therefore satisfy the URLLC and EMBB needs outlined for 5G.

Multiple access techniques are other prominent research topics that allow more users to share network radio resources, without enduring much loss in terms of interference. Traditional multiple access techniques like Time Division Multiple Access (TDMA) or Frequency Division Multiple Access (FDMA) can accommodate as much as the time sampling or the bandwidth spectrum can allow for in an orthogonal way (i.e., without interference between signals). However, enabling non-orthogonality can theoretically break the ceiling set by orthogonal methods. Indeed, Non-Orthogonal Multiple Access (NOMA) promises to outperform the Orthogonal Multiple Access (OMA) methods by offering better spectral efficiency, and thus higher data-rates. NOMA allows for the superposition of multiple users in time and frequency. Then, using the Successive Interference Cancellation (SIC) scheme for decoding, it deals with removing the interference between users. With 5G still using Orthogonal-Frequency Division Multiple Access (OFDMA), NOMA arises as a strong mul-

tiple access technique for future use [6].
Nevertheless, NOMA SIC decoding can suffer from error propagation and a loss in performance when increasing the number of users. In order to reduce inter-user interference, users can be clustered appropriately in groups that perform NOMA in different sub-carriers.

On the other hand, AI is one of the most capable tools to make use of the huge amount of data collected from the connected devices, whether it concerns images, texts, sounds, or excel sheets. Recent breakthroughs in the field of AI have allowed for some applications that were not thought possible, such as Image recognition, text-to-speech, recommendation systems, and chat bots, illustrated in the hugely popular ChatGPT. Machine Learning (ML) is in the center of this advancement, for the way it is able to learn from data and recognize complex patterns that are otherwise extremely hard to conceive with hand-crafted methods. ML can be classified into different categories that depend on the type of data and task, namely Supervised Learning, Unsupervised Learning, and Reinforcement Learning (RL).
RL deals with how best to navigate complex environments, by taking actions from observations about states and receiving feedback from the environment in the form of rewards. These rewards inform the RL model about the effectiveness of its actions. The goal in this type of learning is to devise policies that can behave optimally in any environment.

Moreover, Neural Network (NN)s have allowed for boosting the performance of ML models by mimicking the behavior of neurons that fire responses upon receiving signals. From that, graphs can be designed of multiple neurons connected in a layered-fashion to each other. Signals are forwarded between these layers such that complex patterns can be detected in input data, and appropriate outputs are produced (after training on a specific dataset). Deep Neural Network (DNN)s takes this concept further by adding more layers to the network, allowing it to capture even more high-level patterns, and approximating even more complex functions.
More specifically, Deep Reinforcement Learning (DRL) has proven to be a very efficient approach for decision making, by approximating complex and large environments using DNNs, as it allowed for the creation of AlphaGo, the DRL model that beat the world champion in the famous Chinese game 'Go' [7], [8]. Numerous works are published that illustrate the application of RL techniques in general, and DRL ones in particular, to the communications field [9].

We mention some examples of the objectives of these works. Energy efficiency deals with the optimization of the energy consumption of the network devices (end users or at the RAN level), in specific scenarios [10]–[12]. The policy aim in this case is to maximize the reward, which is higher if the energy consumed is less.
Resource Allocation is another example, where the users have to share resources (e.g., bandwidth, power, time) in a wireless communication setup in an efficient manner, given the circumstances presented, and the use of DRL techniques in this problem has proven to be successful [13]–[15]. More specifically, resource allocation problems related to MEC offloading, EH capabilities, and NOMA-enabled communications have been explored in some works in a separate or a joint manner [16]–[22]. The work in [23] presents other examples of the use of DRL techniques in resource allocation problems.

Goal

In this thesis, we consider a multi-user network where the users are allowed to perform NOMA transmission and have energy harvesting capabilities. They are connected to a BS or an access point (cellular or IoT networks), which has more computational and energy resources through a MEC server close to them. The goal of this thesis is to design efficient policies for joint scheduling-offloading in such wireless networks. Data packets arrive at the users' buffers with strict delay constraints, i.e., packets have to be executed before a maximum delay is reached, otherwise the packets are dropped. Therefore, the users have to execute their packets either locally with limited computational powers, or remotely by offloading them to the MEC server, and awaiting the results.

In this case, a policy is needed to determine the best action to take (local or offload execution of packets), which depends on the channel conditions of users, their buffer states, and their battery levels. To resolve the optimization problem, RL methods surface as the relevant techniques to find policies that can choose the best actions in each circumstance. We show that the use of optimal and trial and error-based RL methods, produces the best performing methods when compared to other standard methods. Furthermore, when scaling the system to a larger number of states, a curse of dimensionality arises, and using the same RL used previously is no longer viable. Therefore, we shift to DRL methods due to their generalization capabilities to unseen states using NNs, and thus a performance advantage is maintained above the standard methods.

Moreover, increasing the number of users result in an exponentially larger environment, rendering the task of learning the right policies by the DRL agent harder to achieve. Furthermore, adding more users will cause a performance drop in NOMA, especially in the downlink, where the users do not have the sufficient computational resources to handle the sequential decoding in the limited time. Therefore, clustering the users into groups helps maintain the performance offered by DRL methods, and distribute the task among multiple agents. Moreover, a NOMA-aware clustering will result in better downlink decoding performance using SIC.
In this case, a NOMA clustering method is developed to account for the spatial location of the users, as well as their channel conditions. Additionally, producing policies for each cluster necessitates the use of a distributed learning technique, i.e., Federated Learning (FL) [24]. FL is a learning technique that allows the splitting of learning across multiple nodes that can train on local data without sharing it (thus achieving data privacy), and utilize the collected local information in a central node to produce a global model that is broadcasted back to the local nodes. In our context, FL is used to train an RL agent for each cluster (Federated Reinforcement Learning (FedRL), or Multi-Agent RL), with minimal information communicated between the clusters. Experimental results using Federated DRL techniques have shown good performance compared to standard ones.

Outline

This thesis is composed of 4 chapters:

- The first chapter deals with communication based preliminaries, and introduce the different aspects of our system model, i.e., Edge Computing, Energy Harvesting, Data Modeling, and Non-Orthogonal Multiple Access.
- The second chapter presents some details about Artificial Intelligence in general, and Reinforcement Learning in particular. The chapter details Markov processes that constitute the building block of RL methods, as well as some well-known methods, to be used in the subsequent chapters.
- The third chapter presents our first two contributions, where we develop policies for jointly optimizing scheduling and offloading in a MEC-powered wireless network, with NOMA capabilities, and Energy Harvesting devices. The policies used therein are obtained using RL methods and compared against some other methods. The first part deals with a system that does not incorporate EH at the user level, and thus has lower complexity. DRL methods are used to assess the scalability performance. The second part enables Energy Harvesting capabilities for the users, and proposes to use another powerful DRL method, namely Proximal Policy Optimization (PPO).
- In the fourth chapter, more users are allowed in the network, and a NOMA clustering method is developed to resolve the scalability issues. In this case, Federated Reinforcement Learning policies are produced in a dynamic environment with multi DRL agents. The results show that the use of PPO provides better performance than using standard methods.

Publications

The presented work in this thesis was published in the following articles

Journals

J1. I. Djemai, M. Sarkiss and P. Ciblat, "Federated Reinforcement Learning for Scheduling-Offloading Policies in Multi-Cluster NOMA systems with Mobile Edge Computing and Energy Harvesting", submitted in IEEE Transactions on Green Communications and Networking, January 2024.

International Conferences

C2. I. Djemai, M. Sarkiss and P. Ciblat, "NOMA-based Scheduling and Offloading for Energy Harvesting Devices using Reinforcement Learning", IEEE Asilomar Conference on Signals, Systems, and Computers, October 2023.

C1. I. Djemai, M. Sarkiss and P. Ciblat, "Joint Scheduling-Offloading policies in NOMA-based Mobile Edge Computing Systems", IEEE Wireless Communications and Networking Conference (WCNC), March 2023.

Part I

Preliminaries

CHAPTER 1

Communication-Based Preliminaries

This chapter introduces communication-related preliminaries, covering different aspects of the system that will be used in chapters 3 and 4. In particular, we consider how data are handled, both at the user and the server level, as well as the energy consumed to process it, and the network access technique to transmit it.

The remainder of the chapter is as follows: Section 1.1 covers Edge Computing. Section 1.2 is dedicated to Energy Harvesting. Section 1.3 discusses Data Models. Lastly, section 1.4 focuses on Non-Orthogonal Multiple Access technique.

1.1 Edge Computing

Conventional wireless network design incorporates a Radio Access Network (RAN), that encompasses the User Equipments (UEs) and the Base Station (BS), through which the UEs connect wirelessly to reach the Core Network. This Core Network is responsible for forwarding the signal transmitted by the UEs to its desired destination via wired connections.

The computational capabilities of the UEs are generally very limited, which prevents them from performing compute-intensive tasks locally, such as Virtual-Reality high-resolution streaming, AI image generation, and all what the newest applications on smartphones rely on. This is the case either due to energy concerns, since processing these tasks consume a sizeable portion of the internal battery's energy, or due to delay issues, where the tasks have some delay constraints that require them to be executed in a limited duration before becoming obsolete. Therefore, solutions are needed to execute these tasks efficiently.

Equipping the UEs with more powerful processing units seems to be the obvious approach to handle higher computational loads, as well as large capacity batteries. However, the downside in this case is the huge cost that comes with it, and the enormous effort needed to update these devices every few years to keep up with the ever-increasing processing-intensive tasks. Moreover, implementing this solution becomes even more problematic as the number of connected devices in the network increases. Indeed, according to Statista [25], the number of connected IoT devices in 2023 is estimated at 15.1 billion and it is expected to reach 29.4 billion by the year 2030.

An alternative solution is to *offload* the tasks to an external server, powerful enough to process them rapidly, without the dependence on limited-sized batteries. More precisely, a *cloud* computing solution. Cloud computing refers to the existence of a set of servers that can handle multiple tasks. Generally accessed through the internet (public cloud), or within a company's Local Area Network (LAN) (private cloud), devices can access the functionalities of the cloud from anywhere by offloading their tasks and retrieving the results. In the context of wireless networks, IoT devices as an example have very limited processing units due to their small size. Thus, they need to offload heavy tasks necessitating an access to a Mobile Cloud Computing (MCC), naturally through the RAN and the core network. Figure 1.1 illustrates a basic example of such an architecture.

Nevertheless, some challenges are risen with the use of MCC technology. Delay concerns are perhaps one of the biggest of issues and are attributed to the fact that MCC servers

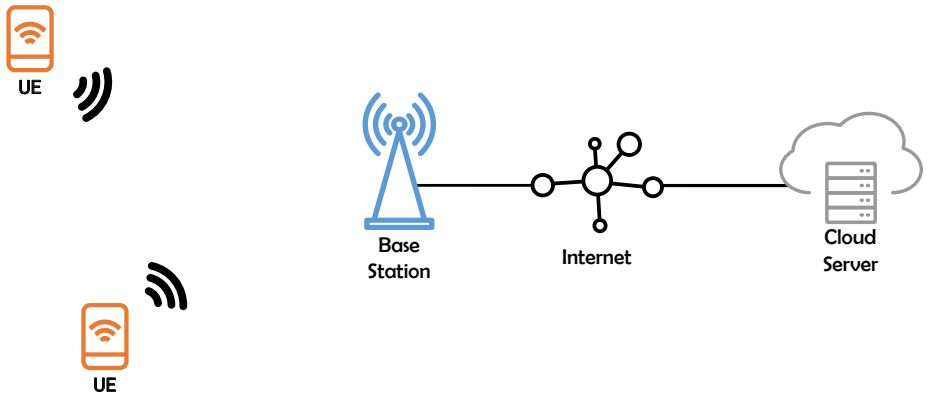

Figure 1.1: Basic mobile cloud computing MCC architecture.

are located far from the UEs. The packets that carry the tasks to be processed can go through a series of networks before reaching the cloud. This shortcoming is brought to light when using real-time applications, which imposes very strict delay constraints. In addition, Quality-of-Service (QoS) is an important aspect to consider for the same type of applications, which is poorly handled as well in an MCC scenario [26]. Other challenges are also faced when using MCCs, such as security constraints, and lag-prone transmission of the data.

Mobile Edge Computing (MEC) is an alternative to MCC, proposed in [27], that aims to reduce the processing delay of offloaded tasks by UEs, by moving the cloud closer to the edge of the network. MEC concept is inspired by cloudlets [28], which are a set of cloud servers used in local Wi-Fi networks, near access points, to facilitate the access to cloud services. They offer a significant delay reduction over public clouds. Similarly, MEC servers are located near the BSs in the RAN, and offer fast and easy access to powerful servers (although not as performing as MCC servers). This MEC technology adds more flexibility, where the MEC servers can be tailored to the RAN's requirements, and improve the QoS of the UEs accessing their services, besides reducing the UE's energy consumption.

Then, enabling each RAN with a MEC server installed near the BS will help tackle the problem of MCC. This comes at the expense of additional hardware cost. A middle-ground design that presents a compromise between cost, performance and latency, is Fog Computing [29]. It consists essentially in aggregating several RANs' tasks into one server. Figure 1.2 showcases the different levels of computing described in this section. Higher the computing unit is located, better the performance is, but at the expense of higher latency.

Discussion

A variety of applications can utilize MEC servers' capabilities to provide better services. These can range from industrial manufacturing, healthcare to autonomous vehicles [3]. In our use case, UEs that can be IoT devices, smartphones, smart vehicles or any mobile users, can all take advantage of the MEC technology to execute their tasks as efficiently as possible. This is illustrated in the contributions of chapter 3 and 4.

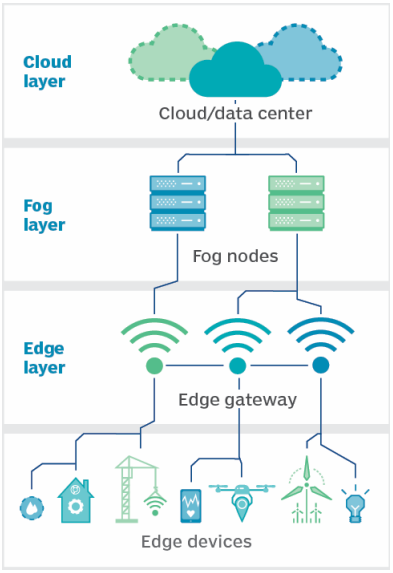

Figure 1.2: Edge to cloud architecture layers. Source: [3].

1.2 Energy Harvesting

With the ever-increasing number of connected devices in networks from IoT devices, smartphones, to connected vehicles, energy concerns are risen. It is estimated according to the International Energy Agency, that the global data centers and transmission networks currently account for 1 to 1.5% of the energy consumption worldwide [30]. Similarly, the CO_2 emissions of these infrastructures, as well as the connected devices, constitute 4% of the global emissions [5]. These numbers suggest that the use of network infrastructures and devices, requires a significant energy deployment and a considerable damage on the environment. Therefore, there is a growing need to handle the energy consumption issue.

Indeed, network vendors and Information and Communication Technologies (ICT) companies are attempting to optimize their products to reduce consumption. Moreover, there is a push for using alternative green energy sources to effectively reduce CO_2 emissions. The energy can be pulled from a variety of sources, including Solar, Wind, or Radio Frequency, reducing thus the need for environmentally harmful fossil-fuel and nuclear-based energy generation.

In addition to environmental impact, energy distribution is a major challenge from a feasibility and cost perspective. Providing power to IoT devices scattered everywhere in the environment is a difficult task to achieve with the exponential growth of the number of devices. The use of limited-size Lithium-Ion batteries that can be recharged appears to be the solution for the current time. It would also be a sustainable solution for the future, especially for devices that emit sparse low power signals. However, many are the devices that require recharging in a frequent manner and can be difficult to attain, and so are the devices that consume huge quantities of power (e.g. smartphones, connected vehicles). Hence, multiple types of devices would benefit from a technology that overcomes this issue.

As an example, sensor devices that capture data from purpose-built sensors (e.g. temperature, movement) have limited-size batteries. Thus, they often rely on MEC servers to

execute their data tasks, either because their local processors are not capable enough to perform them under some delay constraints, or because they need recharging often. Manufacturers of these types of devices optimize the local processing and sensing subsystems through the use of lightweight Operating Systems (or the use of MEC servers) [31]. However, what consumes the most amount of energy is the wireless communication subsystem, which rely on power control to ensure a reliable data transmission to the BS or access point.

Energy Harvesting was presented as a promising alternative to recharging batteries wirelessly and without intervention from the ambient environment (e.g. Solar) or from external sources (e.g. mechanical vibration), using specialized sensors that capture the energy and store it in batteries. This would allow for a green sustainable energy solution with less reliance on the existing electrical grid, as well as a more infrastructure-wise flexible design and deployment of devices.

1.2.1 Energy Harvesting Sources and Modeling

It exists different sources of energy harvesting that devices can utilize to recharge up their energy storage. According to [31], the sources are divided into two categories. The first category of sources comes from *ambient environment*, where the energy is collected free of charge from the environment. Examples of these types of sources are solar, radio frequency (RF), and wind. The second category comes from other *external sources*, that are deployed specifically to harvest energy from. Mechanical based sources such as vibration and pressure are examples of external sources.

Each type of energy harvesting source behaves differently than the other in the environment. Therefore, adequate modeling of each source, mathematically or computationally, is required to predict with accuracy future energy arrivals and build systems around it. Energy harvesting models are classified in [32] into three categories:

1. *Deterministic models*: The transmitter knows the amounts and the instants of energy arrival in advance. This modeling is useful in designing optimal energy strategies, and thus providing benchmarks to other sub-optimal approaches.

2. *Stochastic models*: The energy arrival is modeled as a random process. They can be time-uncorrelated models (e.g. Poisson process, uniform process) or time-correlated models (e.g. Markov-chain based process, Poisson-counting and non-negative uniform process).

3. *Other models*: They often model RF energy using different models and hybrid systems that utilize power supplies from the electrical grid, or RF energy harvesting to power their devices.

In what follows, we describe two ambient energy sources, i.e., Solar-based and RF-based, and some of their existing models in the literature that are associated with them.

Solar Sources

Solar energy, abundant in the environment, emerges as an affordable and eco-friendly solution that can address the aforementioned energy challenges. The use of the photovoltaic effect in the design of panels allows in the presence of sunlight, to transform the solar arrays into DC power. Due to the absence of such source at night, the focus is on maximizing the efficiency during the day to ensure power sustainability. This downside is what makes solar energy harvesting in indoor environment challenging. Though collecting energy for internal light sources is still possible but with lower efficiency. Diverse implementations of solar energy harvesting sensors exist. They differ based on solar panel types, battery specifications, and circuit complexities for recharging [31].

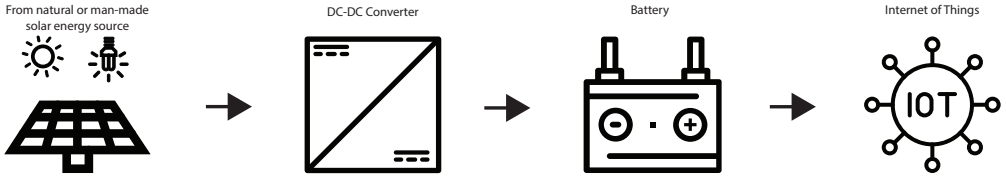

Figure 1.3: Solar energy conversion.

Modeling solar power produced some interesting works in the literature. For example, in [33], Real-world data are exploited to derive a Gaussian mixture hidden Markov model that quantifies the energy arrivals into different states. Measurements taken in 5-minute intervals were used to collect the data in different conditions (sunny, cloudy, at night, etc..). Then, they were utilized to build the hidden Markov chain, i.e., solar states that correspond to different energy arrivals, and their empirical transition probabilities.

Kraemer et.al. showed in their paper [34] that it is possible to predict solar energy variations with limited data thanks to Machine Learning (ML) techniques, such as Support Vector Machines (SVM), and Artificial Neural Networks (ANNs). Real-world data was used to assess the performance of ML prediction models.

Radio Frequency Sources

Radio Frequency-based energy harvesting converts ambient electromagnetic energy captured from radio waves into DC power, used either to recharge the device battery, or to run direct power to the device. Multiple factors play a role in the amount of harvested power, e.g. the distance between the source and the destination and transmission power. Typically, an RF energy harvesting device, such as a sensor node, can either use a single radio antenna for communicating and collecting energy, or use independent radio antennas for each purpose, and that depends on the size of the device and the complexity of the system. The conversion efficiency of RF to DC differs based on all these parameters [35].

RF energy can be utilized in a variety of applications, in both indoor and outdoor environments. These applications range from agriculture, smart homes, Radio Frequency IDentification (RFID), to monitoring. Indoor applications are especially beneficial compared to solar power, as the latter suffers substantially in such an environment [31].

Many works have been published to model RF energy sources. The authors in [36]

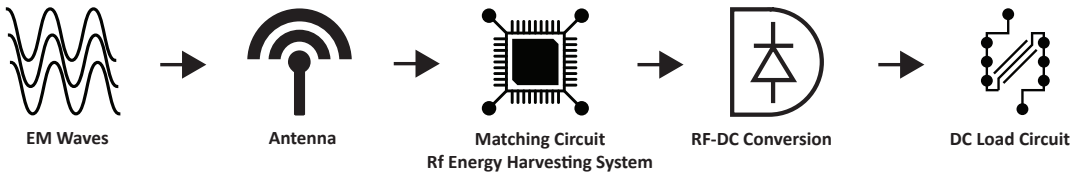

Figure 1.4: RF energy conversion.

studied the performance of RF energy harvesting wireless sensors in cellular networks by analyzing a point-to-point transmission between the sensors and a data sink. To geometrically model the distribution of the ambient energy sources, they used the Ginibre α-determinantal point process (DPP) [37] and derived the expected RF energy harvested at each sensor using the Friis equation [38]. Furthermore, upper-bounds were derived for both power outage probability and transmission outage probability.

The work in [39] utilized real-world measurements to model the RF energy harvested using Machine Learning techniques, namely Linear Regression and Decision Trees. First, harvested power was calculated from the collected measurements for each time slot (a maximum of 100 times slots in 2 hours was used), and for different frequency bins (a maximum of 192 bins are used per frequency band). Then, the computed powers were used as labels to train the ML models. The results showed that Linear Regression performs better than Decision Trees, by offering a more accurate prediction of RF energy.

In [40], a kernel density estimation technique was proposed to estimate the mobile service channels power density and a moving average-based prediction method. Six different frequency bands that correspond to different transmission standards were used to collect real-world data, in outdoor and indoor scenarios, at a rate of one data point per second. To model these data points, kernel density estimator was used, being considered as non-parametric and central estimator, as for the moving-average predictor, current point in time was computed by averaging n previous observations.

Discussion

The aforementioned models in both solar and RF energy harvesting, specifically the ones that rely on real-world data, suffer from sparsity in the data points collected. The time interval between the data points in the used databases varies from one second to minutes or hours, which is not suitable to our use case. For RF energy harvesting, the frequency ranges do not correspond to the ones used in our work. Therefore, we opted not to use them and subsequently use a stochastic process, similar to other works that investigated scheduling and allocation in wireless networks with EH [15], [19], [41], [42]. More specifically, we use in chapters 3 and 4 Poisson Random Process as our energy arrival model for its simplicity of use and its scalability.

1.3 Data Models

The growth in data transmitted, processed and stored across the internet as a whole is unprecedented. This can be explained by the increase in smartphone usage, deployment of

IoT devices, including connected objects. In addition to that, modern applications provide more advanced services, and thus consume more data. Data are collected from the users in order to recommend better content, display targeted ads, analyse the behavior and enhance the performance. Therefore, data has become the center of attention in today's networks and applications.

Studying and simulating systems that handle data in any way necessitates building a model of how the data behaves in such a system. In some resource scheduling scenarios, dedicated resources are assigned to users at certain times depending on the conditions of the environment, to perform their data-related tasks. By modeling the behavior of the data in this setup, the scheduling algorithm can predict the data quantity to handle in the next steps.

Data models are thus based on real-world measurements of applications and systems. Through the analysis of data behavior, a prediction can be made using these models of the next states of data is in the system. Exploiting the structure that generates the data is important to produce a better model. For instance, in [43] the authors utilized the video encoding schemes, based on the concept of Groups-of-Picture (GoP), to encode video frames based on the difference between them and an initial frame gathered in one GoP. This information was further used to build a Markov model that can be exploited to generate subsequently its own video traffic. The approach was proven to be encoder-independent, which makes it more general compared to other works in the field.

The work in [44] utilized real-world data from smartphone applications collected in the MIRAGE-2019 dataset [45] to build a hidden Markov model. The approach analyzed the packets transmitted from these applications in terms of payload length, inter-arrival time (considered as a hidden feature), and payload direction when building the model. The model is then used to predict the traffic size and direction of each studied application.

A survey on 5G scenarios was given in [46], including traffic models for the different standards adopted in 5G, mainly from ITU, 3GPP, IEEE and WiMAX. Traffic analysis and modeling was listed for each 5G standard, depending on the type of application/service. For example, FTP traffic in 3GPP was modeled as a truncated lognormal distribution for the file size, and exponential distribution for the reading time.

Discussion

The modelings brought up in the mentioned examples are very important to our use case. We can assume that the data arriving to the devices and need to be processed are modeled in terms of arrival quantity following these models. Further details on the used models are in the chapter 3 and chapter 4.

1.4 Non-Orthogonal Multiple Access

Fifth Generation (5G) of wireless cellular network promised great improvements over its 4G predecessor in terms of scalability of the number of users, connection speeds, and reliability. To reach such expectations, new standards and techniques were studied, designed and implemented for this generation and beyond. 5G support for massive IoT deployment,

delay sensitive applications, more bandwidth and connection speeds are enabled by Massive Machine Type Communications (MMTC), Ultra-Reliable Low Latency Communications (URLLC), and Enhanced Mobile BroadBand (EMBB) technologies, which are already deployed in the current generation.

To achieve URLLC and EMBB requirements, multiple access is an important technique to focus on and improve. By controlling the utilization of the spectral resources, an increase in data rates can be achieved. When using multiple access technique in wireless networks, it allows for the sharing of resources in time and frequency by multiple users when communicating with the BS.

Time division multiple access (TDMA), frequency division multiple access (FDMA), and code division multiple access (CDMA) are some of the existing and well-established techniques used in previous generations of cellular networks. They are classified as Orthogonal Multiple Access (OMA) techniques, which translates to the orthogonal nature of accessing the communication channel and minimizing the interference between the users. OMA techniques exploit network resources that are orthogonal, either in time (TDMA), frequency (FDMA), code domain (CDMA), or a combination of the previous ones (Orthogonal Frequency Division Multiple Access (OFDMA)), to allocate them for each user.

As an example, the bandwidth can be divided into several sub-carriers of equal size (a portion of the spectrum), and each sub-carrier is allocated to a user to send its signals through, via the communication channel. Multiple user signals are sent simultaneously in time but using the different sub-carriers. The decoding of each signal is simply done by considering each resource block on its own to decode an individual user's signal.

Nevertheless, with the minimal inter-user interference constraints, OMA techniques suffer from performance issues when the number of users that communicate increases. More specifically, the orthogonality aspect imposes that the users do not share any resource blocks (in time or frequency), and therefore the resources allocated for each user get smaller with every newly added user. This issue makes the use of OMA techniques challenging to satisfy 5G requirements in terms of rates and latency. Hence, Non-Orthogonal Multiple Access (NOMA) emerged as a solution to overcome this issue. The concept of NOMA in the context of 5G was initially introduced in [6], and its benefits were showcased in subsequent works [47], [48].

Although not adopted in 5G standards (OFDMA was kept as the multiple access technique of choice), NOMA remains a powerful technique to be considered in future generation wireless networks. More specifically, NOMA enables multiple users to share the same resource blocks, tolerating thus inter-user interference. To overcome the interference issue, Successive Interference Cancellation (SIC) decoding scheme is implemented.

NOMA can be divided into two categories; Code-domain NOMA, which acts in a similar way to classic CDMA, where each user is assigned a time-frequency block (a portion in bandwidth and time). Code-domain NOMA may differ from CDMA in the use of sparse spreading sequences, or non-orthogonal low cross-correlation sequences [49].

Power-domain NOMA is the second category, and the more interesting multiple access technique for our work. This type of NOMA superposes multiple user signals in time and frequency with power being the distinction at the receiver. The receiver then decodes the

signal using the SIC algorithm. We describe in what follows the Uplink (UL) and Downlink (DL) cases of NOMA transmission. Figure 1.5 shows the way multiplexing of users is done.

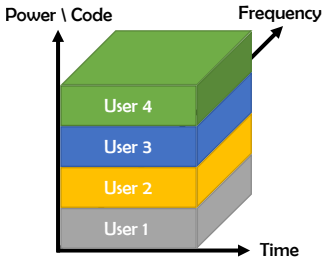

Figure 1.5: NOMA multiplexing.

1.4.1 Uplink NOMA

We consider as an example that two UEs, $\text{UE}^{(1)}$ and $\text{UE}^{(2)}$ are connected to a BS, and communicate with power-domain NOMA. The received signal $y^{(bs)}$ at the BS is expressed as:

$$y^{(bs)} = \sqrt{p^{(1)}} \cdot h^{(1)} \cdot s^{(1)} + \sqrt{p^{(2)}} \cdot h^{(2)} \cdot s^{(2)} + w^{(bs)}, \tag{1.1}$$

where $s^{(j)}$ is the signal transmitted from each UE, $p^{(j)}$ is the corresponding transmission power, and $w^{(bs)}$ is the Additive White Gaussian Noise (AWGN). In this configuration, $\text{UE}^{(1)}$'s signal will be the stronger one between the two. Figure 1.6 illustrates the SIC decoding process at the BS level, where the BS will be decoding in a cascading fashion, from the strongest signal to the weakest. In the example, the BS starts by decoding $\text{UE}^{(1)}$'s signal, i.e., $s^{(1)}$, while considering $s^{(2)}$ as interference. After that, the decoded signal $\hat{s}^{(1)}$ is subtracted from the received signal $y^{(bs)}$, and then the decoding of $s^{(2)}$ is done without any inter-user interference.

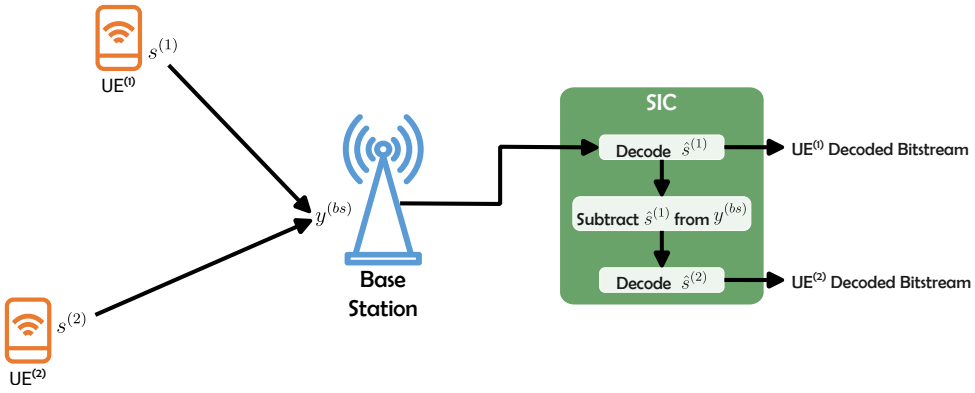

Figure 1.6: Uplink NOMA.

If the UEs channel gains are sufficiently distinct, they can utilize the full power to send their signals. However, if the channel gains are too similar, power control can be performed to boost the performance of the stronger UEs [50].

The rate expressions $\mathcal{R}_{noma}^{ul,(i)}$ for each UE, following the SIC operation at the BS, are

formulated as follows:

$$\mathcal{R}_{noma}^{ul,(1)} = \mathcal{W}^{ul} \cdot \log_2 \left(1 + \frac{p^{(1)} \cdot |h^{(1)}|^2}{p^{(2)} \cdot |h^{(2)}|^2 + \mathcal{W}^{ul} \cdot \mathcal{N}_0} \right), \tag{1.2}$$

$$\mathcal{R}_{noma}^{ul,(2)} = \mathcal{W}^{ul} \cdot \log_2 \left(1 + \frac{p^{(2)} \cdot |h^{(2)}|^2}{\mathcal{W}^{ul} \cdot \mathcal{N}_0} \right), \tag{1.3}$$

where $\mathcal{W}^{ul}$ is the UL bandwidth, and $\mathcal{N}_0$ is the noise spectral density.

1.4.2 Downlink NOMA

The DL case of NOMA follows a similar procedure to the one in the UL, with some differences. First, we assume that UE$^{(1)}$ has a better channel gain[1] than UE$^{(2)}$, $|h^{(1)}|^2 \geq |h^{(2)}|^2$. The SIC decoding is done at the UE level, in our case, UE$^{(1)}$ with its stronger channel gain. UE$^{(1)}$ proceeds into decoding using SIC, while UE$^{(2)}$ decodes its own signal directly. In the general case, the SIC decoding is done at all the UEs except for the weakest one, and the stronger the UE, the more signals from other UEs it has to decode. This is due to the fact that the BS allocates more power to weaker UEs in order to achieve reasonable rates. According to [50], the gain in performance when using NOMA is perceived if and only if the BS power allocation is inversely proportional to the UEs' channel gains.

The received signal at each UE in our example is expressed as:

$$y^{(1)} = h^{(1)} \cdot \sqrt{p^{(bs)}} \cdot s^{(bs)} + w^{(1)}, \tag{1.4}$$

$$y^{(2)} = h^{(2)} \cdot \sqrt{p^{(bs)}} \cdot s^{(bs)} + w^{(2)}, \tag{1.5}$$

where $s^{(bs)}$ is the signal broadcasted by the BS. Figure 1.7 illustrates the SIC decoding process at the user level, where UE$^{(1)}$ has to decode UE$^{(2)}$'s, subtract it from $y^{(1)}$, and then decode its own signal. For UE$^{(2)}$, the decoding is done directly while considering inter-user interference from UE$^{(1)}$, which is the case due to the weaker power allocated to the strong UE.

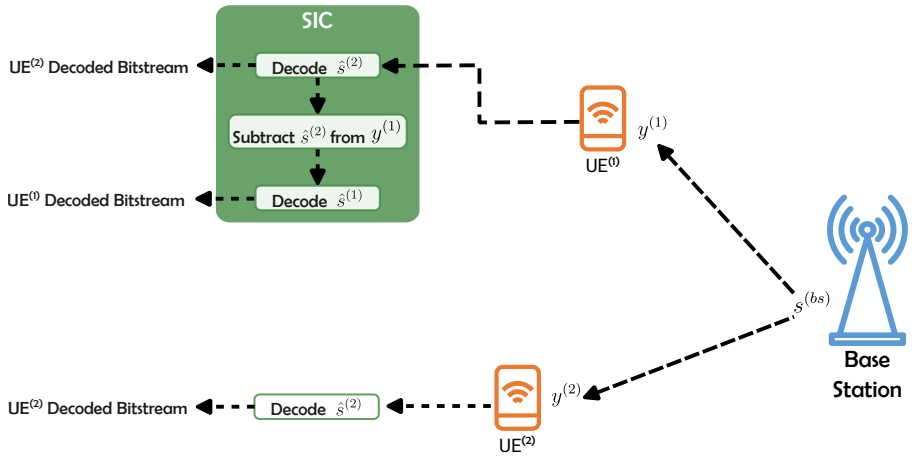

Figure 1.7: Downlink NOMA.

