Deep Learning Applications in Beyond 5G Mobile Networks

Ignas Laurinavicius

Supervisors:

Dr. Huiling Zhu,

Prof. Jiangzhou Wang

University of Kent

Thesis submitted for the degree of

Doctor of Philosophy

in Electronic Engineering

June 5, 2023

Canterbury, United Kingdom

Acknowledgements

This thesis would be incomplete without support from many people that I have met since starting out. I would like to show my major appreciation to Dr. Huiling Zhu and Prof. Jiagnzhou Wang. Their immeasurable support and patience has been invaluable in completion of the thesis.

I am grateful for working with Dr. Philippos Assimakopoulos, Dr. Shabnam Noor, Mr. Mohsan Niaz Chughtai. Their amazing collaboration and discussions helped me deepen my knowledge in centralised mobile networks.

I would like to recognize my family, friends and colleagues for letting me talk about my research without stopping me. This helped me stay motivated and excited about my thesis. A special thank you to my brother and sister for being there through all the highs and lows.

Abstract

The thesis is motivated by the evolution of mobile networks needing to match novel applications. Cloud radio access network (C-RAN) is a network architecture that solves power consumption issues, improves scaling, and reduces the operational expenditures for operators. Due to capacity limitations of fronthaul links, C-RAN fails to meet the expectations placed on it in solving the network objectives. Fog radio access network (F-RAN) has been proposed to solve the fronthaul loading. By placing radio processing and caching functionality at the edge nodes latency can be reduced and fronthaul experiences less load, increasing performance gains of the cloud infrastructure.

F-RAN faces multiple issues since the fog access points (FAPs) are a novel addition to the network architecture. FAPs have to communicate with each other to solve optimisation problems and provide a satisfactory quality of service (QoS). Further, compared to the baseband unit (BBU), FAPs are equipped with much smaller caches, so algorithms have to proactively cache content that users would like to access. Due to the stringent low latency requirement set for the future generation of mobile networks, some information cannot be exchanged, thus FAPs have to make assumptions about the global state of the environment. For this reason, the thesis looks at machine learning options, which are known to be able to predict and generalise outcomes based on historical data.

Reinforcement learning (RL) is chosen as it learns from an environment that follows a Markov decision process (MDP). Double deep Q-network (DDQN) is known to be one of the best performers, as it provides convergence guarantees

for MDP environments and improved learning stability through the use of experience replay. The thesis uses C-RAN architecture as an environment for a DDQN deployment to control flexible ethernet (FlexE) calendar slot allocation as well as remote unit (RU) load-balancing. The operation of the method is assessed and its performance is compared to auto regressive integrated moving average (ARIMA) algorithm, which uses statistical network information to perform resource allocation for FlexE links.

Assuming information about the global state is unknown, the environment becomes partially observable. This is expected to be the case for F-RAN considering the latency constraints. In a partially observable Markov decision process (POMDP) this leads to uncertainty in making decisions, as the outcomes are perceived as stochastic by the agent. With the goal of maximising the number of users served by multiple FAPs in a POMDP under stringent constraints, DDQN fails to converge to a solution. For this problem, a multi-agent double deep Q-network (MA-DDQN) solution is proposed, in which FAPs exchange information about their experiences after every learning step. The method is tested in a simulated F-RAN environment and its performance is discussed.

While focusing on allocating resources provides satisfactory performance it has been shown to only be a half of the complete solution. Since FAPs are capable of caching data or applications close to the user, controlling what is being cached is equally as important. This makes the problem a capacitated facility location problem (CFLP). A method is proposed that chooses which services to cache in the fog layer in the first step and how to distribute the content to the users in the second step. An algorithm based on MA-DDQN, termed focused batch DDQN (FB-DDQN) is used to solve the optimisation problem of improving the cache hit ratio in the fog layer. FB-DDQN focuses on areas of the state space that the agents know they are uncertain with and requests other agents to share their solutions to similar states, which improves the generalisation of the less knowledgeable agent. The results are compared to the base MA-DDQN algorithm, as well as a

RL algorithm that is written for this task.

List of Publications

- **I. Laurinavicius**, H. Zhu, J. Wang, "Beam Squint Exploitation for Linear Phased Arrays in a mmWave Multi-Carrier System", *2019 IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, 2019, pp. 1-6
- M. Niaz Chughtai, S. Noor, **I. Laurinavicius**, P. Assimakopoulos, N. J. Gomes, H. Zhu, J. Wang, X. Zheng, Q. Yan , "User and resource allocation in latency constrained Xhaul via reinforcement learning", *J. Opt. Commun. Netw. 15, 219-228 (2023)*

Contents

Acknowledgements iii

Abstract v

List of Publications ix

1 Introduction 1

1.1 Motivation 1

1.1.1 Objectives of the Research 7

1.1.2 Contributions of the Thesis 9

1.2 Thesis Outline 11

2 Background Study and Literature Review 13

2.1 Network Architectures 13

2.1.1 Cell-based Radio Access Networks 13

Cellular deployment 13

Distributed Radio Access Networks 14

Centralised Radio Access Networks 16

Coordinated Multipoint Communication 18

2.1.2 Cell-free Radio Access Networks 19

Cloud Radio Access Networks 19

Fog Radio Access Networks 21

2.2 Machine Learning 23

Supervised Learning 24

Unsupervised Learning 25

2.2.1 Reinforcement Learning . . . 27
Model-free learning . . . 29
Open problems of RL . . . 31
2.3 DRL Applications in Mobile Networks . . . 32
2.3.1 Dynamic network access . . . 34
2.3.2 Resource control . . . 34
2.3.3 Wireless caching . . . 34
2.3.4 Data offloading . . . 35
2.4 Summary . . . 35

3 Modelling a latency constrained RAN and Xhaul 37
3.1 Introduction . . . 37
3.2 Network Model . . . 39
3.3 CoMP and Latency Model . . . 43
3.4 Approach to Simulation . . . 46
3.5 Simulation Results for FlexE workers . . . 52
3.6 Simulation Results for offloader . . . 54
3.7 Conclusion . . . 57

4 User Association in Fog Radio Access Networks 59
4.1 Introduction . . . 59
4.2 System Model . . . 64
4.2.1 Transmission model . . . 64
4.2.2 Cache and Scheduling Model . . . 67
4.2.3 Problem Formulation . . . 69
4.3 Environment Design . . . 71
4.4 Neural Network Distribution . . . 74
4.5 MA-DDQN Algorithm Design . . . 77
4.6 Simulation Results . . . 79
4.6.1 Training Results . . . 80

4.6.2 Performance Metric Results . . . 85
User Selection . . . 85
Power Consumption and Service Delay Results . . . 88
4.7 Summary . . . 90

5 Distributed Cache Access in Fog Radio Access Networks 91
5.1 Introduction . . . 91
5.2 System Model . . . 95
5.2.1 Ricean Fading Transmission . . . 95
5.2.2 Dynamic Service Caching Model . . . 98
5.2.3 Problem Formulation . . . 99
5.3 Focused Batch Double Deep Q-Network . . . 100
5.4 Environment Design . . . 102
5.5 Simulation Results . . . 104
5.6 Summary . . . 108

6 Conclusions and Future Work 111
6.1 Introduction . . . 111
6.2 Conclusions . . . 111
6.3 Future Work . . . 115

Bibliography 119

List of Figures

2.1 A distributed RAN (D-RAN) system with Frequency Reuse factor of 3. 14
2.2 An example of centralised RAN with a BBU pool serving multiple cells. 17
2.3 A Markov decision process showing interactions between the agent and a state 28

3.1 C-RAN architecture with CoMP. 40
3.2 FlexE Xhaul network architecture. 41
3.3 Markov Decision Process for a FlexE agent. Background traffic is introduced as a simulation parameter. 42
3.4 Markov Decision Process for a Controller Agent. Latency is dependent on the calendar slots allocated by FlexE agents. 42
3.5 RU coordinates in x,y plane (Case-1) [67]. 47
3.6 Reward vs difference of allocated slots and input traffic demand. . 50
3.7 Training results of controller agent, total achievable bit rate (a), number of users in RU groups (b) for RU group position in Figure 3.5. 52
3.8 FlexE Agent slot allocation (a), input traffic (b). 53
3.9 Controller agent deployment results, total achievable bit rate (a), number of users in RU groups (b) for RU positions in Figure 3.5. . . 55

4.1 System model showing a F-RAN deployment as an extension to the cloud. Links between FAPs (short dashed lines) allow for CRRM. 65

4.2 Experience Sharing helps agents learn from other agents' experiences as if they were their own through off-policy learning. 75
4.3 Neural network used for DDQN and MA-DDQN algorithms. . . . 80
4.4 Training results over entire training period for DDQN and MA-DDQN algorithms, higher is better. 1 epoch is equal to 20 training steps. 82
4.5 Difference between predicted reward and received reward over entire training period for DDQN and MA-DDQN algorithms. 83
4.6 Simulation output showing number of UEs served under required constraints. 85
4.7 Simulation output showing the number of users selected when the number of cached objects changes. 87
4.8 Simulation output showing the average powers consumed per user. Training steps T = 50000, $\tau = 119$. 87
4.9 Delay performance for 1 and 3 cacheable objects (c.o.). 89

5.1 Recorded probabilities of each service request 99
5.2 Number of batches sent over the network 106
5.3 Reward comparison between FB-DDQN, MA-DDQN and DCC-MAB algorithms 107
5.4 Cache hit-rate for algorithms under test 108
5.5 Number of users served by the fog layer 108

List of Tables

3.1 Simulation Parameters 48
3.2 Functional Split Upscaling Parameters 48
3.3 Deep-Q Network Parameters 49
3.4 Queueing Delay Parameters 50
3.5 Background Traffic effects on RU group offloading. 54

4.1 Training results comparison for a large and a small network using MADDQN 84

5.1 Normalized application requirements 104
5.2 Simulation parameters and hyperparameters 105

List of Abbreviations

A2C	Advantage Actor Critic
AR	Augmented Reality
A3C	Asynchornous Advantage Actor Critic
AP	Access Point
ARIMA	Auto regressive Integrated Moving Average
AWGN	Additive White Gaussian Noise
BBU	Baseband Unit
c-RAN	Centralised Radio Access Network
C-RAN	Cloud Radio Access Network
CFLP	Capacitated Facility Location Problem
CoMP	Co ordinated Multi point
CSI	Channel State Information
CU	Core Unit
CPU	Central Processing Unit
CUA	Cummulative Under Allocation
CRRM	Coordinated Radio Resource Management
C.O.	Cached Object
CB	Coordinated Beamforming
DRL	Deep Reinforcement Learning
DQN	Deep Q- Network
DDQN	Double Deep Q- Network
D-RAN	Distributed Radio Access Network
DU	Distributed Unit
EB	Exa bytes
eNB	evolved Node B
FAP	Fog Access Point
F-RAN	Fog Radio Access Network
FlexE	Flexible Ethernet
IDDQN	Individual Double Deep Q Network
JP	Joint Processing
JT	Joint Transmission
LC	Line Card
MOA	Mean Over- Allocation
MIMO	Multiple Input Multiple Output
MARL	Multi- Agent Reinforcement Learning
MDP	Markov Decision Process
MA-DDQN	Multi- Agent Double Deep Q-Network
M-MIMO	Massive Multiple Input Multiple Output
MSC	Mobile Switching Center

MILP	Mixed Integer Linear Programming
O-RAN	Open Radio Access Network
OPEX	Operation Expenditure
POMDP	Partially Observable Markov Decision Process
RRH	Remote Radio Head
RL	Reinforcement Learning
RU	Radio Unit
RAN	Radio Access Network
SINR	Signal to Interference plus Noise Ratio
SVM	Support Vector Machine
TD	Temporal Difference
uRRLC	Ultra Reliable Low Latency Communication
UE	User Equipment
XR	Mixed Reality
ZF	Zero Forcing

Chapter 1

Introduction

1.1 Motivation

The Internet is so much more than sites on the web and phones no longer serve a single purpose. It would surprise the original inventors of the first mobile phone, the Motorola Dynatac 8000X, which made its first call in 1973 (released in 1984), how the device they have engineered has evolved from a clunky rectangle the size of a small bag. As it stands now, mobile phones have become smartphones, tiny gadgets, that fit inside of pockets of a light jacket, with processing power equivalent to that of a personal computer. This evolution is necessary, since the conception of the first mobile phone, consumers have started requiring their daily usage device to be able to do much more than make call or send a text message, both of which are low-bandwidth applications. Phone and mobile network service evolution can be divided into generations, each lasting roughly a decade, with mobile analog communications being the first generation (1G), between 1979 and mid-1980s [1]. At the time of writing this document, fifth generation (5G) of mobile networks is being rolled out for widespread use. Continuous evolution is crucial for both mobile networks and devices using them. Networking is as important to the infrastructure of a country as roads, being unable to support user requirements or losing service can lead to dramatic consequences.

Since the adoption of mobile networks by the public, the number of active users has continued to increase. According to the Ericsson Mobility Report [2],

the number of active mobile subscriptions is 8.3 billion, which exceeds the global population of approximately 8.02 billion. It should be noted that not all of these subscriptions are unique, as it is estimated that about 60% of the population has a mobile subscription. Furthermore, non-human users such as machines, sensors, and vehicles are now being considered as active users of 5G cellular IoT networks. The report predicts that by 2028, the number of active cellular IoT devices will double from 2.7 billion to 5.5 billion. Currently, the global network traffic consumption stands at 146 exabytes (EB) per month. Despite the steady increase in the total number of network users expected in the coming years, the network traffic itself will experience exponential growth, reaching 452EB by 2028 [2]. This is primarily due to the fact that phones are predominantly used for media consumption, with only 13% of the traffic being attributed to text and voice usage. The majority of the traffic consists of video and audio traffic, social networking, and various software downloads. Additionally, multimedia applications, particularly virtual reality, are gaining consumer traction and are projected to increase their market value by 15% [3]. Research in virtual reality is expected to pave the way for augmented reality (AR) and mixed reality (XR) applications as well. Since these applications have different network consumption profiles compared to voice and text data, network service providers and equipment manufacturers must seek novel solutions to address associated challenges. The conventional distributed networks used in 4G LTE are no longer sufficient to meet the requirements of high bandwidth, coverage, and user experience. This has led to the proposal of the Cloud Radio Access Network (C-RAN) architecture.

C-RAN is primarily a cell-based mobile network architecture that moves processing from base stations, also known as evolved Node Bs (eNBs) into a centralized baseband unit (BBU) pool [4]. With most heavy computation taking place at the BBU, remote radio head (RRH) units are used for radio reception and transmission in a cell. These units convert between radio frequencies and baseband

data, which is transmitted via a powerful fronthaul to the BBU. This network architecture has been proved to reduce the overall network power consumption, improve cell spectral efficiency [5] and for manufacturers, improve the service time to market – how fast operators can deploy upgrades to the network infrastructure. The benefits for user experience are present, too. Cloud is capable of caching data that users would like to access, which leads to reduced latency [6]. Because BBUs control many RRHs there are coordination solutions that can reduce inter-cell interference, leading to greater data rates experienced by individual users.

While C-RAN appears to have addressed the critical challenge of supporting novel applications that require low latency and high bandwidth, there are still some outstanding issues with this architecture. The goal of improving data rates for users extends beyond just enhancing the network architecture. Mobile network stakeholders are now exploring new spectrum ranges, particularly carrier frequencies above 20GHz, known as mmWave spectrum [7]. Unlike the sub-6GHz bands, mmWave is more susceptible to various path loss effects, including attenuation caused by oxygen particles and rain. To mitigate these challenges, the usage of massive multiple input multiple output (M-MIMO) is proposed. M-MIMO utilizes a large number of antenna elements to amplify the signal gain within a small angular area. However, it is important to note that M-MIMO requires significant processing power and is also vulnerable to additional parasitic effects, such as beam squint [8], depending on the hardware used. As a result of the deployment of numerous remote radio heads (RRHs), a significant amount of raw data will be transmitted over fronthaul links, particularly from RRHs supporting M-MIMO antenna arrays. This substantial data volume cannot be overlooked. In the absence of efficient methods for handling channel information, there is a risk of introducing additional latency. Furthermore, the parasitic effects and the requirements of beamforming algorithms in M-MIMO communications necessitate additional processing, which further exacerbates the negative impact

on user quality of service.

Open RAN (O-RAN) is a virtualised approach to C-RAN that has recently been standardized by multiple vendors in parallel to third generation partnership project (3GPP) [9]. By virtualising the C-RAN architecture it is possible to unbind processing units in the BBU from specific RRHs. By doing this, BBU can use the processors not in use by inactive RRHs to be used where there are heavier loads[10]. Such virtualization is beneficial, as flexibility is needed to tackle the challenges with C-RAN. Separating software from hardware also makes cell-free communications possible [11]. Cell-free means that users experience no cell boundaries during transmissions, with macro diversity leading to consistent data rates across the radio deployment area, which is an issue with conventional cellular networks. Since joint processing requires some coordination, cell-free networks inherently benefit from centralized RAN deployment, which benefits in return from low-complexity processing.

Fog RAN (F-RAN) is a specific future deployment of O-RAN where general purpose processors are deployed not only within the BBU, but also within the RRHs, leading to a hybrid centralised-distributed network architecture [12]. Additionally, RRHs can be equipped with caches, bringing computation and caching closer to the users, solving network latency and fronthaul congestion issues that are common in centralised-only architecture. This evolved smart and cache-capable RRH is known as a fog access point (FAP). Since FAPs are equipped with more powerful processors compared to its earlier generation counterpart, FAPs are capable of performing some baseband radio processing tasks [13], and if they are connected with each other through backhaul, they can perform cooperative radio signal processing and radio resource management. Because the architecture is a mixture of centralised and distributed architecture, the BBU is still capable of performing tasks that require more processing power, leading to load balancing across the network architecture. With load-balancing, low latency and fronthaul alleviation, F-RAN improves the estimated channel accuracy and the data rates

experienced by the users.

Since F-RAN relies on FAPs cooperating with each other, there are new challenges that need to be addressed before the architecture can fully realize its benefits compared to C-RAN. The first issue is that, unlike centralized caching accessible at the BBU, FAP caches are relatively small, and depending on the content, the cache hit ratio can be low [14]. Additional algorithms are required to support cache access and maximize the hit ratios for cached content. Another challenge to address is the control of content caching locations. Furthermore, in F-RAN, which is a cell-free architecture, multiple FAPs can collaborate to serve the same group of users [15]. If each FAP has a large number of antennas, it is evident that this can result in a combinatorial explosion of potential solutions when determining which users are served by which FAPs. Therefore, it is crucial to develop efficient algorithms that can quickly arrive at solution sets without compromising the optimality of the solution.

It is already known that deep learning is capable of providing solutions to difficult issues. Over the last 5 years machine learning and artificial intelligence algorithms have become advanced enough to solve problems that require creativity and deep domain knowledge [16]. One of the most popular applications using deep learning at the time of writing is ChatGPT, the application utilizes unsupervised learning to teach a recurrent neural network how to perform any task that is requested of it. Unsupervised learning, similar to supervised learning has to learn from a data set. Closer to the domain of mobile communications there are also many novel solutions that use supervised learning to achieve good results. The issue with using such algorithms is that there is a heavy dependency on having a large data set to learn from [17]. In the case of unsupervised or supervised learning for general applications this data can be acquired with questionable integrity from various websites, infringing on copyrights of the existing content. In mobile network applications using these algorithms introduces a novel limitation. The solutions are only as viable or useful as the algorithm used to acquire

the data set, the neural networks cannot find solutions better than what is available in the data set.

A way of breaking through the limitations of learning from data sets is to learn by trial and error [18]. This can be achieved by deploying a learning algorithm in an environment with no prior knowledge about it, providing it with data about the current state of the environment, acquiring its output for action and supplying it with the resulting data about the following state and a reward. This is known as reinforcement learning (RL) [19] and the goal of this learning paradigm is to extrapolate from all of its previous experiences and maximise the rewards. While the most simple forms of RL problems are easily represented by Markov decision processes (MDP) that can be easily solved using tabular methods and optimizing Bellman optimality equations[18], the problems that typically are seen in mobile networks require a deep reinforcement learning (DRL) approach. DRL algorithms are particularly interesting, as they can operate in environments with fine-grained details, having tiny differences between different inputs in datasets. What is more, by approximating Bellman optimality equations using neural networks, it becomes easy to create DRL agents that can understand the problem in a general way. Generalisation is useful, as it explains what agents can expect from state transitions even if they have not been in a particular state yet.

While DRL has many different approaches and can solve many problems it has to be carefully tailored for each application. Single agent algorithms, where an agent tries to maximise its reward in an environment with simple goals, have been very well studied. There are many popular algorithms that tackle different problems in single agent environments: Advantage Actor Critic (A2C) [20], Deep Q-Network (DQN) [21], Double Deep Q-Network (DDQN) [22]. Multiple agent learning is a lot more complicated. There is additional randomness and uncertainty associated with the environment due to the fact that the additional agents are learning simultaneously. This introduces a partial observability problem [23], that a lot of the previously mentioned algorithms cannot handle well. It

becomes challenging for agents to learn in such environments without considering other agents' impact, and maximizing its own reward can affect other agents negatively. Regardless of where deep learning agents are deployed, at the network side or the user side, mobile network reinforcement learning problems are all multiple agent problems. To be able to utilise DRL in future generation mobile networks, modifications to existing algorithms or novel algorithms have to be developed. With these motivations in mind, the objectives of the research presented in the thesis will be discussed in the following section.