[1]This assumption is essential to get the rate expressions in equations 1.6 and 1.7 [51].

The resulting rate expressions with $\mathcal{W}^{dl}$ as the DL bandwidth, and δ as the power allocation coefficient, are:

$$\mathcal{R}_{noma}^{dl,(1)} = \mathcal{W}^{dl} \cdot \log_2 \left(1 + \frac{\delta p^{(bs)} \cdot |h^{(1)}|^2}{\mathcal{W}^{dl} \cdot \mathcal{N}_0} \right), \tag{1.6}$$

$$\mathcal{R}_{noma}^{dl,(2)} = \mathcal{W}^{dl} \cdot \log_2 \left(1 + \frac{(1-\delta) p^{(bs)} \cdot |h^{(2)}|^2}{\delta p^{(bs)} \cdot |h^{(2)}|^2 + \mathcal{W}^{dl} \cdot \mathcal{N}_0} \right). \tag{1.7}$$

The power allocation coefficient controls the portion of power dedicated to each UE. As explained previously, more power is usually allocated to weak UEs. In terms of achievability, Figure 1.8 showcases the region of rates for the two UEs, compared to the FDMA technique.

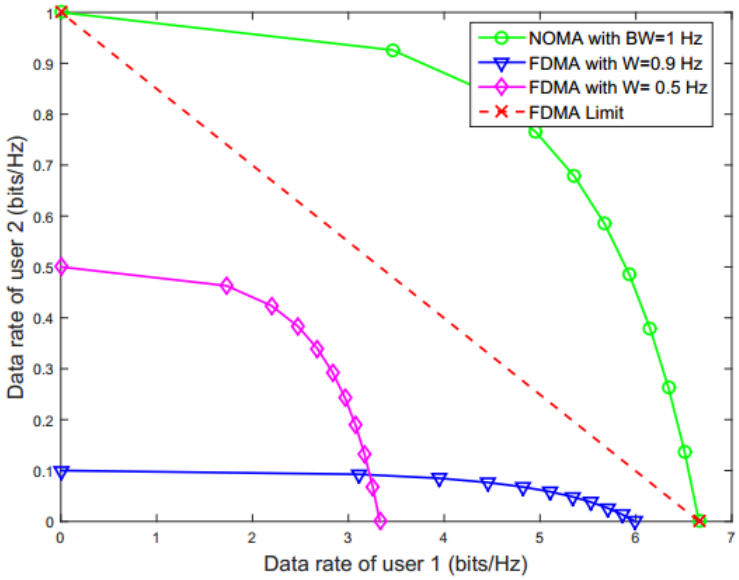

Figure 1.8: Downlink rate region for 2 UEs in NOMA and FDMA. Source: [49].

1.4.3 Clustering NOMA Users

Multiple issues can be highlighted from the equations above that can harm the performance advantage of NOMA. They are brought up to light when increasing the number UEs. One issue is the growing interference at the weaker UEs when decoding, which can degrade the performance badly. In addition to that, SIC can suffer from error propagation in practical systems, which can worsen the performance even more when adding more UEs. Another issue faced is the power allocation that may cause the transmission at low powers for weak UEs. These limitations are shown to reduce the sum rate of NOMA when increasing the number of UEs [52]. The solution can be to use multiple clusters with different bandwidths (or sub-carriers) and to group UEs in these clusters that can boost the performance of NOMA as a whole.

Formally, for n number of UEs connected to a BS, with $|h^{(i)}|^2 \geq |h^{(i+1)}|^2, \quad i \in$

$\{1, 2, \cdots, n\}$, the equations for UL and DL cases become as follow :

$$\mathcal{R}_{noma}^{ul,(i)} = \mathcal{W}^{ul} \cdot \log_2 \left(1 + \frac{p^{(i)} \cdot |h^{(i)}|^2}{\sum_{j=i+1}^{n} p^{(j)} \cdot |h^{(j)}|^2 + \mathcal{W}^{ul} \cdot \mathcal{N}_0}\right), \tag{1.8}$$

$$\mathcal{R}_{noma}^{dl,(i)} = \mathcal{W}^{dl} \cdot \log_2 \left(1 + \frac{\delta^{(i)} p^{(bs)} \cdot |h^{(i)}|^2}{\sum_{j=1}^{i-1} \delta^{(j)} p^{(bs)} \cdot |h^{(i)}|^2 + \mathcal{W}^{dl} \cdot \mathcal{N}_0}\right), \tag{1.9}$$

where $\delta^{(i)}$ is the allocation coefficients for each $\text{UE}^{(i)}$, with $\sum_{i=1}^{n} \delta^{(i)} = 1$. From these equations, the increasing inter-user interference is apparent for strong UEs in the UL case, and weak ones in the DL case. In particular, the increasing interference terms are an issue in the DL phase due to the limited computational capabilities of the UEs, especially in delay sensitive tasks. This issue further motivates the clustering of UEs for better performance.

Perhaps the most adequate method to cluster NOMA users is based on their channel gain, where depending on the number of users and allowed clusters, each cluster will contain a strong user and a weak user channel gain-wise. This will allow for better performance via power control [53].

Some works have been done on the clustering of NOMA users in the literature. For example, the work in [53] considered optimal clusters as the result of a joint optimization problem for global throughput maximization, and operation constraints for SIC decoding, under power and rate constraints. The derived solution is compared with the aforementioned channel-gain method, and DL NOMA performance was assessed with regard to the number of users per cluster. Jingjing et. al. [54] used K-means clustering method, to cluster NOMA users in Millimeter-wave systems. The clustering was done based on the spatial locations of the users, and could accommodate dynamic changes in the number of the clustered users. The authors in [55] proposed a system setup with the distinction between MMTC and URLLC devices. A NOMA clustering method was developed, where MMTC and URLLC devices are clustered together to avoid grouping multiple MMTC devices together. The clustering process also considered intra-cluster interference, transmission power and QoS requirements. The work in [56] designed an adaptive strategy for clustering EMBB and IoT users aiming to maximize the use of NOMA and optimize the used power. In the proposed scenario, NOMA was employed selectively by comparing its performance to OMA. The introduced clustering and power optimization algorithm proved more efficient than non-adaptive multiple access techniques and other clustering solutions.

Discussion

Analyzing the existing clustering methods, we can make the observation that combining spatial and channel-gain based clustering methods can be a good unexplored solution. It can maximize the performance of NOMA in the network, while considering the spatial proximity of the users. Following these observations, we propose a clustering method in chapter 4.

CHAPTER 2

Learning-Based Preliminaries

NOWADAYS, AI has quickly become the center of attention due to its impressive abilities to handle information and learn from collected data. Various applications ranging from health to robotics and communications have demonstrated the great potential of using AI in designing systems. AI can provide accurate predictions, navigate cleverly around complex environments, or answer questions in the most human possible way , following the breakthrough of Large Language Models (LLMs) like ChatGPT. Yet at the core of AI resides a multitude of statistical techniques that were used for a long time (e.g. regression). Indeed, the interesting combinations and additions of some tools (e.g. convolutions, attention mechanisms) in the AI models is what makes the resulting algorithms as powerful as they are now.

The main idea of AI is to produce models that learn from provided data in a certain way suited for the desired task. These models can be used for inference, prediction, classification, decision-making, and many other potential applications and use cases. Naturally, AI is divided into several branches that cover the different tasks mentioned. Figure 2.1 covers some of the branches.

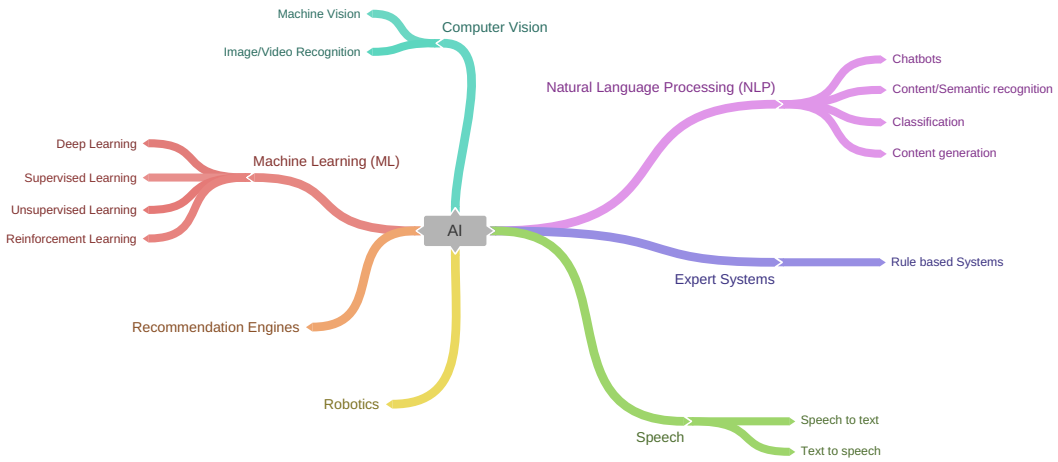

Figure 2.1: Artificial intelligence AI branches.

Machine Learning (ML) in particular has been one of AI's main areas that made it as successful as it is today. ML is concerned with building systems that improve the performance on a given task given examples of how the data is structured, or what the results should look like with certain inputs. It is subdivided into three types of learning, namely *supervised learning*, *unsupervised learning*, and *reinforcement learning*.

Supervised learning uses labeled data to learn the model. The dataset has a set of input $\mathbf{x}$ and corresponding output examples $\mathbf{y}$. Given an input x_t, the supervised learning model $\mathcal{F}^{\mathbf{w}}$ with parameters $\mathbf{w}$ outputs a prediction $\hat{y}_t = \mathcal{F}^{\mathbf{w}}(x_t)$, and a loss value $\mathcal{L}(y_t, \hat{y}_t)$ is computed using the original label. The model then updates its parameters in an iterative way to reduce the loss value until reaching a convergence point where the difference between the model output and the desired one is close [57]. Following the training process, a separate test data is used to assess the performance of the produced model and whether or not it can generalize to never seen before examples.

Supervised Learning can also be divided into 2 subcategories:

- *Regression*, where the prediction function gives a numerical approximation of a certain task in a continuous space. For example, modeling a random distribution of points with a polynomial function and predicting new points' positions.
- *Classification*, which is concerned with predicting a specific class of the output in a discrete space. For example, predicting the type of a captured traffic sign.

On the other hand, unsupervised learning models operate differently than supervised ones. The main goal is to learn the internal representation of the input data to further analyze and understand its structure. This approach helps uncover hidden correlations and patterns in the provided data and serve as a prediction model to label any new data that arrives. Clustering and dimensionality reduction are examples of such learning models.

As for Reinforcement Learning, section 2.2 details its structure and learning procedure, as well as some algorithms in the literature. The rest of the chapter explores Neural Networks in section 2.1, and an overview of Federated Learning is given in section 2.3.

2.1 Neural Networks

Perhaps one of the most used techniques of ML in its different categories are Artificial Neural Networks (ANNs), also referred to as just Neural Networks (NNs). This brain-inspired idea relies on the weighted interconnection of different nodes together in the form of layers to form a complex graph. The weights of the connections between each neuron is what is learned during training the machine learning model, yielding to a model that can perform the required task. The structure of NNs is organized in 3 parts, namely the input layer, hidden layer(s) and an output layer, as shown in Figure 2.2.

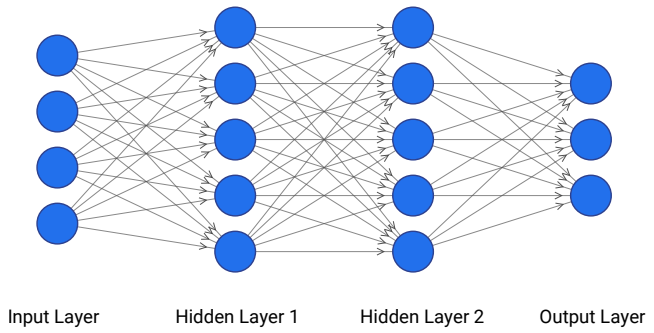

Figure 2.2: Artificial neural network ANN example.

Formally, we consider a NN with n layers, with each layer i containing d_i neurons. We model the connections weights between layer $i-1$ and layer i as a matrix $\mathbf{w}_i \in \mathbb{R}^{d_i \times d_{i-1}}$. Given an input vector $\mathbf{x}_{i-1} \in \mathbb{R}^{d_{i-1}}$, produced from layer $i-1$, we output the vector $\mathbf{x}_i \in \mathbb{R}^{d_i}$ as follows:

$$\mathbf{x}_i = \sigma\left(\mathbf{w}_i \times \mathbf{x}_{i-1} + \mathbf{b}_i\right), \tag{2.1}$$

where $\mathbf{b}_i \in \mathbb{R}^{d_i}$ is the scalar bias vector for layer i, and σ is an activation function, used to introduce non-linearity to the forward pass. The described equation is used to compute the output of each layer, until reaching the final output layer.

Training an NN model is performed using the back-propagation technique. It takes the loss computed between the produced output and the desired one, and propagates it throughout the network's weights. In fact, it computes small steps of corrections in the opposite direction of the loss using gradient-descent optimization. This helps the model to converge to a local minimum that can satisfy the accuracy and generalization requirements (i.e., validation/testing accuracy with new data).

Given a loss $\mathcal{L}$, the update formula for the weight of neuron j of i^{th} layer $w_{i,j}$ is:

$$w_{i,j} \leftarrow w_{i,j} - \alpha \nabla w_{i,j}, \tag{2.2}$$

with $\nabla w_{i,j} = \frac{\partial \mathcal{L}}{\partial w_{i,j}}$ being the gradient of the loss with respect to the weight $w_{i,j}$. Intuitively, the weights are updated in the opposite direction to their gradients. The update is multiplied by a factor α called the *learning rate*, that limits its size and thus its impact on the weights. This is done to avoid the issues of divergence or overshooting from the local/global minimum. The update step is done repeatedly with different input examples, until convergence.

Neural Networks construct the essential building block for Deep Learning since deep neural networks are simply neural nets with two or more hidden layers. However, deep learning encompasses more than this basic learning structure. Many architectures and designs that have been proposed are optimized to specific tasks. For instance, Convolutional Neural Networks (CNNs) [58] and Recurrent Neural Networks (RNNs) [59] were built respectively for visual and textual/audio data. Lately, deep learning has been an area of focus in Artificial Intelligence generally and Machine Learning specifically due to its ability to extract automatically relevant features for the required tasks.

2.2 Reinforcement Learning

Reinforcement Learning (RL) can be considered as a semi-supervised technique of ML that resolves sequential decision-making problems. In RL, an agent is trained in an environment to learn its dynamics and to produce a model that takes decisions or actions based on observations of that environment. What differs from supervised learning techniques is the feedback from the system. Instead of having labels that tell the agent exactly what the prediction should be, a reward/cost is given by the environment to estimate the quality of the action taken, in the short term or the long term.

To better understand RL problems, we can formalize them into Markov Decision Process (MDP) problems. We describe in the following the Markov modeling of the environment, then we present the Markov model-based and Markov model-free RL techniques.

2.2.1 Markov Modeling

The environment in an RL problem is a set of different representations of a system, namely *states* **S**. These states provide a sufficient description of the system. To model the transition and interaction between such states, Markov modeling is a powerful tool.

2.2.1.1 Markov Process

Markov Process or Markov Chain (MC) is a stochastic model that describes the relation between the different states of the environment. The main aspect is the Markov property, dictating that the occurrence probability of a state S_t only depends on the previous state S_{t-1}.

$$p(S_t = s_t|S_{t-1} = s_{t-1}, S_{t-2} = s_{t-2}, \cdots, S_1 = s_1) = p(S_t = s_t|S_{t-1} = s_{t-1}). \tag{2.3}$$

Accordingly, a chain can be constructed to model the interactions between the states and the corresponding probabilities of transition. Figure 2.3 illustrates a Markov Chain with 3 states $\mathbf{S} = \{s_1, s_2, s_3\}$.

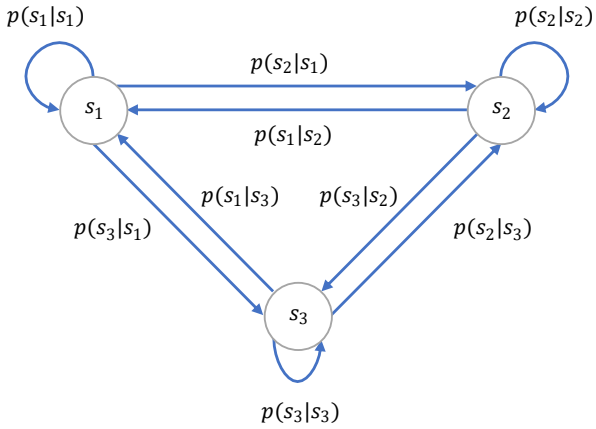

Figure 2.3: Markov chain example with 3 states.

A matrix $\mathbf{T}$ can also be computed to store the transition probabilities between the states. The sum of probabilities for each row of the matrix is $\sum_{j=1}^{n} p(s_j|s_i) = 1, \quad i \in [1, n]$.

$$\mathbf{T} = \begin{bmatrix} p(s_1|s_1) & p(s_2|s_1) & \cdots & p(s_n|s_1) \\ p(s_1|s_2) & p(s_2|s_2) & \cdots & p(s_n|s_2) \\ & \cdots & & \\ p(s_1|s_n) & p(s_2|s_{n-1}) & \cdots & p(s_n|s_n) \end{bmatrix}. \tag{2.4}$$

2.2.1.2 Markov Reward Process

Markov Reward Process (MRP) is a Markov Process that assigns a reward value for each state in the environment. We describe the reward model by a vector $\mathbf{R} = [R_1, R_2, \cdots]$, a function that outputs a reward for every state in the MRP. Figure 2.4 illustrates an MRP with reward associated to each state $\mathbf{R} = [R_1, R_2, R_3]$.

2.2.1.3 Markov Decision Process

Markov Decision Process (MDP) adds a set of actions $\mathbf{A} = [a_1, a_2, \cdots]$ to the MRP. The agent can take these actions from given states and the environment transitions to new states, with a given probability of transitioning. The transition matrix $\mathbf{T}$ becomes a 3D

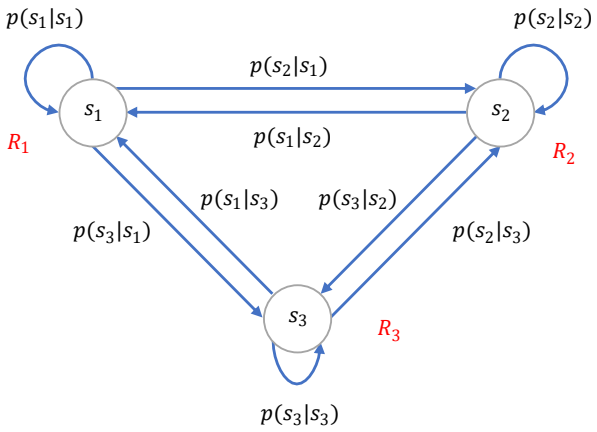

Figure 2.4: Markov reward process example.

tensor for the probabilities for each state, next state and action $p(s'|s,a)$. Figure 2.5 illustrates an example of an MDP, where for three states $\mathbf{S} = \{s_1, s_2, s_3\}$, two actions are available $\mathbf{A} = \{a_0, a_1\}$.

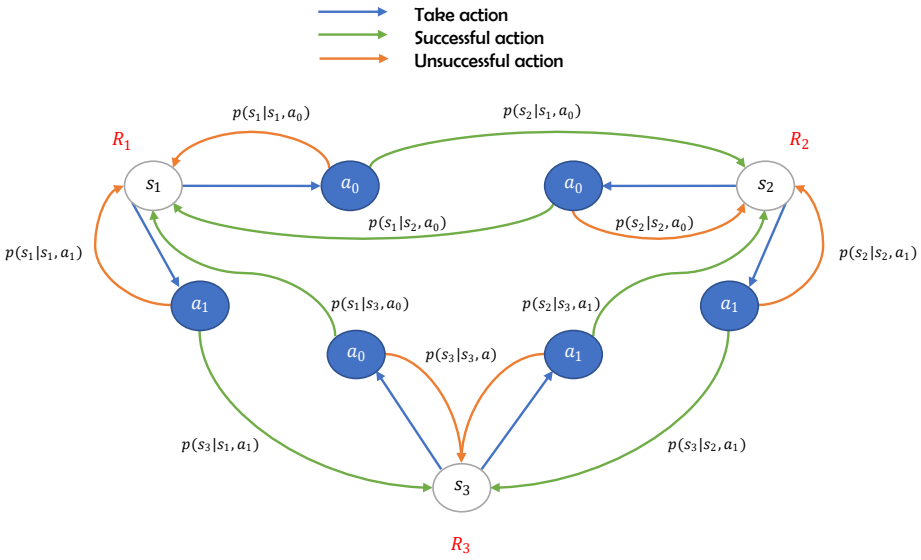

Figure 2.5: Markov decision process example.

With the modeling of an MDP $(\mathbf{S}, \mathbf{A}, \mathbf{R}, \mathbf{T})$, we have a complete description of the environment. The RL agent tries to learn the environment dynamics and understand the transition between the states by taking the appropriate actions, using the reward feedback from the environment. Figure 2.6 illustrates the RL agent interaction with the environment.

2.2.2 MDP-related functions

In order to navigate the environment for a certain duration in the best possible way, the agent's goal is to take the actions that transition to the states with the highest reward. Therefore, it needs to find the best policy π that dictates what action to take at each state by maximizing the cumulative reward.

The policy π is a probability distribution of actions $a \in \mathbf{A}$ from a state $s \in \mathbf{S}$. We

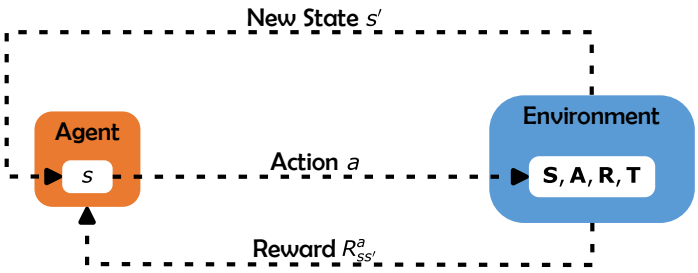

Figure 2.6: RL agent interaction with the environment.

assume that the reward for each state depends on the action taken, as well as the next state $R^a_{ss'}$, i.e., the reward value is received after transitioning to a new state s', when starting from state s and taking an action a. The policy π can be deterministic in the sense that each state has a corresponding best action. To assess such policy, we can compute the expected reward (return) R^π following the policy π during k number of transitions between states (or timesteps):

$$R^\pi = \mathbb{E}^\pi \left[R_0 + \pi(a|s) \cdot \sum_{t=1}^{k} R^a_{ss'} \cdot p(S_t = s' | S_{t-1} = s, A_{t-1} = a) \right], \tag{2.5}$$

where R_0 is the initial reward when starting from state $S_{t=0}$. When the number of transitions k is finite, the duration is called an *episode*, and the MDP is in the finite-horizon case. Whereas for the MDP infinite-horizon case, the number of transitions tends to infinity $k \to \infty$. In the latter case, we use a discount factor $0 < \gamma \leq 1$ that limits the expected reward to avoid having a reward that diverges to infinity. The reward expression is then[1]:

$$R^\pi = \mathbb{E}^\pi \left[R_0 + \pi(a|s) \cdot \sum_{t=1}^{k \to \infty} (\gamma)^i R^a_{ss'} \cdot p(S_t = s' | S_{t-1} = s, A_{t-1} = a) \right]. \tag{2.6}$$

Adjusting the value of the discount factor controls how much the future rewards, states and action impact the expected reward. When $\gamma = 1$, all the future rewards are considered equally, whereas for γ values close to 0, the expected return will be heavily biased towards immediate rewards. Obviously, the aim is to maximize the reward over many episodes[2]. We consider the optimal policy $\pi^\star$ is the one that achieves this:

$$\pi^\star = \arg\max_\pi R^\pi. \tag{2.7}$$

The optimal policy can be found using direct analytical expressions that produce an exact solution to the problem. However, with a environment being more complex, optimization techniques are needed to converge to the solution in an iterative way. This is where training an RL agent becomes essential.

To describe if it is good or bad to be in the environment states, and obtain the optimal policy, a set of functions are introduced. State value function (Value function for short),

[1] In what follows, only the infinite-horizon case is considered.

[2] Since the infinite horizon MDP does not have an episode end, we define it simply as a certain number of transitions before resetting the initial state

State-Action Value function (Q-function) and Advantage function are examples that serve this purpose.

Value function is simply the discounted sum of instantaneous rewards R_t following a policy π, starting from a state $S_t = s$:

$$V^{\pi}(S_t = s) = \mathbb{E}^{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s]. \tag{2.8}$$

In other terms, the value function estimates the quality of being in a state s, while following the policy π.
state-action value function, on the other hand, is the discounted sum of instantaneous rewards R_t following a policy π, starting from a state $S_t = s$ and taking an action $A_t = a$:

$$Q^{\pi}(S_t = s, A_t = a) = \mathbb{E}^{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s, A_t = a]. \tag{2.9}$$

Similarly to the value function, Q-function evaluates the quality of being in a state s and taking an action a. For a finite number of states and actions, we obtain a vector representing the value function $\mathbf{V}^{\pi} \in \mathbb{R}^{|\mathbf{S}|}$, and a matrix for the Q-function $\mathbf{Q}^{\pi} \in \mathbb{R}^{|\mathbf{S}| \times |\mathbf{A}|}$, following the policy π, with $|\mathbf{S}|$ and $|\mathbf{A}|$ representing the state space and action space sizes, respectively.
Accordingly, the optimal policy $\pi^{\star}$ is obtained by taking the sequences of actions that maximize the value function for all states or the Q-function for all state-action pairs:

$$V^{\star}(s) = \max_{\pi} V^{\pi}(s) = V^{\pi^{\star}}(s), \ \forall s \in \mathbf{S}, \tag{2.10}$$

$$Q^{\star}(s, a) = \max_{\pi} Q^{\pi}(s, a) = Q^{\pi^{\star}}(s, a), \ \forall s \in \mathbf{S}, \ \forall a \in \mathbf{A}. \tag{2.11}$$

Bellman provided recursive forms to evaluate both functions $V^{\pi}(s)$ and $Q^{\pi}(s, a)$. The Bellman expectation equations are expressed as:

$$V^{\pi}(s) = \sum_{a \in \mathbf{A}} \pi(a|s) \sum_{s' \in \mathbf{S}} p(s'|s, a) \Big(R^{a}_{ss'} + \gamma V^{\pi}(s') \Big), \tag{2.12}$$

$$Q^{\pi}(s, a) = \sum_{s' \in \mathbf{S}} p(s'|s, a) \Big(R^{a}_{ss'} + \gamma \sum_{a' \in \mathbf{A}} \pi(a'|s') \cdot Q^{\pi}(s', a') \Big). \tag{2.13}$$

Notably, there is a dependence between the current value of a state s, and the value of its successor state s'. Dynamic programming algorithms can benefit greatly from this recursive form to derive an iterative solution that improves the equations constantly. In addition, a relation can be derived between the value function and the Q-function, shown in Equation 2.14 and Equation 2.15. Moreover, Figure 2.7 shows this relation where the left structure represents the value function is different Q-functions for each action taken from the policy $\pi(a|s)$, while the Q-function on the right is a sum of value functions after taking an action a and receiving a reward $R^{a}_{ss'}$:

$$V^{\pi}(s) = \sum_{a \in \mathbf{A}} \pi(a|s) \cdot Q^{\pi}(s, a), \tag{2.14}$$

$$Q^{\pi}(s, a) = R^{a}_{ss'} + \gamma \sum_{s' \in \mathbf{S}} p(s'|s, a) \cdot V^{\pi}(s'). \tag{2.15}$$

Figure 2.7: Relationship between the value function and the Q-function.

The advantage function can be defined as the difference between the Q-function and the value function. Intuitively, it can indicate whether the taken action a outputs a reward value better than the expected one following the policy π:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s). \tag{2.16}$$

It can be written as function of the value function and the reward obtained by taking the action a in state s and transitioning to state s', denoted as $R^a_{ss'}$:

$$A^\pi(s, a) = R^a_{ss'} + \gamma V^\pi(s') - V^\pi(s). \tag{2.17}$$

Optimal Value and Q-functions are defined by the Bellman optimality equations:

$$V^\star(s) = \max_{a \in \mathcal{A}} \left(\sum_{s' \in \mathbf{S}} p(s'|s, a) \left(R^a_{ss'} + \gamma V^\star(s') \right) \right), \tag{2.18}$$

$$Q^\star(s, a) = \sum_{s' \in \mathbf{S}} p(s'|s, a) \left(R^a_{ss'} + \gamma \cdot \max_{a' \in \mathbf{A}} Q^\star(s', a') \right). \tag{2.19}$$

They yield a deterministic policy expressed as:

$$\pi^\star(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in \mathbf{A}} \sum_{s' \in \mathbf{S}} p(s'|s, a) \left(R^a_{ss'} + \gamma V^\star(s') \right) \\ 0 & \text{otherwise} \end{cases}, \tag{2.20}$$

$$\pi^\star(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in \mathbf{A}} Q^\star(s, a) \\ 0 & \text{otherwise} \end{cases}. \tag{2.21}$$

2.2.3 Model-based Methods

In this section, we discuss model-based methods for finding the optimal policy to navigate the environment with the highest reward. Model-based refers to the full knowledge of the transition matrix $\mathbf{T}$ and the reward model $\mathbf{R}$ of the MDP representing the environment. The convergence of these methods is guaranteed, making them optimal.

2.2.3.1 Value Iteration

The first optimal method is Value Iteration (VI). The principle of VI is to incrementally improve a randomly initialized value function in the form of a vector $\mathbf{V}_0(s)$. It applies

iteratively the Bellman optimality equation until converging to the optimal solution with some stopping criteria. It is described by:

$$\mathbf{V}_k(s) = \max_{a \in \mathbf{A}} \left(\sum_{s \in \mathbf{S}} p(s'|s,a) \left(R^a_{ss'} + \gamma \mathbf{V}_{k-1}(s') \right) \right), \tag{2.22}$$

where $\mathbf{V}_k(s)$ takes the maximum value possible by an action a for state s at iteration k. It depends on $\mathbf{V}_{k-1}(s')$, the optimal value of successor state s' obtained at the previous iteration $k-1$.

2.2.3.2 Policy Iteration

The second optimal method is Policy Iteration (PI). This algorithm runs two phases iteratively until convergence: Policy Evaluation and Policy Improvement. The first phase evaluates the value function at iteration k using the value function vector $\mathbf{V}_{k-1}(s)$ and the policy $\pi_{k-1}(a|s)$ obtained in the previous iteration $k-1$. It is described as:

$$\mathbf{V}_k(s) = \sum_{a \in \mathbf{A}} \pi_{k-1}(a|s) \left(\sum_{s \in \mathbf{S}} p(s'|s,a) \left(R^a_{ss'} + \gamma \mathbf{V}_{k-1}(s') \right) \right). \tag{2.23}$$

The second phase improves the policy $\pi_k(a|s)$ by choosing the actions a that maximize the value function at each state s according to:

$$\pi_k(a|s) = \begin{cases} 1 & \text{if } a = \arg\max\limits_{a \in \mathbf{A}} \sum_{s' \in \mathbf{S}} p(s'|s,a) \left(R^a_{ss'} + \gamma \mathbf{V}_k(s') \right) \\ 0 & \text{otherwise} \end{cases}. \tag{2.24}$$

The algorithm converges when the policy stops changing, i.e., $\pi^\star = \pi_k = \pi_{k-1}$.

2.2.4 Model-free Methods

In contrast to model-based, model-free methods have no knowledge of the dynamics of the environment, i.e., $\mathbf{T}$ and $\mathbf{R}$, but only the state space $\mathbf{S}$ and the action space $\mathbf{A}$. This can happen in most RL problems where the environment is complex, or where it is computationally hard to model the dynamics of their environments. In this case, the agent attempts to learn the optimal policy by exploring the environment, i.e., by taking different actions and getting feedback through rewards. This is denoted as a "trial-and-error' approach.

On one side, exploring the environment indefinitely does not necessarily yield to an optimal policy. On the other side, only attempting to take the best action at each state without exploring other actions and states in the environment is not a good approach neither. Therefore, a balance between exploration and exploitation of the obtained information is needed for the agent to optimize its actions and produce a good policy.

2.2.4.1 Q-Learning

Q-Learning (QL) [60] is a widely popular model-free algorithm that tries to understand the environment by filling its Q-function in matrix form $\mathbf{Q}$ when navigating samples of

states s and taking actions a from those states of the environment. QL learns to assess the quality of the action taken at each state to update the Q-matrix $\mathbf{Q}(s,a)$ in small steps iteratively.

At any step, the action taken a at state s is decided by an ϵ-Greedy algorithm. It chooses the best action that gives the highest reward with probability $(1-\epsilon)$ (this corresponds to *exploitation*), and chooses a random action with probability ϵ to explore the action (*exploration*). Updating the matrix $\mathbf{Q}$ at any iteration is expressed as:

$$\mathbf{Q}_{new}(s,a) = \mathbf{Q}_{old}(s,a) + \alpha \cdot \underbrace{\Big(R^a_{ss'} + \gamma \cdot \max_{a' \in \mathbf{A}} \mathbf{Q}_{old}(s',a') - \mathbf{Q}_{old}(s,a)\Big)}_{\text{Temporal Difference}}, \tag{2.25}$$

where α is the *learning rate*, a scaling factor to limit the size of the update for the Q-matrix. The Temporal Difference (TD) highlighted in the equation above is a measurement of the difference (error) between the Q-matrix value $\mathbf{Q}(s,a)$ and its Bellman equation estimation when sampling an action and a next state, as shown previously in equation 2.13. By progressively exploring and exploiting the environment, the agent fills up the Q-matrix and learns a policy. QL has a guarantee of convergence with enough iterations if the set of state-action pairs are sufficiently visited.

2.2.4.2 Double Q-Learning

One of the issues of the Q-learning algorithm is the overestimation of its Q-matrix values, otherwise known as the maximization bias. The problem happens when updating the matrix with new values where the actions that yield good rewards, and thus a bigger TD, explode the corresponding state-action values in the Q-matrix $\mathbf{Q}_{new}(s,a)$. This is due to the use of the max operation on the estimates of the Q-function, rather than the exact ones. The constant occurrence of such phenomenon slows down the convergence of the algorithm, and does not produce good estimations of the Q-function [61].