1.1.1 Objectives of the Research

Deep learning algorithms have been proposed for route optimization in fronthaul for C-RAN [24] [25]. Because DRL can be very efficient in producing outputs, it was shown to work well for ultra-reliable low latency (uRLLC) communications as well as F-RAN under latency constraints. Previous studies did not acknowledge that channel estimation is dependent on the speed of the Xhaul connecting the network, latency affects the maximum achievable data rate of the system. Additionally, they focus on the effects of a single source of delay, such as queueing or propagation. It is not accurate to assume that either can be ignored, as a combination of all sources can negatively affect the performance of a network.

So, in this thesis, firstly it is proposed to look at the C-RAN network through a combined model of the RAN and Xhaul. For this network model, a DRL algorithm is suggested to optimize the route selection and resource allocation with the goal of maximizing experienced data rate. Using Flexible Ethernet (FlexE) for Xhaul enables the use of DRL at each aggregation node. The use of DRL aims to accurately allocate resources by predicting network traffic states and adding or removing bandwidth from the links. In the RAN, RUs use DRL agents to decide how to balance users and off-load them to other groups of RUs which are experiencing lower Xhaul latency.

F-RAN has been proposed as a possible solution to fronthaul loading issues in

C-RAN, by placing cooperative radio resource management and cell-free MIMO at the edge of the network. User-FAP association is a mixed integer linear programming problem, which is known to be NP-Hard. This problem is addressed in [26] and [27]. However, these studies failed to address the limited cache issue within FAPs. There are contents that cannot be cached and such interactive content is also delay-sensitive. Due to these factors, solution sets for UE-FAP association are exponentially large. Additionally, studies do not take into account that due to latency requirements deep learning agents do not have time to exchange information prior to making a decision, which makes the environment a POMDP.

Secondly the thesis presents an analysis of F-RAN as a POMDP. Because of the strict latency constraints of future mobile networks, a fully observable MDP cannot be considered. In such environments, agents have to be able to predict how other agents deployed in the same environment would behave under a given observation. Low complexity algorithms and conventional DDQN fail to converge to a good solution.

From the above, because individual DDQN algorithms fail to adapt to a POMDP efficiently, the thesis proposes to implement a novel multi agent DDQN algorithm as a third contribution. This algorithm uses experience exchange to improve off-policy learning by introducing a diverse set of experiences to learn from. The proposed algorithm is deployed in an environment that models F-RAN with multiple FAPs and a large number of UEs making random requests. Its goal is to maximise the number of users that can be served successfully while optimising network KPIs such as power consumption and user latency. The solution is analysed for both training-related performance as well as practical performance.

Network performance has limited improvement when only one objective is being optimized at a time. When the caches are static and cache selection is optimal, or on the other end of the spectrum, if the cache management is smart and cache selection is random, the cache-hit ratio is limited. In [14] authors addressed

cache management by using a tabular Q-learning RL method, under the assumption of random user association, whereas [28] studied user association using a hard-coded two step algorithm. These studies showed good results, but it is hard to judge the optimality of the solutions, since their optimality can depend on the performance of the other algorithm. This is notably mentioned in [29] in which a tabular RL method is used to select which services to cache at the edge and using a low complexity method to guide requests to adjacent edge computing nodes or the cloud. Some practical issues are ignored in the aforementioned papers. When FAPs are used to run services, there are memory and CPU limitations which were not considered. What is more, in [29] the states of adjacent nodes are ignored, which could result in reduced quality of service. It is clear why, considering this extra information leads to a complex problem that is difficult to solve using hard-coded or tabular algorithms.

Motivated by this, as a fourth objective, a study is performed on how efficient deep learning algorithms can optimize multiple objectives at the same time. A two step optimisation algorithm is developed, which utilizes the CU to synchronize data available to the FAPs and performs cache management in the first step and cache selection in the second. Since CU is providing the observations to each FAP, they are aware of some states of their adjacent FAPs, however they don't see the entire network state, which makes this a POMDP problem. The thesis proposes an extension to the MADDQN algorithm to select experiences in a focused manner by analysing which experiences the agent struggled to predict accurately and requesting other agents to provide similar state transition experiences.

Using these objectives as motivation, the contributions of the thesis are summarised in the next subsection.

1.1.2 Contributions of the Thesis

This thesis provides the following significant contributions:

- A deep reinforcement learning algorithm for C-RAN path selection and load balancing. The method is applied to an environment modelled after a joint RAN and Xhaul network. By efficiently allocating bandwidth to the Xhaul, latency is reduced, which maximises the possible achievable sum-rate by the RAN, due to an accurate channel estimation. The method aims to not over-allocate resources as it would lead to greater operational expenditures, the RAN maximises its performance by off-loading users between RU groups to both reduce the channel estimation latency and to maximise the number of users served.

- An analysis of F-RAN as a POMDP. Due to latency constraints, a fully observable MDP may not be available, as such, the thesis presents an analysis which is reproducible and can be used as an environment to study multi-agent deep learning algorithm performance.

- A multi-agent DDQN algorithm that uses naive experience exchange between multiple agents to improve system performance. In the thesis this algorithm is used to make UE-FAP association decisions at the FAP. Because DRL algorithms are agnostic to constraints, additional algorithms have to be employed to enforce them, this is included in the thesis.

- A multi-step solution using MADDQN with focused experience exchange is used to optimize multiple objectives simultaneously. In the first step, agents using MADDQN algorithm place in-demand services in the cache, while in the second step, agents perform cache selection. During the cache selection phase, users make service requests to the fog and agents make decisions whether to serve them themselves or to off-load them to other FAPs which would have resources for the operation. In this phase agents also collect information for the first step.

1.2 Thesis Outline

The rest of the thesis is composed as follows:

Background literature and state of the art in the field are presented in Chapter 2. In this chapter cell-based and cell-free network architectures are introduced. The introduction includes both the overview of these architectures as well as their benefits and limitations. Reinforcement learning is attractive as it can support novel radio network architectures, a brief introduction to reinforcement learning is provided. Due to performance limitations of pure reinforcement learning, deep reinforcement learning has been important in recent studies and is introduced in the chapter. Finally, it will present state of the art applications of deep reinforcement learning in mobile network applications.

Chapter 3 offers a DRL solution for a latency constrained C-RAN environment. The chapter models a RAN/Xhaul network which uses FlexE links for the fronthaul. In the Xhaul, an agent is placed in each FlexE aggregation node, to control the FlexE calendar and allocate slots to links. The goal is to provide just enough resources for the links, as over-allocations are wasting resources and under-allocations lead to queueing and reduced performance. A separate controller agent controls the loading on RU groups in the RAN, as additional load is introduced on the links that are used in the fronthaul. The results are discussed and compared with an algorithm that uses statistical data to allocate resources.

Chapter 4 will introduce the F-RAN environment as a POMDP. Due to a unique environment configuration a new solution must be provided. As such a solution for a user-FAP association problem using a novel multiple-agent double deep Q-network algorithm is introduced. The performance of this algorithm is compared to independent double deep Q-network algorithm. The comparison of the two algorithms is studied for both practical metrics as well as deep learning specific results.

Chapter 5 presents an multiple step algorithm based on MADDQN to solve a

joint cache population and service selection problem. This problem is described as a capacitated facility location problem (CFLP). CFLP is proven to be NP-hard, due to the combined nature of the solution it is a complex task that cannot be easily solved with greedy or disjointed algorithms that optimize a single objective at a time. Results of the proposed algorithm versus another deep learning-based algorithm are presented.

Finally, Chapter 6 concludes the contributions of the thesis and presents possible future work that the thesis is a baseline for.

Chapter 2

Background Study and Literature Review

2.1 Network Architectures

2.1.1 Cell-based Radio Access Networks

Cellular deployment

When the first commercial mobile network launched in 1979, it was a cellular network. A cell is one of the most fundamental entities in RAN [30], as it defines the effective area of communications from a radio unit or base station. Commonly, cells are represented as hexagons, more realistically, the shape of a cell is dependent on the geographic topology and channel conditions. Compared to a single-transmitter network, cell-based communications have a few advantages: smaller area communications, which need less power and can utilize more frequency resources, to increase network coverage additional base stations can be deployed to create new cells. Typically, base stations are deployed with some cell boundary overlap to support smooth handovers between cells, but if base station cells are close and they share the same frequency they can cause significant interference to each others' users.

To avoid inter-cell interference, frequency reuse was suggested. By choosing a frequency reuse factor a pattern in cell deployment can be generated to minimise

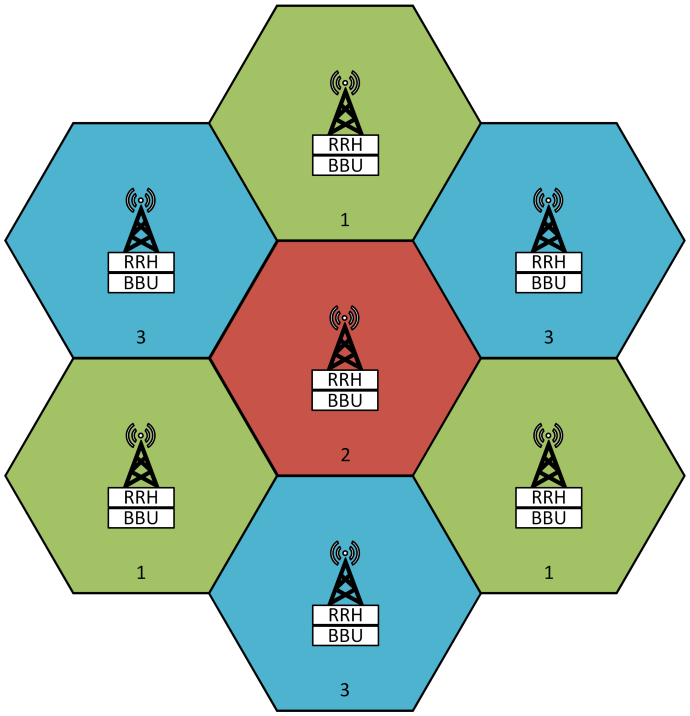


FIGURE 2.1: A distributed RAN (D-RAN) system with Frequency Reuse factor of 3.

the interference between cells that share the same frequency, as shown in Figure 2.1. This also leads to a higher capacity in the system, as users can call on the same frequency, if the closest base station does not have the frequency available, the second nearest can be used. This was called resource borrowing and there were supervisors, such as the mobile switching center (MSC), that prevent resources from being borrowed in a cell that is also under heavy utilisation.

Distributed Radio Access Networks

Cell-based RAN is implemented as distributed RAN (D-RAN), which has been used for a couple of decades. D-RAN is an umbrella architecture term that encompasses GSM, E-UTRA and LTE [31]. In this architecture the base station is composed of an RRH and a BBU. The RRH is responsible for transmitting, receiving and converting digital and analog signals, whereas the BBU is used for digital processing. In these networks, it is common that one user equipment (UE) connects to one base station. Since the initial uses for mobile networks were to handle

voice calls, a data use-case that can require between 10-100 kilobits per second (kb/s) [30], the D-RAN architecture was great as it provided enough threshold signal to noise ratio to support this application. However, as the network requirements are becoming greater due to emerging applications, there have been few innovations that could improve the network performance over the entire range of the cell.

The first innovation was spectrum utilization. As mobile networks got data traffic demands, standards required more spectrum to meet them. GSM, also known as 2G, the first data traffic enabled standard increased the channel bandwidth from 30kHz to 200kHz. With user requirements increasing, 3G introduced 5MHz channels using WCDMA and in 4G LTE the channel can be up to 20MHz. As data traffic demands are increasing rapidly, the current frequency spectrum is starting to get crowded for sub-6GHz frequencies. For future mobile communications mmWave communications are being proposed which will utilise spectrum far beyond the current range, by targeting frequencies over 30GHz with single-channel bandwidths as high as 250MHz and total aggregated channel bandwidth of 1.75GHz.

The second innovation was densified networks. By increasing the number of base stations that are deployed in a single area the number of served users could be increased. To support denser networks, cell sizes had to be reduced by decreasing the transmit power of base stations. In turn, because the cell sizes are smaller, users are much closer to base stations, which improves link quality and the resulting data rates. The significant difference in network evolution is that as small-cells were used for network off-loading, in 4G they are an important component of the D-RAN architecture, creating a heterogeneous network which uses large and small-cells to tackle traffic loads.

The third innovation was an increase in the number of antennas and directional transmission. The benefit of this is two-fold. As the frequency spectrum increases towards mmWave communications, it is difficult to implement a mobile

network that can support cells of the same size as sub-6GHz. Because of path-loss that is significantly greater than sub-6GHz, due to penetration losses, absorption by gas particles and rainfall effects, high directionality produced by a large number of antennas can mitigate the effect. Additionally, as the number of antennas increases, with digital and analog processing arrays produce high-gain beams, which introduces an additional resource which can be used to serve more users. Indeed, through the use of multi-user MIMO, the same frequency resource can be allocated to many different beams, increasing the overall network capacity.

This architecture still has several limitations. The first one is limited capacity, as a result of the limited number of resources that are available for use. When all of the available channels are utilized additional users cannot be served. The coverage of D-RAN is another issue and is limited by the range of a base station. When a user moves between base stations a handover must take place. Handovers can result in calls being dropped or service being interrupted. Mobile operators are also under pressure due to increasing operational expenditures. D-RAN is deployed to support peak-time traffic, so when the traffic is off-peak, available resources are not used, leading to a waste of resources.

Centralised Radio Access Networks

Some of the issues of D-RAN can be solved by creating a centralised radio access network. By deploying the RRH close to the user to transmit and receive signals, the BBU can be set-up far away connected via common public radio interface (CPRI) and oversee the network remotely, as shown in Figure 2.2. When applied to many different cells, there remains one central location with a BBU pool, which communicates with many RRHs that are serving their respective cells. Without performing any virtualisation, centralised RAN can control RRHs to tackle traffic on-demand. There is no longer a need to maintain peak-hour traffic for the entire operational run-time. As the traffic requirements change, BBU pool can adapt by

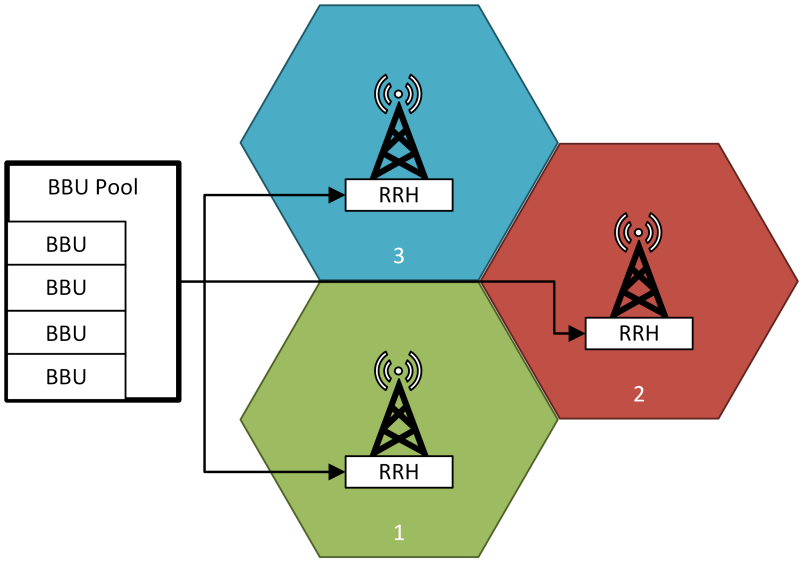


FIGURE 2.2: An example of centralised RAN with a BBU pool serving multiple cells.

using the resources of inactive cells to support the cells that are experiencing high traffic.

Because centralised RAN controls many RRHs, which can be adjacent the available resource problem can be addressed as well. With coordination within the BBU pool, RRHs can be aligned to share spectrum resources [32] as well as reduce interference between each other [33]. Additionally, inactive RRHs can be switched off [10], which is a power-saving measure as well as a method of controlling interference, especially at the edges of the cell. Finally, moving between cell-sites leads to less service drops, as the BBU pool has superior mobility management, by coordinating the handover between different RRHs.

Although there are benefits of implementing centralised RAN, it is difficult to implement. One of the key components of C-RAN is Xhaul, the underlying network composed of fronthaul, midhaul and backhaul. More research is needed to address challenges in the Xhaul, to improve latency and resource utilization, as well as addressing the operational expenditures of running centralised RAN. One of the more attractive implementations of centralised RAN is cloud RAN, which is based on the concept of virtualisation, running on general purpose hardware,

rather than purpose built. As general purpose hardware can have varying resources, efficient algorithms have to be developed to support centralisation.

Coordinated Multipoint Communication

One important feature for centralised RAN is known as coordinated multipoint (CoMP) transmission [34]. Introduced in 3rd generation partnership project (3GPP) Release 9, CoMP is a method to tackle the growing traffic requirements. One of the biggest challenges in the physical layer is serving users at the cell edge at a satisfactory quality of service. Users at the cell edge create a unique scenario as they are affected by very high interference from other cells and due to the high distance from the base station of their own cell. At a high level, using CoMP multiple base stations can serve a single UE which not only mitigates interference, but elevates the possible data rates experienced by the cell-edge user.

There are a couple of different approaches to base station coordination [35]:

- Centralized coordination is controlled by the BBU pool. By collecting the necessary channel state information (CSI) and user information the BBU pool can perform advanced scheduling and processing. This in turn allows for significantly reduced inter-cell interference. One of the most significant challenges for this implementation is the high fronthaul delay.

- Distributed coordination is performed by linking the distributed units (DUs) via an Xn or an enhanced X2 interface as shown in [36], following the ETSI terminology, the thesis refers to this as a backhaul link [37]. Neighbouring RRHs can perform exchange of CSI to facilitate joint transmission, as in [38]. Typically in this approach, one of the eNBs takes the role of the primary eNB, taking on the resource management tasks and communicating with the centralised unit, while the other eNBs are secondary.

CoMP has many different implementation schemes, for both uplink and downlink communication. In this thesis joint processing (JP-CoMP) is adopted. Joint

processing is further divided into two types: joint transmission (JT), and dynamic point selection [35]. In support for the thesis joint transmission is further explained. Each cooperating base station receives CSI, such as channel quality, precoding matrix and rank information from the user, and they handle it mutually. With this, various cells can transmit the same data to the user. Since the data is the same, it is said that the user experiences power and signal-to-noise ratio gains, which improve the quality of service. JT can be achieved by introducing an additional logical layer to the network, the DU. With centralised coordination, the DUs are controlled via CU, whereas user data transmission happens by routing data through the primary DU to the secondary DUs.

2.1.2 Cell-free Radio Access Networks

Based on the concept of CoMP a new communications paradigm can be explored. Rather than serving users in pre-determined cells, remove the area limitation of network and use CoMP mechanisms to jointly serve users as long as they are within reach of an RU, this is known as a cell-free network [39]. Architecturally, the cell-free network is similar to cell-based, i.e. access points or RUs are located at the edge, close to the user, while there are processors, also known as DUs connected to the central unit (CU) [40]. The difference between cell-based and cell-free is theoretically invisible to the user, as most changes affect the infrastructure and signal processing at the network. Two specific network architectures can be applied to implement cell-free RAN: C-RAN and F-RAN.

Cloud Radio Access Networks

To tackle centralised RAN requirements, network functions can be virtualised. Virtualisation is achieved by abstracting hardware resources in a way that many different applications can share one piece of hardware [31]. A common example of virtualisation is in operating systems, a single storage device, such as a hard

disk drive can be partitioned into many different drives for varying functions. Such partitioning does not create new hard disk drives, but virtually they are accessed as separate locations. The process of virtualisation in RAN applies to all of the layers of the RAN architecture, the central unit (CU), the distributed unit (DU) and the radio unit (RU). Just as storage or processing can be virtualised so can the network and the radio resources. As stated at the start, C-RAN is a centralised RAN architecture, this means that the benefits of centralised RAN apply to C-RAN as well.

Different from rigid centralised RAN, C-RAN supports heterogeneous network architectures that integrate different types of RRHs, such as macrocells, small cells and massive MIMO arrays, thus providing seamless and ubiquitous connectivity to users. Another significant benefit is the scalability and cost-effectiveness of C-RAN [35]. Due to virtualisation and use of general purpose processors (GPPs), C-RAN deployment incurs lower CAPEX and OPEX, allowing operators to meet consumer demands by acquiring more GPPs and simply plugging them in, versus planning, buying and deploying a full cell. It also introduces a high reliability into the network. As resources are virtualised, it does not matter if the physical hardware fails, as it can be easily replaced by another functional hardware and since everything is managed from a single cluster, recovery and robustness can be easily automated.

As C-RAN is becoming standardized and well-studied, there are multiple challenges that are widespread in literature:

- Latency requirements are a major issue, mobile network evolution requirements are placing stringent constraints on the mobile network systems. It can be easy to see that in order to provide service to a user, the data signal has to traverse large distances, both physically and logically. Compared to distributed networks, there are additional processing requirements and the network is composed of multiple hops before it reaches the BBU. This is one of the key aspects that is being addressed in research [41], [42].

- Data rate requirements at fronthaul: since fronthaul has to handle both user data as well as the necessary CSI it is important to have physical links capable of supporting these large requirements. Optical fiber is one of the enablers for such fronthaul transmission. On one hand this is can be a significant expense for mobile network providers that do not own optical fiber infrastructure. On the other hand it may not be necessary to implement optical fiber as solutions are being developed to compress the data that traverses the links. Another suggestion is to replace conventional common public radio interface (CPRI) links with other technologies, such as FlexE.

Fog Radio Access Networks

Fog radio access networks (F-RANs) are a novel network architecture that combines fog computing and radio access networks (RANs) [12] to provide low-latency, high-reliability and energy-efficient wireless services for the fifth generation (5G) and beyond [43]. F-RANs evolve the architecture of C-RAN by replacing the RRH with a computationally more powerful FAP to form a hybrid distributed and centralised architecture. FAPs are connected to the core network via fronthaul links. FAPs are edge devices that perform signal processing, caching, computation offloading and network management functions.

Caching and computation offloading are achieved in literature by introducing a content-centric network (CCN) layer between application and packet data convergence protocol layers [44]. Instead of reading the host address, it reads the content identification and uses each node in the network as an in-network cache. [44] provides additional information of how content requests with mobile network caching work. Additionally, in [45] an example method for coordination of FAPs under a content request is provided, by using an edge node to perform some centralisation functionality. However, in F-RAN BBU pool still provides global coordination and resource allocation for FAP groups.

By moving some of the signal processing, caching and computation tasks from centralised servers to FAPs, F-RANs can reduce the end-to-end delay and improve the quality of service for latency-sensitive applications such as augmented reality, virtual reality and online gaming [15]. By exploiting the diversity and redundancy of FAPs and RRHs, F-RANs can enhance the robustness and resilience of wireless communications against channel fading, interference, congestion and failures [26]. By optimizing the resource allocation and power consumption of FAPs and RRHs, F-RANs can reduce the energy cost and carbon footprint of wireless networks.

A unique feature of F-RAN is that of multiple transmission modes [12]. By fully utilising cell-free communication via joint transmission, F-RAN supports both centralised and distributed communication. When FAPs are connected to each other, they can perform joint transmission via coordinated radio resource management [45]. The link that would be used is not standardized, but it should be able to transport user plane data between FAPs and perform some X2 interface coordination functionality. If the area of communications is large and FAPs are not interconnected, the underlying cloud radio access network can perform the necessary coordination [46]. Cloud coordination can also be used when a user requests data that is not available at the fog layer. This makes the network flexible to different data traffic requirements.

F-RANs also face some challenges, such as mobility management, interference mitigation and resource optimization [12]. As users move across different FAPs, F-RANs need to handle the handover, load balancing and user association issues efficiently and seamlessly. As multiple FAPs share the same spectrum, F-RANs need to coordinate the transmission and reception of signals among FAPs to avoid or minimize interference. As FAPs have limited resources such as storage, computation and bandwidth, F-RANs need to allocate these resources among FAPs dynamically and adaptively according to the network conditions and user demands.

In summary, F-RANs are a promising network architecture for 5G and beyond that can provide low-latency, high-reliability and energy-efficient wireless services. However, there are still many open issues and future trends in F-RANs that need further investigation and exploration.

2.2 Machine Learning

In this subsection, machine learning is described in general. Supervised and unsupervised learning are introduced here as well. Using this subsection the thesis aims to broadly explain the literature behind these two learning paradigms. By looking at them critically, it will be explained why neither learning method applies well for mobile network deployments.

Machine learning, or artificial intelligence, was first developed by [47]. In the work, artifical neural networks are described for the first time, in which neurons would be "turned on" or "turned off" depending on a stimulation from an input which came from adjacent neurons. This work showed that it is possible to map any computable function to a solution by using an artificial neural network. This neural network had one drawback - it could not learn. Donald Hebb implemented the first updating neural network [48], which used a mathematical rule to change the strength of connection between neurons.

Machine learning is a method of soft computing. Instead of hard-coding a solution to a problem, the goal is to deploy a method that can approximate a solution over time [49]. Similar to how humans learn, machines can get knowledge by observing example inputs and outputs and making predictions for not yet seen inputs and outputs. However, it is rare for machines to develop general knowledge over a subject and extrapolation is very nearly impossible for classical machine learning. Instead, patterns can be discovered for a lot of the data and there are ways to fit predictions along a line, an area, or a cluster. Patterns exist in mobile networks, as channels are representative of environments and there

are many similar environments where mobile networks can be deployed. Additionally, many patterns are time-based, networks see the biggest increases in data consumption in the morning and at night, when people are engaging with media the most.

Supervised Learning

Supervised learning is one of the main subsets of machine learning. Using clearly labelled examples and feeding them into an approximation function or, more commonly, a neural network, agents can learn to predict an output for a new example with high confidence. This is the most popular method of learning to date and is used in most popular every day applications, such as generative AI. Since supervised learning is a broad field, there are many ways to implement it, for example:

- Decision trees [50] are the rudimentary approach. Learning is done by analysing the features and creating a tree with the shortest path to a solution. This can be achieved by choosing the most impactful feature to make a decision on as the top node and further divide the problem into a list of recursive sub-problems.

- Naive Bayes classifiers [51] are a family of algorithms and methods that derive a probabilistic model for the input data to produce output data. Examples are presented split into two parts, the feature data and a response vector. By processing the examples and finding out the probability of making a decision for independent features, the algorithm can make a decision for combinations of inputs it hasn't seen before.

- Support vector machines (SVM) [52] are the most popular approach to non-neural network supervised learning. Similar to decision trees, the idea for SVM is to find a split in the data. The split concept is known as a maximum

margin separator, a boundary that is equally spaced between all examples, as a result generalising the possible classification.

There have been applications developed using supervised learning in mobile networks. In [53] an SVM approach is used to determine the direction of arrival for an antenna array. Using supervised learning with training examples achieved by implementing MUSIC [54], the system is capable of detecting angles of arrival with high accuracy, although suboptimal compared to MUSIC. Similarly, SVM is used in [55], by using user location information as a feature vector and the corresponding beam as a fitting label. Trained by using a two-step beam sweeping method to generate examples, the algorithm finds a way to separate the different user locations to allocate a beam with a high degree of accuracy, but does not reach the optimum performance of the parent algorithm.

The main drawback of this type of learning is that the data has to be clearly labelled from the beginning and the data sets required to teach such algorithms are very large. There are options of using online-offline learning, using which new data, including a label, gets generated during run-time. Solutions from such algorithms are only beneficial for learning from the most accurate algorithms when trying to reduce execution time. Supervised learning is commonly only as good as the data it learns from.

Unsupervised Learning

Unsupervised learning is a type of learning that does not need clearly labelled examples, but relies on expert knowledge to draw conclusions on the outputs. By providing the features of whatever data is present, unsupervised learning can create clusters of similar data. Alternatively it can create associations between data, e.g. for one chosen item, find a fitting match. Unsupervised learning is useful for finding patterns in data that is not initially clear. The approaches commonly used for unsupervised learning are as follows:

- K-means clustering [56] is a method that collects samples together around other similar samples. The letter K is representative of how many centroids, points that are statistical averages of the cluster, there are around which the samples should be clustered. The aim of the algorithm is to find where those centroids would be for the provided dataset.

- Isolation forest [57] is a method for detecting anomalies in data. This algorithm analyses the dataset by partitioning it, aiming to isolate any points on the graph. The less partitions it takes to isolate any given point the more likely it is to be an anomaly. This algorithm provides additional information to people monitoring data by means of giving a score to a given input, a high score means data is more likely to be an anomaly.

Unsupervised learning has been used as a method for power control and allocation in C-RAN [58]. In this paper, without knowing a lot about the environment, the neural network could group channel statistics to output desired power levels to combat slow fading. The benefit of the solution is that it does not base its performance on a parent algorithm, the way that supervised learning does. In [59] anomaly detection in LTE is presented. In the work, authors present a process for processing the data, extracting KPIs of the network and choosing the reference point from which to judge anomalies. The framework uses K-means to model the dataset and a three sigma rule to judge a data point's likelihood of being an anomaly. The framework aids in detecting anomalies and provides a visual output of when an anomaly spike is likely.

The significant drawback of this method is that the neural network only outputs which grouping a data sample likely belongs to or how likely it is to be an anomaly. It is up to the expert generating the solution to decide what to do with this grouping output. Additionally, if there are new groups introduced, the training process has to be restarted from the beginning.

2.2.1 Reinforcement Learning

Reinforcement learning (RL) is an area of machine learning that deals with how agents should act in an environment to maximize a reward [19]. RL agents learn from their own experience and feedback, without explicit supervision or labels. It is also unlike unsupervised learning, since the goal of unsupervised learning is to find structure in data, while RL tries to find fitting actions to maximise a reward. This is commonly done by solving Bellman optimality equations, first introduced by Richard Bellman. Once these non-linear equations are solved, the agent can act optimally in the environment. This applies to any environment that the agent can be deployed in.

Reinforcement learning is composed of these main elements:

- Policy, which is how an agent will act. Typically, this is an agent's reaction to an input. In practice it can range between a table, an approximation function, a search process. In more novel adaptations the policy can be represented as a neural network. The policy is known to be the core of reinforcement learning. Policies can be stochastic or deterministic, in case of stochastic policies, they show the probability of agent taking a specific action, while in deterministic policies, there is only one action chosen for a given state.

- Reward is what gives the method its name. It is how the agent can understand the environment and its actions through reinforcement. A reward signal is produced when an agent makes a decision in a state. Since the primary goal of the reinforcement learning agent is to maximize the reward, it directly impacts how the policy develops over time. When an agent makes a decision that provides a low reward (negative reinforcement), it will be more likely to choose an action that will give a higher reward (positive reinforcement) in a similar state.

- Value is the last significant component of reinforcement learning. Generally speaking, the value is taking into account the sum of all the possible rewards from a given state. This allows the agent to have some foresight when solving a problem. After replaying over the environment many times, the agent can predict how the current decision will impact all future decisions. In a concrete example, sometimes a greedy decision may yield a high reward, but the following best rewards will be low, but a worse decision at the same state could lead to high rewards in every following state. By optimising for the value, the agent would choose the worse decision with good following rewards.

- In the case of model-based learning, there is an additional component – the model. A model is a representation of the environment that the RL agent is deployed in. Using a model the agent can predict how the environment will behave, which is beneficial for planning and making complex decisions. The model can be known a priori, as a static probability distribution of state transitions or it can be learned, by adjusting the probability distribution as the environment plays out.

One of the most important concepts for RL is a Markov decision process (MDP). It is a mathematical formulation of solving a problem by learning about it. In RL, MDPs are composed of an agent and an environment, as shown in Fig.

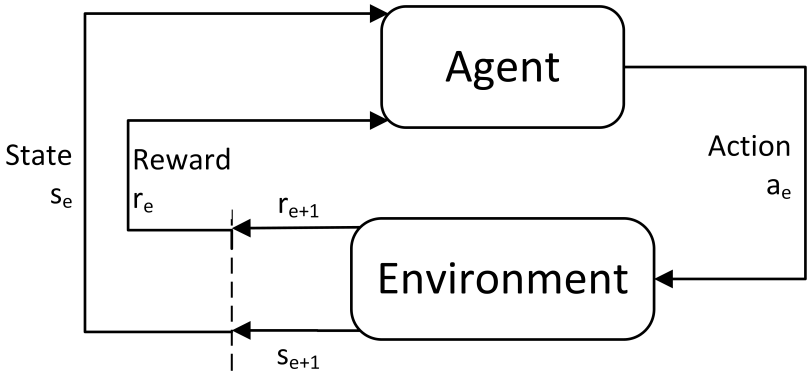


FIGURE 2.3: A Markov decision process showing interactions between the agent and a state

2.3. The definitions are straightforward, the agent is whatever entity is acting and making decisions. An environment is an entity from which the agent can get observations from and interact with. The entities interact with each other in a loop: the environment presents observations to which the agent reacts and makes actions, in return the environment presents new observations and the outcome of the previous action as a reward.

RL agents can adapt to complex and dynamic situations, and solve problems that require long-term planning and exploration. If a problem is being solved with planning, it is called model-based learning. In this method algorithms use a model of the environment, which can be either formulated a priori or learned by the agent, to plan ahead and improve the policy. Examples are dynamic programming, Monte Carlo tree search, and model predictive control. The opposite of this is model-free learning, such algorithms do not use a model of the environment, but learn directly from experience. This can be called true trial and error learning. Common model-free algorithms are Q-learning, SARSA, and policy gradient methods.

Model-free learning

Model-free RL is an important branch of reinforcement learning. While many applications can be solved using exact tabular methods with finite, small-to-medium sized search spaces, a lot of applications cannot. When the state can't be defined by discrete variables, it becomes continuous. This makes the search space grow indefinitely large. For such state spaces it is impossible to find an optimal solution. Considering that there is a very large amount of possible inputs, the issue is not the physical resources required to store them, rather it is understanding how to map them correctly; every time a state transition occurs, it is likely that the agent has not visited the new state before.

Generalisation is what helps RL make sense of a continuous environment. While states that the agent visits have never been seen before, some previously

visited states are similar enough to approximate a solution. And from this, a set of model-free RL methods are called approximate solution methods. This means that it is possible to take examples of the value function from previous results and apply them to create a solution which approximates the complete value function.

Algorithms that are capable of such learning split into on-policy and off-policy, also known as action-value methods. Their use of policy for learning is an important distinction. On-policy algorithms learn directly from actions being made by the current policy, the agent makes decisions assuming the current policy will continue to be followed. Off-policy algorithms instead focus on what the value of the next state will be if a greedy policy will be followed, hence the current policy does not determine the future outcomes.

Off-policy algorithms are currently the most popular and were one of the first ones to be introduced for model-free learning. Q-learning [49] is the most famous off-policy algorithm and is researched in the most depth. Many modifications are readily available that improve the variance in the observed value of the approximations as well as improving the convergence of the algorithm. The algorithm itself is based on learning the Q-value. From the earlier definition of value, Q-value is the expected reward of taking a value while being in a certain state. This is evaluated by predicting the reward of taking an action and adding it to the possible upcoming rewards if a greedy action is taken. Since the greedy action is not considered to be a part of the policy, this algorithm creates the name off-policy learning.

SARSA is one of the most famous modifications to Q-learning. Proposed in [60], the algorithm is an improvement on classical Q-learning. SARSA aims to improve the convergence of Q-learning by focusing on the possible rewards if the current policy keeps being followed. In [60] it is argued that traditional Q-learning assumes wrong rewards from the very beginning, as there is no knowledge of states available and as learning continues, it overestimates the possible rewards. To solve this, a probability of taking all actions for the current state is

necessary, this reduces the over-estimation of the value of taking an action in a state, which in return stabilizes learning. Since the probability of taking all actions is based on the current policy this algorithm is known as on-policy.

In addition to action-value methods, policy gradient methods are also used. Particularly useful for continuous states and actions, policy gradient methods are usually used in control tasks. These algorithms learn the policy parameter, that is the optimal probability distribution of taking a certain action for a provided state. A value function can still be used to learn the policy, but it is not used to select the value. Actor-Critic algorithm [61] is one of the best examples of this. In this algorithm, an actor implements a stochastic policy, using which it chooses actions with some probability, determined by the policy. The critic, which is a model-based reinforcement learning algorithm, determines how good the actions taken by the actor are. The actor uses the responses from the critic to improve its policy. Since the critic provides a gradient response which is used to update the policy, actor-critic is known as a policy-gradient method.

Open problems of RL

While reinforcement learning solves a lot of challenges that can come with optimisation problems, there are also new challenges to overcome, which introduce new research on a consistent basis. Some of the challenges and open problems in RL are:

- Exploration-exploitation trade-off [62]: RL agents can be deployed in environments that are expected to change and evolve. A challenge that has to be solved is how to balance between trying new actions to discover better ones, and exploiting the current knowledge to maximize the reward.

- Partial observability [63]: due to environment limitations a full state might not be available to the RL agent. Fully observable MDP assumptions do not fully apply to such scenarios. In addition to this, there is new uncertainty

that agents have to deal with. Methods need to be developed to deal with situations where the agent cannot access the full state of the environment, but only some observations or features.

- Multi-agent systems [64]: studying mobile networks in particular, many environments that researchers are faced with are multi-agent. In such systems, multiple RL agents are deployed in the same environment and share or compete for the same resources. In such environments the additional complexity requires modifications to existing algorithms to ensure that agents are coordinated and can benefit the system.

2.3 DRL Applications in Mobile Networks

Deep reinforcement learning (DRL) is a combination of reinforcement learning and deep learning that can solve complex control and decision-making problems in dynamic and uncertain environments. DRL has been applied to various domains such as computer games, robotics, natural language processing, and computer vision. The algorithms that are used to represent deep reinforcement learning are Double Deep Q-Network (DDQN) [22] and Advantage Actor Critic (A2C) [20].

DDQN is a deep neural network approach to Q-learning, useful for discrete or continuous inputs, but only discrete actions, such as discrete control tasks or resource allocation. It is termed as double because it uses an estimation network and a target network [22], both of which are used to learn about the environment. It uses two networks with the aim of combating overestimation that is common for Q-learning. Rather than estimating the possible value by only using the estimation network, the greedy policy is still estimated by the estimation network, while the target network is estimating the value of this policy. While estimation

network is the one that learns consistently with each interaction, the target network must be updated periodically. Target network update frequency is an open challenge and requires careful optimisation for the best results.

A2C is an algorithm that is best used in environments where both the inputs and outputs are continuous, such as continuous manipulation and locomotion [20]. A2C is similar to DDQN in a sense that multiple neural networks are used, one for estimating the policy of the network and one to estimate the value of this policy for a given state. However, instead of replaying experiences, multiple actor neural networks are deployed on multiple separate instances of the environment, from which they all learn together, as they experience different states at the same time. The algorithm also shows a lot of practical benefits, as both the actors and the critic can be relatively small compared to networks used in DDQN, which improves the execution speed and does not use as many resources.

From the previous, DRL has several benefits over conventional RL [65], such as:

- DRL can handle high-dimensional and complex state and action spaces, which are common in real-world problems such as robotics, games, and natural language processing.
- DRL can learn from raw sensory inputs, such as images or sounds, without requiring handcrafted features or domain knowledge.
- DRL can leverage transfer learning and meta-learning to generalize across tasks and domains, and to adapt to new environments and situations.
- DRL can incorporate prior knowledge and constraints into the learning process, such as human demonstrations, expert feedback, or safety rules.

Further in this section, the focus is on the applications of DRL in mobile networks, especially for 5G and beyond. Mobile networks are facing many challenges such as increasing data traffic, diverse service requirements, heterogeneous network architectures, and limited spectrum resources [39]. To cope with these challenges, mobile networks need to be more intelligent, adaptive, and autonomous. DRL can provide a promising solution for mobile networks by enabling network entities (such as base stations, users, devices, etc.) to learn from their interactions with the environment and optimize their actions accordingly [61]. The following subsections outline the on-going research in usage of DRL applications in mobile networks.

2.3.1 Dynamic network access

DRL can help network entities to select the best access point or frequency band based on their current state and network conditions [66], [67]. For example, DRL can be used to implement cognitive radio systems that can dynamically access the spectrum without interfering with licensed users.

2.3.2 Resource control

DRL can help network entities to adjust their transmission rate or power based on their channel quality and network congestion [68], [69]. For example, in [68] DRL can be used to predict network conditions and adjust the power of transmitters proactively.

2.3.3 Wireless caching

DRL can help network entities to decide what content to cache locally or prefetch from the network based on their popularity and demand [65]. For example, DRL can be used to implement proactive caching strategies that can reduce the latency and bandwidth consumption of content delivery.

2.3.4 Data offloading

DRL can help network entities to decide when and how to offload their data to other devices or networks based on their energy consumption and cost. As an example in [70], DRL is used to implement mobile edge computing that can enhance the performance and efficiency of mobile applications, by taking into account the latency and computing cost of offloading computations. The algorithm that is proposed can effectively adjust the IRS parameters to handle changing channel conditions as well as choose the time allocation for tasks.

2.4 Summary

As applications are becoming more demanding, operators and vendors have to make sure the network infrastructure evolves as well. The 5G evolution aims to implement cloud RAN to support the current network requirements and some research suggests a movement away from cell-based architecture. While cloud RAN can still support cell-free communication, it imposes heavy loads on the fronthaul as both user and control plane data traverses to the cloud servers for processing. Fog RAN implementation is shown to have features necessary to deal with cacheable content and content that requires close to real-time latency.

Fog RAN tackles a lot of the issues of Cloud RAN. By introducing a hybrid distributed and centralised architecture fog RAN alleviates the heavy loads on fronthaul links, which results in higher data rates, due to accurate channel estimation. Additionally, edge caching allows for reduced latency. Importantly, cloud layer is still present in fog RAN, so the evolved architecture is flexible for various traffic requirements.

As fog RAN becomes desirable, it introduces challenges that have to be tackled. Resource optimization, such as cooperative radio resource allocation and cache placement are key challenges in fog RAN deployment. Deep learning has

been proven to solve unique challenges in many different industries and is an attractive choice for mobile networks as well. Supervised, unsupervised and reinforcement learning have advantages and disadvantages for implementation, but reinforcement learning is preferred as it can handle the random environments that mobile networks find themselves in.

Deep reinforcement learning introduces added benefits of being able to deal with high complexity of data, and is more efficient at generalising solutions that conventional reinforcement learning fails to do. In addition to the suggested applications, DRL can also be used for other purposes such as traffic routing, resource sharing, and data collection in mobile networks. However, applying DRL in mobile networks also faces many challenges such as partial observability, non-stationarity, multi-agent coordination, and scalability. Therefore, more research is needed to overcome these challenges and explore the full potential of DRL in mobile networks.

Chapter 3

Modelling a latency constrained RAN and Xhaul

3.1 Introduction

Coordinated multi-point (CoMP) beamforming is a technology that is capable of improving the signal quality and datarate experienced by users [71]. To take advantage of this coordination, remote units (RUs) have to collect information about the channel between them and users. This information is fed back via fronthaul to distributed units (DUs), which are responsible for coordinating the signaling between multiple RUs. The DUs are interconnected with each other and with the central unit (CU) by a midhaul that allows them to exchange information relevant for coordination. CU is the core of this network architecture. After collecting information from the DUs about channel conditions it is responsible for processing the data and calculating the beamforming weights to coordinate the RU transmissions with reduced interference. Fronthaul and midhaul compose some of the transport network, which is sometimes abstracted and referred to as the Xhaul [72]. Xhaul abstractions are beneficial when considering cell-free MIMO [39] and CoMP, as it allows modelling the variation in bit rate caused by beamforming method, user mobility, user allocation and varying path selection through the Xhaul. It has been shown in [71] and [73] that Xhaul delays influence

achievable beamforming gains and user bit rate.