To enhance the results of Q-Learning and speed up the convergence, the authors in [62] propose Double Q-Learning (2QL). Instead of updating one Q-matrix estimation of the true Q-function, 2QL algorithm trains with two matrices $\mathbf{Q}^A(s,a)$ and $\mathbf{Q}^B(s,a)$. When updating the value of one Q-matrix, it uses its values for selecting the next best action, and used the second for estimating the state-action value for the selected next action. The updating process is detailed in what follows, where $N = \{A, B\}$ is the index of the Q-matrix to be updated, and the one that selects the next best action, while $\bar{N}$ is the index of the Q-matrix that performs the estimation of the Q-value of said action:

$$\mathbf{Q}^N_{new}(s,a) = \mathbf{Q}^N_{old}(s,a) + \alpha \cdot \Big(R^a_{ss'} + \gamma \cdot \mathbf{Q}^{\bar{N}}_{old}(s', \arg\max_{a' \in \mathbf{A}} Q^N(s',a')) - \mathbf{Q}^N_{old}(s,a)\Big), \tag{2.26}$$

where the term $\arg\max_{a' \in \mathbf{A}} \mathbf{Q}^N(s',a')$ represents the action chosen from the matrix $\mathbf{Q}^N$ and used for evaluating the state-action value of $\mathbf{Q}^{\bar{N}}$. 2QL algorithm randomly chooses a Q-matrix between the two to update at each timestep, and averages both matrices to obtain a final one upon convergence.

2.2.4.3 Deep Q-Learning

When the state and actions spaces grow larger, modeling them using Q-matrices can be memory consuming. In addition, the time required for exploring the state space sufficiently to determine what action to take for each circumstance increases, and thus making it harder for QL and 2QL to converge on a policy. Therefore, Deep Q-Learning [63] or Deep Q-Network (DQN) was introduced to tackle the curse of dimensionality that traditional algorithms face, by using NNs to model the environment rather than matrices. Thanks to the generalization abilities of NNs, DQN is able to approximate the Q-matrix using a function of parameters $\mathbf{w}$, even without exploring the entirety of the state space $\mathbf{S}$.

Specifically, a given state s is fed into the DQN's Neural Network of parameters $\mathbf{w}$. The output is an estimation of the Q-values denoted as $\mathbf{Q}^{\mathbf{w}}(s,a)$, which indicates the state-action values for the state s. An action a is chosen based on the obtained Q-values (in training, the action is sampled following the ϵ-Greedy method, whereas during the test, the action is the one that achieves the highest Q-value) and the environment transitions to a new state s', with a reward $R^a_{ss'}$, as illustrated in Figure 2.8.

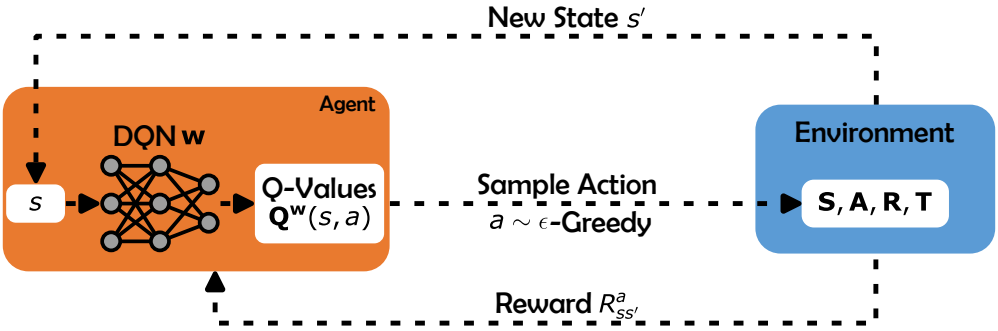

Figure 2.8: DQN agent exploring the environment.

To train the DQN agent, a loss function $\mathcal{L}^{\mathbf{w}}_{DQN}$ is used to evaluate the quality of the produced Q-values. More specifically, the estimate $\mathbf{Q}^{\mathbf{w}}(s,a)$ for a state s, is used to take an action a, transition to a next state s' and receive a reward $R^a_{ss'}$. The result is then compared with the Bellman equation of the Q-function using the DQN Q-values estimation of the next state $\mathbf{Q}^{\mathbf{w}}(s',a')$, expressed in the following equation:

$$\hat{\mathbf{Q}}^{\mathbf{w}}(s,a) = R^a_{ss'} + \gamma \cdot \max_{a' \in \mathbf{A}} \mathbf{Q}^{\mathbf{w}}(s',a'). \tag{2.27}$$

From that, the loss function computes the error between the two estimates $\mathbf{Q}^{\mathbf{w}}(s,a)$ and $\hat{\mathbf{Q}}^{\mathbf{w}}(s,a)$, and the DQN agent uses that to update its parameters $\mathbf{w}$ with Gradient Descent, and improve its estimation quality.
However, using the parameters $\mathbf{w}$, initially used to estimate $\mathbf{Q}^{\mathbf{w}}(s,a)$, to compute the Q-values of the next state $\mathbf{Q}^{\mathbf{w}}(s',a')$ when evaluating the Bellman estimation $\hat{\mathbf{Q}}^{\mathbf{w}}(s,a)$, to later update the same parameters can cause the issue of a moving target, i.e., the convergence point is constantly moving and reaching it becomes more difficult. To combat this issue, a second NN can be used with different parameters $\mathbf{w}'$. This network will be used to compute the Bellman equation and offer more stability to the training of the parameters $\mathbf{w}$ of the DQN agent. Notably, the first NN is named the Evaluation Network (EN), while

the second NN is named the Target Network (TN). To avoid training two separate NNs, the TN parameters $\mathbf{w}'$ will be a copy of the EN parameters $\mathbf{w}$ that do not get updated as often. More specifically, for a certain number of episodes, the parameters $\mathbf{w}'$ are fixed to allow the parameters $\mathbf{w}$ to converge towards a fixed target. After that, an update on the TN parameters is done to make them equal to the EN parameters $\mathbf{w}' = \mathbf{w}$. This is done to improve the Bellman estimation using an updated version of the Q-values $\mathbf{Q}^{\mathbf{w}'}(s, a)$. The process is repeated until convergence.

Moreover, the sequential nature of MDP problems creates a correlation in the observed states used to train the DQN. If the system is unstable, the correlation aspect can yield to high variance updates of the agent's parameters, and therefore impact the performance and convergence of the DQN [61]. From that, DQN uses Experience Replay (ER), a technique implemented to train the agent more efficiently, and with more stability. ER consists of collecting the transitions of the agent in the environment, and storing them in a memory buffer. This memory buffer is then used to replay the transitions played in a randomized way (to break the correlation), and compute the loss function. Following this method, training the DQN agent becomes more stable against the variations in the transitions of the environment.

In what follows, we detail the training procedure of the DQN:

1. Starting with an initial state $S_t = s$ at timestep t, we compute the Q-values $\mathbf{Q}^{\mathbf{w}}(s, a)$ using the EN, choose an action $A_t = a$ following the ϵ-Greedy approach, transition to a new state $S_{t+1} = s'$ and receive a reward $R^a_{ss'}$. The tuple $\left(s, \mathbf{Q}^{\mathbf{w}}(s, a), s', R^a_{ss'}\right)$ is saved in the ER memory buffer, and the system transitions to a new timestep. The next state s' becomes the current state s and the process is repeated until the end of the episode, where a new initial state is used to start a new episode.

2. Once the memory buffer is filled with a sufficient number of tuples to form a mini-batch, the learning process starts, in which the sampled tuples are shuffled to break the correlation between the transitions to form a mini-batch.

3. The obtained mini-batch, denoted as b and of size $|b|$, is then used to compute the Bellman step using the TN for each tuple $\left(s_i, \mathbf{Q}^{\mathbf{w}}_i(s_i, a_i), s'_i, R^a_{ss',i}\right)$ of index i in the mini-batch, following the equation:

$$\hat{\mathbf{Q}}^{\mathbf{w}'}_i(s_i, a_i) = R^a_{ss',i} + \gamma \cdot \max_{a' \in \mathbf{A}} \mathbf{Q}^{\mathbf{w}'}_i(s'_i, a'). \tag{2.28}$$

4. The obtained Bellman estimations of the mini-batch b are then used to compute the loss $\mathcal{L}^{\mathbf{w},\mathbf{w}',b}_{DQN}$, which is the Mean Squared Error (MSE) between the Bellman estimates and the Q-values outputs of the EN. The loss function is expressed as follows:

$$\mathcal{L}^{\mathbf{w},\mathbf{w}',b}_{DQN} = \frac{1}{|b|} \sum_{i=1}^{|b|} \left(\hat{\mathbf{Q}}^{\mathbf{w}'}_i(s_i, a_i) - \mathbf{Q}^{\mathbf{w}}_i(s_i, a_i)\right)^2. \tag{2.29}$$

5. The parameters $\mathbf{w}$ are then updated using the loss $\mathcal{L}^{\mathbf{w},\mathbf{w}',b}_{DQN}$ following the Gradient Descent method with a learning rate α:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} \mathcal{L}^{\mathbf{w},\mathbf{w}',b}_{DQN}. \tag{2.30}$$

6. The parameters $\mathbf{w}'$ of the TN are updated every few episodes, to be the same as $\mathbf{w}$. The process described in the outlined steps is repeated until convergence.

Other features were added to DQN in order to improve its performance and accelerate its convergence. The work in [64] used prioritized ER to sample experiences that have high expected learning progress from the memory buffer with higher probability compared to experiences with lower expected learning progress. The estimation of the expected learning progress was done using TD-error of the experiences. Moreover, the paper [65] introduced a dual architecture to the DQN, where one network estimated the state value function and a separate network estimated the advantage function. Both network outputs were aggregated to produce the Q-values. The advantage of this approach was to make the distinction between actions in states where immediate reward is preferred (i.e., maximize the Advantage function), and actions where the long-term reward is better (i.e., maximize the value function).

2.2.4.4 Proximal Policy Optimization

Proximal Policy Optimization (PPO) [66] is another NN-based method, which falls under the family of Policy Gradient, Actor-Critic methods. Policy Gradient algorithms are a family of RL methods that attempt to find the policy directly by iteratively optimizing an objective [67]. In order to overcome the curse of dimensionality, this policy is a parameterized NN function with parameters $\mathbf{w}$. Therefore, the output of this kind of approach is a policy $\pi^{\mathbf{w}}(a|s)$ (Actor part). Using an objective function $\mathcal{L}^{\mathbf{w}}$, the quality of the current policy is measured, and the parameters $\mathbf{w}$ of the policy are updated with Gradient Ascent in the direction of the gradient $\nabla_{\mathbf{w}}\mathcal{L}^{\mathbf{w}}$. In addition to the policy update stage, an analysis of the value function can be done to provide more insights into finding a good policy (Critic part).

The Proximal Policy Optimization (PPO) is related to an Actor-Critic scenario where the actor part is the policy estimation with parameters $\mathbf{w}$ and the critic part is the value function with other parameters $\mathbf{w}'$.

More precisely, PPO uses a policy gradient method where Neural Networks approximate the policy and the value function with weights $\mathbf{w}$ and $\mathbf{w}'$, respectively. This algorithm was built upon a previous work called Trust Region Policy Optimization (TRPO) [68] where the goal was to limit the update step for the policy in order to improve the training stability. It forced the Kullback-Leibler (KL) divergence between the old and the new policies to be smaller than a predefined threshold, allowing thus small but monotonic updates of the policy.

Formally, assuming that we are in state $S_t = s$ at timestep t, and having taken an action $A_t = a$, the system transitions to a next state $S_{t+1} = s'$. The TRPO objective function is defined as follows:

$$\mathcal{L}^{\mathbf{w},\mathbf{w}'}_{TRPO} = \hat{\mathbb{E}}\left[\frac{\pi^{\mathbf{w}}(a|s)}{\pi^{\mathbf{w}_{old}}(a|s)}\hat{\mathbf{A}}^{\mathbf{w}'}\right], \tag{2.31}$$

with $\hat{\mathbb{E}}$ being the expectation over batch of samples, and $\hat{\mathbf{A}}^{\mathbf{w}'}$ being the truncated version of the advantage function for state S_t derived from the value function calculating with

weights $\mathbf{w}'$. This advantage function is defined as:

$$\hat{\mathbf{A}}^{\mathbf{w}'}(S_t = s) = \sum_{\tau=t}^{T-1} (\gamma\lambda)^{\tau-t} \left(R^{A_\tau}_{S_\tau S_{\tau+1}} + \gamma \mathbf{V}^{\mathbf{w}'}(S_{\tau+1}) - \mathbf{V}^{\mathbf{w}'}(S_\tau) \right), \tag{2.32}$$

where λ is the generalized advantage estimation factor. The last term in brackets is related to Bellman equation where $S_{\tau+1} = s'$. Then the TRPO update works as follows:

$$\begin{array}{ll} \max_{\mathbf{w},\mathbf{w}'} & \mathcal{L}^{\mathbf{w},\mathbf{w}'}_{TRPO} \\ \text{subject to} & \hat{\mathbb{E}}\left[\mathrm{KL}\left(\pi^{\mathbf{w}_{old}}(a|s), \pi^{\mathbf{w}}(a|s)\right)\right] \leq \delta. \end{array} \tag{2.33}$$

PPO simplifies this concept by replacing the constraint on KL divergence with a clipping function, i.e., a function $f(x)$ that puts a threshold on its input x in the following way:

$$f(x) = \mathrm{clip}(x, a, b) \implies (x, a, b) \in \mathbb{R}^3 : \; f(x) = \begin{cases} x & \text{if } a < x < b, \\ a & \text{if } a \geq x, \\ b & \text{if } b \leq x. \end{cases} \tag{2.34}$$

with a and b being respectively the lower and the upper threshold. The PPO update works as follows

$$\max_{\mathbf{w},\mathbf{w}'} \; \mathcal{L}^{\mathbf{w},\mathbf{w}'}_{PPO}, \tag{2.35}$$

with

$$\begin{aligned} \mathcal{L}^{\mathbf{w},\mathbf{w}'}_{PPO} &= \hat{\mathbb{E}}\left[\min\left(q^{\mathbf{w}}\hat{\mathbf{A}}^{\mathbf{w}'}, \mathrm{clip}\left(q^{\mathbf{w}}, 1-\epsilon, 1+\epsilon\right)\hat{\mathbf{A}}^{\mathbf{w}'}\right)\right], \\ q^{\mathbf{w}} &= \frac{\pi^{\mathbf{w}}(a|s)}{\pi^{\mathbf{w}_{old}}(a|s)}. \end{aligned} \tag{2.36}$$

In contrast to the TRPO algorithm, PPO integrates the constraint into the objective function by clipping the ratio between both policies up to ϵ.

Based on the idea, the PPO agent training procedure is described as follows:

1. Fill up a memory buffer with data from the transitions of the agent when navigating the environment for a certain number of timesteps. Starting with a state s, sample an action $a \sim \pi^{\mathbf{w}_{old}}$ using the actor network, obtain a next state s' and a corresponding reward $R^a_{ss'}$. Calculate the value function vector $\mathbf{V}^{\mathbf{w}'}(s)$ as well, and store the tuple $\left(s, a, \pi^{\mathbf{w}_{old}}, \mathbf{V}^{\mathbf{w}'}(s), s', R^a_{ss'}\right)$ in the memory buffer. The new state s' is now considered the current state as of the next timestep.

2. Repeat the operations in the previous step until the determined number of timesteps is reached, and the obtained sequence corresponds to an episode. Start with a new randomly chosen initial state.

3. Once the memory buffer is completely filled with tuples, it plays the role of batch for training the actor and the critic networks. Given this batch, non-overlapping mini-batches of given sizes are considered by picking up tuples randomly.

4. Looping through all the mini-batches, the advantage functions $\hat{\mathbf{A}}_{b_i}^{\mathbf{w}'}$ for a certain mini-batch b_i are estimated, as well as the ratios between the old policy and the new one $q_{b_i}^{\mathbf{w}}$.

5. Calculate the objective function $\mathcal{L}_{PPO}^{\mathbf{w},\mathbf{w}',b_i}$ associated with the mini-batch b_i. In fact, this function is obtained as in (2.36) where the expectation has been replaced with the expectation over the tuples in the considered mini-batch b_i. It is used to update the weights of both networks with gradient ascent:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \nabla_{\mathbf{w}} \mathcal{L}_{PPOa}^{b_i}, \tag{2.37}$$

$$\mathbf{w}' \leftarrow \mathbf{w}' + \alpha \nabla_{\mathbf{w}'} \mathcal{L}_{PPOa}^{b_i}, \tag{2.38}$$

   where α is the learning rate and $\mathcal{L}_{PPOa}$ is the *augmented* objective function defined as:

$$\mathcal{L}_{PPOa}^{b_i} = \mathcal{L}_{PPO}^{\mathbf{w},\mathbf{w}',b_i} + \beta_1 \mathcal{L}_{VF}^{\mathbf{w}',b_i} + \beta_2 \mathcal{S}^{\mathbf{w},b_i}, \tag{2.39}$$

   with $\mathcal{L}_{VF}^{\mathbf{w}',b_i}$ a squared error term for the value function, and $\mathcal{S}^{\mathbf{w},b_i}$ the entropy loss that encourages exploration. β_1 and β_2 are some hyperparameters to tune.

6. After visiting all the mini-batches b_i, the previous steps in the training operation are repeated on the same batch with newly shuffled mini-batches for $\mathbf{I}$ iterations. Afterwards, the stored batch is wiped out and a new process is started once more, filling up the memory buffer, and training the agent again (this corresponds to a new epoch), with the obtained policy becoming the old one.

7. After a certain number of epochs, the algorithm converges to a policy $\pi^{\mathbf{w}}(a|s)$ which will be applied during the test phase.

2.3 Federated Learning

Federated Learning (FL) is a relatively new concept in ML introduced in [24]. The main aim of FL is to distribute learning over multiple nodes while conserving data privacy, and learn a joint model. To train FL models, a set of nodes that communicate with a central node perform local updates and operations using locally produced data (without sharing it) on the same NN structures. Following few iterations, the nodes send either their updated weights, the batch loss, or the gradients of the weights to the central node, where a global aggregation of the received information from participating node is done and a new global model is produced. The global model is then broadcasted to the local nodes to continue training and the process is repeated until global convergence. An illustrative example is shown in Figure 2.9. Note that distributed training is different than FL in the sense that there is not one model trained by multiple nodes in a distributed way using the same pool of data, but rather multiple models trained separately, using local data, and averaged in one central node.

The produced FL model has great generalization abilities across the heterogeneous data that the nodes might have. Indeed, the global update procedure can differ from

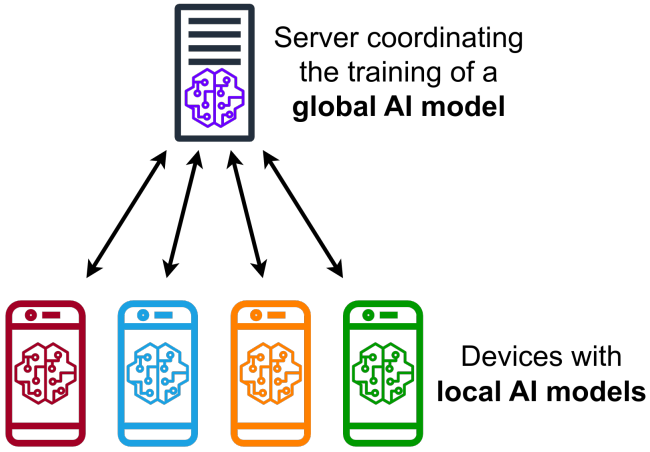

Figure 2.9: Federated learning general structure. Source [69].

one approach to another, involving weighted sums, batch loss, gradients or other useful parameters, depending on the application and the local nodes computational capacities.

FL is categorized into three categories; Horiztonal FL (HFL), Vertical FL (VFL), and Federated Transfer Learning (FTL) [70]:

- *HFL* : The datasets of the nodes in this case have a small overlap, while the data features are mostly aligned.
- *VFL* : The nodes have different targets to achieve, thus less correlated data features and models, but they serve a large portion of users towards a common goal.
- *FTL*: The nodes are not aligned in this particular scenario, and the aim is to transfer the knowledge from one source domain with rich resources, to a less rich domain, by exploiting the invariance between these domains.

A concrete example of FL modeling is in communication networks, where the BS (or the MEC server next to it) can be considered as a central node, while the local nodes are the devices connected to the BS (IoT devices for example). The devices train local models without sharing the data generated or collected. This approach is particularly useful when utilizing a Federated Learning Multi-Agent setup of Reinforcement Learning as introduced in [71] and [72], and further investigated in [73]. The work in Chapter 4 implements this concept.

Part II

Contributions

CHAPTER 3

Optimal Scheduling-Offloading Policies

The generational leap in wireless communications standards from 4G to 5G brought with it numerous advantages and improvements to the existing infrastructure and applications. 5G offers a significant increase in data rates, response times, and system capacity for handling multiple connected users. Many new applications and services are enabled that were not possible to implement in previous generations, e.g. Massive IoT and Autonomous Driving. This was done thanks to the addition of technologies like Mobile-Edge Computing (MEC) and Energy Harvesting EH.

However, with the exponential growth in the number of connected devices, optimizing the existing resources will limit the costly infrastructure changes, in terms of the used equipment and the energy consumed that need to be available in order to accommodate the large numbers of users.
One of the ways to perform such optimization is to allocate the resources of the network among the users in an efficient and fair way, while maintaining a level of performance throughout the system. Resource Allocation can target different aspects of the network resources, e.g. transmit power, bandwidth, energy consumption, computational offloading or scheduling tasks.

Subsequently, multiple scenarios of resource allocation can arise that differ from each other depending on the components of the system. As an example, in a scenario where a Base Station (BS) serves multiple users with a limited budget power of Downlink transmission, the goal is to distribute this power optimally to satisfy the different needs of the users. By considering the Uplink communication, the objective may become related to bandwidth allocation, user scheduling, or to a joint-setup of all the aforementioned goals. Additionally, when the users / BS have energy constraints, the problem may consider the energy efficiency of the communications between the system devices.

The resulting optimization problem can be of a non-convex nature, that requires complex tools to solve it analytically. However, when the system becomes too complex, Markov Decision Processes MDPs can be used to model the system environment, to be further solved with the deployment of Reinforcement Learning RL methods. RL produces the policies that can dictate the optimal allocation of resources for each scenario that may present itself. Moreover, for even larger and more complex MDP environments, Deep Reinforcement Learning DRL algorithms offer better performance through the use of NNs and their ability to model complex functions.

From that, the resource allocation problem that we tackle in the two contributions of this chapter and solve using RL techniques, is concerned with a system containing a Base Station (BS) serving User Equipments (UEs). Each UE tries to execute the flow of data packets that arrive at its buffer, which have strict delay constraints, i.e., the data packets are dropped once reaching an age limit. In addition, a limited-size battery is fitted to each UE that can harvest energy from ambient sources (e.g. Solar, RF). Moreover, a Mobile Edge Computing (MEC) server is installed near the BS, to which the UEs can offload their data packets to execute and send back the result (local execution is possible as well with a lower computing capacity). In the case where multiple UEs offload their data packets at the same time, Non-Orthogonal Multiple Access (NOMA) is enabled to allow the sharing of channel resources. The resulting resource allocation problem is a joint

Scheduling-Offloading one, where the goal is to minimize the number of dropped packets.

In this first contribution, Energy Harvesting is not considered in the system, rendering the users able to execute their data packets without energy concerns. The problem is therefore less complex to model using an MDP, and the RL methods used to solve it are optimal. Comparing the used methods (model-based and model-free) against some standard ones shows their performance advantage in terms of the packet error-rate. When the system grows larger in size, the curse of dimensionality makes using optimal methods impractical, and thus only model-free methods (with DRL ones in particular) are viable. The scaling of the system shows the benefits of using DRL-based techniques, due to their impressive generalization capabilities to unseen scenarios using NNs.

In the second contribution, Energy Harvesting is added to the system, which results in a more complex system to solve. The subsequent problem formulation changes, and using model-based methods becomes infeasible due to the exponential increase in the environment size. Therefore, The MDP model is solved using a state-of-the-art DRL algorithm named Proximal Policy Optimization (PPO) [66]. Results show that PPO performs better than the standard methods, while consuming less overall energy.

Some recent works were proposed in this scope, e.g. a heuristic algorithm was studied in [74] to minimize the task average execution delay by optimizing NOMA's Successive Interference Cancellation (SIC) ordering and resource allocation in IoT networks. An online packet offloading algorithm was also developed in [75] that maximized the long term system throughput and minimizes the signaling overhead. A multi-base station and multi-user framework was considered in [76] where a Multi-Agent Deep Reinforcement Learning (DRL) was used to jointly optimize the packet offloading, sub-station and sub-channel resource allocation, and minimize the energy consumption in the system. In [77], joint optimization of offloading policy including a packet splitting for partial offloading and channel resource allocation (NOMA/OMA sub-carrier allocation) using DRL was also investigated. Moreover, a DRL-based online packet offloading in multi-carrier NOMA-enabled MEC networks was developed in [78] to optimize sub-carrier allocation. The work in [79] focused on an ultra-dense network with small BSs and clustering to jointly optimize user clustering and resource and power allocation. Finally, the RL-based approach studied in [80] was tailored for computation offloading and energy transmission in a MEC-EH based system. They minimized the energy consumption and execution delay by decomposing the problem into sub-problems to overcome the dimensionality curse.
Regarding the works that considered PPO in their works, an extension to MIMO-MEC has been done in [81] by using Deep Q-Network [63] and PPO. In [82], the authors used PPO in a multi-user environment with multiple MEC servers. An actor critic DRL model was developed in [83] for a resource allocation problem in a multi-cell NOMA system. The proposed model is a dynamic one that could switch between multiple NNs depending on the network loads. multi-cell NOMA with Actor Critic DRL was shown to be more performant compared to an OMA scheme.

The aforementioned works differ from ours in the strict delay aspect, that imposes constraints on the MDP environment configuration, and the resulting RL policies. Additionally, a scenario adopting of MEC capabilities, EH-enabled devices and NOMA transmission

is not used in any of these works. These contributions are an extension of the one done in [42], with the addition of multiple UE, NOMA transmissions, and the use of model-free methods including DRL ones.

The remainder of the chapter is organized as follows:

- Section 3.1 describes the system model and the subsequent data buffer and channel models, as well as the scheduling decisions, energy equations for each decision, and the time constraints.
- Section 3.2 presents the formulation of the optimization problem as an MDP and solves it using various Reinforcement Learning algorithms.
- Section 3.3 presents and analyzes the numerical results.
- Section 3.4 explores the extension to an Energy Harvesting-based system at the UE level, with an energy and battery model, problem formulation and resolution, as well as simulation results.
- Section 3.5 gives some concluding remarks.

3.1 System Model

We consider 2 user equipments UEs served by a base station BS with a MEC server. The UEs are equipped with buffers to store the application data that need to be processed within a strict delay. The decision for offloading or processing locally is made at the BS level at the beginning of each timestep of size $\mathcal{T}$. The decision is then broadcasted to the UEs for free.

Therefore, at each timestep, the UEs process either their data packets *locally* using their processor, or *remotely* by offloading them to the BS , to be processed at the MEC before receiving the results. When both UEs offload their packets to the MEC server, NOMA is considered as the multiple access technique. In the sequel, we provide more details on the buffer model and strict delay constraint, the channel model, and the scheduling decisions with their corresponding time constraints.

Figure 3.1 showcases the adopted system model.

3.1.1 Data Buffer Model

The data buffer at each $\text{UE}^{(1)}$ and $\text{UE}^{(2)}$ is used to store the packets awaiting their execution. It is modeled as a vector $\mathbf{d}$ of size $\mathcal{S}^d$ where each component d^k, $k \in [0, \mathcal{S}^d)$, represents a data packet by its age (-1 is for an empty buffer slot). The data packets are ordered in a descending order with respect to (w.r.t.) their age (old packets shift to allow for the arrival of new packets).

The number of packets in the buffer at a given timestep t is denoted by $\mathcal{N}_t^d \leq \mathcal{S}^d$. The arrival of data packets is following an independent and identically distributed (i.i.d.) random Poisson process with mean μ^d. The probability distribution is given by:

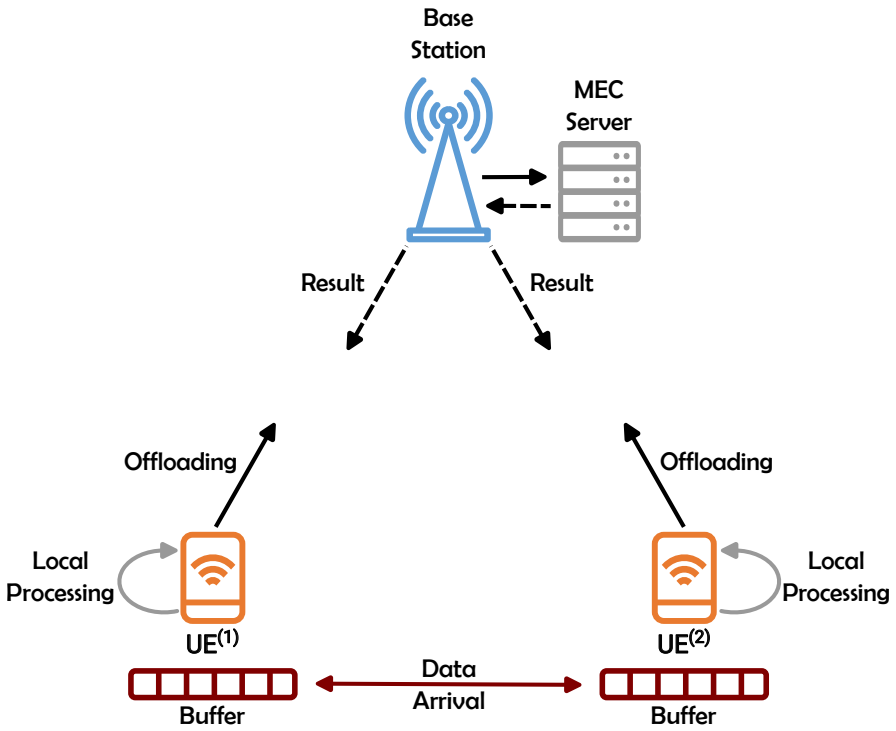

Figure 3.1: System model.

$$p(q_t = \mathcal{D}) = e^{-\mu^d} \cdot \frac{(\mu^d)^{\mathcal{D}}}{\mathcal{D}!}, \tag{3.1}$$

where q_t is the number of arrived packets at timestep t. If q_t exceeds the available $\mathcal{S}^d - \mathcal{N}_t^d$ slots in the buffer, buffer overflow occurs and the excess will be dropped. If the packets present in the buffer reach a maximum pre-fixed delay, ∇, due to strict delay constraint, delay violation occurs and these packets will be dropped as well. The buffer model is illustrated in Figure 3.2.

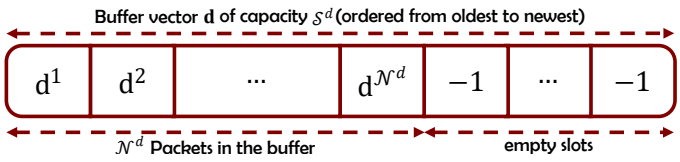

Figure 3.2: Buffer model.

3.1.2 Channel Model

The channel between the UEs and the BS is modeled as a Rayleigh flat-fading channel, with an additive white Gaussian noise AWGN of power spectral density $\mathcal{N}_0$. The bandwidth is denoted $\mathcal{W}^{dl}$ for the Downlink (DL) and $\mathcal{W}^{ul}$ for the Uplink (UL). We assume full knowledge of the Channel State Information at the Transmitter (CSIT). The channel response is constant during a timestep, with $h^{(i)}$ the complex amplitude of the channel for UE$^{(i)}$, $i \in \{1, 2\}$, and $x^{(i)} = |h^{(i)}|^2$ its gain. We assume no channel correlation across successive timesteps. We model the variation of the channel gain by the continuous random

variables following the exponential distribution with mean μ^c:

$$p(x_t = \mathcal{C}) = \frac{1}{\mu^c} e^{\frac{-\mathcal{C}}{\mu^c}}. \tag{3.2}$$

We consider that the BS makes its decision based on a quantized version of the channel. In practice, the BS broadcasts a training sequence, the UEs then estimate their channel and send back a quantized version in order to limit the size of the overhead. The quantized channel gain for UE$^{(i)}$ is defined as $\underline{x}^{(i)} = \mathcal{Q}(x^{(i)})$, where $\mathcal{Q}$ is the quantization function and $\underline{x}^{(i)}$ is the lower value of the interval in which $h^{(i)}$ belongs to. We denote $\overline{x}^{(i)}$ as the upper value of that interval which can be deduced from $\underline{x}^{(i)}$. The finite set of the quantized channel states is $\mathcal{X}$. At each timestep t, the gain $x^{(i)}$ for UE$^{(i)}$ will be projected into the discrete space and the obtained value $\underline{x}_t^{(i)}$ will be used further in the calculations.

Notably, the new NOMA equations for the UL and DL case, taking into account the channel gain quantization, are as follows:

$$\mathcal{R}_{noma,t}^{ul,(1)} = \mathcal{W}^{ul} \cdot \log_2 \left(1 + \frac{p_t^{\mathbf{o},(1)} \cdot \underline{x}_t^{(1)}}{p_t^{\mathbf{o},(2)} \cdot \overline{x}_t^{(2)} + \mathcal{W}^{ul} \cdot \mathcal{N}_0} \right), \tag{3.3}$$

$$\mathcal{R}_{noma,t}^{ul,(2)} = \mathcal{W}^{ul} \cdot \log_2 \left(1 + \frac{p_t^{\mathbf{o},(2)} \cdot \underline{x}_t^{(2)}}{\mathcal{W}^{ul} \cdot \mathcal{N}_0} \right), \tag{3.4}$$

$$\mathcal{R}_{noma,t}^{dl,(1)} = \mathcal{W}^{dl} \cdot \log_2 \left(1 + \frac{\delta p_t^{\mathbf{b},(bs)} \cdot \underline{x}_t^{(1)}}{\mathcal{W}^{dl} \cdot \mathcal{N}_0} \right), \tag{3.5}$$

$$\mathcal{R}_{noma,t}^{dl,(2)} = \mathcal{W}^{dl} \cdot \log_2 \left(1 + \frac{(1-\delta) p_t^{\mathbf{b},(bs)} \cdot \underline{x}_t^{(2)}}{\delta p_t^{\mathbf{b},(bs)} \cdot \underline{x}_t^{(2)} + \mathcal{W}^{dl} \cdot \mathcal{N}_0} \right). \tag{3.6}$$

For these equations, we consider a worst case scenario when formulating the interference of the weaker UE, w.r.t. the channel gain, with $\overline{x}_t^{(2)}$ the upper-bound value of the quantization in Equation (3.3).