FlexE is a transport technology that could replace common public radio interface in the Xhaul [74], but it is sensitive to the over and under-allocation of slots in the FlexE calendar. This can affect serialization and buffering delays and the total latency of the Xhaul. Xhaul latency affects the accuracy of channel state information that is sent to the DU/CU where beamforming vectors are computed if CoMP is employed [73], [75]. Therefore, [67] proposes that to maximize the achievable user bit rate, a resource allocation scheme that can anticipate the traffic in advance and appropriately allocate resources is required. Further, some coordination of control between RAN and Xhaul is desirable. Achievable user bit rates through the selection of low-latency paths over the Xhaul can be maximised. For this, [67] suggests that a mechanism to offload users from RUs with higher Xhaul latency to those with lower latency will contribute to the maximisation of the achievable user bit rate.

Optimal route finding in Xhaul has been addressed in [76] and Xhaul latency effects have been studied in [77], [78] as a problem in optimising operational expenditure (OPEX). In [77] and [78] only RU queuing delay is considered for modelling latency. Deep reinforcement learning has been shown to be useful for optimal selection of functional splits and routes in fronthaul [24] and for utilising fronthaul with software-defined networking control in [25], with latency constraints. DRL has also been used for resource allocation in an ultra-reliable low latency (uRLLC) network [79] and in Fog RAN networks [69], [65] under the constraints of latency. According to [67]: "The aforementioned studies omitted any mechanism for handling the allocation of users to RUs under the constraints of latency of their paths in the Xhaul, as a means of maximization of their achievable bit rate. Also, the models for latency in these studies were based either on propagation delay, serialization delay, mean queuing delay or a priori end-to-end delay, and thus did not consider the combined effect of all possible sources of delays in their models." In addition, FlexE has not been considered in any of the

aforementioned studies as a transport network technology, which could enable efficient bandwidth utilisation, or studies its influence on latency constraints.

This chapter models looks at how Deep-Q networks (DQN) are used for efficient bandwidth allocation on each FlexE aggregation node to predict input network traffic and allocate bandwidth. DQN aims to minimize over-allocations, which increase network power consumption and OPEX, as well as under-allocations, which would increase Xhaul latency and reduce the achievable user bit rate. The CU has an additional DQN, whose goal is achievable user bit rate maximization by offloading users from RUs with higher Xhaul latency to those with lower latency. The use of multiple DRL agents (in the Xhaul and RAN) results in a multi-agent-based optimisation system for RAN [80].

3.2 Network Model

The network architecture that implements CoMP is shown in Figure 3.1. This network architecture connects many RUs to a single DU via fronthaul and the DUs are connected to the CU using midhaul. It is assumed that the CU and DU are co-located, which eliminates the need to model the midhaul, and instead analysis can be focused on the fronthaul part of the Xhaul, as shown in Figure 3.2. In this network users are connected to RU groups, which allows the RUs to coordinate with each other and perform CoMP beamforming [71], [73]. The layout of the fronthaul transport network composed as described in [67] using multiple layers of aggregation nodes. An ethernet aggregation node is placed close to the edge of the network, aggregating the traffic of all RUs in one group. The traffic is transported via FlexE aggregation nodes to the DU, this traffic can vary depending on the RAN protocol overheads depending on functional split [81] and additional background traffic. Each FlexE aggregation node allocates slots in their calendars to meet traffic requirements and reduce latency in the network.

Since RUs are transmitting user traffic using Ethernet interfaces, the RUs and Ethernet aggregation nodes in Xhaul will introduce queueing and serialisation delays. [67] assumes that the Ethernet aggregation nodes operate with a FIFO scheduling scheme in the queues. Buffers can be used to hold packets while there is congestion at the output of Ethernet and FlexE aggregation nodes. Existence of such buffers introduces delay variations in the traffic. If the FlexE aggregation nodes allocate less rate than required in the calendars, the buffers would hold on to the excess packets [82]. DU/CU will also be equipped with buffers to absorb the delay variations. For this chapter, as in [67], the size of these buffers will be determined by the percentiles of delays that are absorbed by the buffers. Large buffer sizes will ensure a higher percentile delay variation absorption but

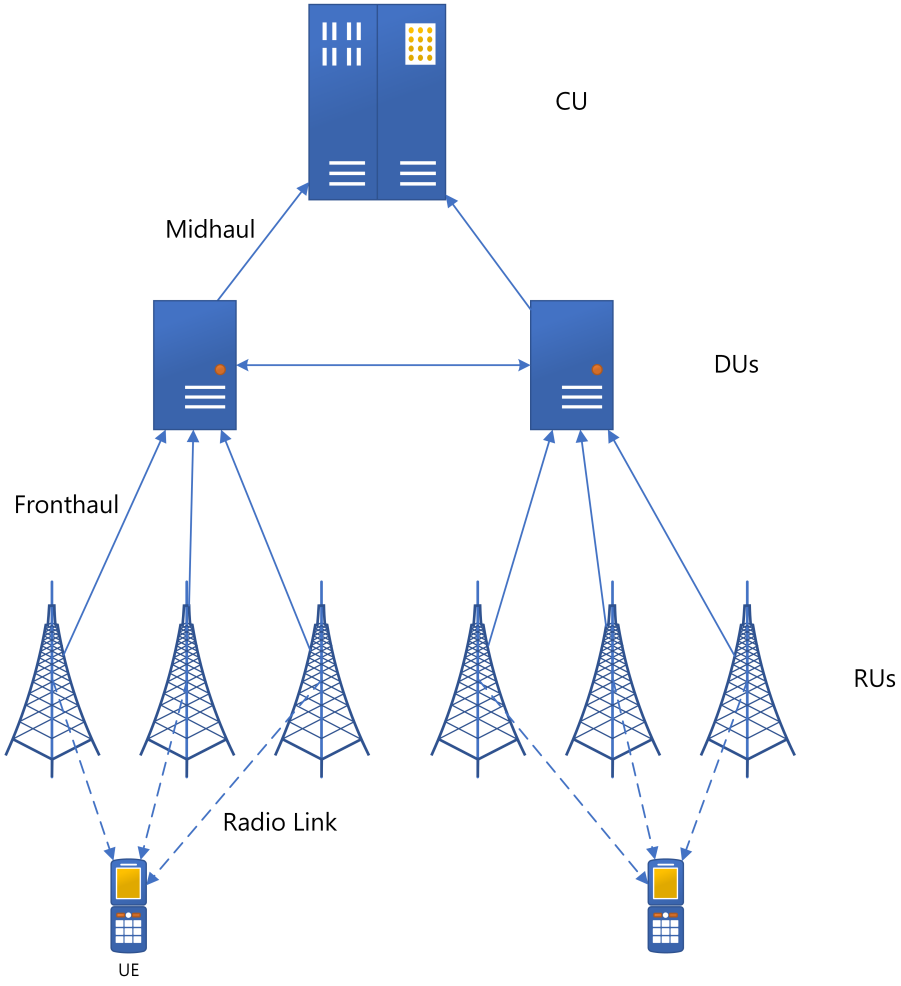


FIGURE 3.1: C-RAN architecture with CoMP.

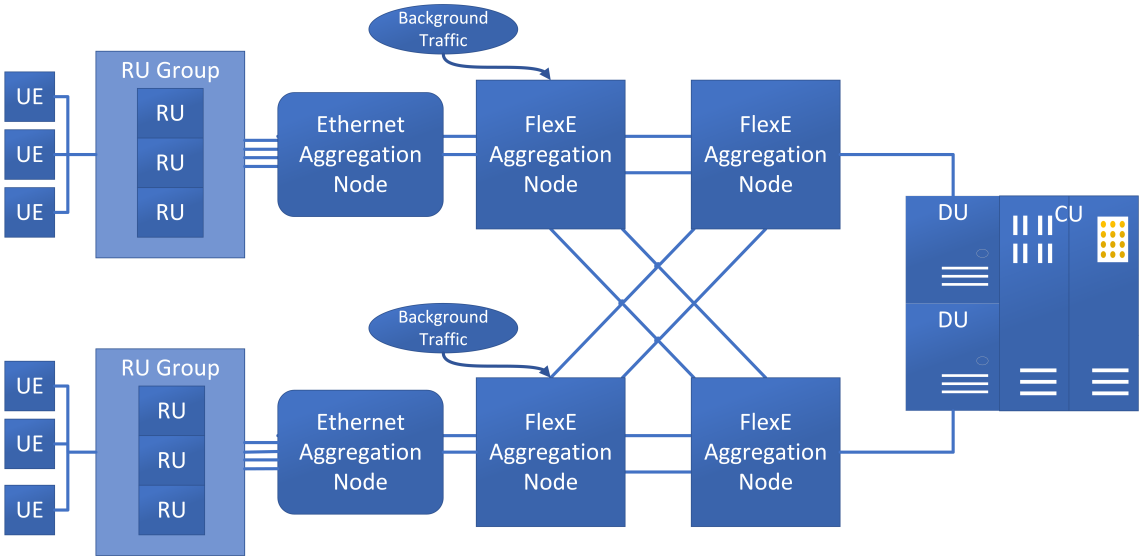


FIGURE 3.2: FlexE Xhaul network architecture.

increased buffering delay [83].

Each FlexE aggregation node manipulates a separate calendar to control traffic flows in different directions. Inbound traffic to a FlexE agent includes the traffic from a preceding Ethernet or a FlexE aggregation node combined with background traffic. The reward function takes into account the inbound traffic and the allocated slots in the FlexE calendar. The state and reward function make up the FlexE environment as shown in Figure 3.3. The observations and rewards are provided every decision period which corresponds to the delay for the reconfiguration of the FlexE calendar. The slot allocations for the incoming traffic are calculated using Deep-Q learning from past actions and rewards in an experience buffer. For this MDP, [67] suggests using Deep-Q learning using the rationale that both input traffic from RAN and other sources in the network has large variations, which would be hard to explore using a tabular RL method.

It is proposed to use a separate agent within the DU/CU pool in Figure 3.2, termed the "controller agent", as shown in Figure 3.4. The controller agent is responsible for offloading users between RU groups to improve the total achievable user bit rate. As shown in Figure 3.4, the MDP that the controller agent is deployed in provides the total user sum-rate and its variations as the state and

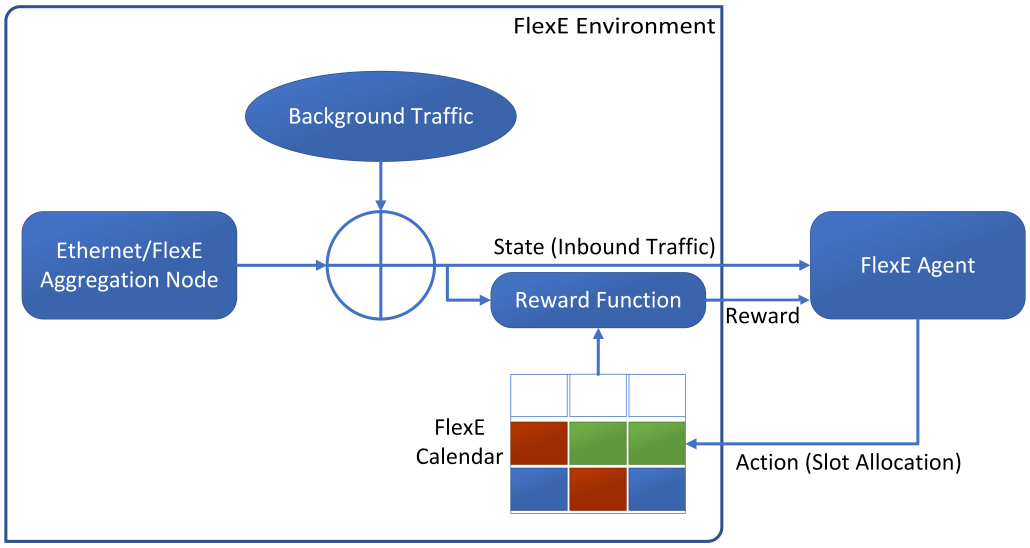


FIGURE 3.3: Markov Decision Process for a FlexE agent. Background traffic is introduced as a simulation parameter.

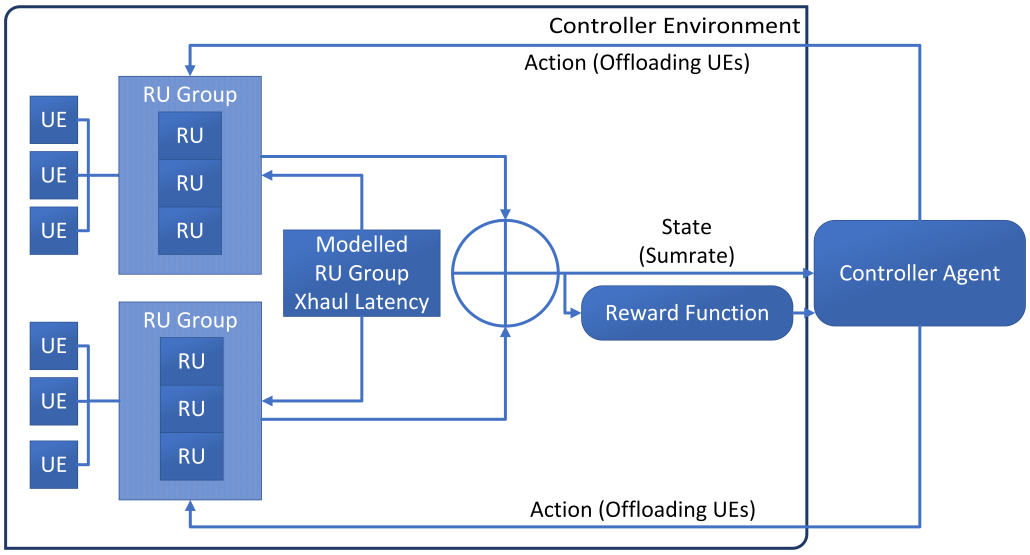


FIGURE 3.4: Markov Decision Process for a Controller Agent. Latency is dependent on the calendar slots allocated by FlexE agents.

the reward is dependent on the sum-rate achieved. The controller agent will perform actions of notifying each RU group of how many users to offload to the other group. Since the sum-rate is sensitive to the delay in the transport network, due to the age of CSI [84], offloading users from the heavily loaded RU group to the lower load one can benefit the network. The controller agent is trained using Deep-Q learning as well due to the large observation space of the RU sum-rate and its variations.

3.3 CoMP and Latency Model

For modelling the achievable user bit rate allocated to RUs incorporating CoMP, the signal-to-interference noise (SINR) γ_k of a user is modelled as follows,

$$\gamma_k = \frac{\sum_j \left| h_{j,k}^{\mathrm{H}} f_{j,k} \right|^2}{\sum_j \sum_{i \neq k} \left| h_{j,k}^{\mathrm{H}} f_{j,i} \right|^2 + \sigma^2} \tag{3.1}$$

where σ^2 is the noise power, $h_{j,k}^{\mathrm{H}}$ is the known channel information for user k from RU j, delivered to the distributed/central unit (DU/CU) for processing, and $f_{j,k}$ is the resulting beamforming vector given as follows,

$$f_{j,k} = (h_{j,k}^{\mathrm{H}} h_{j,k})^{-1} h_{j,k}. \tag{3.2}$$

In Equation (3.1), each user is equipped with a single antenna and Zero-Forcing (ZF) beamforming is used to reduce interference between beams generated by the RUs [69]. As in [84], channel estimation is modelled with delay effects, that is, a perfect channel is not known but is instead estimated and denoted as $\hat{h}(t)$ with error $\eta(t)$, which satisfies a complex normal distribution $\mathcal{CN}\left(0, \sigma_\eta^2\right)$, with zero mean and variance σ_η^2 given as follows,

$$\sigma_{\eta}^{2} = \frac{1}{\text{SNR}d} + 2\left[\pi f_D T_s \left(\frac{2L-d-1}{2}\right)^2\right]. \tag{3.3}$$

Estimation error has the same dimension as $h(t)$; if SNR during channel estimation is decreased, channel accuracy is impacted negatively, which can be mitigated by an increased pilot training length d; this, in turn, affects the total length of payload L. Other significant terms are the Doppler frequency f_D which depends on the parameters of user mobility, and sampling time T_s of the receiver [85]. The complete channel estimate $h(t)$ is given as follows,

$$h(t) = \hat{h}(t) + \eta(t), \tag{3.4}$$

Due to fronthaul and computational delay D, the channel estimate in Equation (3.4) is altered, given as follows,

$$\hat{h}(t) = \rho\hat{h}(t-D) + \epsilon(t) \tag{3.5}$$

where $\rho = J_0(2\pi f_D D)$ is the channel correlation coefficient, portrayed as a Bessel function of order zero [84], affected by user mobility, and $\epsilon(t)$ is the $\mathcal{CN}(0, \sigma_{\epsilon}^2)$ error vector, with variance σ_{ϵ}^2 given as follows,

$$\sigma_{\epsilon}^2 = \sigma_{\hat{h}}^2\left(1 - |\rho|^2\right). \tag{3.6}$$

Then the channel model in Equation (3.5) is substituted into Equation (3.1) for the estimation of SINR. The estimated SINR is then used to estimate the resulting achievable bit rate of a user RZF in the uplink via the Shannon capacity model

[86], is given as follows,

$$R_{\mathrm{ZF}} = \sum_{k=1}^{K} \log_2 \left(1 + \frac{\left| \left(\sqrt{\rho}\hat{h}_k^T + \sqrt{\sigma_{\epsilon,k}^2 + \sigma_{\eta,k}^2} \right) f_k \right|^2}{\sum_{i \neq k} \left| \left(\sqrt{\rho}\hat{h}_k^T + \sqrt{\sigma_{\epsilon,k}^2 + \sigma_{\eta,k}^2} \right) f_i \right|^2 + \sigma_n^2} \right). \tag{3.7}$$

From the transport network perspective, [67] notes that "the traffic is first received at RUs, which add encapsulation overheads dependent on the split option being used. Then further overheads are added by the Ethernet and FlexE-based aggregation nodes in Xhaul." This chapter assumes an intra-PHY functional split option 7.2 [87] that results in upscaling or an increase in the recorded sum-rate. The functional split upscaling factor F_{split} is calculated using the relation given as [88],

$$F_{split} = \frac{2N_{ant}N_{bit}\frac{N_{sc}}{T_{sym}}}{N_{ant}\log_2(M)B} \tag{3.8}$$

where N_{ant} is the number of antenna elements, N_{bit} is the sampling width (no of bits per sample), N_{sc} is the number of data subcarriers comprising the OFDM symbol, M is the modulation order, B is the bandwidth of the signal transmitted by a UE and T_{sym} is the OFDM symbol length in time. The calculations in Equation (3.8) do not account for the encapsulation overheads.

The achievable bit rate of the users is reduced due to the latency in Xhaul discussed in previous subsection and indicated by the factor of D in Equation (3.5). In [67] percentile delay is selected as a benchmark to model the latency effects of the system. Justification for this is to measure the worst-case scenario of latency. Also, as mentioned in the previous subsection, the destination nodes in Xhaul will be equipped with buffers to absorb the percentile delays. The 99.9% percentile delay is substituted as the delay term D in Equation (3.5) to the wireless channel model. For the estimation of percentile delays of a path in the Xhaul, the

mean queuing delay of packets $T_{q,x}$, in RU's Ethernet interface and Ethernet aggregation nodes are estimated using the G/G/1 queueing model given as follows [89],

$$T_{q,x} \approx \frac{\lambda_t}{\mu_r} \frac{\rho}{(1-\rho_{est})} \frac{C_a^2 + C_r^2}{2} \tag{3.9}$$

where x denotes the node in the Xhaul, ρ_{est} is the estimated load on aggregation nodes, λ_t is the arrival rate of packets μ_r is the packet service rate, C_r is the coefficient of variation of the service time, and C_a is the coefficient of variation of the interarrival time. The mean of the delay $\bar{T}$ of a path in the Xhaul in Figure 3.2 is given in [67] as,

$$\bar{T} = T_{s,RU} + T_{s,E} + 2T_{s,F} + T_{q,RU} + T_{q,E} \tag{3.10}$$

where $T_{s,RU}$, $T_{s,E}$ and $T_{s,F}$ are the fixed serialization delays in the RU's Ethernet interface, Ethernet aggregation node and two FlexE aggregation nodes in a path in Xhaul. $T_{q,RU}$ and $T_{q,E}$ are the waiting times in the RU's Ethernet interface and Ethernet aggregation node of the path in Xhaul. The distribution of the total delay of a path is then modelled by a log-normal distribution using the total mean delay in Equation (3.10). This distribution is selected after analysis of packet traces via simulations in NetSim®. From the log-normal distribution, the 99.9 percentile delay is extracted. The fabrication delays are ignored in the analysis.