3.1.3 Scheduling Decisions

Three different decisions can be made at the beginning of a timestep t of duration $\mathcal{T}$: *Idle*, *Local Processing* or *Offloading* for each UE$^{(i)}$, with $i \in \{1, 2\}$.

- *Idle* : UE$^{(i)}$ will not execute any packet, thus the number of packets executed is $m_t^{(i)} = 0$.
- *Local Processing* : UE$^{(i)}$ has the computational capacity to execute locally $m_t^{(i)} \leq \mathcal{M}^{\mathbf{l}}$ packets, $\mathcal{M}^{\mathbf{l}}$ being the maximum number of locally executed packets.
- *Offloading* : UE$^{(i)}$ offloads $m_t^{(i)}$ packets to the BS, $m_t^{(i)} \leq \mathcal{M}^{\mathbf{o}}$, $\mathcal{M}^{\mathbf{o}}$ being the maximum number of offloaded packets. These packets should be processed and returned, all within the timestep duration $\mathcal{T}$. This step comprises 4 steps; transmission, waiting, reception and decoding.

3.1.4 Energy Consumed

When processing packets, different amounts of energy are consumed for each scheduling decision. These energy values are related to the time constraints, and they are derived as the following:

- *Idle* $\mathbf{i}$: $\mathrm{UE}^{(i)}$ does not consume any energy:

$$\mathcal{E}_t^{\mathbf{i},(i)} = 0. \tag{3.7}$$

- *Local Processing* $\mathbf{l}$: $m_t^{(i)}$ are processed locally for $\mathrm{UE}^{(i)}$ with power $\mathcal{P}^{\mathbf{l}}$ per processed packet. The corresponding energy is:

$$\mathcal{E}_t^{\mathbf{l},(i)} = m_t^{(i)} \cdot \mathcal{P}^{\mathbf{l}} \cdot \mathcal{T}. \tag{3.8}$$

- *Offloading* $\mathbf{o}$: $\mathrm{UE}^{(i)}$ offloads $m_t^{(i)}$ packets with power $p_t^{\mathbf{o},(i)}$ that cannot exceed the maximum offloading power $\mathcal{P}^{\mathbf{o}}$.

  – If only one UE is offloading, the consumed energy at $\mathrm{UE}^{(i)}$ is:

$$\mathcal{E}_t^{\mathbf{o},(i)} = m_t^{(i)} \cdot \Big(\underbrace{\frac{\mathcal{L}^{ul} \cdot p_t^{\mathbf{o},(i)}}{\mathcal{R}_{su,t}^{ul,(i)}}}_{\text{Transmission}} + \underbrace{\mathcal{T}^{\mathbf{w}} \cdot p^{\mathbf{w}}}_{\text{Waiting}} + \underbrace{\frac{\mathcal{L}^{dl} \cdot p^{\mathbf{r}}}{\mathcal{R}_{su,t}^{dl,(i)}}}_{\text{Reception}} + \underbrace{\rho \cdot \frac{\mathcal{L}^{dl} \cdot p^{\mathbf{d}}}{\mathcal{R}_{su,t}^{dl,(i)}}}_{\text{Decoding}} \Big), \tag{3.9}$$

  where $i \in \{1, 2\}$, $\mathcal{L}^{ul}$ and $\mathcal{L}^{dl}$ are the lengths in bits of the transmitted data packets in UL and DL, respectively. $\mathcal{T}^{\mathbf{w}}$ is the waiting time at UEs for receiving the results. $p^{\mathbf{w}}$, $p^{\mathbf{r}}$ and $p^{\mathbf{d}}$ are the consumed powers for waiting, reception and decoding. ρ is a scaling factor for efficient decoding. The optimal rates for single-user offloading in UL and DL are expressed as:

$$\mathcal{R}_{su,t}^{ul,(i)} = \mathcal{W}^{ul} \cdot \log_2 \left(1 + \frac{p_t^{\mathbf{o},(i)} \cdot \underline{x}_t^{(i)}}{\mathcal{W}^{ul} \cdot \mathcal{N}_0} \right), \tag{3.10}$$

$$\mathcal{R}_{su,t}^{dl,(i)} = \mathcal{W}^{dl} \cdot \log_2 \left(1 + \frac{p_t^{\mathbf{b},(bs)} \cdot \underline{x}_t^{(i)}}{\mathcal{W}^{dl} \cdot \mathcal{N}_0} \right). \tag{3.11}$$

  – On the other hand, when both UEs offload at the same timestep, NOMA is used to allow simultaneous transmission. The expressions at both UEs of the consumed energy are:

$$\mathcal{E}_t^{\mathbf{o},(1)} = m_t^{(1)} \cdot \Big(\frac{\mathcal{L}^{ul} \cdot p_t^{\mathbf{o},(1)}}{\mathcal{R}_{noma,t}^{ul,(1)}} + \mathcal{T}^{\mathbf{w}} \cdot p^{\mathbf{w}} + \frac{\mathcal{L}^{dl} \cdot p^{\mathbf{r}}}{\min\{\mathcal{R}_{noma,t}^{dl,(1)}, \mathcal{R}_{noma,t}^{dl,(2)}\}} + \rho \cdot \big(\frac{\mathcal{L}^{dl} \cdot p^{\mathbf{d}}}{\mathcal{R}_{noma,t}^{dl,(1)}} + \frac{\mathcal{L}^{dl} \cdot p^{\mathbf{d}}}{\mathcal{R}_{noma,t}^{dl,(2)}} \big) \Big), \tag{3.12}$$

$$\mathcal{E}_t^{\mathbf{o},(2)} = m_t^{(2)} \cdot \Big(\frac{\mathcal{L}^{ul} \cdot p_t^{\mathbf{o},(2)}}{\mathcal{R}_{noma,t}^{ul,(2)}} + \mathcal{T}^{\mathbf{w}} \cdot p^{\mathbf{w}} + \frac{\mathcal{L}^{dl} \cdot p^{\mathbf{r}}}{\mathcal{R}_{noma,t}^{dl,(2)}} + \rho \cdot \frac{\mathcal{L}^{dl} \cdot p^{\mathbf{d}}}{\mathcal{R}_{noma,t}^{dl,(2)}} \Big). \tag{3.13}$$

  The reception time in equation 3.12 is tuned according to the slowest communication rate. In addition, the decoding time has two terms since SIC is applied.

3.1.5 Time Constraints

The above energy equations imply that offloading decisions only occur when the transmission, waiting, reception and decoding are completed within the timestep duration $\mathcal{T}$. For the NOMA offloading case, we have the following equations (single user case is similar to the one of $\text{UE}^{(2)}$ in Equation (3.15)):

$$m_t^{(1)} \cdot \Big(\frac{\mathcal{L}^{ul}}{\mathcal{R}_{noma,t}^{ul,(1)}} + \mathcal{T}^{\mathbf{w}} + \frac{\mathcal{L}^{dl} \cdot p^{\mathbf{r}}}{\min\{\mathcal{R}_{noma,t}^{dl,(1)}, \mathcal{R}_{noma,t}^{dl,(2)}\}} + \rho \cdot \big(\frac{\mathcal{L}^{dl}}{\mathcal{R}_{noma,t}^{dl,(1)}} + \frac{\mathcal{L}^{dl}}{\mathcal{R}_{noma,t}^{dl,(2)}} \big) \Big) \leq \mathcal{T}, \quad (3.14)$$

$$m_t^{(2)} \cdot \Big(\frac{\mathcal{L}^{ul}}{\mathcal{R}_{noma,t}^{ul,(2)}} + \mathcal{T}^{\mathbf{w}} + \frac{\mathcal{L}^{dl}}{\mathcal{R}_{noma,t}^{dl,(2)}} + \rho \cdot \frac{\mathcal{L}^{dl}}{\mathcal{R}_{noma,t}^{dl,(2)}} \Big) \leq \mathcal{T}. \quad (3.15)$$

From these equations, we can derive the corresponding offloading power, by forcing equality and using the explicit forms of $\mathcal{R}_{noma,t}^{ul,(1)}$ and $\mathcal{R}_{noma,t}^{ul,(2)}$:

$$p_t^{\mathbf{o},(1)} = \frac{\left(2^{\zeta^{(1)}} - 1\right) \left(p_t^{\mathbf{o},(2)} \cdot \overline{x}_t^{(2)} + \mathcal{W}^{ul} \cdot \mathcal{N}_0\right)}{\underline{x}_t^{(1)}}, \quad (3.16)$$

$$p_t^{\mathbf{o},(2)} = \frac{\left(2^{\zeta^{(2)}} - 1\right) \mathcal{W}^{ul} \cdot \mathcal{N}_0}{\underline{x}_t^{(2)}}, \quad (3.17)$$

with

$$\zeta^{(1)} = \left(\frac{\mathcal{L}^{ul}}{\mathcal{W}^{ul} \left(\frac{\mathcal{T}}{m_t^{(1)}} - \mathcal{T}^{\mathbf{w}} - \frac{\mathcal{L}^{dl}}{\min\{\mathcal{R}_{noma,t}^{dl,(1)}, \mathcal{R}_{noma,t}^{dl,(2)}\}} - \rho \left(\frac{\mathcal{L}^{dl}}{\mathcal{R}_{noma,t}^{dl,(2)}} - \frac{\mathcal{L}^{dl}}{\mathcal{R}_{noma,t}^{dl,(1)}} \right) \right)} \right), \quad (3.18)$$

$$\zeta^{(2)} = \left(\frac{\mathcal{L}^{ul}}{\mathcal{W}^{ul} \left(\frac{\mathcal{T}}{m_t^{(2)}} - \mathcal{T}^{\mathbf{w}} - \frac{\mathcal{L}^{dl}}{\mathcal{R}_{noma,t}^{dl,(2)}} - \rho \frac{\mathcal{L}^{dl}}{\mathcal{R}_{noma,t}^{dl,(2)}} \right)} \right). \quad (3.19)$$

The resulting offloading powers are limited by a maximal offloading power $\mathcal{P}^{\max} = \mathcal{P}^{\mathbf{o}}$ that can forbid the transmission when breached.

3.2 Problem Formulation and Resolution

3.2.1 Problem Formulation

In this section, we formulate our scheduling problem (how many packets to schedule for computing and at which level: locally or remotely) as an MDP. The problem boils down to exhibiting policies that minimize the overall data packet loss due to either buffer overflow or delay violation. These policies output an action to be made at the beginning of each timestep, given some information on the system (e.g. channel conditions, number of packets in the buffer and their age, multiple access).

In our problem, the data buffer and channel dynamics satisfy the Markov property. Therefore, we can formulate the problem as an MDP. The MDP environment can be described with a state space $\mathbf{S}$, an action space $\mathbf{A}$, a reward model $\mathbf{R}$ and a transition model $\mathbf{T}$. In what follows, we define each component in our MDP problem.

- The state space $\mathbf{S}$ contains the information on the buffer and channel status for both users. It is defined as:

$$\mathbf{S} = \{\mathbf{d^{(1)}}, \mathbf{d^{(2)}}, \underline{x}^{(1)}, \underline{x}^{(2)}\}, \tag{3.20}$$

where $\mathbf{d^{(i)}} = [d^{1,(i)}, d^{2,(i)}, \cdots, d^{\mathcal{S}^d,(i)}]$ is the buffer state for UE$^{(i)}$ represented as a vector with $\mathcal{S}^d$ being the buffer size, and $\underline{x}^{(i)}$ is the corresponding quantized channel gain for that UE, with $i \in \{1, 2\}$.

With a finite number of possible combinations, the size of the state space $\mathbf{S}$ can be upper-bounded by the following expression:

$$|\mathbf{S}| \leq \left[|\mathcal{X}| \cdot (\nabla + 2)^{\mathcal{S}^d}\right]^2, \tag{3.21}$$

where $|\mathcal{X}|$ is the number of quantized channel states in the space $\mathcal{X}$, and ∇ is the maximum delay for the packets in the data buffer. This upper-bound can be further reduced, by ordering the packets in the buffer from oldest to newest.

- The action space $\mathbf{A}$ contains the possible joint decisions at both UEs that the agent can take at any given state. The actions are the following ones: offloading packets, processing locally packets or staying idle, along with the number of packets $m^{(i)}$ to be processed for each UE$^{(i)}$ within the limit defined for each type ($\mathcal{M}^{\mathbf{o}}$ for offload and $\mathcal{M}^{\mathbf{l}}$ for local).

We denote the size of the action space as $|\mathbf{A}|$ and we order the set of action indices a as follows: $a = 0$ refers to the following action
$\{\text{type 1 : 'idle'}, \text{ type 2 : 'idle'}, \ m^{(1)} = 0, \ m^{(2)} = 0\}$, and $a = |\mathbf{A}| - 1$ is associated with $\{\text{type 1 : 'offload'}, \text{ type 2 : 'offload'}, \ m^{(1)} = \mathcal{M}^{\mathbf{o}}, \ m^{(2)} = \mathcal{M}^{\mathbf{o}}\}$. The resulting action space size is:

$$|\mathbf{A}| = (\mathcal{M}^{\mathbf{l}} + \mathcal{M}^{\mathbf{o}} + 1)^2. \tag{3.22}$$

To perform the mapping between the index a and the corresponding action couple, we convert the basis of the index from the decimal base to a new base $u = \mathcal{M}^{\mathbf{l}} + \mathcal{M}^{\mathbf{o}} + 1$ that corresponds to the number of possible actions per UE. We use each digit to determine the action of each UE:

$$\overline{a}^{10} = \overline{a_1\, a_2}^{u} = a_1 \cdot (u)^0 + a_2 \cdot (u)^1. \tag{3.23}$$

The fact that we have $a_i \in \{0, \cdots, u-1\}$ for each $i \in 1, 2$ means that each a_i can represent all possible actions for a specific UE. The corresponding action for UE$^{(i)}$ is then as follows:

  - $a_i = 0$: translates to the idle action, with $m^{(i)} = 0$
  - $0 < a_i \leq \mathcal{M}^{\mathbf{l}}$: Local action with $m^{(i)} = a_i$
  - $\mathcal{M}^{\mathbf{l}} < a_i \leq \mathcal{M}^{\mathbf{o}}$: Offloading action with $m^{(i)} = a_i - \mathcal{M}^{\mathbf{l}}$

The table 3.1 details the ranges of each joint action.

Table 3.1: Action Space Ranges

a	**Action 1**	**Action 2**
$a = 0$	idle	idle
$0 < a \leq \mathcal{M}^{\mathbf{l}}$	local	idle
$\mathcal{M}^{\mathbf{l}} < a \leq \mathcal{M}^{\mathbf{l}} + \mathcal{M}^{\mathbf{o}}$	offload	idle
$\mathcal{M}^{\mathbf{l}} + \mathcal{M}^{\mathbf{o}} < a \leq \mathcal{M}^{\mathbf{l}} + \mathcal{M}^{\mathbf{o}} + \mathcal{M}^{\mathbf{l}} \cdot \mathcal{M}^{\mathbf{o}}$	offload	local
$\mathcal{M}^{\mathbf{l}} + \mathcal{M}^{\mathbf{o}} + \mathcal{M}^{\mathbf{l}} \cdot \mathcal{M}^{\mathbf{o}} < a \leq \mathcal{M}^{\mathbf{l}} + \mathcal{M}^{\mathbf{o}} + \mathcal{M}^{\mathbf{l}} \cdot \mathcal{M}^{\mathbf{o}} + (\mathcal{M}^{\mathbf{l}})^2$	local	local
$\mathcal{M}^{\mathbf{l}} + \mathcal{M}^{\mathbf{o}} + \mathcal{M}^{\mathbf{l}} \cdot \mathcal{M}^{\mathbf{o}} + (\mathcal{M}^{\mathbf{l}})^2 < a \leq 2 \cdot \mathcal{M}^{\mathbf{l}} + \mathcal{M}^{\mathbf{o}} + \mathcal{M}^{\mathbf{l}} \cdot \mathcal{M}^{\mathbf{o}} + (\mathcal{M}^{\mathbf{l}})^2$	idle	local
$2 \cdot \mathcal{M}^{\mathbf{l}} + \mathcal{M}^{\mathbf{o}} + \mathcal{M}^{\mathbf{l}} \cdot \mathcal{M}^{\mathbf{o}} + (\mathcal{M}^{\mathbf{l}})^2 < a \leq 2 \cdot \mathcal{M}^{\mathbf{l}} + 2 \cdot \mathcal{M}^{\mathbf{o}} + \mathcal{M}^{\mathbf{l}} \cdot \mathcal{M}^{\mathbf{o}} + (\mathcal{M}^{\mathbf{l}})^2$	idle	offload
$2 \cdot \mathcal{M}^{\mathbf{l}} + 2 \cdot \mathcal{M}^{\mathbf{o}} + \mathcal{M}^{\mathbf{l}} \cdot \mathcal{M}^{\mathbf{o}} + (\mathcal{M}^{\mathbf{l}})^2 < a \leq 2 \cdot \mathcal{M}^{\mathbf{l}} + 2 \cdot \mathcal{M}^{\mathbf{o}} + 2 \cdot \mathcal{M}^{\mathbf{l}} \cdot \mathcal{M}^{\mathbf{o}} + (\mathcal{M}^{\mathbf{l}})^2$	local	offload
$2 \cdot \mathcal{M}^{\mathbf{l}} + 2 \cdot \mathcal{M}^{\mathbf{o}} + 2 \cdot \mathcal{M}^{\mathbf{l}} \cdot \mathcal{M}^{\mathbf{o}} + (\mathcal{M}^{\mathbf{l}})^2 < a \leq 2 \cdot \mathcal{M}^{\mathbf{l}} + 2 \cdot \mathcal{M}^{\mathbf{o}} + 2 \cdot \mathcal{M}^{\mathbf{l}} \cdot \mathcal{M}^{\mathbf{o}} + (\mathcal{M}^{\mathbf{l}})^2 + (\mathcal{M}^{\mathbf{o}})^2$	offload	offload

- The reward model $\mathbf{R}$ is defined as the negative cost in an infinite-horizon MDP problem with with γ being the discount factor. In fact, the cost represents the sum of the number of dropped packets during a timestep for both UEs. It is expressed as:

$$\mathbf{R}^{\pi} = -\lim_{\mathbf{T}\to\infty} \mathbb{E}^{\pi}\left[\sum_{t=0}^{\mathbf{T}} \sum_{i\in\{1,2\}} (\gamma)^t \left(c_{1,t}^{(i)} + c_{2,t}^{(i)}\right)\right], \tag{3.24}$$

  where

  - π is the policy for which the reward function is evaluated.
  - $c_{1,t}^{(i)} = w_t^{(i)} - m_t^{(i)}$ is the instantaneous cost at UE$^{(i)}$ due to delay violation, i.e., the number of packets that reach the maximum delay ∇ and are discarded. We denote $w_t^{(i)} = \max(m_t^{(i)}, r_t^{(i)})$ as the number of packets that leave the buffer of UE$^{(i)}$ after taking an action a_t, where $r_t^{(i)}$ is the number of packets that have reached the maximum delay ∇ after incrementing the age of all packets by 1.
  - $c_{2,t}^{(i)} = \sum_{j=\mathcal{S}^d-\mathcal{N}_t^{d,(i)}+w_t^{(i)}+1}^{+\infty} (\mathcal{N}_t^{d,(i)} - w_t^{(i)} + j - \mathcal{S}^d) \cdot e^{-\mu^d} \cdot \frac{(\mu^d)^j}{j!}$ is the instantaneous cost due to buffer overflow, which is the number of arrived packets following the Poisson distribution that could not enter the buffer. It follows the Cumulative Distribution Function (CDF) of the Poisson distribution at the point where the arrival of packets causes a buffer overflow.

- The transition model $\mathbf{T}$ defines all the transition probabilities from a state $S_t = \mathbf{s}_t$ to a state $S_{t+1} = \mathbf{s}_{t+1}$ depending only on the current state $\mathbf{s}_t$ and the current action $A_t = a_t$. They are given by:

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t) = p(\mathbf{d}_{t+1}^{(1)}|\mathbf{d}_t^{(1)}, a_t) \cdot p(\mathbf{d}_{t+1}^{(2)}|\mathbf{d}_t^{(2)}, a_t) \cdot p(\underline{x}_{t+1}^{(1)}) \cdot p(\underline{x}_{t+1}^{(2)}), \tag{3.25}$$

  where $p(\underline{x}_{t+1}^{(i)})$ is the distribution of the channel gain $\underline{x}_{t+1}^{(i)}$, and $p(\mathbf{d}_{t+1}^{(i)}|\mathbf{d}_t^{(i)}, a_t)$ is the probability transition between two data buffer states for UE$^{(i)}$.

We hereafter define the range of possible actions and next state transitions, where for a state $\mathbf{s}_t$, each next state $\mathbf{s}_{t+1}$ and action a_t have to satisfy the below-mentioned conditions.

- The offloading powers corresponding to the chosen action a_t are less than or equal to the maximum offloading power, $p_t^{\mathbf{o},(i)} \leq \mathcal{P}^{\mathbf{o}}$.
- The number of processed packets is less than or equal to the size of the buffer, $m_t^{(i)} \leq \mathcal{N}_t^{d,(i)}$.
- The age difference between the same buffer slot in the states $\mathbf{s}$ and $\mathbf{s}_{t+1}$ is less than or equal to 1,

$$d_{t+1}^{j,(i)} \leq d_t^{j,(i)} + 1,\ j = \{0, \cdots, \mathcal{S}^d\}.$$

- The size of the next buffer state is greater than or equal to the difference between the size of the current buffer and the number of packets leaving the buffer, $\mathcal{N}_{t+1}^{d,(i)} \geq \mathcal{N}_t^{d,(i)} - w_t^{(i)}$.
- The age of the next state packets is bigger than the same current packets ages by 1,

$$\textbf{if } d_t^{j+w_t^{(i)},(i)} \neq -1 \textbf{ then } d_{t+1}^{j,(i)} = d_t^{j+w_t^{(i)},(i)} + 1,\ j = \{0, \cdots, \mathcal{S}^d - w_t^{(i)}\}.$$

- The age of the next state packets is less than or equal to 0 for empty slots in the current buffer state,

$$\textbf{if } d_t^{j+w_t^{(i)},(i)} = -1 \textbf{ then } d_{t+1}^{j,(i)} \leq 0,\ j = \{0, \cdots, \mathcal{S}^d - w_t^{(i)}\}.$$

- The empty slots in the current buffer, when $w_t^{(i)} \neq 0$ packets leave the buffer, have an age of 0 or less in the next state,

$$\textbf{if } \mathcal{N}_t^{d,(i)} = \mathcal{S}^d \textbf{ and } w_t^{(i)} \neq 0$$

$$\textbf{then } d_{t+1}^{j,(i)} \leq 0,\ j = \{\mathcal{S}^d - w_t^{(i)}, \cdots, \mathcal{S}^d\}.$$

If these conditions are satisfied, the probabilities of transition for the buffers are given by:

if $\mathcal{N}_t^{d,(i)} < \mathcal{S}^d$ **then**:

$$p(\mathbf{d}_{t+1}^{(i)} | \mathbf{d}_t^{(i)}, a_t) = e^{-\mu^d} \cdot \frac{(\mu^d)^{\mathcal{N}_{t+1}^{d,(i)} - \mathcal{N}_t^{d,(i)} + w^{(i)}}}{(\mathcal{N}_{t+1}^{d,(i)} - \mathcal{N}_t^{d,(i)} + w^{(i)})!}, \tag{3.26}$$

else:

$$p(\mathbf{d}_{t+1}^{(i)} | \mathbf{d}_t^{(i)}, a_t) = 1 - e^{-\mu^d} \sum_{j=0}^{\mathcal{S}^d - \mathcal{N}_t^{d,(i)} + w^{(i)} - 1} \frac{(\mu^d)^j}{j!}. \tag{3.27}$$

Our goal is to find the optimal policy $\pi^\star$ that maximizes the reward function defined in Equation (3.24) as:

$$\pi^\star = \arg\max_{\pi}\ \mathbf{R}^\pi. \tag{3.28}$$

3.2.2 Problem Resolution

To solve the optimization problem in Equation 3.28, We use the MDP model-based and model-free Reinforcement Learning algorithms described in chapter 2:

- Value Iteration (VI) : A model-based method with guaranteed convergence, as explained in section 2.2.3.1. It involves improving the value function vector $\mathbf{V}(s)$ in small steps starting from a initial $\mathbf{V}_0(s)$ and using the dynamic programming approach.

- Policy Iteration (PI) : Another model-based method detailed in section 2.2.3.2. PI attempts to find directly the optimal policy (rather than the optimal value function as in VI), by iterating between two steps: policy evaluation (using the current estimate of the value function) and policy improvement (using the valuation to improve the actions taken).

- Q-Learning (QL) : The most well-established model-free algorithm explained in section 2.2.4.1. QL relies on trial and error to build a model of the environment without having access to the transition model $\mathbf{T}$ or the reward model $\mathbf{R}$. The State-Action value matrix (Q-matrix) is used to record the experiences and learn from them. A balance of exploration and exploitation is needed to achieve convergence.

- Double Q-Learning (2QL) : Double Q-Learning solves the issue of over-estimation of Q-Learning as explained in section 2.2.4.2. 2QL promises faster convergence and better estimates of the Q-matrix by using two Q-matrices during training, while alternating in updating them.

- Deep Q-Network (DQN) : A Neural Network-based method that attempts to approximate the output of Q-Learning as introduced in section 2.2.4.3. The NN is used as an approximation to the Q-matrix by sampling states as input, and producing the corresponding State-Action values. A memory buffer is used to store the experiences, to be further used to train the NN (i.e., the Evaluation Network) with the help of a second network (i.e., Target Network). Double DQN can also be used in the same logic as 2QL.

3.3 Simulation Results

We consider a buffer of size $\mathcal{S}^d = 3$, a maximum delay $\nabla = 2$ and a timestep duration of $\mathcal{T} = 1$ ms. The maximum number of packets that can be offloaded is $\mathcal{M}^{\mathbf{o}} = 3$, while the maximum number of packets that can be executed locally is $\mathcal{M}^{\mathbf{l}} = 1$. The offloaded packets are of size $\mathcal{L}^{ul} = 1000$ bits and are sent with an offload power not exceeding $\mathcal{P}^{\mathbf{o}} = 2$ mW, through a quantized Rayleigh-faded channel with $|\mathcal{X}| = 3$ channel gain states $\mathcal{X} = \{-20, -1.487, 1.492\}$ dB. Uplink UL and downlink DL bandwidths are set to $\mathcal{W}^{ul} = 1$ MHz and $\mathcal{W}^{dl} = 5$ MHz, respectively. The channel noise spectral density is $\mathcal{N}_0 = -87$ dBm / Hz. We consider $\mathcal{T}^{\mathbf{w}} = 0.1$ ms when waiting for the results from the MEC server, with power $p^{\mathbf{w}} = 0.1$ mW.

The size of the packets in DL is $\mathcal{L}^{dl} = 100$ bits and these packets are broadcasted back to the UEs with a power $p^{\mathbf{b}} = 50$ W and a power allocation coefficient $\delta = 0.5$ (in the case of NOMA). The decoding efficiency ρ is set to 1. The power consumed at the receiver device is $p^{\mathbf{r}} = 3$ mW, and the power for decoding is $p^{\mathbf{d}} = 5$ mW. Local processing, on the other hand, is done using power $\mathcal{P}^{\mathbf{l}} = 150\mu$W.

Concerning the algorithms, the stopping criterion for the iterations on the VI algorithm is a gap on the value function update of $\nu = 10^{-7}$. In addition, QL and 2QL were trained over $\mathbf{E} = 5 \cdot 10^4$ episodes of $\mathbb{T} = 5000$ timesteps, and the learning rate α is decayed linearly throughout the training, from $\alpha_0 = 10^{-2}$ to $\alpha_{\mathbf{E}} = 10^{-3}$. DQN was trained over $\mathbf{E} = 10^3$ episodes of $\mathbb{T} = 5000$ timesteps, with a fixed learning rate $\alpha = 5 \cdot 10^{-3}$, a buffer size of $256 \cdot 10^2$ tuples, and a batch size of 256. Moreover, the used NN was a Multi-Layer Perceptron with 3 hidden layers of 64 neurons. QL, 2QL and DQN use an exploration rate ϵ that decreases along the training, from $\epsilon_0 = 1$ to $\epsilon_{\min} = 0.01$. The function that governs the evolution of epsilon is described as:

$$\epsilon_k = \max(\epsilon_{k-1} \cdot \epsilon_{\min}^{2/\mathbf{E}}, \epsilon_{\min}). \tag{3.29}$$

The aim is to decrease the exploration rate to its minimum when reaching 50% of the training procedure (episode $\mathbf{E}/2$), in order to achieve a better exploration of the environment. The discounted factor γ is set to 0.99.

In following figures, we consider the next algorithms: Value Iteration VI, Policy Iteration PI, Q-Learning QL, Double Q-Learning 2QL and Deep Q-Network DQN, compared to some naive methods. These naive methods are: naive local (NL), where the UEs are only allowed to execute their packets locally, naive offload (NO), where the UEs can only offload their packets to the MEC server, and naive random (NR), where the UEs choose a random action at each timestep. The figures display the overall discounted cost $\mathbf{R}^{\pi}$ (negative reward) averaged over $\mathbf{E} = 10^3$ episodes of $\mathbb{T} = 20000$ timesteps, for different average packet arrival rates μ^d at the data buffer.

In Figure 3.3, we consider the NOMA case, i.e., both UEs can offload at the same time. We reiterate that VI and PI are optimal since they have access to the transition and cost matrices, and thus have a guarantee of convergence. Indeed, VI and PI achieve the lowest negative rewards, outperforming the model-free methods (QL, 2QL and DQN). We notice that QL, 2QL and DQN offer remarkable performance while they are model-free, with QL performing better than DQN (since the latter is an approximation of the former). 2QL produces slightly better results compared to QL with the same number of training steps, proving thus the faster convergence claims. All these RL algorithms outperform the naive methods, proving the effectiveness of using a sequential decision-making or a reinforcement learning approach.

In Figure 3.4, we consider only the TDMA case, where only one UE is allowed to offload at a given time while the other one can still process its packets locally. We have similar comments except that the QL, 2QL and DQN achieve the optimality since the space to explore is strongly reduced without NOMA operations.

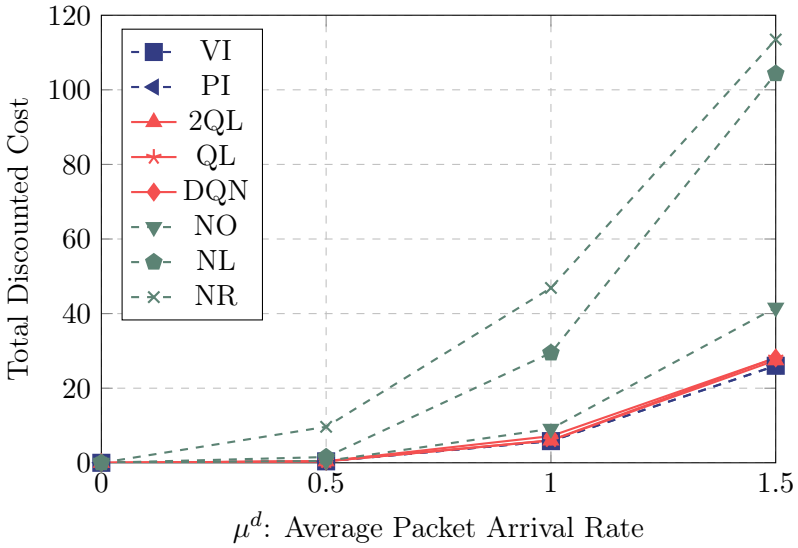

Figure 3.3: Total discounted cost (negative reward) averaged over 1000 episodes vs average packet arrival rate μ^d, with NOMA.

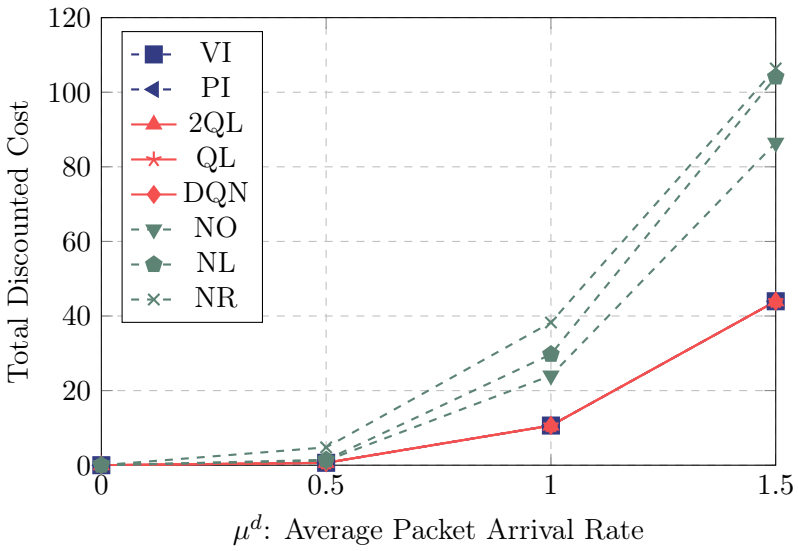

Figure 3.4: Total discounted cost (negative reward) averaged over 1000 episodes vs average packet arrival rate μ^d, with TDMA.

However, there is a noticeable difference between the performance of the used RL techniques in the NOMA setting and the TDMA setting. Indeed, Figure 3.5, Figure 3.6 and Figure 3.7 show that the introduction of NOMA in the system improves the overall discounted cost, and gives a cost up to 1.8 times lower compared to TDMA. Therefore, we illustrate that the use of NOMA is beneficial for our system even with a higher state space configuration, hence a slower convergence time.

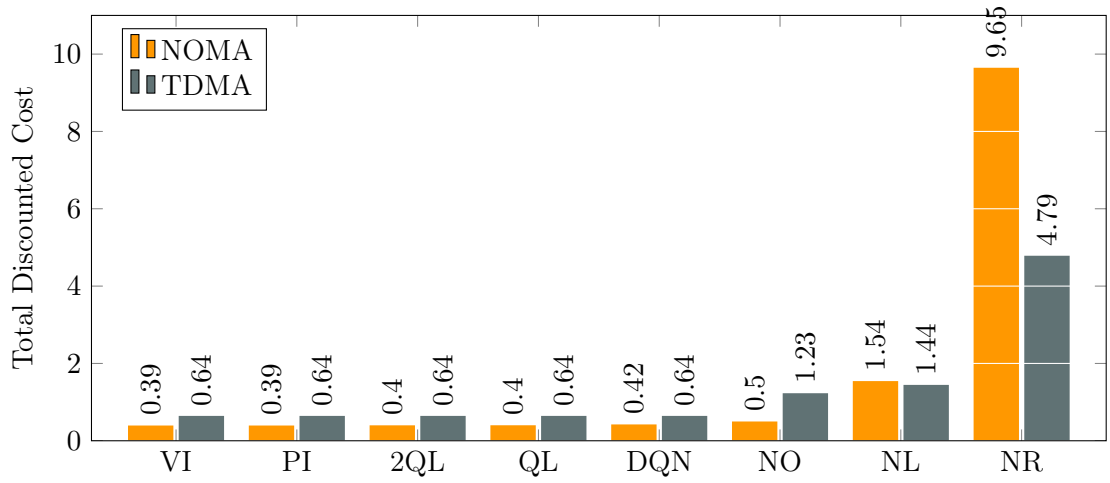

Figure 3.5: Total discounted cost comparison between NOMA and TDMA with $\mu^d = 0.5$.