3.4 Approach to Simulation

Simulink MATLAB® is used to model wireless communications based on CoMP as described in Section 3.3. The parameters for the wireless system are listed in Table 3.1. The RAN model is initialized by assuming 15 users in each RU group

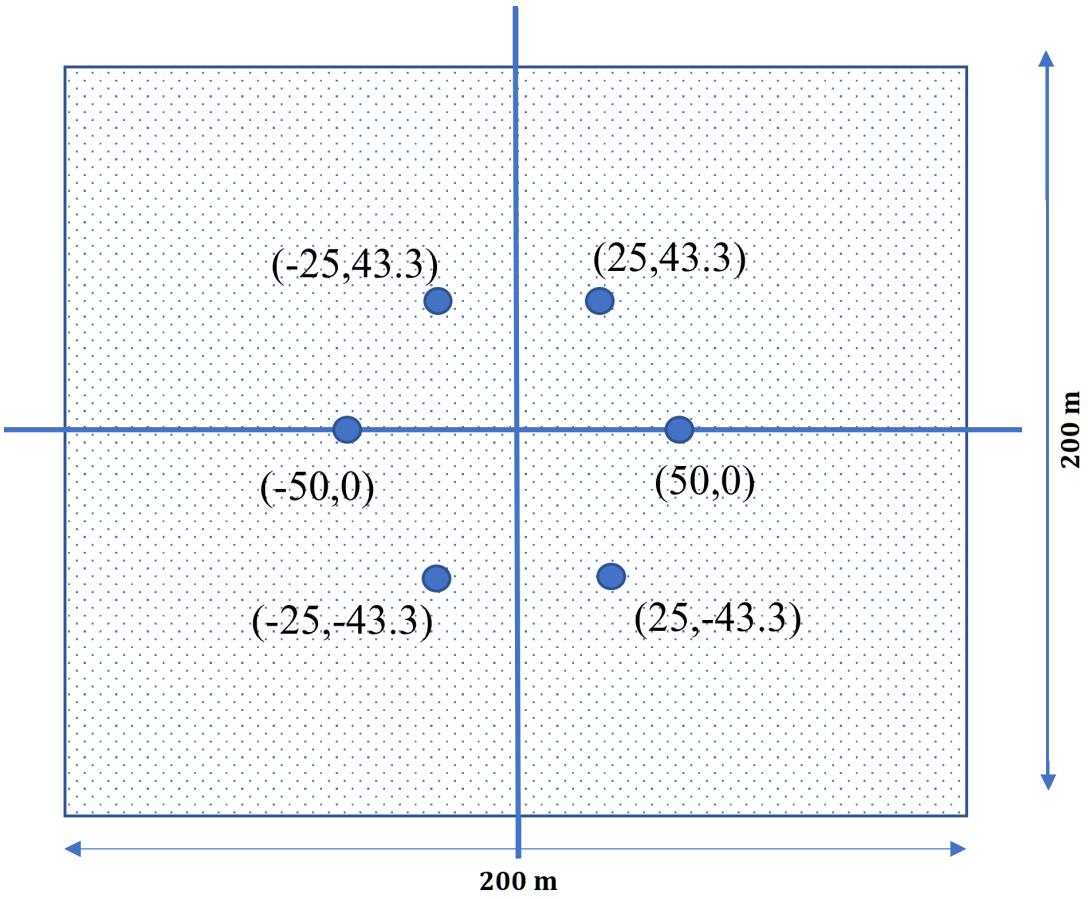


FIGURE 3.5: RU coordinates in x,y plane (Case-1) [67].

(totalling 30 users in the two groups). Each RU is modelled to transport 16 independent beams' data in Xhaul. The locations of users follow a uniform random distribution over a 200x200m grid, shown in Figure 3.5. RUs are deployed in a hexagonal formation with an estimated radius of 50m, this ensures that RUs are equidistant from each other.

Initially, to ensure even latency impacts on all paths, background traffic of 110 Gbps is added into the paths from both RU groups to DU/CU. It is shown in [67] that the resulting total achievable bit rate for both RU groups is 22.3 Gbps. This data rate is upscaled by a factor of 3.5 using Equation (3.8) and parameters from Table 3.2. Further upscaling by a factor of 3 is carried out to accommodate the fact that there are three RUs in an RU group. This upscaling resulted in a load of 234.089 Gbps in the Xhaul. The underlying background traffic pattern is

TABLE 3.1: Simulation Parameters

Number of users per RU group	15
Avg bit rate of RU groups	117.04 Gbps
Functional split factor	10.5
Avg background traffic	110.7 Gbps
Number of RUs Groups	2
Number of RUs per Group	3
SNR per User	13 dB
Antennas per RU	16
Power Constraint per RU	40 W
Bandwidth per user	50 MHz
User Mobilities	6 m/s
Rate of Arrival	10^{-6}
Rate of Departure	4×10^{-6}

TABLE 3.2: Functional Split Upscaling Parameters

Number of antenna elements N_{ant}	15
Number of bits per symbol N_{bit}	8
Number of subcarriers N_{sc} in 50 MHz	250x12
Modulation level (M)	16
Interval of the OFDM symbol T_s	$66.7\mu s$

extracted from the WIDE project packet traces repository [90] specifically for the date of 08/04/2020. The average bit rate of the traffic is 6 Gbps which is presumed less than the expected aggregate background and user traffic in future 5G Xhaul networks. Therefore, in [67] the traffic patterns were increased by multiplication with discrete upscaling factors (5-11) to emulate a high load on the Xhaul aggregation nodes.

The hyperparameters for the Deep-Q network, given in Table 3.3. Discount factor, number of steps to look ahead, mini-batch size, experience buffer length, number of hidden layers and neurons per layer are tuned to provide a good learning performance. Specifically, the aim was to ensure a monotonic increase in average rewards with each training episode and converge towards the end of the training process. The experience buffer size ensured that the agent did not suffer from catastrophic forgetting which may lead agents to take steps that earn negative rewards after earning positive rewards [91]. On the other hand, the discount

TABLE 3.3: Deep-Q Network Parameters

Learning Rate	0.01
Gradient Threshold	1
Number of Hidden Layers	6
Number of Neurons per Layer	98
Sample Time	4×10^{-3}
Experience Buffer length	40×6000
Mini batch size	2048
Epsilon	1
Epsilon Decay factor	0.005
Number of steps to look ahead	
FlexE agent	8
Controller agent	32
Discount factor	
FlexE agent	0.7
Controller agent	0.99

factor for the controller agent is 0.99, but is upper-bound by the number of steps to look ahead, which is set to be 32, preventing large future award approximation functions. The mini-batch size parameter for both the FlexE agent and controller agent is set to 2048 as a compromise between simulation speed and accuracy of convergence of the deep-Q network.

For the FlexE agent, it is assumed that the action space for the FlexE agent is discrete and limited to 80 slots with a granularity of 5 Gbps, so $a(t) = \{0, 5, 10, \cdots, 80\}$ since it is assumed that each FlexE node will support a total of 400 Gbps output line rate. The aim of the reward function is to describe the objective of the system to the agent, so that it would achieve convergence to the desired performance. However, if it is inappropriately designed, it will converge to a local minimum, which may not result in the desired performance, or converge to a bad strategy. The reward function achieving the least standard deviation of the difference between input traffic and allocated traffic is found by [67] as follows,

TABLE 3.4: Queueing Delay Parameters

Average packet length	735 bytes
Standard deviation of packet length	300 bytes
Output links rate	400 Gbps
Standard deviation of inter-arrival time of packets	300 μs
Distance of optical links	1 km

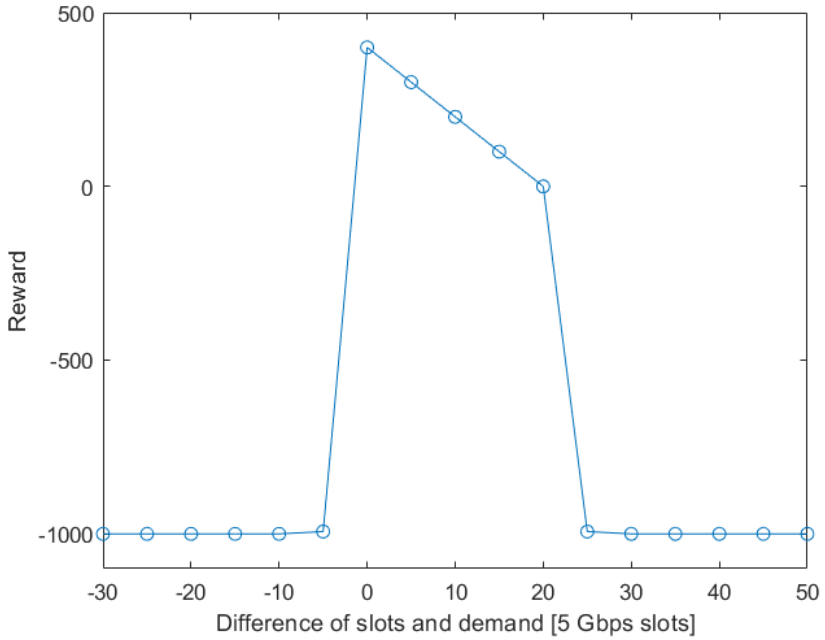


FIGURE 3.6: Reward vs difference of allocated slots and input traffic demand.

$$R_{FlexE} = \begin{cases} -20 + ((S - R_i) - 20) & \forall 0 \leq S - R_i \leq 20 \\ -1000\left(1 - e^{-(S-R_i)}\right) & \forall S - R_i > 0 \\ -100\left(1 - e^{(S-R_i)}\right) & \forall S - R_i < 0 \end{cases} \tag{3.11}$$

where S is the bit rate allocated in the FlexE calendar and R_i is the observed input traffic bit rate. The reward function for the FlexE agent is also plotted in Figure 3.6.

The FlexE agent is trained using Deep-Q learning in the reinforcement learning toolbox in MATLAB®. It is trained to predict the combined traffic patterns

from the WIDE project repository [90] (with upscaling factor of (5-11) and the user traffic patterns from each RU group. The FlexE agent is trained for 40 episodes. Each episode spanned 24 seconds with a step size of 4ms, which is known to be the reconfiguration time frame of the FlexE calendar [91]. The results that are used for Figures 3.8 are from an agent that has been training for 34 episodes.

The controller agent is a trained Deep-Q Network deployed simultaneously with the FlexE agents. The controller agent is being rewarded for maximizing the total achievable bit rate of the two RU groups. It achieves this by offloading users from a path with higher latency to a path with lower latency in the network. The actions of the controller agent are the offloading commands for each RU group, with integer values ranging from 0 to 5. In any training step, the number of users offloaded is limited to 5 users. The choice is made to reach a monotonically increasing average reward earned by the controller agent during the training. The reward function for the controller agent is given as follows,

$$R_{ctr} = 500\bar{R}_{sum}^2 - (5 \times 10^5 \bar{R}'_{sum}) \tag{3.12}$$

where $\bar{R}_{sum}$ is the averaged value over 5 samples of the total achievable bit rate for the two RU groups and R'_{sum} sum is the difference in the bit rate in consecutive timesteps of 4ms, indicating the variations in the total achievable bit rate of the RU groups. It is given as follows,

$$\bar{R}'_{sum} = \bar{R}_{sum}(t) - \bar{R}_{sum}(t - 0.004). \tag{3.13}$$

Due to the second term in Equation (3.13) involving $\bar{R}'_{sum}$ the controller agent is negatively penalized for introducing a high variation into the resulting rate. The first eight seconds of a training episode for a controller agent, which spans

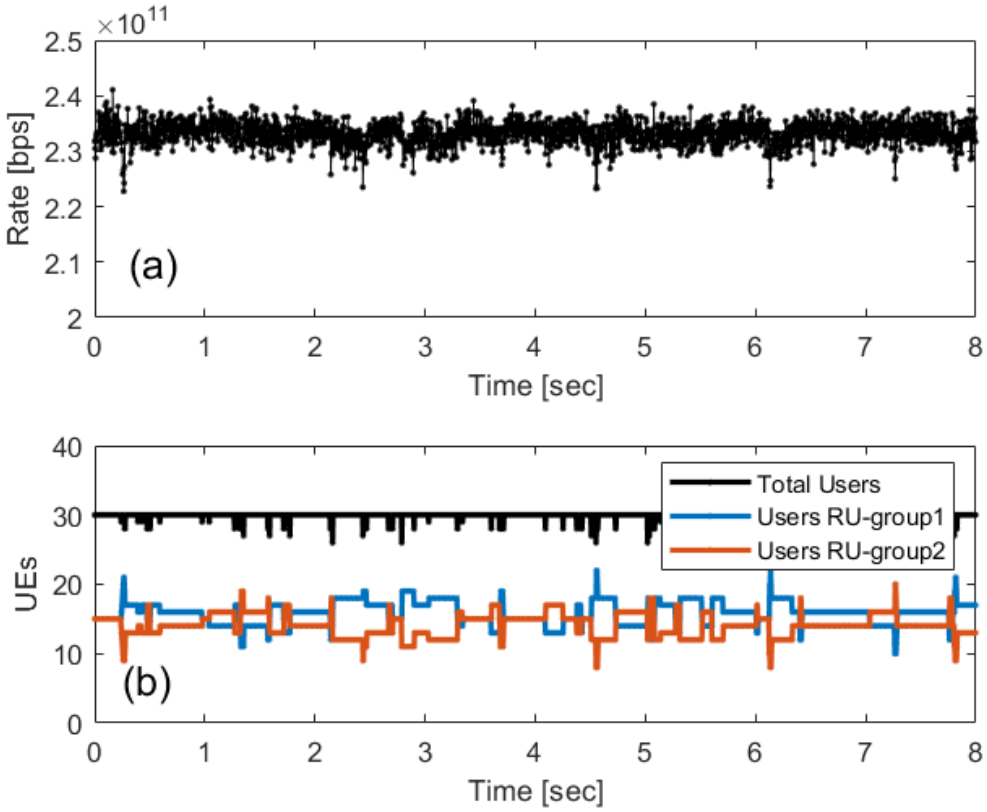


FIGURE 3.7: Training results of controller agent, total achievable bit rate (a), number of users in RU groups (b) for RU group position in Figure 3.5.

over twenty-four seconds in the simulations, is shown in Figure 3.7. The resulting total achievable bit rate of the two RU groups is shown in Figure 3.7(a). It can be seen in Figure 3.7(b), that as the controller agent is learning it constantly swaps the users from one RU group to another. It does so to learn what transitions from one state to another can look like and how it can maximize the total achievable bit rate of the two RU groups. As the agent learns that offloading users to RU-group 2 leads to a lower rate it swaps the users over into RU-group 1, improving the system performance.

3.5 Simulation Results for FlexE workers

The results for the deployment of the trained FlexE agent are shown in Figure 3.8. The input traffic pattern is upscaled by a factor of 10 and normalized to the

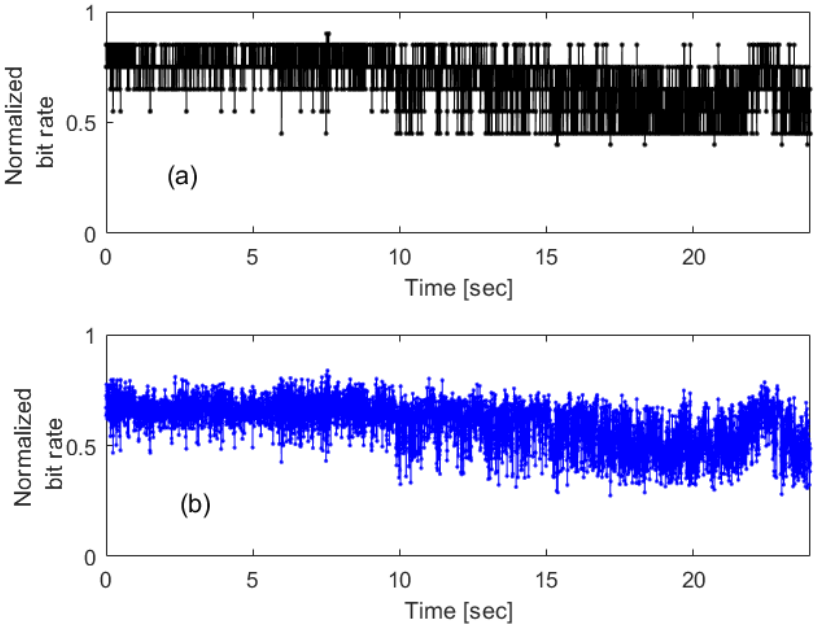


FIGURE 3.8: FlexE Agent slot allocation (a), input traffic (b).

output rate of the FlexE aggregation node. Figure 3.8(b) shows the input traffic and Figure 3.8(a) shows the number of slots allocated to the input traffic. It can be observed in Figure 3.8(a), that the agent tracks the input traffic and allocated slots, including the very rapid changes in the traffic. More results and comparisons are available in [67], for example performance comparison with predictions from the autoregressive integrated moving average model approach [92]. Results were compared in terms of mean over-allocation (MOA) of traffic as a measure of bandwidth over-consumption in the Xhaul. It was observed that the mean over-allocations from DRL-based predictions are 11.6% less than the mean over-allocation from ARIMA-based predictions. A similar result was observed for mean under-allocations. DRL made 25% less under-allocations than using ARIMA-based predictions. Therefore, DRL can outperform ARIMA-based predictions both in terms of efficient bandwidth consumption and minimal delays in Xhaul.

TABLE 3.5: Background Traffic effects on RU group offloading.

RU positions		Background traffic (Gbps)		Number of users		Bit rate (Gbps)		Total bit rate (Gbps)
		RU group		RU group		RU group		
		1	2	1	2	1	2	
1	a	110	110	15	15	117.04	117.04	234.1
	b	285	110	15	15	86.16	115.8	201.8
2	a	110	110	15	15	113.3	115.8	229.1
	b	114	111	14	16	107.8	121.2	229
	c	250	110.7	9	21	77.25	145	222.3
	d	285.5	105.67	7	23	62.9	153.5	216.5
	e	331.5	111.5	3	27	31.38	166.67	198.5
	f	113	265.26	19	11	125	100.2	225.2

3.6 Simulation Results for offloader

The deployment results for the controller agent that is trained for 36 episodes are shown in Figure 3.9 with the RU position shown in Figure 3.5. The average background traffic added to the paths of the two RU groups is the same and around 110 Gbps. The results in Figure 3.9(a) show the total achievable bit rate of the two RU groups is around 234.09 Gbps after applying the functional split factor of 10.5. The results in Figure 3.9(b) show the total number of users in two RU groups, along with users in the individual RU groups. It is apparent from Figure 3.9(b) that the initial number of users in the two RU groups is 13 and 17 as a non-ideal starting distribution of users. The agent then offloads users so that both RU groups have the same number of users. This is so because the total achievable bit rate is maximum when the two RU groups have the same number of users.

The results in Figure 3.9 are reflected in case-1(a) of Table 3.5. In case-1(b), an increased background traffic of 285 Gbps was introduced into the path of RU group-1. It is evident from the results in case-1(b) that the agent did not shift users from RU group-1 to RU group-2. Consequently, this led to a decrease in the total bit rate of the RU groups from 234.1 Gbps to 201.8 Gbps. This illustrates that

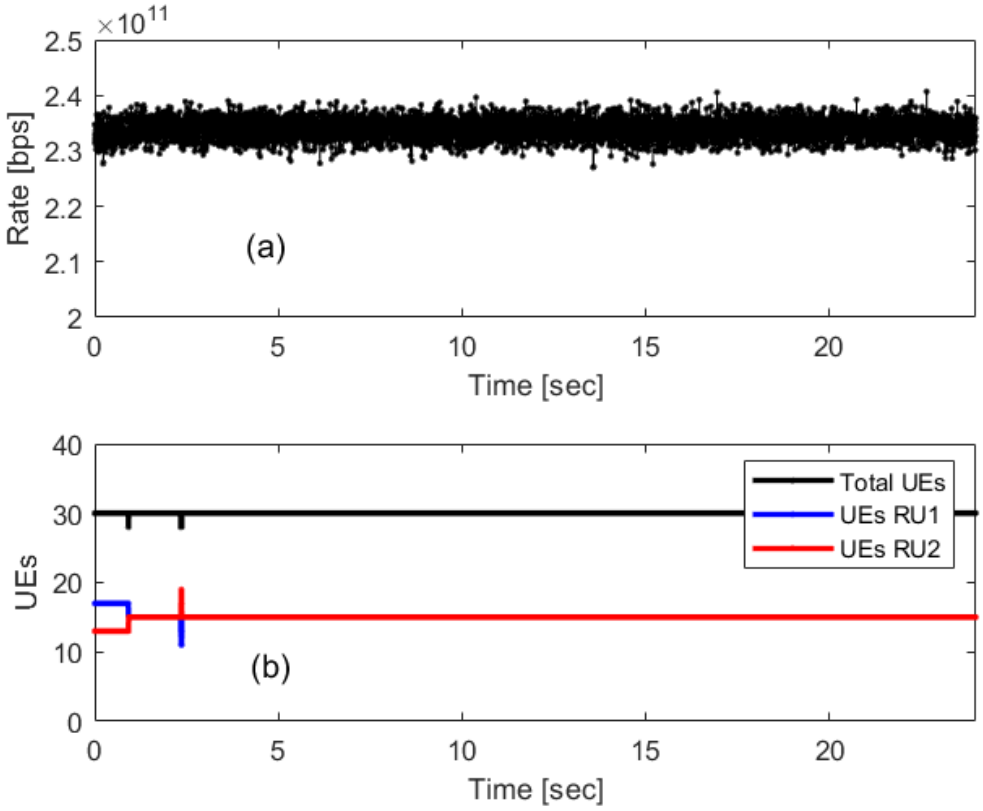


FIGURE 3.9: Controller agent deployment results, total achievable bit rate (a), number of users in RU groups (b) for RU positions in Figure 3.5.

the introduction of higher background traffic leads to increased latency, resulting in a reduction in the total bit rate. Following this, the positions of RUs were altered, placing them in non-ideal locations with varying distances to users across the 200x200m grid. This configuration resulted in a greater average distance between users and RU group-1 compared to RU group-2. Notably, the controller agent was retrained to accommodate this new RU position configuration. The background traffic introduced into the network paths of the RU groups exhibits variability, with their respective values outlined in cases 2(a) through 2(f) within Table 3.5. These values denote the average background traffic sustained over a 24-second deployment period. Notably, the degree of variation in background traffic within the RU group-1 network path exceeds that within the RU group-2 network path, progressing from case 2(a) to case 2(f). Specifically, the background traffic in the RU group-2 network path fluctuates around 110 Gbps. Furthermore, the bit rate and user count for each RU group following the deployment of the trained controller agent are also documented for cases 2(a) through 2(f).