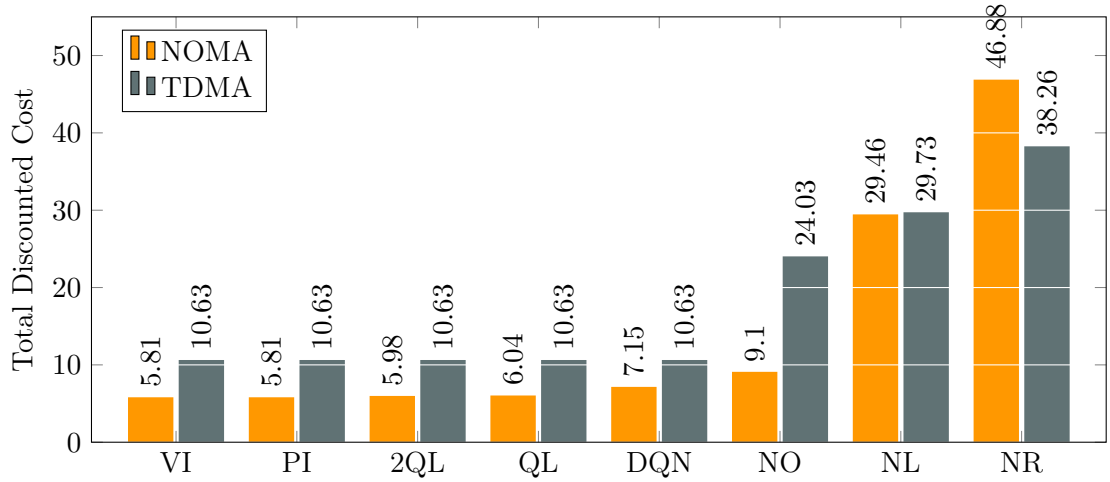

Figure 3.6: Total discounted cost comparison between NOMA and TDMA with $\mu^d = 1$.

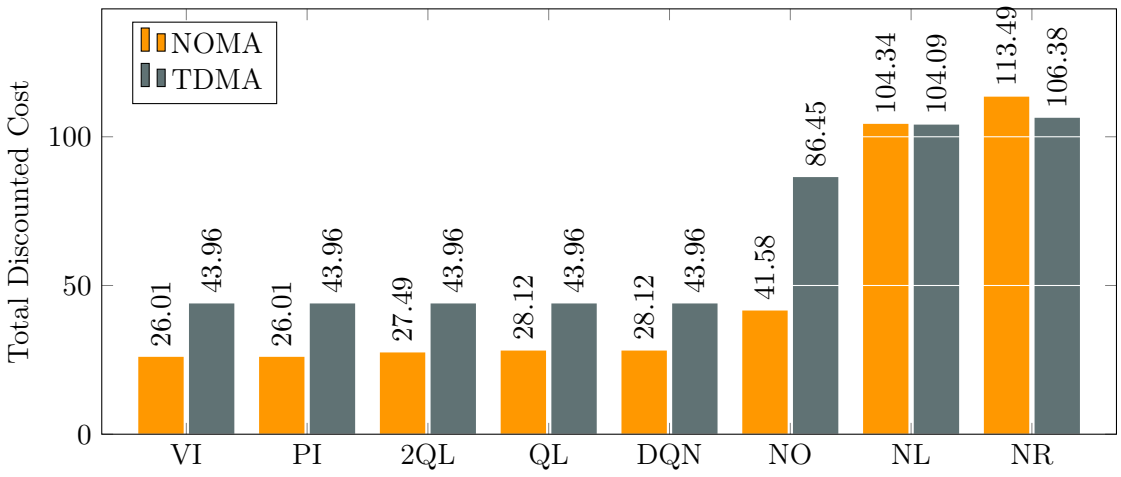

Figure 3.7: Total discounted cost comparison between NOMA and TDMA with $\mu^d = 1.5$.

In Figure 3.8, we plot a pie chart for the percentage of the different actions taken by each algorithm. For instance, Value and Policy Iterations take opposite approaches, in the sense that PI prefers local operations and offloading without NOMA, whereas VI has an "offload while you can" method utilizing more NOMA. QL and DQN balance the processing actions between local and offload. Therefore, NOMA is not used as often as in VI or rarely as in PI.

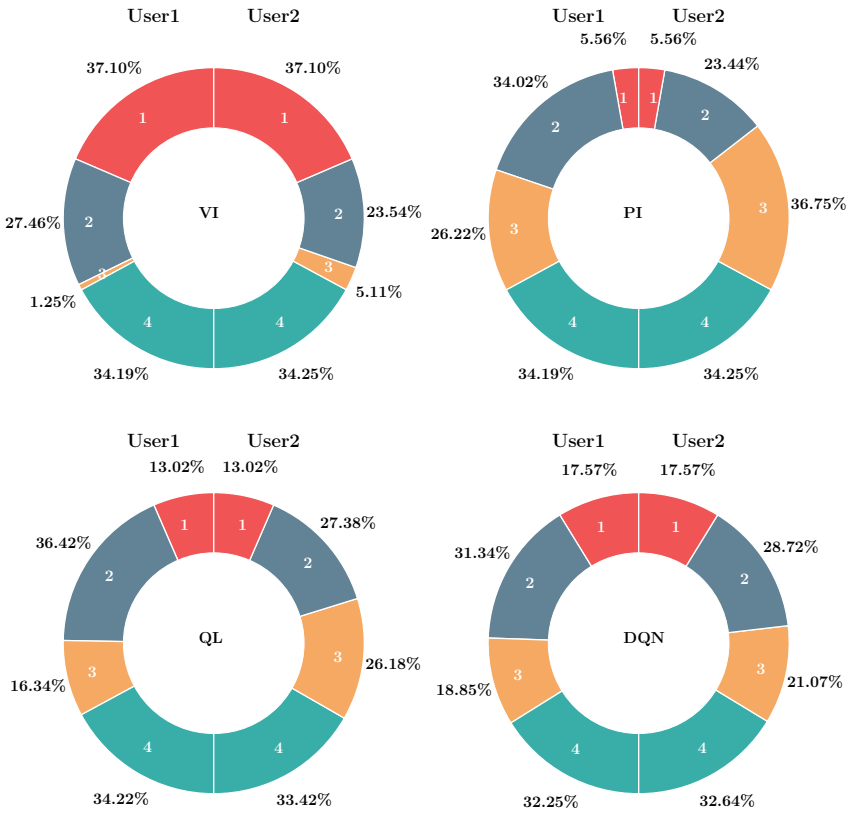

Figure 3.8: Pie chart of the percentage of actions taken during an episode with $\mu^d = 1.0$ for VI (NW), PI (NE), DQN (SW), and QL (SE) algorithms. (1/red) = NOMA, (2/gray) = Regular Offload, (3/orange) = Local, (4/green) = Idle.

We increase the state space size by considering a buffer size $\mathcal{S}^d = 4$, a number of channel states $|\mathcal{X}| = 4$, and a maximum packet delay $\nabla = 3$. The resulting state space is of size $|\mathbf{S}| = 78400$ possible states compared to $|\mathbf{S}| = 3600$ in the previous setup. Figure 3.9 shows the scalability performance of 2QL and DQN. PI and VI are not included as they take exponentially longer time to converge. We can see that DQN still performs better than the naive methods, while 2QL falls behind the NO method. This result affirms that the use of DQN and Neural Networks in general is essential when scaling the system setup.

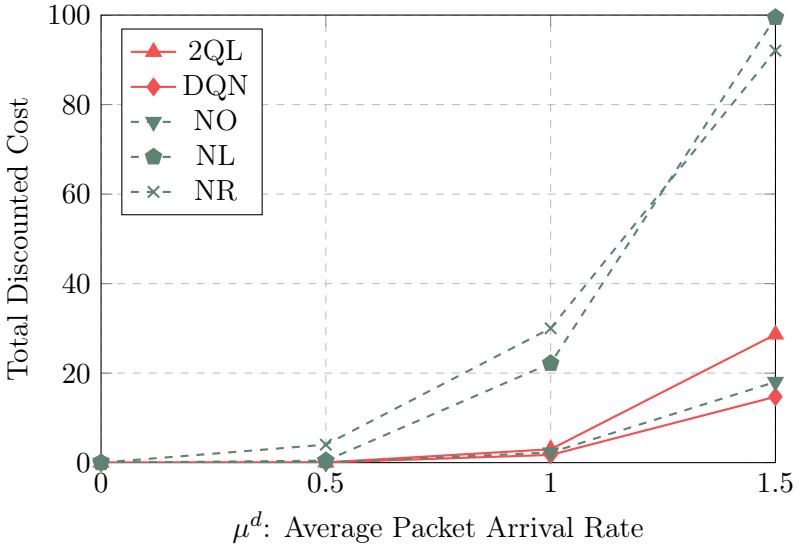

Figure 3.9: Performance of reinforcement learning algorithms : 2QL and DQN with a bigger state space configuration ($\mathcal{S}^d = 4, |\mathcal{X}| = 4, \nabla = 3$).

3.4 Extension to an EH-based System

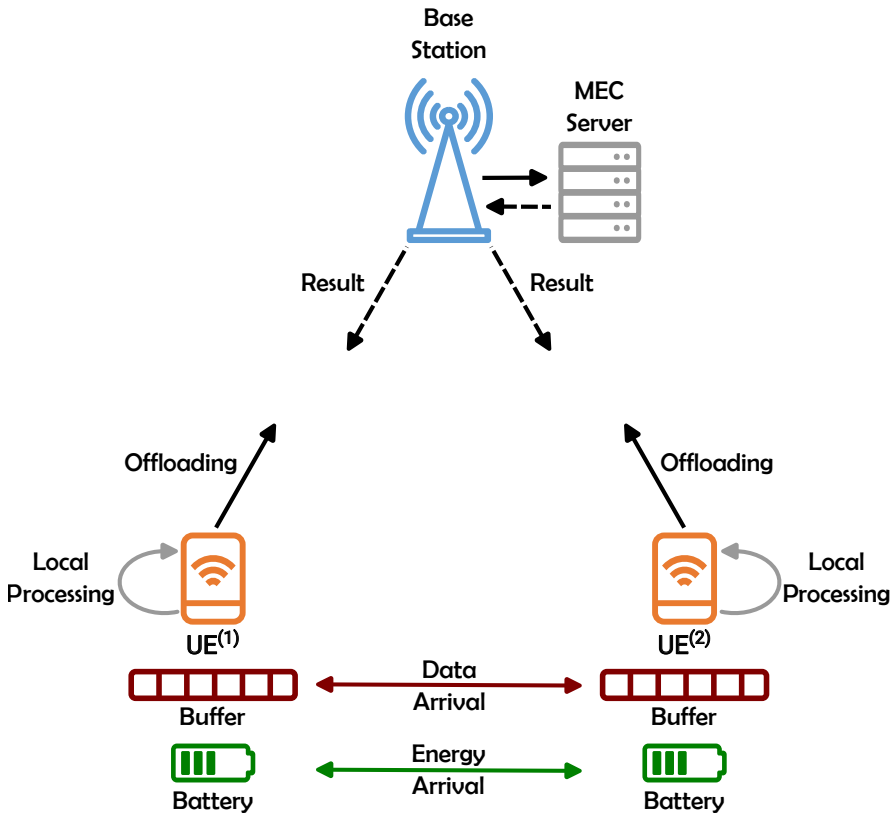

Figure 3.10: System model with energy harvesting.

We have established in the previous simulations that the use of RL methods provides a benefit over using naive methods. In addition, the scalability issue of the system may arise for some algorithms when the state space gets larger, while DRL algorithms scale well.

Therefore, in this section we extend the previous work by adding energy harvesting (EH) capabilities at the device level. More specifically, UEs will have a limited-size battery that can be recharged using ambient sustainable energy (e.g. solar, RF, ...). The system state space is further increased in size w.r.t. data buffer sizes and channel gain quantization steps. To solve the new optimization problem, we use the policy gradient DRL, namely proximal policy optimization PPO to design policies that minimize the packet loss. We compare PPO to the previously studied methods. In the sequel, we describe the changes made to the system model, in terms of energy harvesting EH, as well as the problem formulation and resolution. Then, we showcase the simulation results.

The system model is shown in Figure 3.10 with the addition of batteries, as well as the energy arrivals.

3.4.1 Energy & Battery Model

We suppose that both EH-capable devices $\text{UE}^{(1)}$ and $\text{UE}^{(2)}$ are equipped with batteries of finite capacity of $\mathcal{N}^b$ energy units. Each energy unit corresponds to $\mathcal{S}^e$ Joules. The arrival of energy packets is modeled as an i.i.d. random Poisson process with mean μ^b. At each timestep t, the captured energy, e_t, is stored in the battery while the excess energy is discarded. The battery level is denoted by $b_t^e \in [0, \mathcal{N}^b)$. The probability distribution is given by:

$$p(e_t = \mathcal{E}) = e^{-\mu^b} \cdot \frac{(-\mu^b)^{\mathcal{E}}}{\mathcal{E}!}. \tag{3.30}$$

The energy equations computed in section 3.1.3 are used to find the number of energy units consumed for each decided action.

3.4.2 Problem Formulation and Resolution

3.4.2.1 Problem Formulation

Similarly to the previous formulation, the system is modeled as an MDP. However, it now also incorporates the battery levels of the UEs. The new state space $\mathbf{S}$ is then defined as:

$$\mathbf{S} = \{\mathbf{d^{(1)}}, \mathbf{d^{(2)}}, \underline{x}^{(1)}, \underline{x}^{(2)}, b^{e,(1)}, b^{e,(2)}\}, \tag{3.31}$$

where $b^{e,(i)}$ represents the battery level of energy for $\text{UE}^{(i)}$ with $i \in \{1, 2\}$. The size of the state space becomes bounded by:

$$|\mathbf{S}| \leq \left[|\mathcal{X}| \cdot (\nabla + 2)^{\mathcal{S}^d} \cdot (\mathcal{N}^b + 1)\right]^2. \tag{3.32}$$

In addition to the state space, the transition model $\mathbf{T}$ changes such that it includes transition probabilities for the battery state as well. It is given by:

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t) = \prod_{i \in \{1,2\}} p(\mathbf{d}_{t+1}^{(i)}|\mathbf{d}_t^{(1)}, a_t) \cdot p(\underline{x}_{t+1}^{(i)}) \cdot p(b_{t+1}^{e,(i)}|b_t^{e,(i)}, a_t). \tag{3.33}$$

The energy states of the battery have to satisfy also some conditions.

- Current battery state should be greater or equal to the consumed energy.

$$b_t^e - e_t \geq 0.$$

- Next battery state should at least be equal to the difference between the current battery state and the consumed energy.

$$b_{t+1}^e \geq b_t^e - e_t.$$

The transition probabilities result in:
if $b_{t+1}^{e,(i)} < \mathcal{N}^b$ **then**:

$$p(b_{t+1}^{e,(i)}|b_t^{e,(i)}, a_t) = e^{-\mu^b} \cdot \frac{(\mu^b)^{b_{t+1}^{e,(i)} - b_t^{e,(i)} + \mathcal{N}_{t+1}^{e,(i)}}}{(b_{t+1}^{e,(i)} - b_t^{e,(i)} + \mathcal{N}_{t+1}^{e,(i)})!} \tag{3.34}$$

else:

$$p(b_{t+1}^{e,(i)}|b_t^{e,(i)}, a_t) = 1 - e^{-\mu^b} \sum_{j=0}^{\mathcal{N}^b - b_t^{e,(i)} + \mathcal{N}_{t+1}^{e,(i)} - 1} \frac{(\mu^b)^j}{j!}, \tag{3.35}$$

where $\mathcal{N}_{t+1}^{e,(i)}$ is the number of energy units collected at the next step, which is equal to the integer result of the ratio $e_{t+1}^{(i)}/\mathcal{S}^e$.

3.4.2.2 Problem Resolution

For the problem resolution, we use the proximal policy optimization (PPO), a policy gradient, actor-critic method that uses Neural Networks to approximate a policy, as previously explained in section 2.2.4.4. PPO offers great performance for large environments, which motivates us to investigate it in this problem. Given an input state $\mathbf{s}$, PPO uses two separate NNs, one for generating a policy $\pi^{\mathbf{w}}(.|\mathbf{s})$ (actor part) with weights $\mathbf{w}$, and another for generating the value function $\mathbf{V}^{\mathbf{w}'}(\mathbf{s})$ (critic part) with weights $\mathbf{w}'$. The objective function of PPO is described in Equations 2.36.

In our implementation of PPO, we use a shared layer architecture for both neural networks of the actor and the critic. This means that the first set of layers are shared between the actor and critic networks, while the other layers are independent for each network. Compared to an architecture where all layers are independent for both networks, the former approach led to better results in simulations.

The PPO method is compared with the DRL method (Double DQN) carried out in the previous section results. In particular, we use an improve DQN method by considering a dual architecture, as introduced in [65]. In this case, two networks are used to produce approximations of the value function and the advantage function, and a pooling function outputs the State-Action values from the previous outputs. This architecture allows to take the actions that can privilege values in the short term, or the long-term depending on the current state. Moreover, we implement prioritized Experience Replay (ER) [64] where DQN samples experiences from the memory buffer with higher TD-error more often than the ones with a lower TD-error. This approach helps accelerate the training by learning from unseen experiences more frequently.

3.4.3 Simulation Results

The numerical values used in these simulations are similar to the previous work with some differences, as mentioned what follows. The data buffer size is increased to $\mathcal{S}^d = 6$. The maximum delay is therefore $\nabla = 3$ms. The energy unit is $\mathcal{S}^e = 250$nJ, and the battery can accumulate up to $\mathcal{N}^b = 4$ energy units. The energy unit arrival rate for an EH-capable device is $\mu^e = 0.5$. The channel from UEs to MEC server is quantized into 5 discrete states $\mathcal{X} = [-20, -4.437, -1.487, 0.253, 1.492]$dB. The maximum number of offloaded packets is $\mathcal{M}^{\mathbf{o}} = 4$ and of local-processed packets is $\mathcal{M}^{\mathbf{l}} = 2$. This setup leads to a number of states $|\mathbf{S}|$ approximately equal to 27 million.

To train the PPO, we use two shared layers between the actor and critic networks, while two other layers are each independent for each network. We use 128 nodes for each layer in the network with ReLU activation function in the hidden layers. We train the agent for

10^5 epochs. For each epoch, we have $\mathbf{E} = 128$ episodes of $\mathbb{T} = 128$ timesteps leading to a batch of 16384 tuples. We set the mini-batch size to 128 and the number of iterations per batch to $\mathbf{I} = 10$. We use a fixed learning rate $\alpha = 5 \times 10^{-5}$, a clip factor $\epsilon = 0.2$, a discount factor is $\gamma = 0.99$ and a generalized advantage estimation factor $\lambda = 0.97$.

Moreover, we train the Dual Double DQN (3DQN) with Prioritized ER using a similar architecture to PPO, with two layers shared between the value function and the advantage function estimators, and two layers used for each estimator afterwards. We use for the DQN similar hyperparameters to the ones used for PPO including the number of nodes per layer, number of episodes, timesteps and batch sizes. Moreover, the learning rate and the discount factor remain the same.

In addition, we compare the PPO and DQN to the naive heuristics, namely the naive offload (NO), naive local (NL), naive random (NR), and the Immediate scheduler (IMM) that chooses the operation (local or offload) enabling the instantaneous maximum number of processed packets.

In Figure 3.11 and Figure 3.12, we plot the average percentage of dropped packets versus the packet arrival rate. The test has been done over $\mathbf{E} = 1000$ episodes of $\mathbb{T} = 1000$ timesteps. In the first figure, NOMA is considered, while on the second figure OMA/TDMA is considered. In both cases, the PPO approach outperforms all the naive methods. DQN and IMM approaches give similar performance for lower data arrival rates, but DQN approach experiences more losses for higher data rates.

Moreover, NO manages to perform better than the IMM approach, because IMM often chooses local processing actions, yielding thus to poor performance. In fact, NL executes more packets at the start, since it does not take into account the channel conditions. However, it is limited when the battery gets depleted quickly. On the contrary, NO is preserved from consuming more battery since the offloading operation depends on the channel conditions. Figure 3.13 and Figure 3.14 validate these observations.

Furthermore, the comparison between NOMA and TDMA shows that NOMA is much better in terms of packet loss, which means that the added complexity of using NOMA compared to TDMA due to the state space size (and possible loss in performance due to this complexity) is completely compensated for the better transmission speed with NOMA due to simultaneous users scheduling.

In Figure 3.13, we plot the average energy consumption (through the number of consumed energy units) per episode versus the packet arrival rates. The PPO approach is also better than the naive methods. So the method consumes less energy and loses less packets. DQN consumes slightly more energy than IMM, which could suggest that the DQN performs more local operations. NL consumes more than NO while performing poorly, which further highlights the previous comment.

For the TDMA case, the energy consumed is plotted in Figure 3.14 showing that using TDMA yields to higher energy consumption compared to NOMA. Indeed, the UEs have to

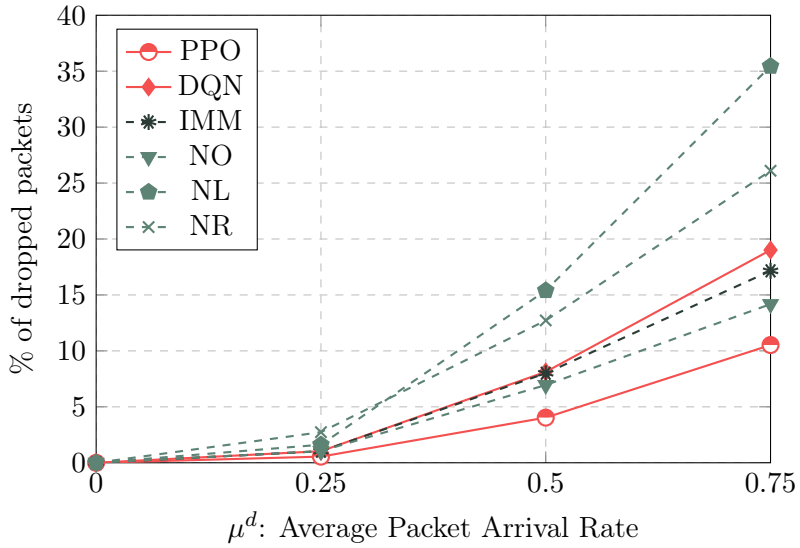

Figure 3.11: Percentage of dropped packets vs μ^d for NOMA.

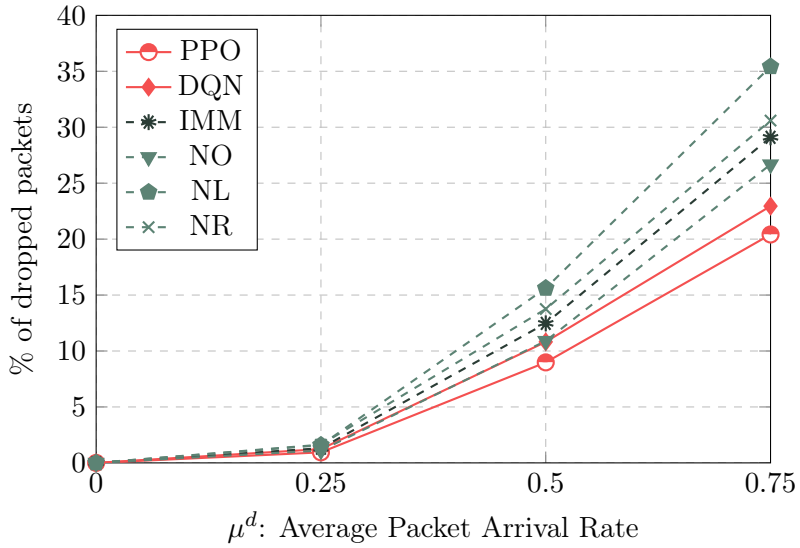

Figure 3.12: Percentage of dropped packets vs μ^d for TDMA.

offload as much as possible whenever they have the allocated resources, and perform more local processing when offloading is not allowed. This behavior makes NOMA more energy efficient while reducing the packet loss.

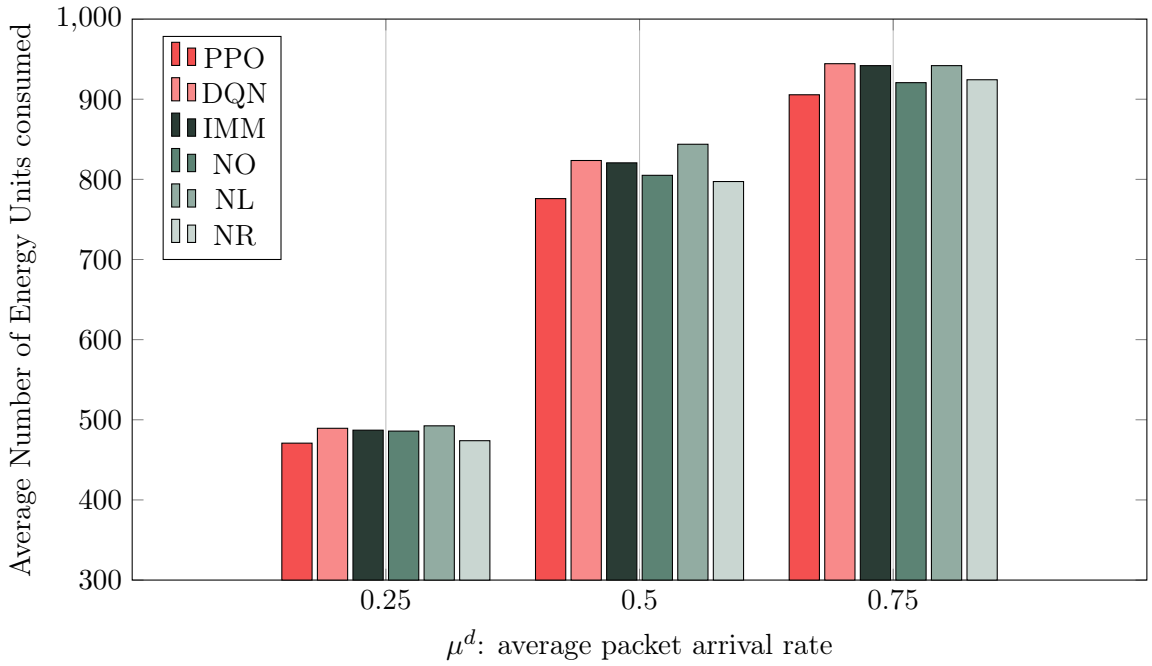

Figure 3.13: Number of consumed energy units per episode vs μ^d with NOMA.

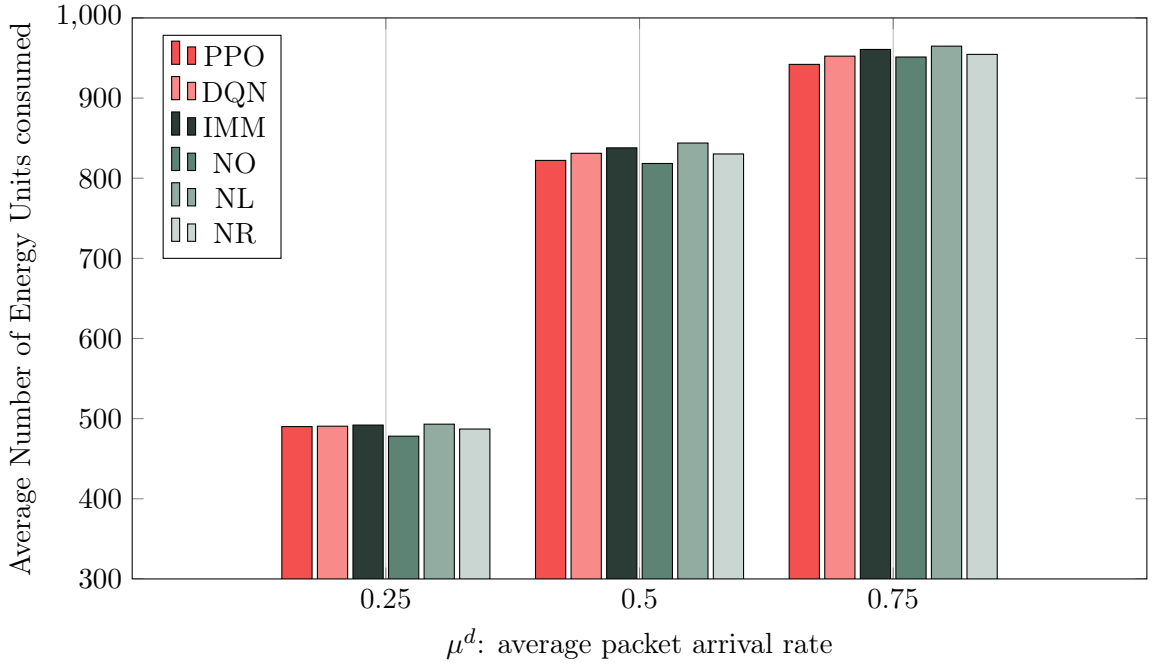

Figure 3.14: Number of consumed energy units per episode vs μ^d with TDMA.

3.5 Conclusion

This chapter has addressed the problem of jointly optimizing scheduling and offloading in a NOMA environment with 2 UEs and a MEC server at the base station. It is solved by using Reinforcement Learning techniques including optimal model-based value iteration/policy iteration algorithms and other model-free algorithms. In that context, NOMA technique has offered better performance than TDMA. Nevertheless, if the state space gets larger, scaling issue arises with some of the used algorithms, which makes the use of Neural Network-based RL methods even more important in larger systems.

Moreover, considering EH capabilities and finite size batteries at the devices further renders the problem more complex, thus justifying the need for Deep Reinforcement Learning methods. PPO has been used in this setup to learn the policies that allow for a lower packet loss compared to DQN and naive methods, while preserving more battery. The simulation results further reaffirm the previous claims about the use of NOMA when compared to TDMA. This work was published in the contributions **C1** and **C2**.

CHAPTER 4

Multi-Cluster System with Federated Reinforcement Learning

NOWADAYS, it is well known that MDP modeling of the environment in RL problems, especially how the state is represented, suffers from a curse of dimensionality issue. This occurs when the number of states increases exponentially with every component added to the model. Therefore, traditional RL model-based and model-free methods (e.g. Q-Learning) can no longer be viable due to the necessity to store every state and action values. In chapter 3, to overcome this issue, we have used DRL methods that rely on neural networks and produce RL models to learn the environment and act accordingly to the different scenarios that they might encounter. We have implemented variants of DRL methods to allow efficient exploration of the environment. In particular, we have considered DQN that approximates the Q-function and PPO actor-critic method that approximates the value function and the policy. The latter method has shown good potential in handling larger state spaces, achieving thus better performance.

However, even with the scalability advantage of DRL methods, the increase in the number of UEs in our setup causes the state space to be exponentially large. Training a DRL agent on such a large space requires a thorough exploration, and thus a longer convergence time. A solution to this issue is to separate the UEs into different *clusters* and to assign a DRL agent for each cluster. This is referred to as *Multi-Agent Reinforcement Learning* setup. While allowing for cooperation between the agents, this setup reduces the size of the state space per cluster, which can yield to faster convergence times.

One type of cooperation is enabled by using federated learning principle FL [24]. FL allows the distributed learning of multiple nodes with local weights and different datasets to converge onto one. Specifically, each local node uploads periodically its weights to a central node (a global one) where the weights are averaged and a new global model is broadcasted back to the local nodes. This process is repeated until convergence.

When applying FL to DRL methods, otherwise known as Federated Reinforcement Learning (FedRL), the global model produces a policy that takes into account the various configurations of the nodes and output actions accordingly. The first FedRL works were introduced in [71] and [72]. The latter is a DRL version of FedRL, in which two deep Q-networks were used to exploit separate environments and a global model was learned by using the output of each DQN.

By using FL in training DRL agents, a level of heterogeneity in the clusters can be learned and the global model can remain invariant to the differences between clusters. For instance, the clusters can have different numbers of UEs and each UE can have a different data arrival model representing an application's behavior (e.g. video streaming or web browsing). Such added functionalities in the system further contribute to making the problem more complex, and necessitating the FedRL techniques to converge on a general model, especially with the dynamic nature of the UEs.

On the other hand, allowing NOMA access between users when designing scheduling-offloading policies provides a boost in performance in terms of effective transmission rate, as we have observed in our previous contributions. However, we have considered in our prior system model only two UEs. In that case, the interference was mitigated relatively

easily via SIC decoding. With the objective of generalizing the system model, adding more UEs can hinder the performance of NOMA and its advantage over OMA methods. More specifically, sharing the channel resources between more UEs causes a performance drop in the achieved sum rates. This performance drop is due to the increasing interference in the decoding phase [52]. Indeed, SIC fails in decoding signals in practical systems due to the error propagation, which can be even more significant with more signals to decode. Moreover, performing the SIC in a sequential way from the strongest signal to the weakest at the UE level (Downlink) can consume considerable computational resources, resulting hence in higher decoding time.

The workaround to these issues is the separation of the UEs into clusters that utilize their own bandwidth in an FDMA setup. In this way, the amount of interference is reduced per cluster and SIC's decoding performance is improved. As aforementioned, the SIC decoding in DL is limited by the computational resources at UEs level. However, it is not the case for the Uplink case where the BS has the necessary computational power to handle the decoding for all the UEs. Therefore, we consider FDMA only in the DL case.

To fully exploit the advantages of this clustering, the common method is to cluster UEs with contrasting channel conditions together, meaning that a UE with good channel is paired with a UE with a bad channel. This type of clustering allows for power allocation in the DL to be efficient, and thus results in good spectral efficiency. In addition, the ever-changing channel conditions of the UEs and the dynamics of the number of active users require a dynamic clustering process. Some works [53]–[56] have been cited in chapter 1.

In this chapter, we address a NOMA multi-cluster system incorporating various numbers of UEs connected to a BS with a nearby MEC server. DRL methods are ought to adapt to this different system configuration. Thus, we consider a multi-agent DRL setup where each cluster has its own agent and learns the environment contained within the cluster. With the dynamic nature of the clusters, the agents learn a generic model that can adapt to each configuration of the UEs.