In case 2(a), where both network paths feature identical background traffic levels of 110 Gbps, RU group-2 exhibits a higher bit rate than RU group-1, primarily due to the users' closer proximity to RU group-2 on average. However, the controller agent does not transfer users from RU group-2 to RU group-1 in this scenario. Controller agent opts to offload users only when additional background traffic is introduced into the RU group-1 network path, a trend that becomes increasingly apparent from cases 2(b) through 2(e). Furthermore, it becomes apparent that with the escalation of background traffic, a rise in latency leads to a decrease in the bit rate of RU group-1 and, subsequently, the overall bit rate of the two RU groups. This highlights that the impact of latency in terms of diminishing the overall bit rate of the RU groups is more pronounced than the impact of users locations and distances from the RU groups. Additionally, a comparison between case 1(b) and case 2(d), where background traffic levels in both network paths are identical, reveals that the total bit rate in case 2(d) (216.5 Gbps) surpasses that in

case 1(b) (201.8 Gbps) by up to 7.3%. This illustrates that the trained controller, by reallocating users between RU groups, can enhance the attainable system bit rate. Finally, in case 2(f), an increase in background traffic within RU group-2's network path results in the controller agent offloading users from RU group-2 to RU group-1, demonstrating the system's symmetry and the controller agent's ability to adapt to offloading users in both directions.

3.7 Conclusion

The chapter proposed to use DQN in a centralised RAN capable of CoMP communication. DQN is used for both user and resource allocation in a central controller and FlexE agents, respectively. In the controller agent, DRL is used to reduce the effects of latency in RU groups by offloading users to less active RU groups. This agent learned the policy required to reduce the latency in the Xhaul and increased the resulting sum-rate experienced by the users and when comparing cases with equal bandwidth, but different user distributions a rate improvement of 7.3% was achieved. In FlexE aggregation nodes, agents are used for traffic prediction and performing slot allocation. From the results it can be concluded that DRL can allocate resources in FlexE based Xhaul more efficiently than a classical approach based on ARIMA allocation scheme, with 11.6% less over-allocations and 25% less under-allocations which would improve the OPEX when used in a real world scenario. Further, the FlexE agent could learn a wide variety of traffic patterns with different properties of mean and variance, to the least, and is able to predict underlying patterns in incoming traffic.

Chapter 4

User Association in Fog Radio Access Networks

4.1 Introduction

Cloud radio access network (C-RAN) has been an important and widely used network architecture in current wireless communications. In this architecture, a baseband unit (BBU) with high processing capability is connected to multiple remote radio heads (RRHs) via fronthaul links and undertakes all baseband processing tasks from RRHs. This architecture can implement cell-free massive multiple input multiple output (MIMO) [5]. Cell-free means that users experience no cell boundaries during transmissions. From the network point of view, RRHs cooperate with cloud synchronization to jointly process and transmit user data. RRH cooperation leads to large gains at the cell-edge [11]. In practice, a BBU needs to collect all channel state information (CSI) through fronthaul links to perform its tasks [93], which can overload the fronthaul links and degrade the network performance. Furthermore, considering that a BBU is far-away from the RRHs, with supported distances of more than 10km [94], fronthaul data transmission introduces additional latency and significant power consumption in C-RAN [95].

A fog radio access network (F-RAN) has been proposed as an evolution of

C-RAN architecture [12], aiming to alleviate the strain on fronthaul links and provide low latency by replacing RRHs in C-RANs with fog access points (FAPs) which have application processing and caching capabilities [44]. With this evolution, time-critical computing, such as cooperative radio resource management (CRRM) and cell-free MIMO radio signal processing, is placed at the edge of the network [45] to support low-latency services. In an F-RAN, cloud layer is still present [15] to process the resource-intensive tasks and larger quantities of data. As processing and caching are enabled at FAPs, which are at the edge of the network, this architecture reduces the amount of channel information being transmitted over the long fronthaul links to the cloud for network control, e.g. resource management, and allows content acquisition and network control to be done in a coordinated manner at the fog network access layer. Therefore, the F-RAN architecture can improve channel accuracy, resulting in higher transmission rates and low latency [12]. However, it leads to unique challenges before it can be mature enough for operation [43]. With radio resource management and content acquisition capabilities enabled at the edge, cooperative methods have to be developed among FAPs for the distributed system. Since FAPs can be deployed densely to serve a large number of user equipments (UEs), without cooperation among FAPs, strong interference can be introduced among the coverage areas of different FAPs. Therefore, joint UE-FAP association and transmit power allocation is essential to manage available resources efficiently.

UE-FAP association in F-RANs has been addressed in [26] and [27] by considering the signal to interference plus noise ratio (SINR) [26] or received signal power [27]. However, these studies ignored that an FAP only has limited resources to accommodate UE demands for services and content. If one FAP connects to many UEs, it can overload, leading to a significant deterioration of performance in the fog network. As such, load balancing algorithms have to be taken into account to ensure a smooth experience for UEs [46]. Additionally, most research focusing on UE-FAP association including data caching [96] did

not consider that a high percentage of Internet contents are generated from delay-sensitive applications, such as interactive content and banking applications [97]. When users connecting to fogs need to access this kind of delay-sensitive data, it is essential to address the latency in the UE-FAP association.

Considering the fronthaul load, cache prioritisation, as well as edge computing, UE-FAP association is a high dimensionality problem, to which the solutions in the existing research cannot solve. When the number of UEs is large, solving this problem to achieve optimal performance using traditional algorithms would be near-impossible due to the high computation complexity. On the other hand, deep learning has shown the capability of solving extremely complex problems, which require abstract understanding and creativity, like the algorithm that plays Go at a grandmaster level [98]. Sharing information between agents has been investigated in deep learning based solutions [99]–[101]. In [99], agents with different roles in a given environment, communicated with each other to increase the number of observations available to each one. Experience sharing is possible in A3C [100], an algorithm that simultaneously performs policy optimisation as well as policy evaluation, through the actor and critic respectively. In A3C, experiences are shared among homogeneous agents interacting with an instance of the same environment where they do not interact with each other and do not affect the decisions of other agents. This makes the environments identical to single-agent learning. In more recent multi-agent reinforcement learning (MARL) settings, multiple agents exchange information about the environment to make joint observations and actions. [101] studied how multiple agents can learn cooperatively to improve learning speeds. However the results are only applicable to fully observable or complete Markov decision process (MDP) environments, where agents are deployed and interact with separate instances of the same environment. In above studies, if agents exist within each other's environment, they require a lot of communication before making a decision [23].

However, to meet latency requirements under the F-RAN architecture, practically, FAPs can only partially observe the environment, leading to a partially observable MDP (POMDP), as exchanging observation information to form a joint observation and joint action could lead to significant overheads. Some work has been done in the area of MARL in POMDP, [102], which used a method similar to hyper-Q-Learning, which does not focus on the area of experience sharing or exchange. Instead, agents learn about the policies of the other agents through their own observations.

A potential solution to this problem would be to train a double deep Q-network (DDQN) to learn about the environment. DDQN is a reinforcement learning (RL) method that uses two networks, an estimation network and a target network. It was proposed in [22] to overcome the problems of classical deep-Q-network learning, which only uses one estimation network to learn. By introducing a target network that is slowly updated, the deep learning agent becomes more stable. The target network is assumed to be closer to the ground truth, and the estimation network has an entity against which it can compare decision results to. This solution works well for single-agent environments, as it can closely estimate MDP state transitions and rewards, but its performance degrades in multi-agent POMDPs [102]. If multiple agents explore a shared POMDP environment, with each maximising their own reward, they will fail to cooperate with other agents effectively, because estimations are not able to reflect other agents' decisions. Therefore, there is higher uncertainty, resulting in lower stability when learning, even with slow updating target networks and relatively long training durations.

This chapter aims to tackle UE-FAP association in the F-RAN architecture as a partially observable distributed MARL problem where each FAP is an independent agent, but also works collaboratively with other FAPs to maximize the number of served UEs under some constraints. Due to the changing nature of the environment, agents must learn that trade-offs are necessary to accommodate the

maximum number of UEs and adapt the solution to satisfy UE requirements. The main contributions of this chapter are summarized as follows:

1. Upon analysis it is clear that in the distributed F-RAN architecture agents may not have enough time to exchange entire state information due to the low latency requirement in future wireless communications. Since a fully observable MDP cannot be developed due to strict constraints, a POMDP environment is formulated. Deep learning agents must not only learn how their actions affect the system state, but also how to predict what actions other agents would take in the same state. It is shown that the traditional low-complexity algorithms perform poorly in the F-RAN environment, since they consider greedy UE associations to benefit individual FAPs, not the entire system.

2. A solution for a POMDP environment called multi-agent distributed double deep Q-network (MA-DDQN) is proposed, which uses an experience exchange method to trade information, such as the states, actions, rewards, and following states between agents in the environment. This method is particularly helpful in making deep learning agents diverge from a local optimum solution, as agents can build a larger knowledge base required for learning than they would independently. Subject to system constraints, it is shown how agents use the experience exchange mechanism with no changes to the hyper-parameters of the neural networks, and collect better rewards than low-complexity hard-coded methods. The simulation results show that the proposed algorithm outperforms the conventional DDQN algorithm which uses no experience exchange.

The remaining structure of the chapter is as follows. In Section 4.2, the system model is introduced as an iteration of C-RAN, with distributed coordinated

multi-point radio resource allocation. Section 4.2 also introduces the main optimisation problem and the constraints, which are presented as a POMDP environment in Section 4.3. In Section 4.4 the rationale of using distributed neural networks, rather than centralised is explained. Based on an overview of the concept of RL, a novel deep RL algorithm to solve partially observable multi-agent problems is proposed in Section 4.5. In Section 4.6, the results of training and simulation are presented and discussed, and the chapter is concluded in Section 4.7.

4.2 System Model

4.2.1 Transmission model

Consider an F-RAN architecture including a set of FAPs $\mathcal{J}$ with a total of $|\mathcal{J}| = J$ FAPs, each equipped with M antennas, as shown in Figure 4.1. FAPs are connected to the cloud network through the fronthaul links. They are also connected to each other via high-speed fibre links in a mesh topology [103]. In F-RAN, network access layer requires horizontal connectivity to allow FAPs to exchange information required for cell-free CRRM [40], as such, links between FAPs are utilized to share network information with low latency. FAPs can connect to multiple UEs by using cell-free MIMO methods for joint transmission and coordinated beamforming[104]. The set of UEs to be served is denoted as $\mathcal{K}$, with a size of $|\mathcal{K}| = K$. It is assumed that each UE has a single antenna. Each FAP can associate with a number of UEs through multiple beams, and the association factor between UE $k \in \mathcal{K}$ and FAP $j \in \mathcal{J}$ is denoted as a variable $x_{j,k}$, given by

$$x_{j,k} = \begin{cases} 1, & \text{if } k \text{ is associated to FAP } j, \\ 0, & \text{if not.} \end{cases} \tag{4.1}$$

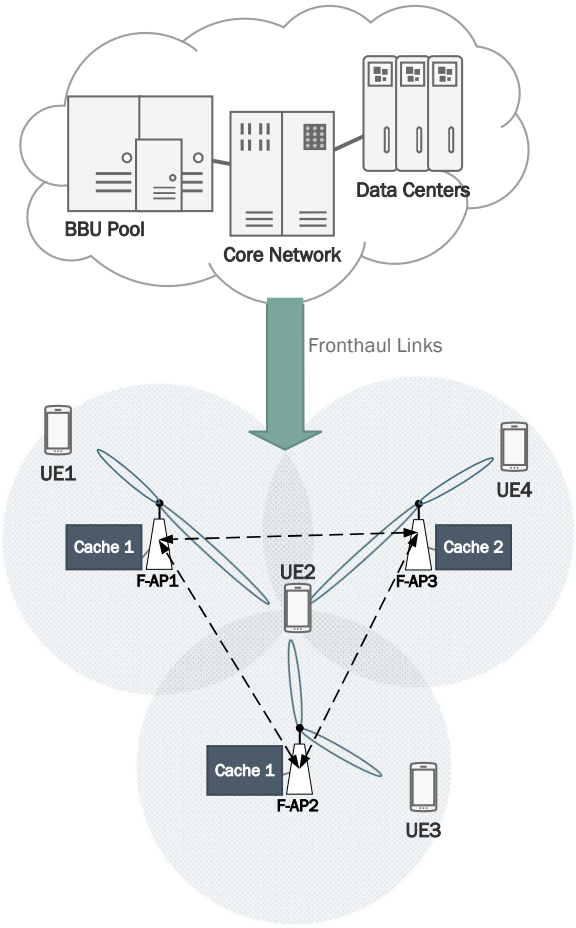


FIGURE 4.1: System model showing a F-RAN deployment as an extension to the cloud. Links between FAPs (short dashed lines) allow for CRRM.

The sum of associations $\sum_k x_{j,k}, \forall j \in \mathcal{J}$ is the cardinality of set $\mathcal{K}^{(j)}$ that includes all the UEs associated with FAP j. The union of all association sets produces a set of UEs associated with the fog $\mathcal{K}_\Phi$ and is shown as $\bigcup_{j=1}^{J} \mathcal{K}^{(j)} = \mathcal{K}_\Phi$.

Note that even if the interest lies in associations within fog with global synchronisation between FAPs assumed, some information may not be known by the rest of the FAPs. In a downlink communication scenario, UEs estimate the channel condition and feed it back. Assuming that the channel condition is perfectly known, UE k requests for content $c_k = [0, 1, \cdots, \phi]$ of size b_ϕ. If the content request is fulfilled, UE k is considered as served. When the UE service request

is done, it is included in a set $\mathcal{K}_d$, and the number of served UEs will be given as $|\mathcal{K}_d| = x_d$. Using cell-free MIMO [40] [105], the file would be jointly transmitted to the UE in multiple streams of simultaneous message signals s_k. With coordination, interference alignment between APs is possible, and UE k can be associated with multiple FAPs, while UEs that are not associated with all FAPs can also receive a signal with reduced interference from other FAPs[106]. Since the UEs are static, the main source of interference is the inter-stream interference between FAPs. As such the received symbol at UE k is given by

$$y_k = \sum_j x_{j,k} p_{j,k} \boldsymbol{h}_{j,k}^{\mathrm{H}} \boldsymbol{w}_{j,k} s_k + \sum_j \sum_{i \neq k} x_{j,i} p_{j,k} \boldsymbol{h}_{j,k}^{\mathrm{H}} \boldsymbol{w}_{j,i} s_i + n, \tag{4.2}$$

where n is the additive white gaussian noise (AWGN), satisfying $\mathcal{N}(0, \sigma^2)$, $p_{j,k}$ is the waterfilling power allocation from FAP j to user k. $\boldsymbol{h}_{j,k}$ is the $\mathbb{C}^{M \times 1}$ static complex Gaussian channel model composed of M channel responses, denoted as

$$\boldsymbol{h}_{j,k} = \begin{bmatrix} \mathrm{PL}_{j,k}^{\mathrm{ABG}} h_{j,k,1} & \mathrm{PL}_{j,k}^{\mathrm{ABG}} h_{j,k,2} & \cdots & \mathrm{PL}_{j,k}^{\mathrm{ABG}} h_{j,k,M} \end{bmatrix}^{\mathrm{T}} \tag{4.3}$$

where each entry $h_{j,k,m}$ represents the channel of UE k to antenna m of FAP j. Defining $\mathrm{PL}_k^{\mathrm{ABG}}$ as a pathloss coefficient that follows the ABG urban micro open space pathloss model [107] for high frequency communications, the pathloss coefficient for frequency f, distance $d_{j,k}$ from UE k to FAP j, with log-normal shadowing $\chi_\sigma^{\mathrm{ABG}}$ is evaluated as follows

$$\mathrm{PL}_{j,k}^{\mathrm{ABG}}(d_{j,k}) = 10 \cdot 2.6 \cdot \log_{10}\left(\frac{d_{j,k}}{1\mathrm{m}}\right) + 24 + 10 \cdot 1.6 \cdot \log_{10}\left(\frac{f}{1\mathrm{GHz}}\right) + \chi_\sigma^{\mathrm{ABG}}. \tag{4.4}$$

In (4.2), $\boldsymbol{w}_{j,k}$ is the $\mathbb{C}^{M \times 1}$ transmitter beamforming weights vector. For simple analysis, zero-forcing is considered for the transmit beamforming vectors [108],

given by

$$\boldsymbol{w}_{j,k} = (\boldsymbol{h}_{j,k}^{\mathrm{H}} \boldsymbol{h}_{j,k})^{-1} \boldsymbol{h}_{j,k} \tag{4.5}$$

FAPs have limited instantaneous transmit power, denoted as P_j^{TRI}, given by

$$P_j^{\mathrm{TRI}} = \sum_k P_{j,k}^{\mathrm{TRI}} = \sum_k x_{j,k} p_{j,k} \left| \frac{\boldsymbol{w}_{j,k}}{|\boldsymbol{w}_{j,k}|} \right|^2, \tag{4.6}$$

where the precoding vector $\boldsymbol{w}_{j,k}$ is normalized by its magnitude to ensure that only interference cancellation is being performed by the precoding vectors.

The resulting signal to interference plus noise ratio (SINR) experienced by UE k is given by

$$\gamma_k = \frac{\sum_j x_{j,k} p_{j,k} |\boldsymbol{h}_{j,k}^{\mathrm{H}} \boldsymbol{w}_{j,k}|^2}{\sum_j \sum_{i \neq k} x_{j,i} p_{j,i} |\boldsymbol{h}_{j,k}^{\mathrm{H}} \boldsymbol{w}_{j,i}|^2 + \sigma^2}. \tag{4.7}$$

The corresponding data rate of UE k is denoted R_k, given by

$$R_k = \log_2(1 + \gamma_k), \tag{4.8}$$

4.2.2 Cache and Scheduling Model

Consider that the F-RAN has caching capabilities at FAPs for an improved UE experience. FAPs will be located within a small area to support the transmission of delay-sensitive contents. Further consider that FAPs only have limited caching capability and the caches are populated by a uniform random distribution. Content availability in FAP j is denoted as a binary value $c_{j,\phi} \in \{0, 1\}$, and $\sum_\phi c_{j,\phi} \leq C_j$, where C_j represents the cache capacity. To guarantee the latency requirement, δ_{th}, for the contents, it needs to analyse the user experienced delay, depending on whether content is present in the caches. As a UE demands content ϕ of size b_ϕ, each FAP is aware if it is present in the cache, as such, the delay for

UE k is split into three cases

$$\delta_k = \begin{cases} b_\phi \left(\frac{1}{\Psi_j} + \frac{1}{R_k} \right), & \text{if } c_{j,\phi} = 1 \\ b_\phi \left(\frac{1}{\Psi_j} + \alpha \frac{1}{R_{\text{bh}}} + \frac{1}{R_k} \right), & \text{if } c_{l,\phi} = 1 \, , l \neq j \\ b_\phi \left(\frac{1}{\Psi_b} + \frac{1}{R_{\text{fh}}} + \alpha \frac{1}{R_{\text{bh}}} + \frac{1}{R_k} \right), & \text{otherwise.} \end{cases} \tag{4.9}$$

In the first case in (4.9), content is available at FAP j that received the request, Ψ_j denotes the processing speed of the FAP. In the second case, the content is not immediately available at the FAP, but can be obtained from the adjacent FAP l. R_{bh} is the backhaul link rate, and α denotes the minimum number of hops required to acquire the content. In the final case, the content is unavailable at the adjacent FAPs; therefore, it must be acquired from the BBU pool through the fronthaul. Ψ_b is the backhaul processing speed. Fronthaul rate is defined as R_{fh}. R_{fh} and R_{bh} are assumed to be constant. Since δ_k is dependent on R_k and, as a result on how many FAPs are associated with user k, some FAPs may need to wait before content is acquired before transmitting the content. As such, if the FAPs associated with the UEs have different contents in caches, the highest delay value is used.

Corresponding to three cases, FAPs that are associated with UE k will consume a different amount of power to acquire the content through the network. Fronthaul and backhaul consume power at a time instant, modeled as

$$P_{j,k}^{\text{XH}} = \begin{cases} 0, & \text{if } c_{j,\phi} = 1 \\ \alpha \left(P_{\text{ONU}} + P_{\text{LS}} \right) + N_{\text{LC}} P_{\text{LC}}, & \text{if } c_{l,\phi} = 1 \, , l \neq j \\ \alpha \left(P_{\text{ONU}} + P_{\text{LS}} \right) + N_{\text{LC}} P_{\text{LC}} + m P_{\text{SW}} + P_{\text{SWB}}, & \text{otherwise.} \end{cases} \tag{4.10}$$

In the first case in (4.10), content does not need to traverse the network and is present in the cache, meaning no Xhaul power is being consumed. A linecard

(LC) is a piece of Xhaul hardware used for a multitude of functions, such as digital-to-analog conversion within switches and local communication between FAPs and the cloud. In the second case, N_{LC} and P_{LC} represent the number of LCs used in the network and the power consumed by each LC, respectively. LCs can be shared in the scenario of sharing the same wavelength, as such, no more than one LC will be used in this model. In the same case, P_{ONU} and P_{LS} are the power consumed by optical network units, used for high-speed data transmission, and each switch along the path in the backhaul, respectively [109]. In the final case, P_{SW} denotes consumed power at each switch port in the cloud and m is the number of switch ports used in the content's path. P_{SWB} is the baseline power consumption of the cloud switch.