More specifically, the goal of this chapter's work is to devise policies capable of jointly scheduling and offloading computations of the UEs by utilizing FedRL with PPO. The buffer, channel and battery states of each UE are used as information to decide the actions to take. Additionally, the UEs are defined by their location on the grid, in terms of angles and distance from the BS [1], and NOMA clustering is performed based on this information. Each cluster has a DRL agent stored in a Cluster Head (CH), which is a randomly selected UE from the cluster, and the global model will be stored in the MEC server. The CH produces a decision for all the UEs in the cluster (including itself), and the process is continued until re-clustering whenever the location of the UEs change.

We briefly overview in what follows some works related to our work, that utilize the multi-agent RL or the FedRL, where both methods differ in the way the agents cooperate. While FedRL utilizes a global model that aggregates the local models, multi-agent RL refers to a more general type of cooperation (or lack thereof).

[1] the distance is expressed in terms of the channel gain

With regards to FedRL technique, the work [84] presented a concurrent federated reinforcement learning scheme for resource allocation, where the main goal was to preserve the privacy of the edge hosts and the server. The distributed agents decided in a concurrent way while sharing their outputs and rewards and not their models. The authors in [85] proposed an intelligence ultra-dense edge computing framework for AI and Blockchain applications. It jointly optimized resource allocation, application partitioning and server caching. The edge user can offload its tasks either to an edge server, a mobile device in proximity, or to a cloud server. To train the model, FL was used with DRL, where each agent's model, located at the edge device level is trained locally, and the weights are sent into the macro base station where they are averaged and a new global model is broadcasted, in a similar way to our scenario. The paper [86] tackled the problem of vehicular communications. It investigated the reuse of cellular channels that were already allocated to perform the communication, whilst not interrupting the existing cellular operations and avoiding collisions with other vehicle-to-vehicle links. Federated multi-agent DRL was used to get policies that determine the transmission powers and the cellular channels. The double DQN with dual architecture was considered and the FL scheme consisted of sharing the local weights with the central node followed by broadcasting the global averaged model. The work in [87] addressed challenges in ensuring high-quality healthcare in the 6G era through the integration of wearable medical devices into the Internet of medical things. Leveraging wireless body area network and MEC technologies, the study focused on optimizing the quality of service with ultra-reliable data transfer, processing at low latency, and energy usage. The proposed FedRL task offloading approach relied on the sharing of local weights with the central node to perform the averaging operation.

As for Multi-agent RL methods, the authors in [88] explored a computation offloading problem in IoT networks with edge computing for multiple "selfish" users. They addressed challenges in resource allocation and competition for spectrum and radio access technologies resources. A multi-agent RL framework using Q-Learning was used to solve the problem with the RL agents located at the users without sharing information (non-cooperative). The work in [89] investigated the use of Unmanned Aerial Vehicles (UAVs) as aerial BSs for cost-effective and on-demand wireless communications. A dynamic resource allocation scheme was formulated where the UAVs select the communicating users, power levels, and sub-channels, without any information exchanged between them. Multi-agent RL with Q-Learning was also used to find a solution. The authors in [90] designed an indoor wireless communication architecture. The mobile users were served by access points using multiple simultaneously transmitting and reflecting re-configurable intelligent surfaces with NOMA capabilities. The problem of NOMA user pairing was solved by relying on the channel correlation between the users and by using K-Means to perform the clustering. In addition, the beamforming problem was solved by a Multi-agent PPO method to optimize the beamforming vector.

Notably, both categories of works perform a multi-agent RL setup. However, our contribution differs from the rest in the dynamic aspect of the nodes. In our case, the clusters could have varying numbers of UEs, and thus the local model has to accommodate

for that change by having a generic representation of the state.

The remainder of the chapter is organized as follows:

- Section 4.1 presents the system model and each UE structure, with the channel model, transmission model, data buffer model, and battery and energy model. In addition to that, the scheduling decisions, the corresponding consumed energy and the time constraints are also highlighted.
- Section 4.2 details the problem formulation as an MDP and its resolution using Federated PPO, with a detailed description of the clustering and learning processes.
- Section 4.3 describes the simulation parameters and the results obtained when running the Federated PPO compared with the standard methods.
- Section 4.4 concludes the chapter with some final remarks.

4.1 System Model

We consider a system model consisting of a BS, with a close MEC server, serving $\mathcal{N}^{\mathbf{UE}}$ active UEs $\{\mathbf{UE}_1, \cdots, \mathbf{UE}_i, \cdots, \mathbf{UE}_{\mathcal{N}^{\mathbf{UE}}}\}$ with EH capabilities, limited-size batteries and data buffers. Due to the large number of UEs, we separate them into $\mathcal{N}^{\mathbf{C}}$ distinct clusters $\{\mathbf{C}_1, \cdots, \mathbf{C}_k, \cdots, \mathbf{C}_{\mathcal{N}^{\mathbf{C}}}\}$ to facilitate the communication and reduce the complexity in decision-making. We set the maximum number of clusters to $\mathcal{N}^{\mathbf{C}} \leq \mathcal{N}^{\mathbf{C}}_{\max}$ and the number of UEs per cluster $\mathbf{C}_k$ to $\mathcal{N}^{\mathbf{UE}}_{\min} \leq \mathcal{N}^{\mathbf{UE}}_{k} \leq \mathcal{N}^{\mathbf{UE}}_{\max}$. Before the start of each episode of transmission e with $\mathbb{T}$ timesteps, a clustering process is done and the UEs are distributed among the clusters based on their channel gains and their angles w.r.t. the BS. Figure 4.1 illustrates the system model.

We assume that each cluster $\mathbf{C}_k$ has a cluster head CH, randomly selected among the UEs in the cluster. At the start of each timestep t of size $\mathcal{T}$, the buffer, channel and battery information of all the UEs in the cluster are shared with the CH. Such information enables it to decide on the action to take by each UE. The decision is then broadcasted to these UEs for free.

Furthermore, at the start of the episode, the average channel gain of all the $\mathbf{UE}_{ik}$ within a cluster $\mathbf{C}_k$ is transmitted to the other clusters via the BS. After that, no information is shared between the clusters for the remainder of the episode.

We consider NOMA operations in both the UL and the DL with some differences. In the UL phase, we assume that all the UEs from all clusters can send simultaneously their data packets to the BS using the whole available bandwidth, and the SIC operation is performed globally on all UEs. Obviously, inter-cluster and intra-cluster interference terms are present and have to be dealt with. The CH cannot compute the inter-cluster interference precisely since it has information only on the average channel gain of the channel states of other clusters' UEs. Therefore, an estimation is done to account for this interference, and underestimating it results in transmission errors (i.e., the transmission rate will be higher than the Shannon limit). On the counterpart, DL operations are done in

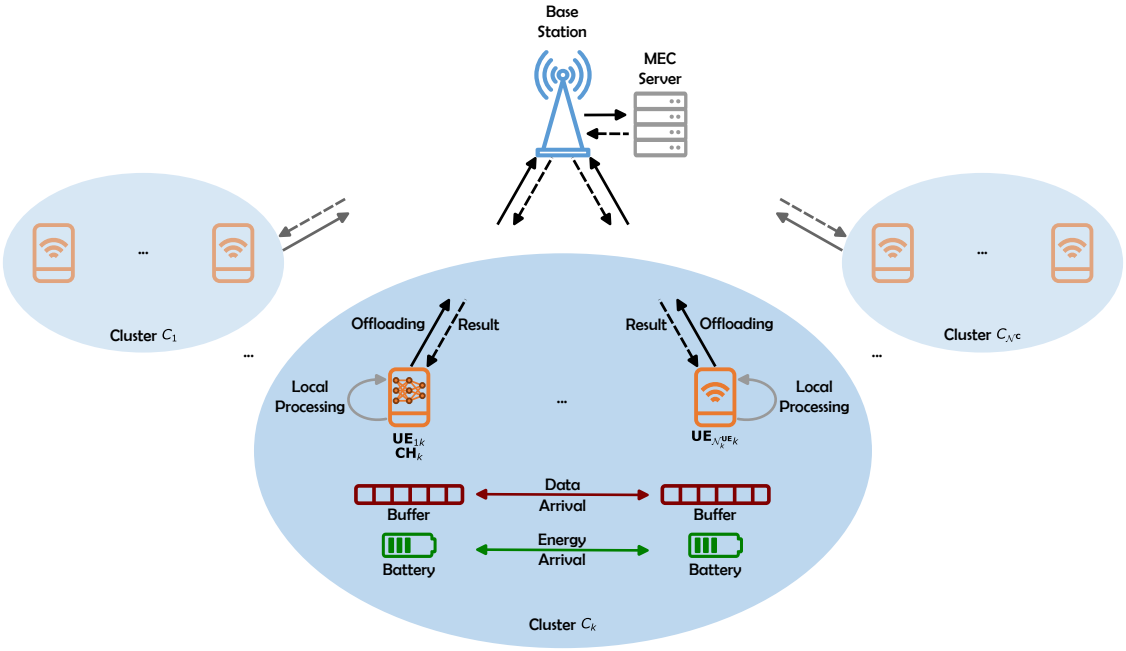

Figure 4.1: Multi-cluster system model with one cluster head CH for each cluster, which has the decision-making model.

NOMA for each cluster separately. This means that FDMA is performed on a cluster-basis to eliminate the inter-cluster interference and enhance the performance of SIC decoding. The differences in UL and DL scenarios stem from the computational advantage of the BS compared to the CH.

Similarly to our previous works, each UE can either process the data arrived at its buffer locally, or remotely by offloading the packets to the MEC server, whilst accounting for packets delays, channel gains, and battery levels. The EH and battery models are carried over from the previous contributions. However, the channel, transmission and data buffer models are different. Therefore, we detail them in the following subsections.

4.1.1 Channel Model

Following the previous system model in chapter 3, we model the channel between the UEs and the BS as Rayleigh flat-fading channel with AWGN ($\mathcal{N}_0$ being the noise spectral density). For a cluster $\mathbf{C}_k$, each $\mathbf{UE}_{ik}$ with $i \in \{1, \cdots, \mathcal{N}_k^{\mathbf{UE}}\}$ has a channel complex amplitude h_{ik}, and a gain $x_{ik} = |h_{ik}|^2$. The CSIT is assumed at the CH level and the channel response is constant for the duration $\mathcal{T}$ of a timestep t.

We further divide the channel gain range into multiple sub-intervals that refer to categorical channel gain conditions (e.g. bad SNR, mid SNR, good SNR). Each sub-interval is bounded by a maximum and a minimum value, and each UE's channel gain will vary within this sub-interval for the duration of an episode before transitioning to a different channel sub-interval.

Formally, we define $\mathbf{X}$ as the total range of channel gain values, which can take values in $\mathbb{R}$, and $\mathbf{X}^j \in [x^j_{\min}, x^j_{\max}]$ is the j^{th} sub-interval of channel gain, with $x^j_{\min}$ and $x^j_{\max}$ being its lower and upper bounds, respectively. Channel gain variations in time within the

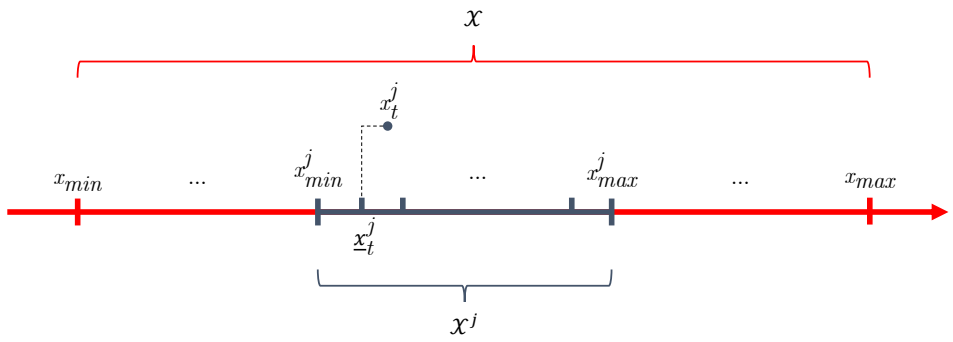

Figure 4.2: Channel modeling and quantization.

sub-interval are modeled as an i.i.d. random process following the exponential distribution with mean μ_j^c:

$$p(x_t^j = \mathcal{C}) = \frac{1}{\mu_j^c} e^{\frac{-\mathcal{C}}{\mu_j^c}}. \tag{4.1}$$

We quantize the values of the channel to make the space finite. For a specific $\mathbf{UE}_{ik}$, a quantization function $\mathcal{Q}$ is defined that projects the UE's channel gain x_{ik}^j into a discrete space with finite values $\underline{x}_{ik}^j = \mathcal{Q}(x_{ik}^j)$. The quantized channel gain $\underline{x}_{ik}^j$ is the lower bound in the interval where x_{ik}^j is contained, i.e., $\underline{x}_{ik}^j \leq x_{ik}^j < \overline{x}_{ik}^j$ with $\overline{x}_{ik}^j$ the interval upper bound. We use the lower bound value to allow the transmission in a worst case scenario even when different value is used (contrary to the upper bound value).

The resulted global range of quantized channel gains is $\mathcal{X}$ and the quantized sub-interval is denoted as $\mathcal{X}^j$, with $|\mathcal{X}^j|$ indicating the number of discrete values in the sub-interval, and $|\mathcal{J}|$ indicating the number of sub-intervals. We also bound the range $\mathcal{X}$ between two values denoted as $x_{\min}$ and $x_{\max}$. Furthermore, we model the transition from one sub-interval to another as a correlated process to simulate a natural change of the channel (due to a location change). Therefore, the probability of transitioning from quantized sub-interval $\mathcal{X}^j$ to $\mathcal{X}^{j'}$ when starting a new episode $e+1$ for $\mathbf{UE}_{ik}$ is defined as follows:

$$p(\mathcal{X}_{ik,e+1} = \mathcal{X}^{j'} | \mathcal{X}_{ik,e} = \mathcal{X}^j) = \frac{\left(1-\rho^{\mathcal{X}}\right)^{|j'-j|}}{\sum_{k=1}^{|\mathcal{J}|}\left(1-\rho^{\mathcal{X}}\right)^{|k-j|}}, \tag{4.2}$$

where $\rho^{\mathcal{X}}$ is the channel range correlation factor. Figure 4.2 represents the described channel.

4.1.2 Transmission Model

For NOMA transmission in the UL, we assume that all UEs can transmit at the same time using the entire available bandwidth $\mathcal{W}^{ul}$. The BS can decode all signals using SIC since it has full knowledge of channel conditions of all UEs. Therefore, intra-cluster as well as inter-cluster interference terms are present in the rate equations in this case. However, in the DL case, we use FDMA between clusters, where the bandwidth $\mathcal{W}_k^{dl}$ is allocated for each cluster $\mathbf{C}_k$. NOMA is then performed on a cluster level eliminating thus the inter-cluster interference in the rate equations.

4.1.2.1 Uplink Transmission

We assume that $\mathbf{UE}_{ik}$ which belongs to cluster $\mathbf{C}_k$ with $i \in \{1, \cdots, \mathcal{N}_k^{\mathbf{UE}}\}$ and $k \in \{1, \cdots, \mathcal{N}^{\mathbf{C}}\}$ is transmitting to the BS, i.e., offloading packets, $o_{ik} = 1$. The action associated is $a_{ik} > \mathcal{M}^{\mathbf{l}}$ (previously explained in subsection 3.2 as we will reintroduce the notations in subsection 4.2). We further suppose, for simplicity of computations, that the UEs in the cluster are ordered from best channel gain to worst , i.e., $\underline{x}_{ik} \geq \underline{x}_{jk}$ if $i < j$. The UL rate for $\mathbf{UE}_{ik}$ is then expressed as:

$$\mathcal{R}_{ik}^{ul} = \mathcal{W}^{ul} \cdot \log_2 \left(1 + \frac{p_{ik}^{\mathbf{o}} \cdot \underline{x}_{ik}}{\mathcal{I}_{ik}^{ul,intra} + \widetilde{\mathcal{I}}_{ik}^{ul,inter} + \mathcal{W}^{ul} \cdot \mathcal{N}_0} \right), \tag{4.3}$$

with

$$\mathcal{I}_{ik}^{ul,intra} = \sum_{j=i+1}^{\mathcal{N}_k^{\mathbf{UE}}} p_{jk}^{\mathbf{o}} \cdot \overline{x}_{jk} \cdot o_{jk}, \tag{4.4}$$

$$\widetilde{\mathcal{I}}_{ik}^{ul,inter} = \sum_{\substack{k'=1 \\ k' \neq k}}^{\mathcal{N}^{\mathbf{C}}} \left[\sum_{j=1}^{\mathcal{N}_{k'}^{\mathbf{UE}}} \left(\mathcal{P}^{\mathbf{o}} \cdot z_{k'} \cdot \mathbb{1} \left(\underline{x}_{ik} > z_{k'} \right) \right) \right], \tag{4.5}$$

being the intra-cluster interference and the inter-cluster interference estimation, respectively. $p_{ik}^{\mathbf{o}}$ is the offloading power of $\mathbf{UE}_{ik}$, which is limited to a maximum value $\mathcal{P}^{\mathbf{o}}$. $z_{k'}$ is the average channel gain for all the UEs in cluster $\mathbf{C}_{k'}$.

Intuitively, the intra-cluster interference term considers all the signals from the offloading UEs with a lower channel gain than $\mathbf{UE}_{ik}$. This term is accurately computed with the assumption that the channel gains and decisions are known within the cluster $\mathbf{C}_k$. The inter-cluster interference, on the other hand, is an estimation based on the information available at the cluster. It considers the maximum offloading power with the average channel gain $z_{k'}$ of all clusters $\mathbf{C}_{k'}$ with $k' \neq k$, if $\mathbf{UE}_{ik}$ channel gain is higher than $z_{k'}$. In fact, we assume that the SIC decoding order in UL starts with the strongest signals and goes to the lowest and that all UEs offload ($o_{jk'} = 1$ for $k' \in \{1, \cdots, \mathcal{N}^{\mathbf{C}}\},\ j \in \{1, \cdots, \mathcal{N}_{k'}^{\mathbf{UE}}\}$).

The true capacity of the channel can be obtained only at the BS by estimating correctly the inter-cluster interference knowing the offloading powers of each UE. It is computed as follows:

$$\mathcal{C}_{ik}^{ul} = \mathcal{W}^{ul} \cdot \log_2 \left(1 + \frac{p_{ik}^{\mathbf{o}} \cdot \underline{x}_{ik}}{\mathcal{I}_{ik}^{ul,intra} + \mathcal{I}_{ik}^{ul,inter} + \mathcal{W}^{ul} \cdot \mathcal{N}_0} \right), \tag{4.6}$$

with

$$\mathcal{I}_{ik}^{ul,inter} = \sum_{\substack{k'=1 \\ k' \neq k}}^{\mathcal{N}^{\mathbf{C}}} \left[\sum_{j=1}^{\mathcal{N}_{k'}^{\mathbf{UE}}} \left(p_{jk'}^{\mathbf{o}} \cdot \overline{x}_{jk'} \cdot o_{jk'} \cdot \mathbb{1} \left(\underline{x}_{ik} > \overline{x}_{jk'} \right) \right) \right]. \tag{4.7}$$

Therefore, if the UL rate $\mathcal{R}_{ik}^{ul}$ used by $\mathbf{UE}_{ik}$ to transmit its data is higher than the channel capacity $\mathcal{C}_{ik}^{ul}$, a transmission error would occur. More specifically, if the estimated interference is less than the exact one, i.e., $\widetilde{\mathcal{I}}_{ik}^{ul,inter} < \mathcal{I}_{ik}^{ul,inter}$, a rate mismatch happens.

4.1.2.2 Downlink Transmission

Similarly to the UL case, we assume the same ordering of the offloading UEs in cluster $\mathbf{C}_k$ based on their channel gains. Therefore, the DL rate equation for $\mathbf{UE}_{ik}$ is as follows:

$$\mathcal{R}_{ik}^{dl} = \mathcal{W}_k^{dl} \cdot \log_2 \left(1 + \frac{\delta_{ik} \cdot p_{bs}^{\mathbf{b}} \cdot \underline{x}_{ik}}{\mathcal{I}_{ik}^{dl,intra} + \mathcal{W}_k^{dl} \cdot \mathcal{N}_0} \right), \tag{4.8}$$

where δ_{ik} is the power allocation coefficient and $p_{bs}^{\mathbf{b}}$ is the total broadcast power by the BS. Thus, $\delta_{ik} p_{bs}^{\mathbf{b}}$ is the portion of the total power allocated to $\mathbf{UE}_{ik}$. We compute the power allocation coefficient based on the channel gain, where a smaller portion of the power is allocated to UEs with stronger channel gains. It is expressed as:

$$\delta_{ik} = \frac{o_{ik}}{\underline{x}_{ik} \cdot \sum_{j=1}^{\mathcal{N}_k^{\mathbf{UE}}} \frac{o_{jk}}{\underline{x}_{jk}}}. \tag{4.9}$$

$\mathcal{I}_{ik}^{dl,intra}$ is the DL intra-cluster interference coming from the other UEs that offload and have stronger channel gains than $\mathbf{UE}_{ik}$:

$$\mathcal{I}_{ik}^{dl,intra} = \sum_{j=1}^{i-1} \delta_{jk} \cdot p_{bs}^{\mathbf{b}} \cdot \underline{x}_{ik} \cdot o_{jk}, \tag{4.10}$$

and

$$\mathcal{W}_k^{dl} = \mathcal{W}^{dl} \cdot \frac{\mathcal{N}_k^{\mathbf{UE}}}{\mathcal{N}^{\mathbf{UE}}}, \tag{4.11}$$

is the allocated bandwidth to the cluster $\mathbf{C}_k$ with FDMA. Obviously, more UEs in the cluster translates to allocating more bandwidth to this cluster.

4.1.3 Data Buffer Model

Each $\mathbf{UE}_{ik}$ is equipped with a limited-size buffer storing the data that arrives and has to be executed within a strict delay. We model the buffer structure in the same way as in the previous chapter, with a vector $\mathbf{d}_{ik}$ of size $\mathcal{S}^d$. An arrival of packets $q_{ik,t}$ into the buffer are set to an age of 0, whereas an empty slot is set to -1. The $\mathcal{N}_{ik,t}^d$ packets in the buffer keep aging with each timestep until reaching a maximum delay ∇, where they are dropped due to delay violation (c^{dv}). In case of insufficient number of empty slots in the buffer when a new batch of data packets arrives, i.e., $q_{ik,t} > \mathcal{S}^d - \mathcal{N}_{ik,t}^d$, the exceeding number of packets is also dropped due to buffer overflow (c^{bo}). Moreover, the packets lost due to the transmission error (c^{te}) are retained in the buffer for the next timestep to be reprocessed [2].

To introduce some heterogeneity in the UEs data, which reflects several realistic applications that the UEs could be running, we model different data arrivals for each $\mathbf{UE}_{ik}$. In particular, we use the data arrival models associated with video streaming, gaming, and other applications, that are described using statistical formulations in [46]. We detail the distributions of these data models in what follows.

[2] A similar structure to the Automatic Repeat Request (ARQ) method, with the packet counter being the same as the maximum delay ∇

Poisson Distribution Suited for IoT applications, Poisson random distribution is also used in our previous work. The arrival of packets with mean $\mu^d_{\mathbf{po}}$ follows:

$$p(q_{ik,t} = \mathcal{D}) = e^{-\mu^d_{\mathbf{po}}} \cdot \frac{(\mu^d_{\mathbf{po}})^{\mathcal{D}}}{\mathcal{D}!}. \tag{4.12}$$

Uniform Distribution According to the survey in [46], the uniform distribution models the packet arrival for gaming applications, as adopted in 3GPP and IEEE traffic models. The arrival of packets is modeled with equal probability for every value between $\mathcal{D}_{\mathbf{u},\min}$ and $\mathcal{D}_{\mathbf{u},\max}$:

$$p(q_{ik,t} = \mathcal{D}) = \frac{1}{\mathcal{D}_{\mathbf{u},\max} - \mathcal{D}_{\mathbf{u},\min}}. \tag{4.13}$$

Lognormal Distribution Truncated Lognormal distribution is used for modeling FTP traffic and web-browsing applications in 3GPP and IEEE traffic standards. The arrival of packets is modeled with mean $\mu^d_{\mathbf{l}}$ (and considering the variance $\sigma^2 = 1$) as follows:

$$p(q_{ik,t} = \mathcal{D}) = \frac{1}{\sqrt{2\pi}\mathcal{D}} \cdot e^{\frac{-(\log \mathcal{D} - \mu^d_{\mathbf{l}})}{2}}. \tag{4.14}$$

The distribution is naturally suited for continuous values, which is not the case in our system as we consider the arrival of discrete number of packets. Therefore, we round the results produced by the distribution to the nearest discrete value.

Pareto Distribution For video streaming applications, the standards 3GPP and IEEE model their traffic following the truncated Pareto distribution. The arrival of packets in this case is modeled with a mean $\mu^d_{\mathbf{pa}}$, and distribution parameters $b > 0$ and $c > 1$, as follows:

$$p(q_{ik,t} = \mathcal{D}) = \frac{b}{1 - (c)^{-b}} \cdot \frac{1}{(\mathcal{D})^{b+1}}. \tag{4.15}$$

The arrival of packets $\mathcal{D}$ is bounded between 1 and c. Similar to the Lognormal distribution, Pareto distribution is for continuous values as well. Therefore a rounding operation is necessary to ensure discrete values of packet arrivals.

In our system model, we assign randomly the distributions to each $\mathbf{UE}_{ik}$ at the start of the transmission process, resulting in clusters with different data arrivals and thus different requirements.

4.1.4 Energy and Battery Model

The energy and battery model remains unchanged from the previous chapter, where each $\mathbf{UE}_{ik}$ has a battery with a capacity of $\mathcal{N}^b$ energy units. The battery can be recharged by harvesting energy from external ambient sources. Each energy unit corresponds to $\mathcal{S}^e$ Joules. We assume that for each timestep t, a $e_{ik,t}$ amount of energy units arrive at $\mathbf{UE}_{ik}$'s battery, following the Poisson distribution with mean μ^b:

$$p(e_{ik,t} = \mathcal{E}) = e^{-\mu^b} \cdot \frac{(-\mu^b)^{\mathcal{E}}}{\mathcal{E}!}. \tag{4.16}$$

The battery level, i.e., number of energy units in it, at timestep t is denoted as $b^e_{ik,t} \in [0, \mathcal{N}^b)$.

4.1.5 Scheduling Decisions

The cluster head CH is the decision center for its own cluster, and thus handles the scheduling decisions for all the UE in this cluster independently from other clusters, then broadcasts them for free. At the start of a timestep t, the CH $\mathbf{CH}_k$ of cluster $\mathbf{C}_k$, upon receiving the necessary information from all $\mathbf{UE}_{ik}$ with $i \in \{1, \cdots, \mathcal{N}_k^{\mathbf{UE}}\}$, produces the scheduling decisions (idle, local processing, or offloading) as well as the number of packets m_{ik} for all these $\mathbf{UE}_{ik}$ as the following:

- *Idle*: $\mathbf{UE}_{ik}$ will not process any packet, thus $m_{ik,t} = 0$.
- *Local Processing*: $\mathbf{UE}_{ik}$ will process locally $m_{ik,t}$ packets, which cannot exceed a maximum number $m_{ik,t} \leq \mathcal{M}^{\mathbf{l}}$.
- *Offloading*: $\mathbf{UE}_{ik}$ will offload $m_{ik,t}$ packets to be processed remotely at the MEC server located near the BS. A limit of $\mathcal{M}^{\mathbf{o}}$ is set on the number of offloaded packets $m_{ik,t} \leq \mathcal{M}^{\mathbf{o}}$.

4.1.6 Consumed Energy

We compute the energy associated with each scheduling decision to determine the number of energy units required to perform these decisions.

- *Idle* $\mathbf{i}$: $\mathbf{UE}_{ik}$ does not consume any energy:

$$\mathcal{E}_{ik,t}^{\mathbf{i}} = 0. \tag{4.17}$$

- *Local Processing* $\mathbf{l}$: $\mathbf{UE}_{ik}$ processes locally $m_{ik,t}$ packets with power $\mathcal{P}^{\mathbf{l}}$ per processed packet. The corresponding energy is thus

$$\mathcal{E}_{ik,t}^{\mathbf{l}} = m_{ik,t} \cdot \mathcal{P}^{\mathbf{l}} \cdot \mathcal{T}. \tag{4.18}$$

- *Offloading* $\mathbf{o}$: $\mathbf{UE}_{ik}$ offloads $m_{ik,t}$ packets with power $p_{ik,t}^{\mathbf{o}}$ that cannot exceed the maximum offloading power $\mathcal{P}^{\mathbf{o}}$. We distinguish two cases:
  - Offloading with no transmission errors (no rate mismatch):
    $\mathbf{UE}_{ik}$ sends its packets, awaits their execution at the MEC server, receives the broadcasted signal by the BS, and decodes the result. The corresponding consumed energy, highlighting each step, is given by:

$$\mathcal{E}_{ik}^{\mathbf{o}} = m_{ik}\Bigg(\underbrace{\frac{\mathcal{L}^{ul} \cdot p_{ik}^{\mathbf{o}}}{\mathcal{R}_{ik}^{ul}}}_{\text{Transmission}} + \underbrace{\mathcal{T}^{\mathbf{w}} \cdot p^{\mathbf{w}}}_{\text{Waiting}} + \underbrace{\max_{i' \in [1, \mathcal{N}_k^{\mathbf{UE}}]} \left[\frac{m_{i'k} \cdot o_{i'k}}{\mathcal{R}_{i'k}^{dl}} \right] \cdot \frac{\mathcal{L}^{dl} \cdot p^{\mathbf{r}}}{m_{ik}}}_{\text{Reception}} + \underbrace{\rho \cdot \left(\frac{\mathcal{L}^{dl} \cdot p^{\mathbf{d}}}{\mathcal{R}_{ik}^{dl}} + \sum_{j=i+1}^{\mathcal{N}_k^{\mathbf{UE}}} \frac{m_{jk}}{m_{ik}} \cdot \frac{\mathcal{L}^{dl} \cdot p^{\mathbf{d}}}{\mathcal{R}_{jk}^{dl}} \cdot o_{jk} \right)}_{\text{Decoding}} \Bigg). \tag{4.19}$$

Note that the reception at $\mathbf{UE}_{ik}$ requires receiving all the offloading UEs' signals. Thus, the reception accounts for the lowest DL rate and the highest number of packets among UEs' signals in the term $\max_{i' \in [1, \mathcal{N}_k^{\mathbf{UE}}]} \left[\frac{m_{i'k} \cdot o_{i'k}}{\mathcal{R}_{i'k}^{dl}} \right]$, which indicates the reception of the signals with the slowest time. In addition, decoding $\mathbf{UE}_{ik}$'s signal requires decoding all the offloading UEs' signals with lower channel gains (due to the higher allocated power) in a sequential manner, following SIC decoding.

– Offloading with transmission errors (rate mismatch):
We assume that $\mathbf{UE}_{ik}$ sends its packets but a timeout happens due to not receiving the ARQ acknowledgement. This indicates that a rate mismatch has occurred. The consumed energy is:

$$\mathcal{E}_{ik}^{\mathbf{o},error} = m_{ik} \left(\frac{\mathcal{L}^{ul} \cdot p_{ik}^{\mathbf{o}}}{\mathcal{R}_{ik}^{ul}} + \mathcal{T}^{\mathbf{w}} \cdot p^{\mathbf{w}} \right). \tag{4.20}$$

4.1.7 Time Constraints

Successful offloading operation occurs only when the offloading, waiting, receiving and decoding processes are performed within the fixed timestep duration $\mathcal{T}$. Following the derivations in chapter 3 and using equation (4.19), we consider the times of the different steps and formulate the inequality as:

$$m_{ik} \left(\frac{\mathcal{L}^{ul}}{\mathcal{R}_{ik}^{ul}} + \mathcal{T}^{\mathbf{w}} + \max_{i' \in [1, \mathcal{N}_k^{\mathbf{UE}}]} \left[\frac{m_{i'k} \cdot o_{i'k}}{\mathcal{R}_{i'k}^{dl}} \right] \cdot \frac{\mathcal{L}^{dl}}{m_{ik}} + \rho \cdot \left(\frac{\mathcal{L}^{dl}}{\mathcal{R}_{ik}^{dl}} + \sum_{j=i+1}^{\mathcal{N}_k^{\mathbf{UE}}} \frac{m_{jk}}{m_{ik}} \cdot \frac{\mathcal{L}^{dl}}{\mathcal{R}_{jk}^{dl}} \cdot o_{jk} \right) \right) \leq \mathcal{T}. \tag{4.21}$$

We can obtain the optimal offloading power $p_{ik}^{\mathbf{o}}$ by forcing equality in the above expression and by using the explicit form of $\mathcal{R}_{ik}^{ul}$ in equation (4.3). Thus, we obtain:

$$p_{ik}^{\mathbf{o}} = \left(2^{\zeta_{ik}} - 1 \right) \cdot \frac{\mathcal{I}_{ik}^{ul,intra} + \widetilde{\mathcal{I}}_{ik}^{ul,inter} + \mathcal{W}^{ul} \cdot \mathcal{N}_0}{\underline{x}_{ik}}, \tag{4.22}$$

with

$$\zeta_{ik} = \frac{\mathcal{L}^{ul}}{\mathcal{W}^{ul} \cdot \left(\frac{\mathcal{T}}{m_{ik}} - \mathcal{T}^{\mathbf{w}} - \max_{i' \in [1, \mathcal{N}_k^{\mathbf{UE}}]} \left[\frac{m_{i'k} \cdot o_{i'k}}{\mathcal{R}_{i'k}^{dl}} \right] \cdot \frac{\mathcal{L}^{dl}}{m_{ik}} - \rho \cdot \left(\frac{\mathcal{L}^{dl}}{\mathcal{R}_{ik}^{dl}} - \sum_{j=i+1}^{\mathcal{N}_k^{\mathbf{UE}}} \frac{m_{jk}}{m_{ik}} \cdot \frac{\mathcal{L}^{dl}}{\mathcal{R}_{jk}^{dl}} \cdot o_{jk} \right) \right)}. \tag{4.23}$$

4.2 Problem Formulation and Resolution

4.2.1 Problem Formulation

Our goal in this chapter is to find optimal scheduling-offloading policies that account for the available information on the data buffers of the UEs, their battery states, as well as their channel conditions, and decide on the actions, i.e., packets' execution locally or remotely, to minimize packet losses. Therefore, similarly to our previous works, we formulate the problem as an MDP with: a state space $\mathbf{S}$ describing the environment information, an action space $\mathbf{A}$ dictating the possible scheduling decisions to take from states, and a reward function $\mathbf{R}$ giving feedback on the actions taken for each state [3]. In the sequel, we define each component of the MDP tailored to our system model.

4.2.1.1 MDP Structure

- State Space $\mathbf{S}$: Each cluster $\mathbf{C}_k$ has a cluster head $\mathbf{CH}_k$, a randomly selected UE in the cluster, that performs the decision-making for all the UEs. The available information at $\mathbf{CH}_k$ is then the buffer, battery, and channel gain states of the cluster UEs, as well as the average channel gains of the other clusters. Therefore, the state representation, denoted as $\mathbf{S}_k$, is specific for each cluster and is defined as:

$$\mathbf{S}_k = \{\mathbf{d}_k, \underline{\mathbf{x}}_k, \mathbf{b}_k, \mathbf{z}_k\}, \tag{4.24}$$

with $\mathbf{d}_k$, $\underline{\mathbf{x}}_k$ and $\mathbf{b}_k$ being respectively the buffer vectors, quantized channel gains and battery levels of the UEs in $\mathbf{C}_k$. $\mathbf{z}_k$ is the set of average channel gains for other clusters $\mathbf{C}_{k'}$ with $k' \in \{1, \cdots, \mathcal{N}^{\mathcal{C}}\}$ and $k' \neq k$. We formulate each component with the following equations:

$$\mathbf{d}_k = \{\mathbf{d}_{1k}, \mathbf{d}_{2k}, \cdots, \mathbf{d}_{\mathcal{N}_k^{\mathbf{UE}}k}\}, \tag{4.25}$$

$$\underline{\mathbf{x}}_k = \{\underline{x}_{1k}, \underline{x}_{2k}, \cdots, \underline{x}_{\mathcal{N}_k^{\mathbf{UE}}k}\}, \tag{4.26}$$

$$\mathbf{b}_k = \{b^e_{1k}, b^e_{2k}, \cdots, b^e_{\mathcal{N}_k^{\mathbf{UE}}k}\}, \tag{4.27}$$

$$\mathbf{z}_k = \{z_1, \cdots, z_{k-1}, z_{k+1}, \cdots, z_{\mathcal{N}^{\mathbf{C}}}\}. \tag{4.28}$$

To compute the number of possibilities for the average cluster channel gains vector $\mathbf{z}_k$, we need to take into account the total number of UEs in the clusters $\mathcal{N}^{\mathbf{UE}}$ that can vary between $\mathcal{N}^{\mathbf{UE}}_{\min}$ and $\mathcal{N}^{\mathbf{UE}}_{\max}$, the number of clusters $\mathbf{C}_{k'} \neq \mathbf{C}_k$, and the number of channel ranges. Therefore, the maximum number of possible states of $\mathbf{z}_k$, denoted as $|\mathcal{Z}|$ counts all the possible average cluster channel gains, and is bounded by:

$$|\mathcal{Z}| \leq \left[\sum_{i=\mathcal{N}^{\mathbf{UE}}_{\min}}^{\mathcal{N}^{\mathbf{UE}}_{\max}} (|\mathcal{J}|)^i \right]^{\left(\mathcal{N}^{\mathbf{C}}_{\max}-1\right)}. \tag{4.29}$$

[3] A transition model is also a part of MDP modeling as we have seen previously, but we deem modeling it not necessary with the use of model-free RL algorithms as we will see in problem resolution

Assuming that the channel sub-intervals have the same number of discrete channel gain values, i.e., $|\mathcal{X}^j| = |\mathcal{X}^{j'}|, \forall\{j, j'\} \in \{1, \cdots, |\mathcal{J}|\}$, we have $|\mathcal{X}| = |\mathcal{X}^j| \cdot |\mathcal{J}|$. The cluster state space size is thus bounded by

$$|\mathbf{S}_k| \leq \left[|\mathcal{X}| \cdot (\nabla + 2)^{\mathcal{S}^d} \cdot (\mathcal{N}^b + 1)\right]^{\mathcal{N}^{\mathbf{UE}}_{\max}} \cdot |\mathcal{Z}|. \tag{4.30}$$

- Action Space $\mathbf{A}$: Each cluster has a specific action space concerning only the UEs of said cluster. It is denoted as $\mathbf{A}_k$ and it represents the type of processing (idle, local or offload) as well as the number of packets m to execute. Therefore, we define a_k as the index that represents the set of actions for all UEs in cluster $\mathbf{C}_k$. a can take values ranging from 0 to $|\mathbf{A}_k| - 1$, where $|\mathbf{A}_k|$ is the action space size bounded by:

$$|\mathbf{A}_k| \leq (\mathcal{M}^{\mathbf{l}} + \mathcal{M}^{\mathbf{o}} + 1)^{\mathcal{N}^{\mathbf{UE}}_{\max}}, \tag{4.31}$$

where $\mathcal{M}^{\mathbf{l}}$ and $\mathcal{M}^{\mathbf{o}}$ are the maximum numbers of packets that can be executed locally or remotely, respectively. Each $\mathbf{UE}_{ik}$ action index a_{ik} can be extracted using a base changing from decimal to base $u = \mathcal{M}^{\mathbf{l}} + \mathcal{M}^{\mathbf{o}} + 1$. This is done in a similar way to the former definition of action space in section 3.2:

$$\overline{a_k}^{10} = \overline{a_{1k} \cdots a_{ik} \cdots a_{\mathcal{N}^{\mathbf{UE}}_k k}}^{u}. \tag{4.32}$$

The specific actions can be then derived for each $\mathbf{UE}_{ik}$ as the following:

  - $a_{ik} = 0$ corresponds to the idle action with $m^{(i)} = 0$.
  - $0 < a_{ik} \leq \mathcal{M}^{\mathbf{l}}$ corresponds to local action with $m^{(i)} = a_{ik}$.
  - $\mathcal{M}^{\mathbf{l}} < a_{ik} \leq \mathcal{M}^{\mathbf{o}}$ corresponds to offloading action with $m^{(i)} = a_{ik} - \mathcal{M}^{\mathbf{l}}$.

- Reward Function $\mathbf{R}$: A reward is associated to each state and action taken from this state in a given cluster, providing a feedback on the state-action pair. We define the reward function $\mathbf{R}_k$ following a policy π_k for a cluster $\mathbf{C}_k$ as the expected negative sum of the packet losses. The losses are due to delay violation c^{dv} and buffer overflow c^{bo}n in addition to the transmission error c^{te} for all UEs in the cluster. The reward is expressed for an infinite horizon model with a discount factor γ as:

$$\mathbf{R}^{\pi}_k = -\lim_{\mathbb{T}\to\infty} \mathbb{E}^{\pi}\left[\sum_{t=0}^{\mathbb{T}} \sum_{i=1}^{\mathcal{N}^{\mathbf{UE}}_k} (\gamma)^t \left(c^{dv}_{ik,t} + c^{bo}_{ik,t} + c^{te}_{ik,t}\right)\right]. \tag{4.33}$$

We aim to find the optimal policy that achieves the maximum expected reward for each cluster. It is obtained as:

$$\pi^{\star}_k = \arg\max_{\pi_k} \mathbf{R}^{\pi}_k. \tag{4.34}$$

Notably, all the clusters' policies converge to the same policy $\pi^{\star}_k = \pi^{\star}_{k'}$, for $(k, k') \in \{1, \cdots, \mathcal{N}^{\mathbf{C}}\}$ through the use of FL, as we shall explain in the next section.

4.2.2 Proposed Resolution

Finding the optimal policy $\pi_k^\star$ for each cluster is hard to achieve analytically, therefore we revert to iterative methods that attempt to converge to the solution. In addition, as we have a complex and large state in this problem, we need to consider model-free methods that can learn the environment by trial and error. Using model-based methods in this setup, where it requires full knowledge of the MDP structure (including the transition function), is unfeasible in practical implementation as it can take exponentially long time to reach the optimal policy.

Moreover, we have established in the previous chapters that DRL-based methods using neural networks can be a powerful tool to get good policies that achieve low packet losses. However, training the DRL agent on a large state space while considering all UEs at once makes the problem very difficult. This has motivated us to introduce clustering of the UEs to eventually simplify the problem for the DRL agent.

In our multi-cluster setup, we require thus a *multi-agent* RL model, where each agent runs on a CH of a cluster. A cooperation process is enabled between clusters by using Federated Learning FL, a learning framework for distributed models. Each model runs locally at the CH level using heterogeneous data different from other models (providing thus data privacy). A global model is then aggregated from all the local models at the central node, which is the MEC server. The changes in the clustering of UEs necessitate that the Federated Reinforcement Learning agents learn flexibility in the system.

In the sequel, we describe the multi-agent training procedure, the clustering model, and the investigated DRL algorithm, namely the proximal policy optimization PPO.

4.2.2.1 Multi-Agent Training Procedure

Learning policies using multi-agent federated reinforcement learning FedRL follows several steps, as illustrated in Figure 4.3. We assume that the information exchange happens in a separate bandwidth from the transmission one, and that the time required to do this is negligible. Each cluster $\mathbf{C}_k$ has a local FedRL agent with weights $\mathbf{w}_k$ that produce a policy π_k, while the MEC server has the global model with weights $\mathbf{w_g}$ and policy $\pi_\mathbf{g}$. At the end of the training procedure, all the clusters have the same global model with weights $\mathbf{w}$ and policy π. Details of each sequential step are listed below.

- First at the start of episode e, the UEs are grouped into $\mathcal{N}_e^\mathbf{C}$ distinct clusters, where one cluster head $\mathbf{CH}_k$ is selected at random to be the decision-making agent for all UEs in $\mathbf{C}_k$. The BS transmits to $\mathbf{CH}_k$ the average of all the UEs' channel gains in other clusters, represented by the vector $\mathbf{z}_k$, and the new episode e starts.
- During the episode e, and at a timestep t, each $\mathbf{UE}_{ik}$ in cluster $\mathbf{C}_k$ with $i \in \{1, \cdots, \mathcal{N}_\mathbf{C}^\mathbf{UE}\}$ shares with $\mathbf{CH}_k$ its buffer vector $\mathbf{d}_{ik,t}$, its quantized channel gain $\underline{x}_{ik,t}$, and its battery level $b_{ik,t}^e$.
- The $\mathbf{CH}_k$ uses the information provided at each timestep to decide on the action to take $a_{k,t}$ using the FedRL model, and broadcasts the individual action $a_{ik,t}$ to each $\mathbf{UE}_{ik}$ in the cluster. The obtained reward is also shared with the cluster head. This information sharing process is highlighted in Figure 4.4.

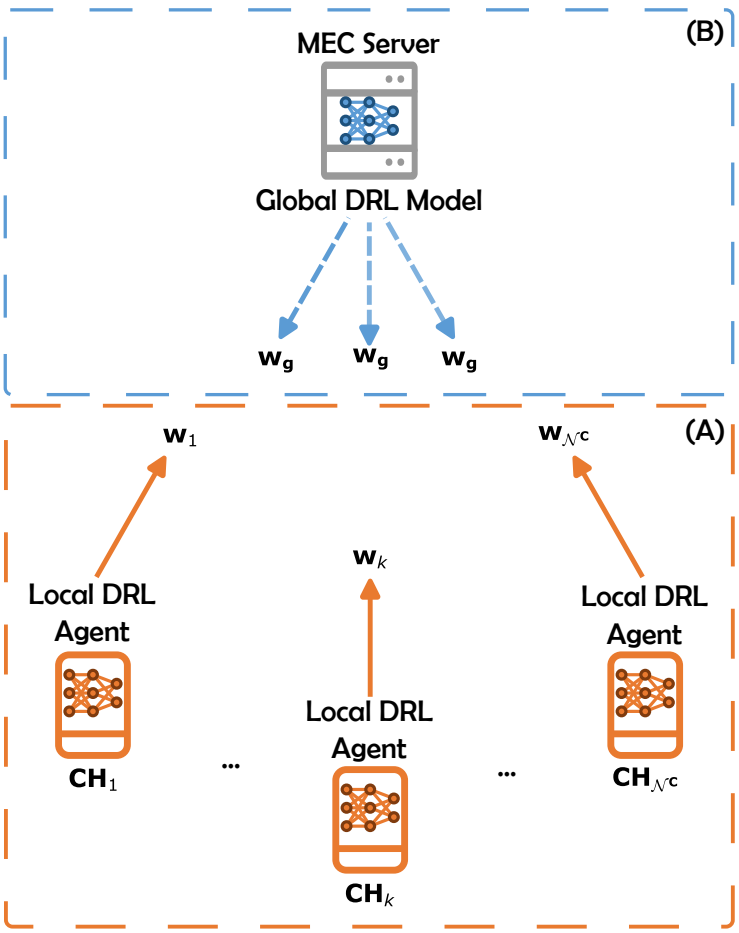

Figure 4.3: Federated reinforcement learning procedure. (A) : The local models trained with clusters information, and transmitted to the central node (MEC Server). (B) : The global model weights aggregated from the received local weights, and broadcasted to the nodes (CHs).

- The information shared by the UEs in $\mathbf{C}_k$ to $\mathbf{CH}_k$ is then used to train the local agent with weights $\mathbf{w}_k$. After a certain number of steps (e.g. after a certain number of timesteps or episodes), the local weights of each cluster head agent are sent to the MEC server.

- After receiving the local weights from the CHs, the MEC server averages the weights and obtains the weights of the global model as:

$$\mathbf{w_g} = \frac{1}{\mathcal{N}^{\mathbf{C}}} \sum_{k=1}^{\mathcal{N}^{\mathbf{C}}} \mathbf{w}_k. \tag{4.35}$$

- The MEC server broadcasts the global model weights to all CHs, and the process repeats until the system converges to a global solution, i.e., $\forall k,\ \mathbf{w} = \mathbf{w}_k,\ \pi = \pi_k$, invariant to different data models and numbers of UEs per cluster. In particular, the re-clustering process takes place at each episode. The learning procedure is illustrated in Figure 4.3.

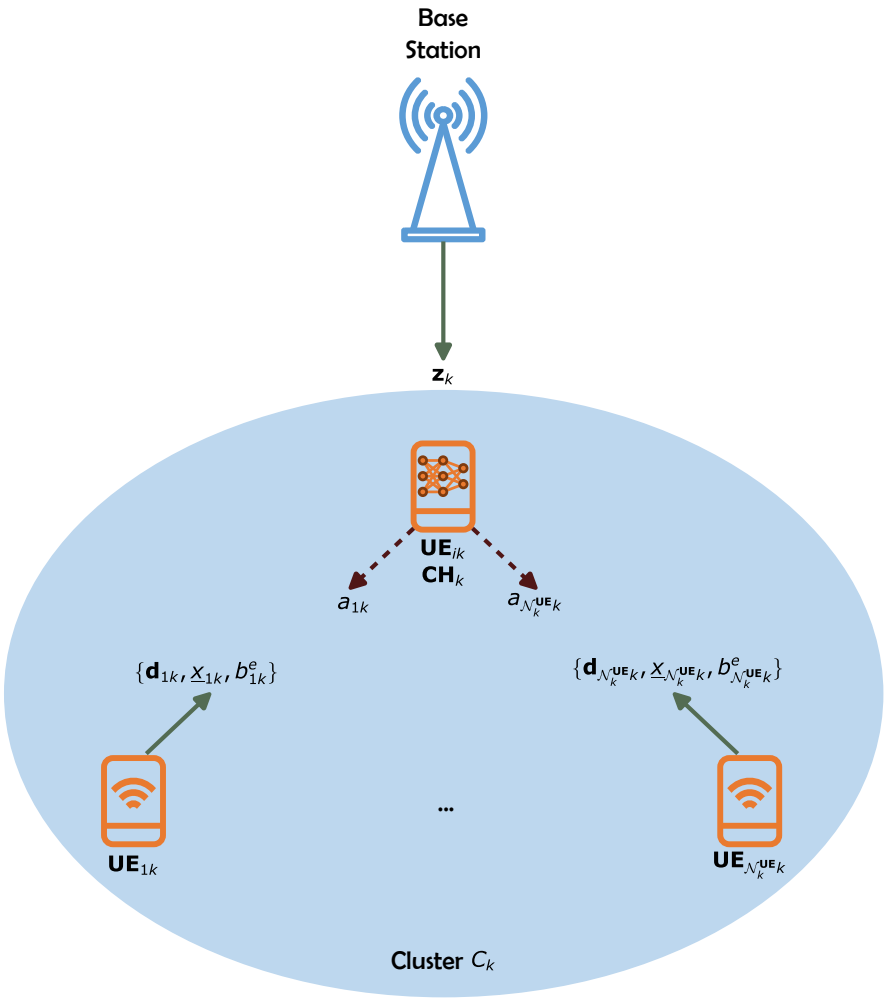

Figure 4.4: Information sharing with the cluster head.

4.2.2.2 Clustering Model

Allowing multiple devices to share the network resources when transmitting signals simultaneously, in time and frequency, is what gives NOMA the advantage in terms of spectral efficiency compared to traditional OMA methods. In addition, as demonstrated in our previous works, the use of NOMA yields to fewer packet losses, and thus better system design.

However, the performance advantage of NOMA decreases when the number of devices simultaneously served by the BS increases. This is due to the SIC decoding especially in the DL, which becomes more time and energy consuming with limited computational power at the UEs compared to BS. Hence, creating multiple NOMA instances that occupy different bandwidths and handling fewer numbers of devices is a good alternative to the performance issues.

Therefore, a clustering process needs to be put in place to group together devices that have desirable characteristics NOMA-wise. The general criterion for clustering NOMA devices is based on channel conditions, where devices with higher SNR values and others with weaker SNRs values are paired together. This is done to maximize the resulted DL

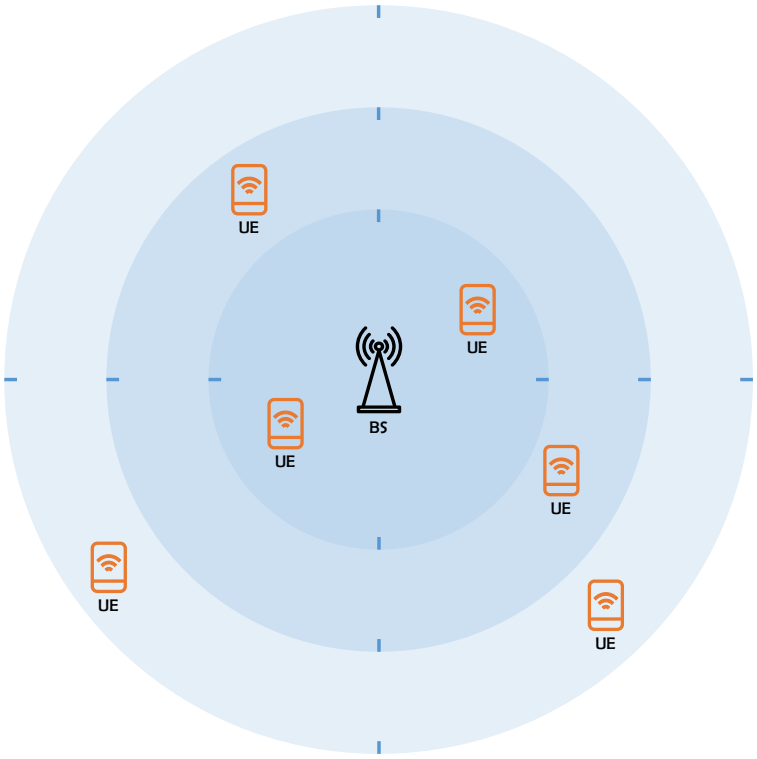

Figure 4.5: Displacement of UEs in the grid around the BS, with different channel sub-intervals (levels) and angles.

rates according to [53].

Some existing works on clustering are mentioned in section 1.4. The main concern with these methods is that the location of the devices with respect to the BS is not considered. This indicates that high SNR devices can be clustered with low SNR devices, even if they are located in opposite directions at either side of the BS. This is an issue in practice especially with the directional beams emitted by the BS's antennas, which can favor one device over the other.

The clustering solution proposed in our work deals with two conflicting issues jointly. The first one is the clustered UEs' channel conditions that have to be contrasting, while the second issue is the need for a spatial proximity between UEs in order to cluster them. In Figure 4.5, we show an example where the UEs can be located. They are defined in our case by their angle w.r.t. the BS. Each circle with a shade of blue corresponds to a quantized channel sub-interval $\mathcal{X}^j$ (distance from the BS), and UEs can vary within the sub-interval as explained in section 4.1.1. Moreover, two UEs can be in the same sub-interval with different angles, which determines the position of the UEs w.r.t. the BS.

In other words, we use the polar coordinates (r, θ) to determine the position of UE in the grid, where r refers to the channel sub-interval and θ to the angle. The polar coordinates allow thus to cluster the UEs that have different r levels and the same (or adjacent) angles θ. The resulting clusters are represented in Figure 4.6, where each cluster is highlighted by the green color.

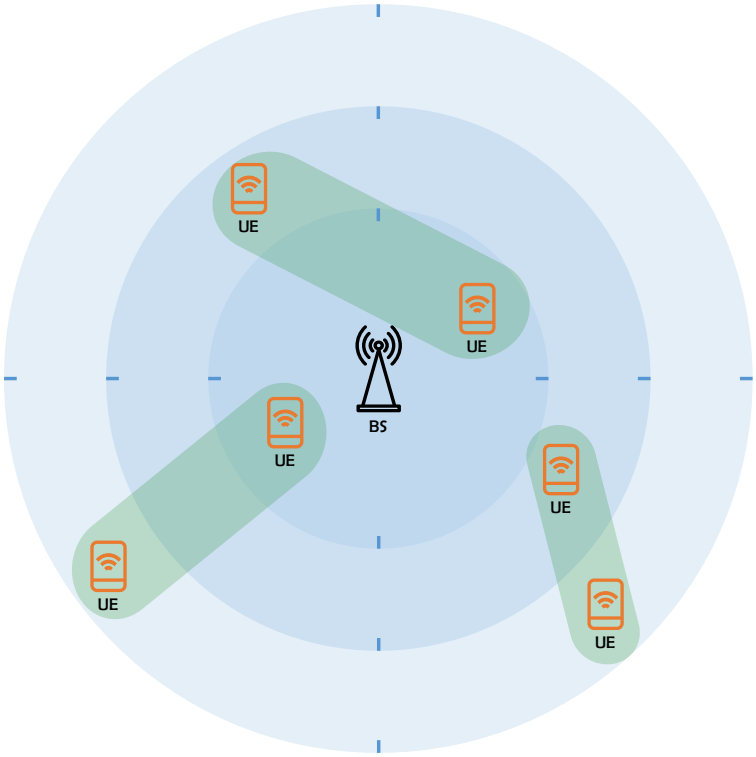

Figure 4.6: Clustering the UEs following their polar coordinates.

Formally, given the set of all UEs in the grid $\{\mathbf{UE}_1, \cdots, \mathbf{UE}_i, \cdots, \mathbf{UE}_{\mathcal{N}^{\mathbf{UE}}}\}$, we quantize the angles space $\Theta = [0, \cdots, 360]$ deg into discrete values $(\overline{\Theta})$ to allow UEs to be in a finite number of angles. Thus, before a re-clustering process at a new episode $e+1$, $\mathbf{UE}_i$ from the set of all UEs will be assigned a discrete channel sub-interval and a discrete angle $(\mathcal{X}_{i,e+1}, \overline{\theta}_{i,e+1})$. It transitions from the previous episodes' parameters $(\mathcal{X}_{i,e}, \overline{\theta}_{i,e})$ following equation (4.2) for the channel sub-interval and the equation below for the angle transition, which is similar to the channel one [4]:

$$p(\theta_{i,e+1} = \theta^{\ell} | \theta_{i,e} = \theta^{\ell'}) = \frac{\left(1-\rho^{\theta}\right)^{|\ell'-\ell|}}{\sum_{k=1}^{|\overline{\Theta}|}\left(1-\rho^{\theta}\right)^{|k-\ell|}}, \tag{4.36}$$

with ρ^{θ} being the angle range correlation factor, meaning that the angle in episode e affects the new angle in episode $e+1$, simulating more accurately the changing in UE location. Then, the clustering algorithm produces the set of clusters $\{\mathbf{C}_1, \cdots, \mathbf{C}_k, \cdots, \mathbf{C}_{\mathcal{N}^{\mathbf{C}}}\}$, with each $\mathbf{C}_k$ has $\mathcal{N}_k^{\mathbf{UE}}$ UEs ranging between $\mathcal{N}_{\min}^{\mathbf{UE}}$ and $\mathcal{N}_{\max}^{\mathbf{UE}}$. In what follows, we describe the clustering process in details, where the steps are organized according to their priority. A verification mechanism is implemented after each step to verify if the produced clusters are valid, in the sense that the number of UEs per cluster is bounded by the minimum and maximum values allowed, and the number of clusters is less than the allowed maximum number $\mathcal{N}^{\mathbf{C}}$. Once the verification is validated, the clustering process stops without continuing through the rest of the steps.

[4] Since the angles define a circle, a rollover functionality is implemented to account for adjacent angles that are in distant deg values

1. Create as many clusters as there are UEs to allow for the merging of clusters in the following steps, and to make the verification step easier :

$$\mathcal{N}^{\mathbf{C}} = \mathcal{N}^{\mathbf{UE}}.$$

2. Combine cluster $\mathbf{C}_k$ with $\mathbf{C}_{k'}$ together if the following conditions are met, with $\{k, k'\} \in \{1, \cdots, \mathcal{N}^{\mathbf{C}}\},\ \{i, i'\} \in \{1, \cdots, \mathcal{N}^{\mathbf{UE}}\}$:

   - The UEs in both clusters are assigned to different channel sub-intervals:

$$\mathcal{X}_i \neq \mathcal{X}_{i'} \text{ if } i \neq i'.$$

   - The UEs in both clusters have adjacent angles:

$$|\overline{\theta}_i - \overline{\theta}_{i'}| \leq \epsilon^{\theta},\ \forall\ \{i, i'\}.$$

     with ϵ^{θ} being the angle quantization step.

   In this step, the clustering is done by choosing one high SNR UE and testing it against all other UEs, starting from low SNR UEs and going up in the SNR scale, i.e., channel sub-interval, while ordering them according to their angle difference with the selected UE.

3. For every $\mathbf{UE}_i$ that was not clustered with other UEs in the previous two steps, assign them to a cluster $\mathbf{C}_k$ if

   - The number of UEs in $\mathbf{C}_k$ is less than the maximum

$$\mathcal{N}_k^{\mathbf{UE}} < \mathcal{N}_{\max}^{\mathbf{UE}}.$$

   - $\mathbf{UE}_i$'s angle is equal to or less than the angle quantization step compared to every $\mathbf{UE}_{i'}$ in $\mathbf{C}_k$:

$$|\overline{\theta}_i - \overline{\theta}_{i'}| \leq \epsilon^{\theta},\ \forall\ i' \in [1, \cdots, \mathcal{N}_k^{\mathbf{UE}}].$$

4. If $\mathbf{UE}_i$ is not assigned to another cluster in the previous step, cluster with it another $\mathbf{UE}_{i'}$ that is already clustered in $\mathbf{C}_k$, after satisfying the following conditions:

   - The number of clusters after this step is less than the maximum number of clusters allowed

$$\mathcal{N}^{\mathbf{C}} \leq \mathcal{N}_{\max}^{\mathbf{C}}.$$

   - The number of UEs in $\mathbf{C}_k$ is more than the minimum

$$\mathcal{N}_k^{\mathbf{UE}} > \mathcal{N}_{\min}^{\mathbf{UE}}.$$

   - the angle difference between the two UEs is less than the angle quantization step

$$|\overline{\theta}_i - \overline{\theta}_{i'}| \leq \epsilon^{\theta},\ \forall\ i' \in \{1, \cdots, \mathcal{N}_k^{\mathbf{UE}}\}.$$

If the resulting clusters are not valid after going through all the steps, the angle and channel sub-interval reassignment step is redone to ensure coherent and valid clusters.

In the example shown in Figure 4.6, the final clusters will be obtained after the second step, since all the UEs that are clustered together have different channel gains, and have equal or adjacent angles.

4.2.2.3 Proximal Policy Optimization

We choose the proximal policy optimization PPO as the DRL technique to train the multi-agent system on a cluster basis. We have previously explained this technique in section 2.2.4.4 and used it in section 3.4. However, the difference here is that each $\mathbf{CH}_k$ will house a different local PPO model with weights $\mathbf{w}_k$ and the federated learning FL is then used to learn a global model with weights $\mathbf{w_g}$.

With the number of UEs changing after each re-clustering operation, the PPO model needs to accommodate this dynamic aspect in the system. More specifically, the inputs and outputs of the PPO's Neural Networks (the actor and the critic networks share the first few layers, therefore their input is the same) have to consider the maximum size of the state representation and the maximum number of actions (in the actor's output). Thus, the PPO input state for $\mathbf{C}_k$, denoted as the vector $\mathbf{s}_k^{PPO}$, is defined as:

$$\begin{aligned}\mathbf{s}_k^{PPO} = \big\{&\mathbf{d}_{1k}, \cdots, \mathbf{d}_{\mathcal{N}_k^{\mathbf{UE}}k}, \cdots, \mathbf{d}_{\mathcal{N}_{\max}^{\mathbf{UE}}k},\\ &\underline{x}_{1k}, \cdots, \underline{x}_{\mathcal{N}_k^{\mathbf{UE}}k}, \cdots, \underline{x}_{\mathcal{N}_{\max}^{\mathbf{UE}}k},\\ &b^e_{1k}, \cdots, b^e_{\mathcal{N}_k^{\mathbf{UE}}k}, \cdots, b^e_{\mathcal{N}_{\max}^{\mathbf{UE}}k}\\ &z_1, \cdots, z_{\mathcal{N}^{\mathbf{C}}}, \cdots, z_{\mathcal{N}_{\max}^{\mathbf{C}}}\big\},\end{aligned} \tag{4.37}$$

where zero padding is performed on the elements from index $\mathcal{N}_k^{\mathbf{UE}}$ to index $\mathcal{N}_{\max}^{\mathbf{UE}}$ if $\mathcal{N}_k^{\mathbf{UE}} < \mathcal{N}_{\max}^{\mathbf{UE}}$, and from index $\mathcal{N}^{\mathbf{C}}$ to index $\mathcal{N}^{\mathbf{C}_{\max}}$ if $\mathcal{N}^{\mathbf{C}} < \mathcal{N}^{\mathbf{C}_{\max}}$. Zero padding is performed in the following way:

$$\mathbf{d}_{(\mathcal{N}_k^{\mathbf{UE}}+1)k} = \mathbf{d}_{(\mathcal{N}_k^{\mathbf{UE}}+2)k} = \cdots = \mathbf{d}_{\mathcal{N}_{\max}^{\mathbf{UE}}k} = \underbrace{[-1, \cdots, -1]}_{\mathcal{S}^d}, \tag{4.38}$$

$$\underline{x}_{(\mathcal{N}_k^{\mathbf{UE}}+1)k} = \underline{x}_{(\mathcal{N}_k^{\mathbf{UE}}+2)k} = \cdots = \underline{x}_{\mathcal{N}_{\max}^{\mathbf{UE}}k} = x_{\min}, \tag{4.39}$$

$$b^e_{(\mathcal{N}_k^{\mathbf{UE}}+1)k} = b^e_{(\mathcal{N}_k^{\mathbf{UE}}+2)k} = \cdots = b^e_{\mathcal{N}_{\max}^{\mathbf{UE}}k} = 0, \tag{4.40}$$

$$z_{\mathcal{N}^{\mathbf{C}}+1} = z_{\mathcal{N}^{\mathbf{C}}+2} = \cdots = z_{\mathcal{N}_{\max}^{\mathbf{C}}} = z_{\max}. \tag{4.41}$$

Basically, we zero pad the remaining spots in the state representation with UEs that have no data in the buffer, the lowest channel gain, and an empty battery. For $\mathbf{z}_k$ that represent information on other clusters, we fill up the spots with the maximum average channel gain (to avoid considering them in the interference estimation for Equation 4.5).

Moreover, we can set the actor's output $\pi_k^{\mathbf{w}}(.|\mathbf{s}_k)$ size as:

$$|\pi_k^{\mathbf{w}}(.|\mathbf{s}_k)| = (\mathcal{M}^{\mathbf{l}} + \mathcal{M}^{\mathbf{o}} + 1)^{\mathcal{N}_{\max}^{\mathbf{UE}}}, \tag{4.42}$$

which is the same as the maximum action space size $|\mathbf{A}_k|$.

4.3 Simulation Results

To assess the performance of the proposed model, we simulate a system with $\mathcal{N}^{\mathbf{UE}} = 6$ UEs. Each $\mathbf{UE}_i$ has a buffer of size $\mathcal{S}^d = 4$ with a maximum delay $\nabla = 2$ on the packets that arrive, as well as EH capabilities with a battery of size $\mathcal{N}^b = 3$ energy units of $\mathcal{S}^e = 1$ μJoules. In addition, we choose a random data model among the models described in section 4.1.3, which are Poisson distribution with mean $\mu^d_{\mathbf{po}} = 0.5$, uniform distribution with $\mathcal{D}_{\mathbf{u},\min} = 0$ and $\mathcal{D}_{\mathbf{u},\max} = \mathcal{S}^d/2$, Lognormal and Pareto distributions with $\mu^d_{\mathbf{l}} = \mu^d_{\mathbf{pa}} = 0$. We assume that the channel gain range is divided into $|\mathcal{J}| = 3$ sub-intervals with $|\mathcal{X}^j| = 3$ quantization steps for each sub-interval $\mathcal{X}^j$, following bounds $\{(-5.23, -0.46); (0, 4.78); (4.91, 6.53)\}$dB. Each sub-interval is further quantized into $|\mathcal{X}^j| = 3$ levels of channel gains, that are used to compute the channel gain of each UE at a given timestep. The angles range is similarly quantized into $|\bar{\mathbf{\Theta}}| = 4$ discrete angles.

We cluster the UEs in at most $\mathcal{N}^{\mathbf{C}} \leq \mathcal{N}^{\mathbf{C}}_{\max} = 3$ clusters, with each cluster $\mathbf{C}_k$ having $2 = \mathcal{N}^{\mathbf{UE}}_{\min} \leq \mathcal{N}^{\mathbf{UE}} \leq \mathcal{N}^{\mathbf{UE}}_{\max} = 4$ number of UEs. The re-clustering is done before the start of each episode.

The cluster $\mathbf{C}_k$ scheduling decisions are taken by the $\mathbf{CH}_k$ at the start of a timestep. It performs either local operations for $\mathbf{UE}_{ik}$ with a maximum of $\mathcal{M}^{\mathbf{l}} = 1$ packets, or offloading packets to the MEC server with a maximum of $\mathcal{M}^{\mathbf{o}} = 3$ packets that can be offloaded at once.