4.2.3 Problem Formulation

Through the UE-FAP association, which generates a set $\mathcal{K}_{\Phi}$, the performance objective is to maximise the number of UEs x_d, which are served by the fog, with their latency requirement, δ_{th} being satisfied. Since the power consumed is different for all UEs, due to varying delays, energy consumption will provide some trade-off if the UEs' requested content is unavailable, but they are experiencing high SINR conditions. The total power consumption can be derived as a sum of the Xhaul power consumption and FAP beamforming power consumption to serve all selected UEs and is expressed as

$$P_{\text{tot}} = \sum_{j} \left(P_{j}^{\text{TRI}} + \sum_{k} P_{j,k}^{\text{XH}} \right). \tag{4.11}$$

Finally, the problem formulation of user-FAP association is given by

$$\text{O1:} \quad \arg\max_{\mathcal{K}_\Phi} x_d, \tag{4.12a}$$

$$\text{s.t.} \quad P_j^{\text{TRI}} \leq P_T, \forall j \tag{4.12b}$$

$$R_k \geq R_{min}, \forall k \tag{4.12c}$$

$$\delta_k \leq \delta_{\text{th}}, \forall k \tag{4.12d}$$

$$P_{\text{tot}} \leq P_{\text{th}} \tag{4.12e}$$

where the first constraint (4.12b) limits the transmit power of FAPs, which also limits how many UEs can be served by individual access points. Constraint (4.12c) guarantees that the UE sets selected by the system have satisfactory rate performance for serving groups of FAPs. Since fronthaul is used for accessing data that is not present within the fog, the latency constraint δ_{th} shown in (4.12d) can help locate UEs with low transmission rate to benefit from files located within the caches in the fog, and place high rate UEs to the fronthaul, in which case the delay introduced by the fronthaul links is endurable. δ_{th} can be chosen differently for different types of contents. Fronthaul links also consume power, which introduces P_{th} as the power constraint in (4.12e). Fronthaul consumes much more power than radio transmission, if unconstrained, selected UE sets can consume tremendous amounts of power [110]. Reducing the amount of power consumed by the UEs means sustaining as many communications within the fog as possible. This benefits the overall goal of improving energy efficiency in the upcoming generations of mobile communications [111]. Further, looking at the delay characteristics in (4.9), if all possible requested UE content is present in caches, the selection becomes rate dependent, rather than relying on reducing the number of links being used for fronthaul transmissions. As such, a fixed solution is not beneficial and must be adaptable to many different factors in the network.

Due to the presence of an integer optimisation variable and multiple floating-point constraints, this problem becomes a non-convex mixed-integer non-linear programming problem. Consider the following factors: 1) there are K UEs, 2) each UE can associate with more than one FAP, 3) a total of M UEs can be admitted at one FAP with M antennas, and 4) the sets of selected UEs can be different at each FAP. Therefore, the computational complexity for an optimal brute-force solution at each FAP j is $O(JK^M)$. With this combinatorial explosion, finding an optimal solution is difficult, and using brute-force search methods is computationally infeasible even at relatively small networks ($M \approx 8$, $K \approx 16$). To find feasible solutions to the problem, a deep RL-based method will be adopted, which has high potential in dealing with dimensionally complex problems.

4.3 Environment Design

Consider an environment following the model defined in Section 4.2. In this section, the states, actions and rewards which are available to each agent are defined in more detail. It is assumed that all UEs enter and remain within the range of their respective FAPs until their content requirement $c_k = \{0, 1, \cdots, \phi\}$ is met. As stated in the system model, UEs provide feedback with CSI holding information about SINR that they are experiencing γ_k, based on the estimated channel strength $|\mathbf{h}_{j,k}|$. The FAP itself holds information about the power P_j, the availability of the content $C_{j,\phi}$, and the expected delay under current conditions $\delta_{j,k}$. In this scenario, an agent is located within each FAP, accessing the required observations. An agent navigates the state-space observing one UE at a time, and a single epoch lasts until all UE-FAP associations have been determined, which is defined as an endgame. The specific variables and values are elaborated as follows

- An endgame $D \in \{0, 1\}$ is said to be reached ($D = 1$) when all users in set $\mathcal{K}$ have been acted on, timestep $k = |\mathcal{K}|$, initially $D = 0$.

- The state observation vector for a single agent j observing UE k is $o_{j,k} = \{\alpha, k, |\mathbf{h}_{j,k}|, R_k, \max \delta\}$ and a collection of observations from all agents is the full state of the system $s_k = \begin{bmatrix} o_{1,k} & o_{2,k} & \cdots & o_{j,k} \end{bmatrix}$. This ensures that the algorithm is flexible for a changing number of UEs.

- An action $a_{j,k} = \{0, 1\}$ is taken by all agents for each UE in range, where action indices correspond to "don't associate with current user", "associate with current user", respectively. A collection of actions from all agents will be $\mathbf{a} = \begin{bmatrix} a_{1,k} & a_{2,k} & \cdots & a_{j,k} \end{bmatrix}$.

- The reward r_k is handed out to the agent after each action and depends on the performance metrics resulting from the decision. The reward function is defined as

$$r_k = \begin{cases} -\frac{P_{\text{tot}}}{P_{\text{th}}} - \sum_k \frac{\delta_k}{\delta_{\text{th}}}, & \text{if } D = 0, \\ x_d - \frac{P_{\text{tot}}}{P_{\text{th}}} - \sum_k \frac{\delta_k}{\delta_{\text{th}}}, & \text{if } x_d > 0 \text{ and } D = 1, \\ r_{min}, & \text{if } x_d = 0 \text{ and } D = 1, \end{cases} \tag{4.13}$$

  where statistic of all UEs within a fog is readily available to the agents.

The way the reward is formulated improves the prospects of cooperation. For the first two conditions, performance metrics are weighted by power and delay thresholds, P_{th} and δ_{th}, respectively. This is used to normalize the impact of each performance indicator in the rewards. Power consumption can be higher by a factor of a thousand than the number of UEs associated with and significantly larger than the delay. The first condition grants the agent intermediary rewards based on power consumption and expected delays for the current number of selected UEs, and grants the agent with an ability to familiarise itself better with state transitions. The second condition assumes an end of association period and grants the rewards based on the end result of the agent's decisions. The first term

is the term that satisfies the primary objective of a number of associations, and the second term describes the condition of energy consumption within the network. The last element of the equation is the sum of the delays that the UEs are experiencing at the time of reward. The third condition prevents the agent from not selecting any UEs, by using a minimum reward, r_{min}, as this undesirable solution theoretically grants null power consumption and does not introduce any delays to other UEs. Hence, a variable reward is chosen that would typically be lower than the rewards from the top two cases. While the first case is useful in helping the agent navigate the state-space, only the second condition is important and shows how good the performance is when the final decision is made. A high reward means that a large number of UEs have been served with power consumption and delay close to the threshold.

Agents only take actions based on the system observation $o_{j,k}$, and are not explicitly aware of the system constraints defined in (4.12b)-(4.12e). To support system constraints, algorithms implementing RL must have approaches to address the constraint handling. The constraint handling method is included in Algorithm 1 of Section 4.5. To make sure that constraints (4.12b)-(4.12d) are satisfied, whenever an agent makes an action that would alter the selected user set, the following procedure should be carried out. Initially, if the FAPs are informed of the association decision, they must send an association signal to UE k. Upon collection of association signals, the number of FAPs wanting to associate is fed back to each FAP. The outcome for a new user set is found by calculating the expected R_k, P_{tot}, and δ_k for all K users. If constraints corresponding to these values, $R_{\text{min}}, P_{\text{th}}, \delta_{\text{th}}$, respectively, would be breached, an action must be prevented and no association takes place. If these conditions are met, the FAP must check if the new user set will breach the association constraint, as the number of served users for a single FAP is constrained by the number of antennas M.

4.4 Neural Network Distribution

When more than one access point can communicate with the same UEs, coordinating their efforts, as opposed to competing to provide resources to the same UE, provides a significant performance boost [34]. Coordinated beamforming is typically performed at a centralized node, such as the BBU of the network, providing the highest boost in terms of sum-rate. The drawback of using centralized coordinated beamforming is the resulting long latency because of high dependency on fronthaul rates [112]. Hence performing beamforming, by aligning beamforming vectors at edge nodes, results in reduced latency, due to more accurate channel information achieved at the edge, which can equivalently achieve better performance than centralized beamforming. The issue imposed by enabling FAPs or edge-nodes cooperation is that overhead is required in exchanging UE and decision information before UEs can be served. This can be solved by predicting other FAPs' decisions using a deep learning method.

Most previous solutions focused on learning locally and then sending data to a global aggregator, e.g. federated learning [113], distributed learning [114]. Keeping data centralised has its cons like reduced performance due to capacity constraints and concerns, such as privacy and security. And, classical distributed learning suffers from having to share model or gradient data between agents, which can grow large with more advanced neural networks present in access points.

In the research presented in the chapter, learning happens by agents exchanging raw experience information $E_{j,k} = \{o_{j,k}, a_{j,k}, r, o_{j,k+1}\}$ with each other over backhaul links. This is most like collaborative evolutionary learning. Exchanging such data between FAPs instead of sending it all to the core network or an aggregator somewhere at the edge reduces loads on fronthaul links. Models are also allowed to develop independently to maximise their performance among themselves. In this method, agent cooperation is not only implicit, but also explicit, by

introducing experience sharing as shown in Figure 4.2. Experience replay buffers are used for single-agent off-policy learning to prevent a high temporal correlation from recent samples in learning and reduce variance in predicted rewards [115]. Sampling experiences in mini-batches using any probability distribution improves the agents' ability to adapt the current policy to previously experienced scenarios. This method of learning is called off-policy learning since the data that is sampled is not representative of the current policy used by the network.

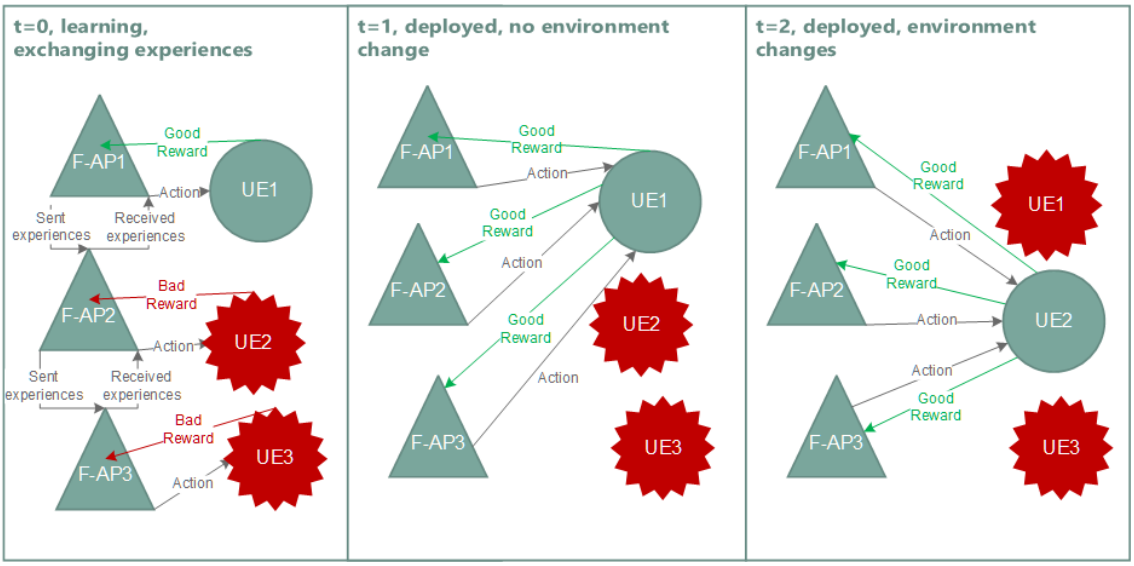


FIGURE 4.2: Experience Sharing helps agents learn from other agents' experiences as if they were their own through off-policy learning.

Since, by definition, off-policy learning can be achieved by sampling decisions made by other policies, it is proposed to extend the algorithm by introducing experience exchange between the learners. When the environment states are not defined by finite integers or finite names, the state-space can become exponentially large or infinite. Even with a limited range, floating-point descriptor values can have infinitesimal, but meaningful contributions to finding optimal or better solutions. Additionally, with a large number of features, the curse of dimensionality becomes prominent [18]. It becomes difficult for agents to learn about the environment efficiently if they learn alone. Due to multiple agents working in the same environment and affecting the reward, the partially observable property

makes the environment even more difficult to resolve [116], as the agents have to predict what actions other agents would make, to maximize the reward. If the agents learn individually, the algorithm could provide good results eventually, with little convergence guarantee. Without having enough data to learn, agents are likely to perform actions that provide a local optimum solution or no solution for an unexplored state. Additionally, independent Q-learning struggles with learning in multi-agent environments [102]. However, learning efficiency reduces with an increase of experience replay size, due to outdated policies being present in the experience replay buffer for a long period of time. On the other hand, a reduced buffer size reduces the stability of learning, as the samples become closely temporally correlated, which results in a high variance in rewards, similar to the problem in [22]. Having agents exchange experience information increases the amount of experiences with policies that are different from the current one and helps agents learn about states they could have previously not visited. An example of this is shown in the model for experience exchange in Figure 4.2. At timestep $t = 0$, agents only have clean experience buffers or experience buffers with their own previous experiences. As time goes on, agents share raw experience data with each other. With the number of shared experiences increasing, agents' decisions that lead to solutions matching other agents with good rewards get reinforced. On the other hand, solutions that would lead to local optimum solutions, but do not benefit the overall goal as much, get pruned, as shown in the $t = 1$ section of the same figure. Furthermore, at $t = 2$, it is shown that due to the generalisation property of Q-learning, agents do not need to get consistently retrained on similar properties. If an observation matches a previously observed solution, agents intuitively find a good solution. In this case, not all agents have to test the experiences individually.

4.5 MA-DDQN Algorithm Design

Off-policy learning is a powerful tool that improves convergence speed and guarantees robustness in novel deep RL algorithms. The loss function with experience replay can be derived as

$$\mathcal{L}(\beta) = \mathbb{E}_{\mathcal{U}\sim(\mathcal{E})}\left[\left(r_k + \rho Q\left(o_{j,k+1}, \arg\max_a Q(o_{j,k+1}, a|\beta)|\beta'\right) - Q(o_{j,k}, a_{j,k}|\beta)\right)^2\right], \tag{4.14}$$

where the addition to the function appears next to the expectation operator. $\mathcal{U} \sim (\mathcal{E})$ is defined as a uniform distribution over the experience database $\mathcal{E}$. The uniform distribution also includes the current state that the agent is experiencing. This solution was fine for single-agent environments, but was not well explored in multi-agent environments. Usually, this was circumvented in literature by implementing more complex algorithms, allowing full tracking of historical states, like DRQN [117], or by performing centralised training and executing distributively [118], which led to large overheads in the network due to the sheer data requirement to download the new models.

Since all agents have the same action space $\mathcal{A}_i = \mathcal{A}_j \; \forall i, j$ and state-space $\mathcal{S}_i = \mathcal{S}_j \; \forall i, j$, sharing experiences between the agents with environments that are sufficiently similar can improve the generalisation for all agents in the system. This can be achieved by hardening the experience replay buffer with various solutions to the same state at the same time. This can be achieved without them explicitly exchanging large model or observation information prior to making a decision, and with low cost in the overhead of the network.

While a similar theorem is straightforward knowledge for algorithms such as A3C [101], which used complete MDPs, it cannot directly be applied to POMDP environments, as agents are interacting with a shared environment. This means

Algorithm 1 Multi-Agent Distributed DDQN in F-RANs

Initialise environment with K UEs, J FAPs each with M antennas, J agents $\beta_J = [\beta_1, \beta_2, \cdots, \beta_J]$, and J empty experience replay buffers $\mathcal{E}_j = \emptyset \quad \forall j$
for k < $|\mathcal{K}|$ **do**
  for all agent j **do**
    Agent receives $o_{j,k}$ and performs action $a_{j,k}$
    if $a_{j,k} = 1$ **then**
      FAP j predicts R_k, P_{tot} and δ_k $\forall k$
      if $R_k > R_{\min}$ and $P_{\text{tot}} < P_{\text{th}}$ and $\delta_k < \delta_{\text{th}}$ **then**
        if $a_{j,k}$ = 1 and $\sum_k x_{j,k} < M$ **then**
          Associate with UE k
        else
          Remove UE $k-1$ associate with UE k
        end if
      end if
    end if
    Agent receives reward r_k, stores result in $E_{j,k} = \{o_{j,k}, a_{j,k}, r_k, s_{j,k+1}\}$ and updates $\beta_j \leftarrow$ (4.14)
    Agent appends $\mathcal{E}_j \cup E_{j,k}$ and transmits $E_{j,k}$ to adjacent agents
  end for
  for all agent j **do**
    j receives experiences and appends them to its own buffer $\mathcal{E}_j \cup \{E_{j-1,k}, E_{j+1,k}\}$
    for all s' sampled by $U \sim (\mathcal{E}_j)$ **do**
      j updates $\beta_j \leftarrow$(4.14)
    end for
  end for
end for

that the action of one agent affects another. This is why DDQN [22] may fail in POMDP environments. Given an observation o_j, agents will behave differently depending on what they have learned and can over or underestimate the reward based on how other agents behaved previously if the amount of information about previous states is limited to only one agent's experiences. Experience replay is already responsible for saving experiences locally and replaying them to improve the prediction of the Q-value by reducing the value of the loss function as shown in (4.14). Experience replay in a multi-agent POMDP is used by defining the extension to the experience saving, transmission, reception, and replay, which will be the basis for an MA-DDQN Algorithm.

This holds for the given environment, after an epoch of steps $K > k$ agents

can exchange their experiences with each other. Agent experience exchange is demonstrated in Algorithm 1. The received experience includes the values for states, actions, rewards and following states. Typically, experience replay memory had to be limited in previous investigations [23] [102], due to the experiences being very distant from the current policy. Using experiences from other agents generates more off-policy and off-environment information that is representative of the current policy and improves sample efficiency without needing importance sampling. Since uniform sampling is used, no additional complexity is introduced with the additional sample efficiency. It is preferred to not limit the exchange range to only agents at adjacent FAPs, as in a realistic deployment scenario the environments might call for different actions to be taken for a similar observation, leading to a better convergence in solution.

4.6 Simulation Results

In this section the performance of the proposed MA-DDQN algorithm compared to a multi-agent individual DDQN (IDDQN) learning method is presented. Both IDDQN and MADDQN agents are defined as neural networks. Both consist of a target and estimation neural network, which are fully interconnected neural networks as shown in Figure 4.3 with an input layer of 5 neurons, 4 hidden layers composed of 64, 128, 3136, 256 neurons in each layer respectively, but can be altered in the process of optimization to reduce resource requirements and achieve different results. The hidden layers are composed of rectified linear unit (ReLU) neurons, where ReLU is a piecewise linear function. These units are standard for deep learning algorithms as they do not suffer from the vanishing gradient problem, which is common in multi-layer learning algorithms. When inputs arrive at neurons they are summed, and the ReLU function will output the input if it is positive, otherwise, the output will be zero. The final layer results are then split into a value layer, the sum of which provides the predicted Q-value for a

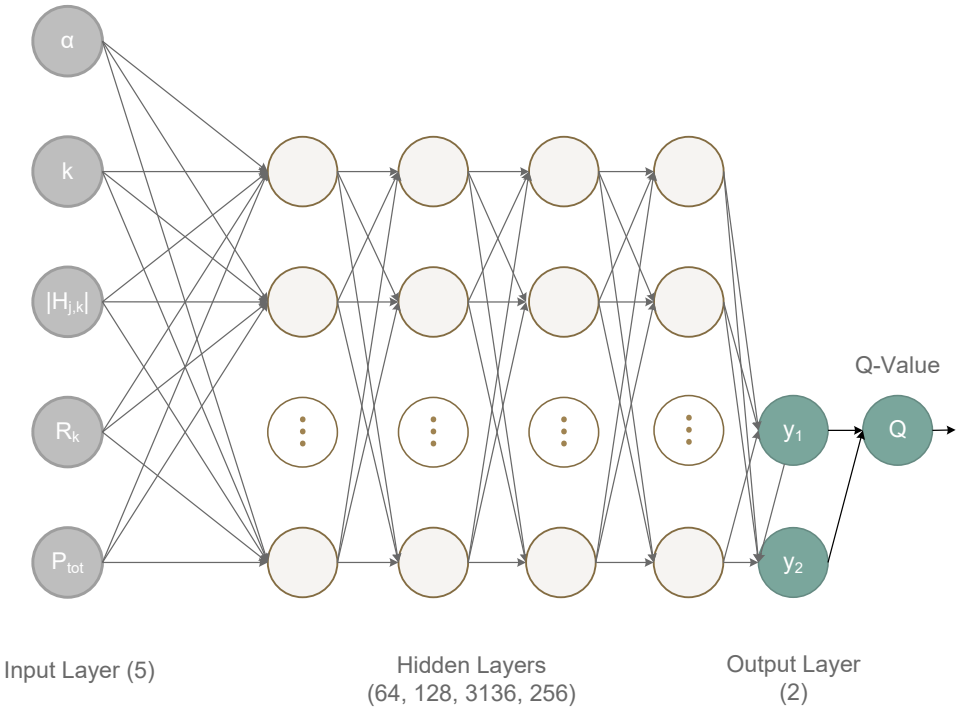


FIGURE 4.3: Neural network used for DDQN and MA-DDQN algorithms.