When processing locally, the power associated with the decision is $\mathcal{P}^{\mathbf{l}} = 180$ μW, whereas the maximum offloading power permitted is $\mathcal{P}^{\mathbf{o}} = 2$ mW. In the offloading operation, $\mathbf{UE}_{ik}$ sends to the MEC server m_{ik} packets of size $\mathcal{L}^{ul} = 500$ bits, over a bandwidth of $\mathcal{W}^{ul} = 4$ MHz, and a channel noise spectral density of $\mathcal{N}_0 = -87$ dBm Hz. It awaits the execution for $\mathcal{T}^{\mathbf{w}} = 0.1$ ms while consuming the power $p^{\mathbf{w}} = 0.1$ mW. For DL transmission, the resulted packets have size $\mathcal{L}^{dl} = 50$ bits and the BS uses a transmission power of $p^{\mathbf{b}}_{bs} = 1$ kW. The total DL bandwidth $\mathcal{W}^{dl} = 10$ MHz is allocated partially to each cluster depending on the number of UEs in it. At the UEs, the reception power is set to $p^{\mathbf{r}} = 3$ mW and the decoding power to $p^{\mathbf{d}} = 5$ mWs. In fact, each $\mathbf{UE}_{ik}$ receives the signals from all the other UEs in the same cluster and performs SIC decoding to decode its signal after decoding the signal of other UEs with weaker channel (higher allocated power). The whole offloading operation has to be done within the timestep size, which is $\mathcal{T} = 1$ ms.

To train the Federated PPO multi-agent setup, each agent will have 2 NNs (actor and critic networks) of 4 layers, the first 2 are shared between them. The number of neurons is 128 per layer with ReLU activation functions. The number of training epochs is the same as the number of episodes $\mathbf{E} = 40 \cdot 10^3$ with each episode having $\mathbb{T} = 1024$ timesteps, and the memory buffer size is 1024 tuples. The batch size when sampling experiences from the memory buffer is 64, the policy clip $\epsilon = 0.2$, the generalized advantage estimation factor $\lambda = 0.97$, the discount factor $\gamma = 0.99$, the learning rate $\alpha = 5 \cdot 10^{-4}$, and the number of iterations of batch sampling is $\mathbf{I} = 10$. Once an episode terminates, the weights are shared with the MEC server, and a re-clustering process is initiated for the next episode.

We compare the Federated PPO method with the same naive methods used previously and extended to a multi-agent setup, namely: the naive offload NO, naive local (NL), naive random NR and the immediate scheduler IMM. We use the percentage of dropped packets and the energy consumption as the performance metrics.

In Figure 4.7, we show the performance of the used methods in terms of the percentage % of dropped packets per cluster, averaged throughout $\mathbf{E} = 1000$ episodes of $\mathbb{T} = 1000$ timesteps. We can see that the Federated PPO method outperforms the other naive methods by achieving less values of all the types of errors that induce packets losses, i.e., delay violation and buffer overflow (transmission error does not cause a loss in packets, only a failure to execute the packets, therefore it is not included). The results illustrate the effectiveness of the adopted learning scheme that renders the federated PPO aware of the inter-cluster interference, as well as the delay, battery and channel information of the UEs in each cluster. Notably, the NO method performs poorly even compared to the NR method. This is due to the high delay violation errors that occur from the lack of cluster awareness, resulting thus in choosing actions that are not available. Meanwhile, the NL method performs better than the NO method, which subsequently indicates that IMM method selects local processing more often.

The energy consumption is evaluated in Figure 4.8, where the number of averaged energy units consumed during an episode per cluster is shown. We can observe that Federated PPO achieves lower energy consumption compared to the naive methods and the immediate scheduler. In addition, the NO achieves less than the other naive methods simply because it stays idle too often. Therefore, the Federated PPO method offers good performance while consuming less power than the other methods.

Investigating the clustering outcomes during the testing episodes reveals that a $\mathcal{N}^{\mathbf{C}} = 3$ clusters configuration with $\mathcal{N}^{\mathbf{UE}} = 2$ UEs each occurs for 65.5% of the episodes. The second highest occurring configuration is the one with $\mathcal{N}^{\mathbf{C}} = 2$ clusters of $\mathcal{N}^{\mathbf{UE}} = 3$ UEs with 28.9%. The two other possible configurations of $\mathcal{N}^{\mathbf{C}} = 2$ clusters of $\mathcal{N}^{\mathbf{UE}} = \{2, 4\}$ and $\mathcal{N}^{\mathbf{UE}} = \{4, 2\}$ UEs happen 3% and 2.6% of the testing episodes, respectively.

In addition, an action analysis is carried out it Figure 4.9, Figure 4.10 and Figure 4.11, where each pie chart represents the percentage of actions taken during an episode (idle actions are omitted) for a UE in a cluster with $\mathcal{N}^{\mathbf{UE}} = \{2, 3, 4\}$ number of UEs, respectively. The figures show that the instances where intra-cluster NOMA is used increases with the number of UEs in the cluster.

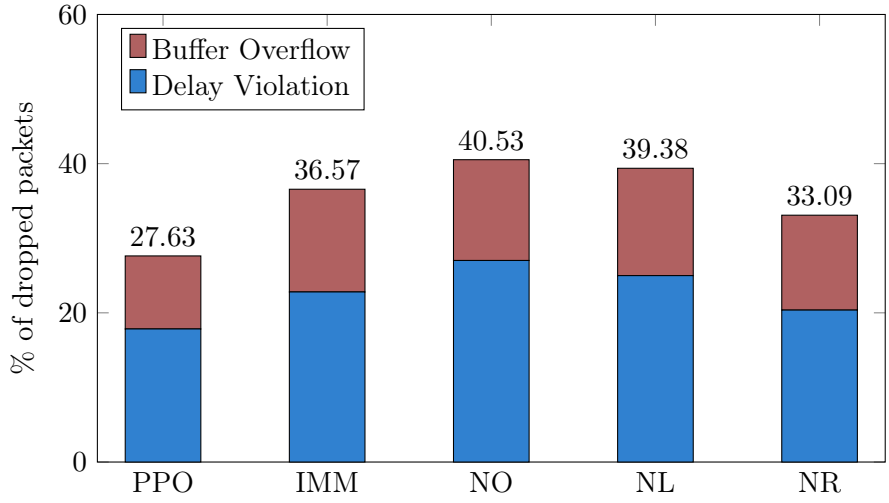

Figure 4.7: Average percentage of dropped packets for each approach.

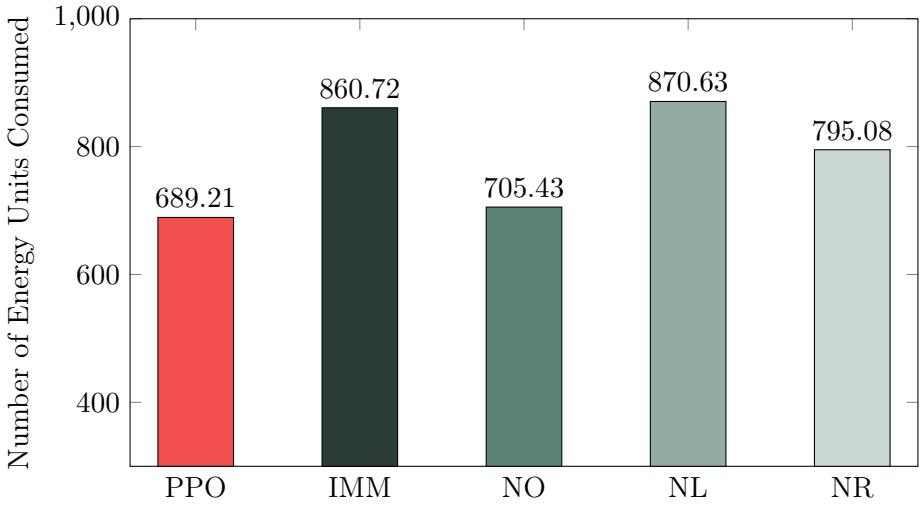

Figure 4.8: Average energy units consumed for each approach.

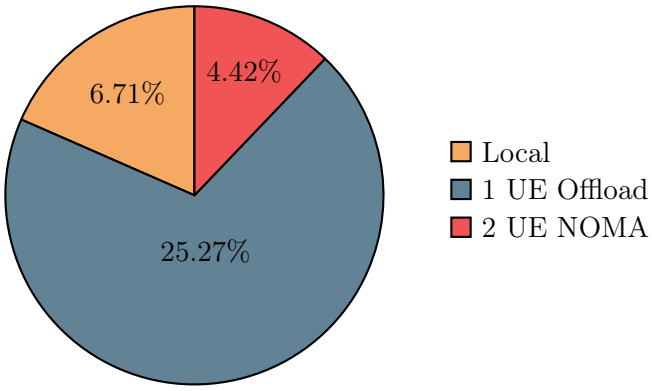

Figure 4.9: Average percentage of taken actions by each UE over all the clusters with $\mathcal{N}^{\mathbf{UE}} = 2$ UEs (idle actions represent 63.1%).

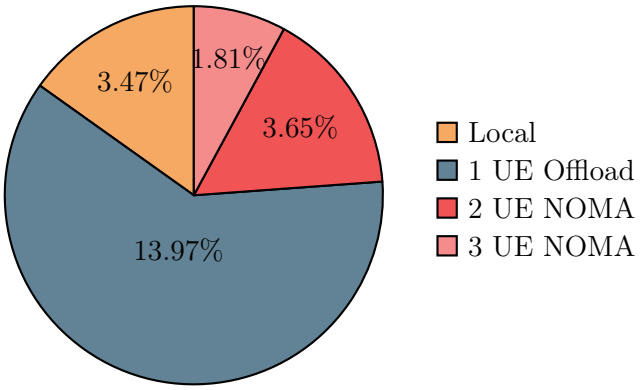

Figure 4.10: Average percentage of taken actions by each UE over all the clusters with $\mathcal{N}^{\mathbf{UE}} = 3$ UEs (idle actions represent 77.1%).

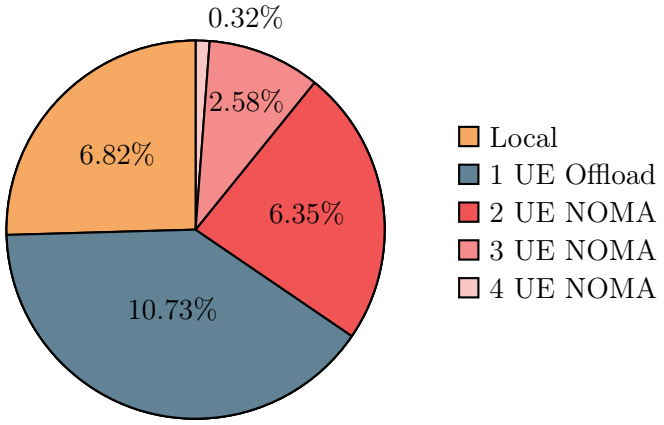

Figure 4.11: Average percentage of taken actions by each UE over all the clusters with $\mathcal{N}^{\mathbf{UE}} = 4$ UEs (idle actions represent 73.2%).

4.4 Conclusion

The work in this chapter has tackled a multi-user problem extending the already studied environment with multiple dynamic UEs due to their mobility. An adaptive NOMA clustering algorithm is proposed that takes into account the channel conditions and the position of the UEs on the grid. The resulted multi-cluster scheduling and computation offloading problem is solved using Federated Learning learning, specifically a multi-agent Proximal Policy Optimization method. Each cluster is trained locally using the heterogeneous UEs' information, and a global model is averaged at the MEC server, and broadcasted to the clusters. The simulation results validated the efficiency of the proposed PPO method when compared to other methods, achieving thus lower packet error rates while consuming less energy units. The presented work will be the subject of a journal submission **J1** by the defense time.

Conclusions and perspectives

THIS thesis addressed the problem of resource scheduling and computational offloading, in a system with multiple user equipments UEs connected to a cellular base station BS. The system had the characteristics listed below.

- Strict delay constraints on the data stored in the UEs.
- Offloading functionalities using mobile edge computing MEC technology.
- Battery recharging from ambient sources using energy harvesting EH capabilities at UEs.
- Multiple access with NOMA-enabled communications.

Reinforcement Learning RL was used to devise optimal scheduling-offloading policies at the UEs that are able to adapt to various situations encountered in the environment. The developed methods were shown to perform better than the standard ones in terms of packet error rate as well as energy consumption across multiple scenarios.

Chapter 1 laid the basics from communication perspective that were later adopted in our works. Specifically, the chapter provided details for:

- How MEC technologies were presented as a solution to the computational capacities of end devices while sacrificing minimal delay.
- EH capabilities that help prolonging the usability of battery-powered devices as well as exploiting energy from sustainable sources.
- NOMA technique that enables simultaneous access of multiple users in time and frequency, in addition to the scaling issues and the clustering that helps overcoming these issues.

Chapter 2 focused on artificial intelligence AI technologies in general, and reinforcement learning RL in particular. The chapter presented Markov modeling of environments and its Markov decision processes MDPs models. Following that, suitable functions (e.g., Value function and Q-function) were described, which can be used to derive expressions for optimal policies. These policies dictate the best actions to take for each state in the environment. Finally, the chapter introduced various algorithms that numerically attempt to find the desired policies. In particular:

- Model-based methods with the guarantee of convergence, e.g., value iteration VI and policy iteration PI.
- Model-free methods, with a trial-and-error based approach, e.g., Q-Learning QL.
- Model-free neural network-based methods, i.e., deep reinforcement Learning DRL methods that approximate the environment using NNs, e.g., deep Q-networks DQN and proximal policy optimization PPO.

Chapter 3 presented the first two contributions of this thesis. The first contribution considered a system setup comprised of two UEs with data buffers storing delay-sensitive tasks and connected to a BS with MEC capabilities. UEs could either execute their packets locally or remotely by offloading them to the MEC server. NOMA communications were enabled when both UEs offload at the same time. Different RL methods were developed to solve the optimization problem and produce policies that output the type of execution as well as the number of packets to execute, given channel and data buffer information of the UEs. As a result, the investigated RL methods showed lower packet loss thus better performance than other methods. When scaling the system to larger state space, it was concluded that DRL techniques were more suited to produce good policies thanks to their generalization capabilities to unseen states.

The second contribution dealt with the extension to an EH-based system, where the UEs have batteries that can be recharged via external ambient and sustainable energy sources. The resulted optimization problem is subsequently more complex and the state space grows larger. Another DRL method, namely the proximal policy optimization PPO was investigated to devise optimal policies. It proved to achieve lower packet error rates compared to the standard methods while consuming less energy.

Chapter 4 extended the previous contributions of chapter 3 to a multi-user system by adding more than 2 UEs in the system. The subsequent challenges arise from this change:

1. The previous DRL methods were no longer viable due to the exponentially large state space, necessitating thus longer exploration times to learn the right policies for all the UEs jointly.

2. NOMA performance especially in the downlink DL phase degraded due to the increasing inter-user interference and loss in SIC decoding performance. The computational capacities of the UEs do not allow for a decoding within the time constraints of the system.

To overcome these challenges, UEs clustering method was proposed in this chapter. The clustering process allows for the creation of smaller groups of UEs that can be handled by one DRL agent. In addition to improving NOMA performance in DL, the clusters were managed separately on an FDMA basis, eliminating thus the inter-cluster interference. Further additions were also considered at the UEs. Namely, various data arrival models were adopted by each UE to induce more heterogeneity in the system. Moreover, mobility of the UEs was introduced to allow for location-based clustering constraints, on top of the channel-based constraints.

Federated learning FL was utilized to aid the learning of a multi-agent setup where each cluster had its own DRL agent at the cluster head. FL helped producing a global model that accounts for all the different cluster configurations in this multi-cluster system. The simulation results showed the advantage of Federated PPO method in finding optimal scheduling-offloading policies compared to other standard methods, with regards to packet loss and energy consumption performance metrics.

Perspectives

We present in the sequel some future perspectives to extend the work of this thesis.

Investigative Study of Data Arrival Models Data arrival models discussing how data arrives at the buffer of UEs were described in section 1.3, and several ones were already used in the work of chapter 4. However, they were not the objective of the outlined work. In this study, we aim at experimenting with the different data arrival models in a similar system as the one used in chapter 3. The goal is to analyze the effect of each data arrival model on the produced RL policy, in terms of actions taken, and resulting packet error rate and energy consumption. Machine Learning (ML) models can be adopted as well to provide a more practical setup and learn the data distributions.

Comparative Study of NOMA clustering methods The NOMA clustering that we developed in section 4.2.2.2 considered the channel conditions of the UEs as well as their positions (in terms of angle) w.r.t. the BS. The purpose of this study is to compare the proposed clustering method to other existing methods as cited in Subsection 1.4.3, by analyzing the NOMA performance, the resulted sum-rates, and the interference levels.

Adaptive Clustering and Learning The proposed NOMA clustering method was performed after each episode of transmission. We considered that the quantized channel sub-interval $\mathcal{X}_i$ and the angle $\bar{\theta}_i$ at every $\mathbf{UE}_i,\ i \in \{1, \cdots, \mathcal{N}^{\mathbf{UE}}\}$ change following a correlated process. Accounting for the mobility of UEs can be further enhanced by designing a more adaptive clustering. The clustering method will have to be able to handle the changing of the clusters at any moment for any UE. The channel and angle ranges will be subsequently used in their entirety with sub-intervals, and different transition models have to be developed to allow the UE to move more freely.

On the other hand, the Federated RL FedRL procedure will become more frequent with the changes in clusters. Therefore, it needs to perform the weights sharing operation whenever possible, e.g., when the changes in the local models are big enough, rather than every episode.

Offloading in a Cooperative Multi-Cell System The objective here is to generalize the system by scaling the setup to multiple cells, with each cell containing UEs connected to the BS with a dedicated MEC server for the cell. Each MEC server may have limited computation resources, for instance, in terms of the number of tasks that can be handled. From that, offloading to MEC servers from other cells becomes necessary under some computation and delay constraints.

Therefore, cooperation between the MEC servers, thus muti-cells, is required to execute the tasks offloaded by the UEs. Moreover, the load at the MEC servers could be used as state information, thus considered in the optimization problem to decide to which server to offload. Another option would be that the MEC servers could offload their tasks when they are fully loaded to the core network, using a Fog Computing server.

Bibliography

[1] M. Report. "5g mobile subscriptions to exceed 5.3 billion in 2029". Ericsson, Ed. (2023), [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-subscriptions-outlook.

[2] F. Duarte. "Number of iot devices (2023)". E. Topics, Ed. (2023), [Online]. Available: https://explodingtopics.com/blog/number-of-iot-devices.

[3] S. J. Bigelow. "What is edge computing? everything you need to know". (2021), [Online]. Available: https://www.techtarget.com/searchdatacenter/definition/edge-computing.

[4] Qualcomm, Ed. "Everything you need to know about 5g." (2023), [Online]. Available: https://www.qualcomm.com/5g/what-is-5g.

[5] T. S. Project, Ed. "Environmental impacts of digital technology : 5-year trends and 5g governance". (2023), [Online]. Available: https://theshiftproject.org/en/article/environmental-impacts-of-digital-technology-5-year-trends-and-5g-governance/.

[6] Y. Saito, Y. Kishiyama, *et al.*, "Non-orthogonal multiple access (noma) for cellular future radio access", in *2013 IEEE 77th vehicular technology conference (VTC Spring)*, IEEE, 2013, pp. 1–5.

[7] D. Silver, A. Huang, *et al.*, "Mastering the game of go with deep neural networks and tree search", *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[8] D. Silver, J. Schrittwieser, *et al.*, "Mastering the game of go without human knowledge", *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[9] C. Zhang, P. Patras, *et al.*, "Deep learning in mobile and wireless networking: a survey", *IEEE Communications surveys & tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[10] H. Fan, L. Zhu, *et al.*, "Deep reinforcement learning for energy efficiency optimization in wireless networks", in *2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, IEEE, 2019, pp. 465–471.

[11] J. Ye and H. Gharavi, "Deep reinforcement learning-assisted energy harvesting wireless networks", *IEEE transactions on green communications and networking*, vol. 5, no. 2, pp. 990–1002, 2020.

[12] D. Shi, F. Tian, *et al.*, "Energy efficiency optimization in heterogeneous networks based on deep reinforcement learning", in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, 2020, pp. 1–6.

[13] N. Zhao, Y.-C. Liang, *et al.*, "Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks", *IEEE Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5141–5152, 2019.

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

[15] X. Chen, Y. Liu, *et al.*, "Resource allocation for wireless cooperative iot network with energy harvesting", *IEEE Transactions on Wireless Communications*, vol. 19, no. 7, pp. 4879–4893, 2020.

[16] T. Yang, Y. Hu, *et al.*, "Deep reinforcement learning based resource allocation in low latency edge computing networks", in *2018 15th international symposium on wireless communication systems (ISWCS)*, IEEE, 2018, pp. 1–5.

[17] J. Li, H. Gao, *et al.*, "Deep reinforcement learning based computation offloading and resource allocation for mec", in *2018 IEEE wireless communications and networking conference (WCNC)*, IEEE, 2018, pp. 1–6.

[18] F. Fang, Y. Xu, *et al.*, "Optimal task assignment and power allocation for noma mobile-edge computing networks", *arXiv preprint arXiv:1904.12389*, 2019.

[19] M. K. Sharma, A. Zappone, *et al.*, "Deep learning based online power control for large energy harvesting networks", in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 8429–8433.

[20] H. Zhou, K. Jiang, *et al.*, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing", *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1517–1530, 2021.

[21] L. Ale, N. Zhang, *et al.*, "Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning", *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 3, pp. 881–892, 2021.

[22] H. Li, F. Fang, *et al.*, "Drl-assisted resource allocation for noma-mec offloading with hybrid sic", *Entropy*, vol. 23, no. 5, p. 613, 2021.

[23] A. Alwarafy, M. Abdallah, *et al.*, "Deep reinforcement learning for radio resource allocation and management in next generation heterogeneous wireless networks: a survey", *arXiv preprint arXiv:2106.00574*, 2021.

[24] J. Konečný, H. B. McMahan, *et al.*, "Federated learning: strategies for improving communication efficiency", in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. [Online]. Available: https://arxiv.org/abs/1610.05492.

[25] L. S. Vailshery. "Iot connected devices worldwide 2019-2030". T. I. E. Topics, Ed. (2023), [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[26] N. Abbas, Y. Zhang, *et al.*, "Mobile edge computing: a survey", *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[27] M. Beck, M. Werner, *et al.*, "Mobile edge computing: a taxonomy", Jan. 2014.

[28] M. Satyanarayanan, P. Bahl, *et al.*, "The case for vm-based cloudlets in mobile computing", *IEEE pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[29] S. Yi, C. Li, *et al.*, "A survey of fog computing: concepts, applications and issues", in *Proceedings of the 2015 workshop on mobile big data*, 2015, pp. 37–42.

[30] "Data centres and data transmission networks". (2023), [Online]. Available: https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks.

[31] F. K. Shaikh and S. Zeadally, "Energy harvesting in wireless sensor networks: a comprehensive review", *Renewable and Sustainable Energy Reviews*, vol. 55, pp. 1041–1054, 2016.

[32] M.-L. Ku, W. Li, *et al.*, "Advances in energy harvesting communications: past, present, and future challenges", *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1384–1412, 2015.

[33] M.-L. Ku, Y. Chen, *et al.*, "Data-driven stochastic models and policies for energy harvesting sensor communications", *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 8, pp. 1505–1520, 2015.

[34] F. A. Kraemer, D. Ammar, *et al.*, "Solar energy prediction for constrained iot nodes based on public weather forecasts", in *Proceedings of the Seventh International Conference on the Internet of Things*, 2017, pp. 1–8.

[35] M. Basim, Q. ul Ain, *et al.*, "A comprehensive review on high-efficiency rf-dc converter for energy harvesting applications", *Journal of Semiconductor Technology and Science*, vol. 22, no. 5, pp. 304–325, 2022.

[36] I. Flint, X. Lu, *et al.*, "Performance analysis of ambient rf energy harvesting: a stochastic geometry approach", in *2014 IEEE Global Communications Conference*, IEEE, 2014, pp. 1448–1453.

[37] L. Decreusefond, I. Flint, *et al.*, "A note on the simulation of the ginibre point process", *Journal of Applied Probability*, vol. 52, no. 4, pp. 1003–1012, 2015.

[38] H. J. Visser and R. J. Vullers, "Rf energy harvesting and transport for wireless sensor network applications: principles and requirements", *Proceedings of the IEEE*, vol. 101, no. 6, pp. 1410–1423, 2013.

[39] F. Azmat, Y. Chen, *et al.*, "Predictive modelling of rf energy for wireless powered communications", *IEEE Communications Letters*, vol. 20, no. 1, pp. 173–176, 2015.

[40] Z. Liang and J. Yuan, "Modelling and prediction of mobile service channel power density for rf energy harvesting", *IEEE Wireless Communications Letters*, vol. 9, no. 5, pp. 741–744, 2020.

[41] Y. Mao, J. Zhang, *et al.*, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices", *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[42] I. Fawaz, M. Sarkiss, *et al.*, "Delay-optimal resource scheduling of energy harvesting-based devices", *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 4, pp. 1023–1034, 2019.

[43] W. Abbessi and H. Nabli, "General approach for video traffic: from modeling to optimization", *Multimedia Systems*, vol. 25, no. 3, pp. 177–193, 2019.

[44] G. Aceto, G. Bovenzi, *et al.*, "Characterization and prediction of mobile-app traffic using markov modeling", *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 907–925, 2021.

[45] G. Aceto, D. Ciuonzo, *et al.*, "Mirage: mobile-app traffic capture and ground-truth creation", in *IEEE 4th International Conference on Computing, Communication and Security (ICCCS 2019)*, 2019.

[46] J. Navarro-Ortiz, P. Romero-Diaz, *et al.*, "A survey on 5g usage scenarios and traffic models", *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 905–929, 2020.

[47] A. Benjebbovu, A. Li, *et al.*, "System-level performance of downlink noma for future lte enhancements", in *2013 IEEE globecom workshops (GC Wkshps)*, IEEE, 2013, pp. 66–70.

[48] A. Benjebbour, K. Saito, *et al.*, "Non-orthogonal multiple access (noma): concept, performance evaluation and experimental trials", in *2015 International conference on wireless networks and mobile communications (WINCOM)*, IEEE, 2015, pp. 1–6.

[49] S. Islam, M. Zeng, *et al.*, "Non-orthogonal multiple access (noma): how it meets 5g and beyond", *arXiv preprint arXiv:1907.10001*, 2019.

[50] H. Tabassum, M. S. Ali, *et al.*, "Non-orthogonal multiple access (noma) in cellular uplink and downlink: challenges and enabling techniques", *arXiv preprint arXiv:1608.05783*, 2016.

[51] L. Salaün, "Resource allocation and optimization for the non-orthogonal multiple access", Theses, Institut Polytechnique de Paris, Mar. 2020. [Online]. Available: https://theses.hal.science/tel-02733385.

[52] "Single carrier noma (sc noma) - how many users can it support?" (2020), [Online]. Available: https://ecewireless.blogspot.com/2020/06/single-carrier-noma-sc-noma-how-good-is_15.html.

[53] M. S. Ali, H. Tabassum, *et al.*, "Dynamic user clustering and power allocation for uplink and downlink non-orthogonal multiple access (noma) systems", *IEEE access*, vol. 4, pp. 6325–6343, 2016.

[54] J. Cui, Z. Ding, *et al.*, "Unsupervised machine learning-based user clustering in millimeter-wave-noma systems", *IEEE Transactions on Wireless Communications*, vol. 17, no. 11, pp. 7425–7440, 2018.

[55] A. Shahini and N. Ansari, "Noma aided narrowband iot for machine type communications with user clustering", *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 7183–7191, 2019.

[56] Z. Wang, M. Pischella, *et al.*, "Clustering and power optimization for noma multi-objective problems", in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, IEEE, 2020, pp. 1–6.

[57] M. Z. Alom, T. M. Taha, *et al.*, "The history began from alexnet: a comprehensive survey on deep learning approaches", *arXiv preprint arXiv:1803.01164*, 2018.

[58] K. O'Shea and R. Nash, "An introduction to convolutional neural networks", *arXiv preprint arXiv:1511.08458*, 2015.

[59] R. M. Schmidt, "Recurrent neural networks (rnns): a gentle introduction and overview", *arXiv preprint arXiv:1912.05911*, 2019.

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

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

[62] H. Hasselt, "Double Q-learning", *Advances in Neural Information Processing Systems*, vol. 23, 2010.

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

[64] T. Schaul, J. Quan, *et al.*, "Prioritized experience replay", *arXiv preprint arXiv:1511.05952*, 2015.

[65] Z. Wang, T. Schaul, *et al.*, "Dueling network architectures for deep reinforcement learning", in *International conference on machine learning*, PMLR, 2016, pp. 1995–2003.

[66] J. Schulman, F. Wolski, *et al.*, "Proximal policy optimization algorithms", *arXiv preprint arXiv:1707.06347*, 2017.

[67] L. Weng, "Policy gradient algorithms", *lilianweng.github.io*, 2018. [Online]. Available: https://lilianweng.github.io/posts/2018-04-08-policy-gradient/.

[68] J. Schulman, S. Levine, *et al.*, "Trust region policy optimization", in *International conference on machine learning*, PMLR, 2015, pp. 1889–1897.

[69] Wikipedia contributors, *Federated learning — Wikipedia, the free encyclopedia*, [Online; accessed 24-November-2023], 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Federated_learning&oldid=1184187973.

[70] C. Zhang, Y. Xie, *et al.*, "A survey on federated learning", *Knowledge-Based Systems*, vol. 216, p. 106 775, 2021.

[71] C. Nadiger, A. Kumar, *et al.*, "Federated reinforcement learning for fast personalization", in *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, IEEE, 2019, pp. 123–127.

[72] H. H. Zhuo, W. Feng, *et al.*, "Federated deep reinforcement learning", *arXiv preprint arXiv:1901.08277*, 2019.

[73] J. Qi, Q. Zhou, *et al.*, "Federated reinforcement learning: techniques, applications, and open challenges", *arXiv preprint arXiv:2108.11887*, 2021.

[74] L. P. Qian, A. Feng, *et al.*, "Optimal SIC Ordering and Computation Resource Allocation in MEC-aware NOMA NB-IoT Networks", *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2806–2816, 2018.

[75] M. Hua, H. Tian, *et al.*, "Online Offloading Scheduling for NOMA-aided MEC under Partial Device Knowledge", *IEEE Internet of Things Journal*, 2021.

[76] W. Liu, Y. He, *et al.*, "Deep Reinforcement Learning-Based MEC Offloading and Resource Allocation in Uplink NOMA Heterogeneous Network", in *2021 Computing, Communications and IoT Applications (ComComAp)*, IEEE, 2021, pp. 144–149.

[77] T. Truong, T. Nguyen, *et al.*, "Partial Computation Offloading in NOMA-assisted Mobile-Edge Computing Systems Using Deep Reinforcement Learning", *IEEE Internet of Things Journal*, vol. 8, no. 17, pp. 13 196–13 208, 2021.

[78] M. Nduwayezu, Q.-V. Pham, *et al.*, "Online Computation Offloading in NOMA-based Multi-access Edge Computing: A Deep Reinforcement Learning Approach", *IEEE Access*, vol. 8, pp. 99 098–99 109, 2020.

[79] L. Li, Q. Cheng, *et al.*, "Resource allocation for NOMA-MEC Systems in Ultra-Dense Networks: A Learning Aided Mean-Field Game Approach", *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1487–1500, 2020.

[80] Z. Hu, J. Niu, *et al.*, "An efficient online computation offloading approach for large-scale mobile edge computing via deep reinforcement learning", *IEEE Transactions on Services Computing*, vol. 15, no. 2, pp. 669–683, 2021.

[81] A. Sadiki, J. Bentahar, *et al.*, "Deep reinforcement learning for the computation offloading in mimo-based edge computing", *Ad Hoc Networks*, p. 103 080, 2023.

[82] L. An, Z. Wang, *et al.*, "Joint task offloading and resource allocation via proximal policy optimization for mobile edge computing network", in *2021 International Conference on Networking and Network Applications (NaNA)*, 2021, pp. 466–471.

[83] A. Alajmi and W. Ahsan, "An efficient actor critic drl framework for resource allocation in multi-cell downlink noma", in *2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, IEEE, 2022, pp. 77–82.

[84] Z. Tianqing, W. Zhou, *et al.*, "Resource allocation in iot edge computing via concurrent federated reinforcement learning", *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1414–1426, 2021.

[85] S. Yu, X. Chen, *et al.*, "When deep reinforcement learning meets federated learning: intelligent multitimescale resource management for multiaccess edge computing in 5g ultradense network", *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2238–2251, 2020.

[86] X. Li, L. Lu, *et al.*, "Federated multi-agent deep reinforcement learning for resource allocation of vehicle-to-vehicle communications", *IEEE Transactions on Vehicular Technology*, vol. 71, no. 8, pp. 8810–8824, 2022.

[87] P. Consul, I. Budhiraja, *et al.*, "Federated reinforcement learning based task offloading approach for mec-assisted wban-enabled iomt", *Alexandria Engineering Journal*, vol. 86, pp. 56–66, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S111001682301027X.

[88] X. Liu, J. Yu, *et al.*, "Multi-agent reinforcement learning for resource allocation in iot networks with edge computing", *China Communications*, vol. 17, no. 9, pp. 220–236, 2020.

[89] J. Cui, Y. Liu, *et al.*, "Multi-agent reinforcement learning-based resource allocation for uav networks", *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 729–743, 2019.

[90] Y. M. Park, Y. K. Tun, *et al.*, "Joint user pairing and beamforming design of multi-star-riss-aided noma in the indoor environment via multi-agent reinforcement learning", *arXiv preprint arXiv:2311.08708*, 2023.

DOCTORAL SCHOOL

Title: Joint Offloading-Scheduling Policies for Future Generation Wireless Networks.

Keywords: Resource Allocation, 5G, Reinforcement Learning

Abstract: The challenges posed by the increasing number of connected devices, high energy consumption, and environmental impact in today's and future wireless networks are gaining more attention. New technologies like Mobile Edge Computing (MEC) have emerged to bring cloud services closer to the devices and address their computation limitations. Enabling these devices and the network nodes with Energy Harvesting (EH) capabilities is also promising to allow for consuming energy from sustainable and environmentally friendly sources. In addition, Non-Orthogonal Multiple Access (NOMA) is a pivotal technique to achieve enhanced mobile broadband. Aided by the advancement of Artificial Intelligence, especially Reinforcement Learning (RL) models, the thesis work revolves around devising policies that jointly optimize scheduling and computational offloading for devices with EH capabilities, NOMA-enabled communications, and MEC access. Moreover, when the number of devices increases and so does the system complexity, NOMA clustering is performed and Federated Learning is used to produce RL policies in a distributed way. The thesis results validate the performance of the proposed RL-based policies, as well as the interest of using NOMA technique.

Institut Polytechnique de Paris
91120 Palaiseau, France