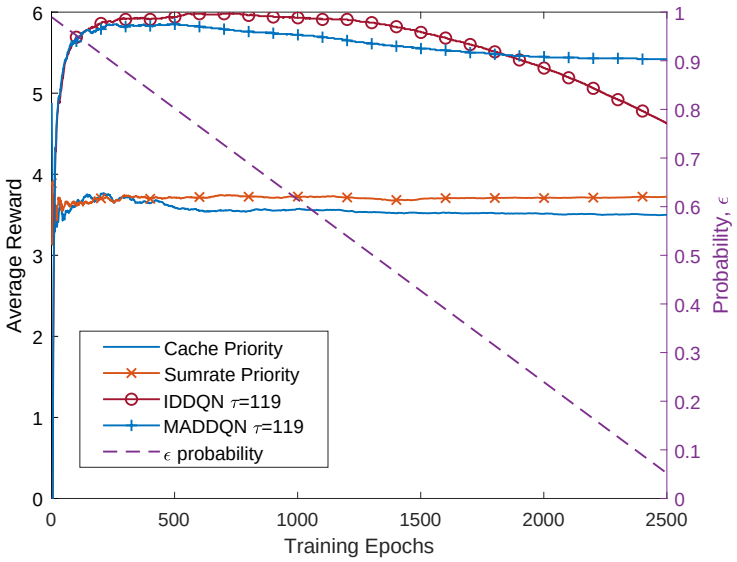
decision. Results have been presented for the DDQN algorithm after 50,000 and 25,000 training timesteps, as specified in figure captions. At the end of training, the network shows predictable converged behaviour and is operating with 5% chance of taking a random action.

4.6.1 Training Results

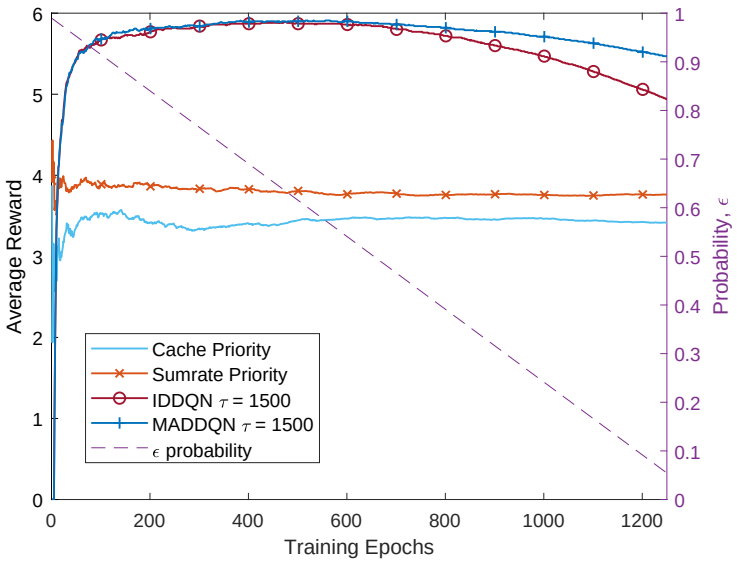
Agents are deployed in the environment as described in Section 4.3 and their neural networks are initialised with arbitrary random parameters at $t = 0$. The learning results are achieved with a discount rate $\rho = 0.99$, learning rate $\mu = 0.00025$, experience batch size $S_{\mathcal{E}} = 512$, pre-training steps $T_{\text{pretrain}} = 512$, probability of random action during training ϵ is chosen in a reducing value over time between the values $[0.99 : 0.05]$, the minimum reward is $r_{\text{min}} = 0$. Simulation parameters for both algorithms are chosen to include $J = 3$ FAPs and agents in the environment, with a maximum transmit power of $P_T = 30$W, the number of users

is $K = 20$ to ensure a quick feedback loop and each user has a minimum rate requirement of $R_{\text{min}} = 5$ Mb/s. The network can consume $P_{\text{tot}} = 1000$ W of power and introduce a delay of $\delta_{\text{th}} = 0.1$s. Training results are shown in Figure 4.4 which are achieved from running 2 different training scenarios. In scenario shown in Figure 4.4(a) the agent updates the target network every $\tau = 119$ steps. This is done deliberately. High density of target network updates destabilizes learning because of high temporal correlation between the two networks. The intention is to see if the algorithm can accommodate training with inherent lack of stability. In the second scenario shown in Figure 4.4(a), the density of target network updates is reduced. It is set $\tau = 1500$ to increase the stability of learning.

Results are smoothed out with a cumulative average of the achieved rewards, to eliminate randomness of the results and observe the trend in convergence behaviour of the algorithms. The proposed MA-DDQN algorithm is compared to the individual learning DDQN (IDDQN) algorithm, which is the only contender capable of solving the problem and is used widely in other works. At the same time, suboptimal greedy solutions are provided as a benchmark in exactly the same random environment determined by an equivalent seed. Since the base algorithm that is used for both IDDQN and MA-DDQN is the same, a similar pattern in training is noticeable in the beginning in both Figures 4.4(a) and 4.4(b). It can be seen that the proposed algorithm shows faster convergence and a higher overall reward at the end of training in both scenarios, when probability ϵ is close to 0, while the standard IDDQN algorithm shows a reduced average performance. As a result of more experiences being available, for the same training scenario as IDDQN, the MA-DDQN algorithm shows convergence to a higher reward as training continues. It can be seen from Figure 4.4(b) that IDDQN agents diverge from the solution more than they do for the scenario with more stable target network updates. The proposed algorithm does not show a similar deterioration, meaning that experience sharing provides an extra layer of stability when learning.



(A) For training steps $T = 50000$.



(B) For training steps $T = 25000$.

FIGURE 4.4: Training results over entire training period for DDQN and MA-DDQN algorithms, higher is better. 1 epoch is equal to 20 training steps.

Figure 4.5 shows a significant statistic to compare the used RL algorithms, which is the learning loss. Learning loss, as stated in (4.14) shows how much the agent deviates from the predicted reward on a new decision. It can be seen that the proposed algorithm shows a better average reward at the end of training in Figure 4.4, however Figure 4.5 shows an approximately 25% better learning performance due to a smaller loss across the entire training period. The two

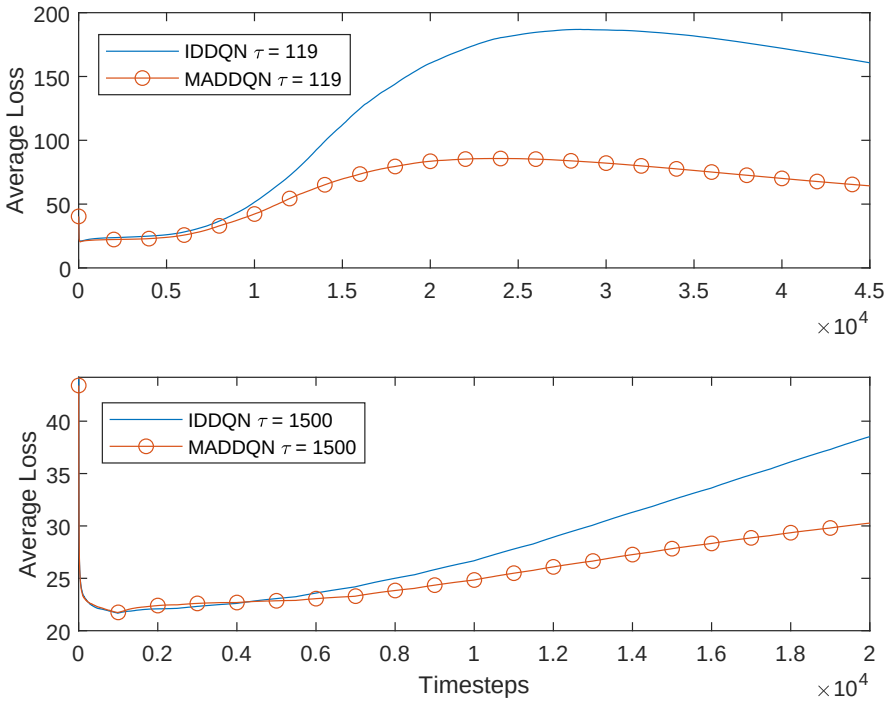


FIGURE 4.5: Difference between predicted reward and received reward over entire training period for DDQN and MA-DDQN algorithms.

figures are connected. Since loss shows how well the agent has generalized its previous experiences, an algorithm with lower loss is less likely to overestimate or underestimate its future solutions. At the end of training, when the number of target update steps is small ($\tau = 119$), the performance of IDDQN is about 3 times worse than the proposed algorithm. Lower loss can be associated with the better decision making. When agents are making decisions with $\epsilon = 0.05$ the proposed algorithm trains agents to associate with more users in Figure 4.4. This proves that exchanging experiences between agents in a POMDP, thus increasing the number of experiences in recent policies, stabilizes and improves training. It is also tested how altering the target network update can affect the two training algorithms, as it is known that larger time difference between target network updates can stabilize learning further. By changing the target network update from happening every 119 steps to every 1500 steps it can be seen that the performances of both algorithms improve, the stability increases and the average

TABLE 4.1: Training results comparison for a large and a small network using MADDQN

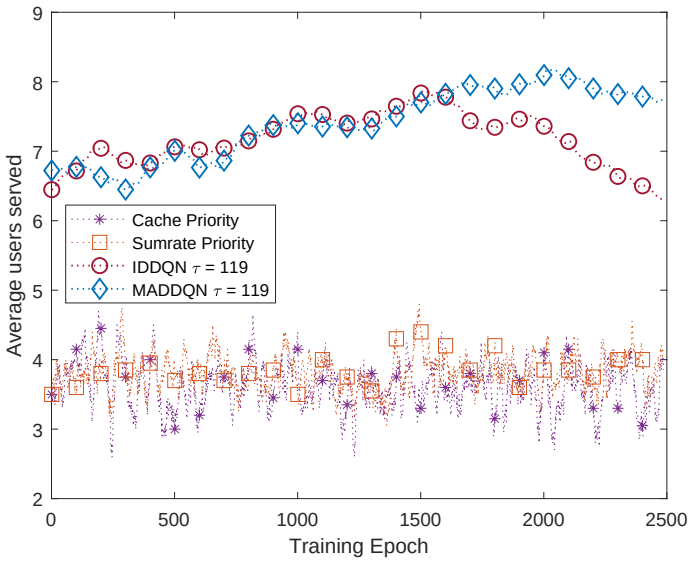
	Users Served	Average Duration	Loss	Single Episode Learning Time
Large Network	7.449	0.0104s	40.1631	0.5625s
Small Network	7.551	0.0109s	42.9084	0.5000s

losses decrease. Although more stable when the target network update happens less often, IDDQN does not catch up to the performance of MADDQN and shows about 30% worse performance in predicting the reward.

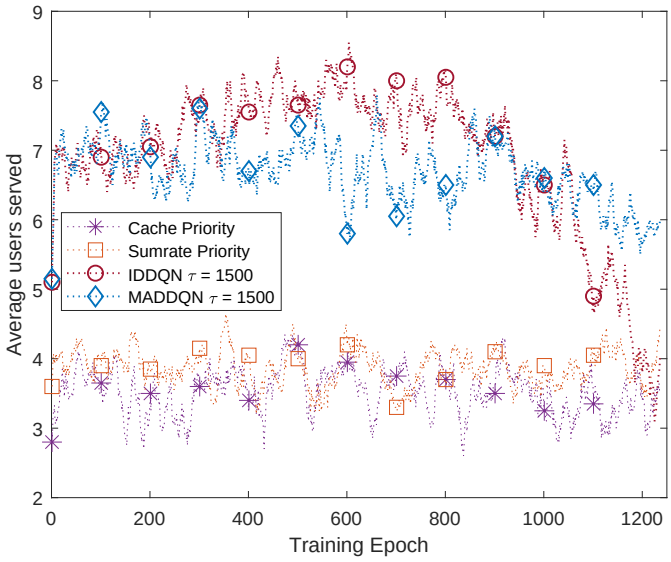
Deep learning network size is a parameter that can be tuned to achieve different results. Using a larger neural network means that the results will take longer to be achieved. It can take careful optimisation to achieve the best results when considering the performance and size trade-off. Table 4.1 shows the comparison of results when using a smaller neural network running a MADDQN method. The results labeled "Large Network" are using the neural network described in the preface to the section and shown in Figure 4.3. The other results are from a smaller neural network, having only 3 hidden layers between the input and output and they are composed of 64, 128, 256 neurons each. It can be seen in the table that the simulation results do not change significantly. The larger network serves a smaller number of users, but chooses users more accurately than the small network. This is backed up by the fact that the duration that each user experiences is lower than that of the small network, as well as the fact that the loss is lower, meaning that the larger network is making the decision while estimating the result more accurately. Depending on the application, the decision on the size can be more impactful. In the environment that the networks are trained in, the extra hidden layer introduces a $0.0625s$ overhead.

4.6.2 Performance Metric Results

User Selection



(A) Training steps T = 50000, $\tau = 119$



(B) Training steps T = 25000, $\tau = 1500$

FIGURE 4.6: Simulation output showing number of UEs served under required constraints.

In this and following sub-sections the results related to individual performance metrics are presented. In this sub-section it is explained how the reward in Figure 4.4 gets influenced by the number of users that have been successfully

served by the system. The system has 20 UEs in total. In this scenario, FAPs have a static cache holding 3 possible cacheable objects out of a total 4, UEs randomly choose 1 out of the 4 files and make a request to the system. In this scenario with a limited size cache, some files are not present at some FAPs and they have to be acquired from the cloud/Internet. If the file reaches the UE completely with specified constraints, the UE is denoted as being served by the system. IDDQN algorithm is deployed in the environment for comparison with MADDQN algorithm as well as simple greedy methods that are aiming to serve users with close cache proximity (Cache Priority) or with higher sumrate (Sum rate priority). Greedy algorithms fail to adapt to the environment and converge to an average 4 users served despite what is being prioritized. Despite seemingly random performance looking at the greedy algorithms it can be observed when better performance would be expected from deep learning methods. A peak in greedy algorithm performance is indicative of favourable conditions for deep learning algorithms as well, since the environment seed is the same for all algorithms. In Figure 4.6(a), as in the reward for 3 cacheable objects (c.o.), the number of users served converges to a solution at the end of training. When the updates of target network are frequent, i.e. $\tau = 119$ the MADDQN algorithm learns about the environment better. Because of a higher loss overall in this scenario, the opportunities to diverge benefit MADDQN algorithm and provide it with more learning experiences. DDQN performs about 40% worse at the end of training, showing that the algorithm finds some strategies for associating with different users across 3 FAPs, but the strategy is not as good as the one found by MADDQN algorithm. A similar conclusion can be drawn for Figure 4.6(b). When the frequency of target network updates decreases to $\tau = 1500$ the performance of the IDDQN algorithm becomes similar to that of the greedy algorithms. At the end of training, the proposed algorithm outperforms the standard algorithm by approximately 40%.

In addition to exploring how changing hyperparameters of training can affect the agents, it is also possible to change the parameters of the environment. One of

the significant parameters to the system is the cache availability. The number of cached objects in the FAPs can be altered to observe if that improves or degrades the performance of deep learning algorithms. Since there is already a reference for greedy algorithms, they can be replaced with curves of algorithms under test for a reduced number of c.o, e.g. 1 c.o. instead of 3. The results for this, shown

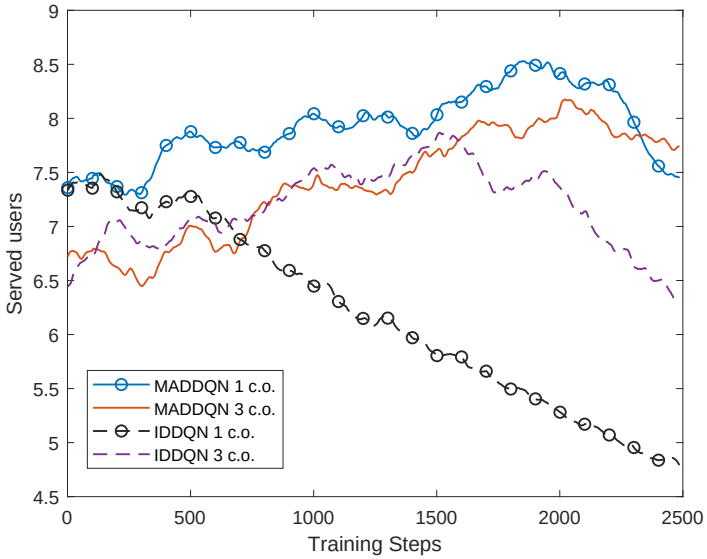


FIGURE 4.7: Simulation output showing the number of users selected when the number of cached objects changes.

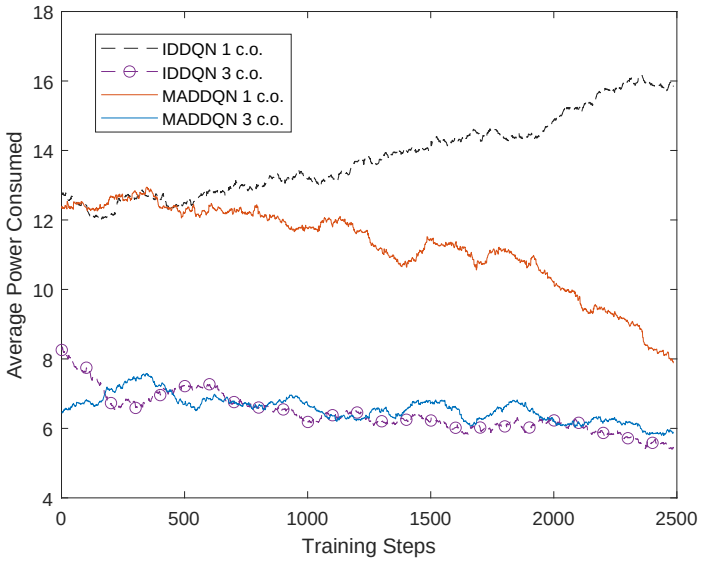


FIGURE 4.8: Simulation output showing the average powers consumed per user. Training steps T = 50000, $\tau = 119$.

in Figure 4.7 are predictable. When algorithms converge, the trend follows the fact that when the number of cached objects decreases, the performances of both algorithms decrease as well. However, the significant improvement of MADDQN is easily visible. When the number of c.o. decreases from 3 to 1, the performance falls only slightly from 7.75 users served to 7.5. That is a large performance increase over IDDQN, showing that the performance decreased from about 6.5 users to less than 5 users served as the cached object number decreases.

Power Consumption and Service Delay Results

Power consumption efficiency of deep learning algorithms for different numbers of c.o. is shown in Figure 4.8. To generate these results, the instantaneous power consumption of each FAP is divided by the number of users served. This normalisation shows how well the solution of each algorithm chooses the users to meet the power consumption constraint. As the cache availability increases, the power consumption performance of both deep learning algorithms converges. The performance of the IDDQN algorithm is nearly identical to MADDQN, however referring to Figures 4.6(a) and 4.6(b) it will become obvious that while the power consumption for IDDQN is identical to MADDQN, it does have additional power to spend to associate with more users. However, MADDQN performance is significantly better when the number of c.o. is lower. The performance of IDDQN degrades as the training continues and when the agent is making most of the decisions it fails to complete requests for users while maintaining a good power efficiency. This is consistent with Figures 4.4 and 4.6, as the reward begins to drastically fall at the end of training, while the agent is making overestimations for its decisions.

The delay experience by the users shows more performance gain when MADDQN is used. When the number of cached objects is high and low, MADDQN finds users to serve to minimize the possible delay which increases the resulting

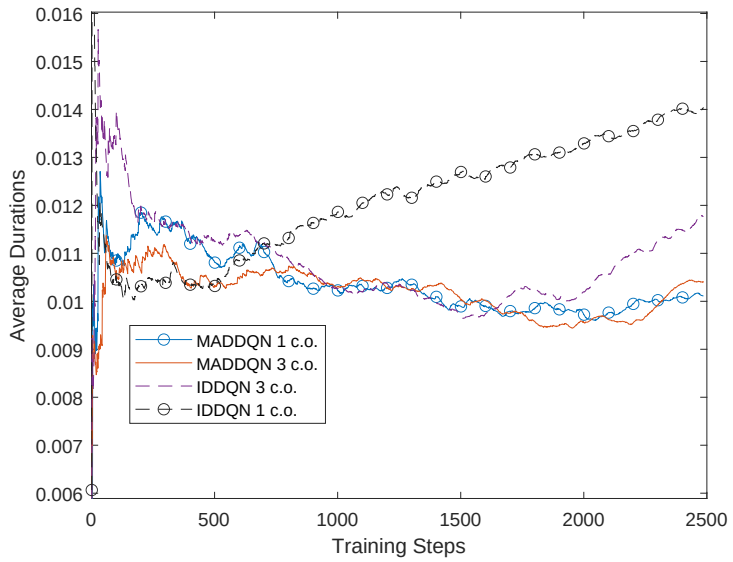


FIGURE 4.9: Delay performance for 1 and 3 cacheable objects (c.o.).

reward. IDDQN fails to find similar user sets. When comparing the average durations, IDDQN is 2*ms* behind MADDQN on convergence and when the number of cached objects is low, IDDQN performance is drastically decreased. As the number of cached files increases, the number of good solutions becomes closely linked to the channel quality experienced by the UEs. However, the delay will also be reduced as it is completely dependent on the high-speed wireless transmission links and is not bottlenecked by the Xhaul processing. Consequently, it becomes harder to find optimal UE combinations that would provide the smallest delay. How this principle affects different algorithms is shown in Figure 4.9. The delay is related to the number of UEs served, as an increase in the size of the served UE set means that the power (and achieved rate) is split between the served UEs, consequently increasing the delay. As shown for greedy algorithms (cache priority and sum-rate priority in the figure), when the number of UEs in the system is large, the delay increases. Just as for power consumption analysis it is preferred that if the number of served UEs is large, the delay should be large as well, as it shows how well the algorithms use the allocated thresholds.

4.7 Summary

In this chapter, the F-RAN system was presented as a POMDP environment, which has stochastic outcomes based on agent decisions. It has been shown that low-complexity greedy-type mechanisms that would typically be used to save computation and transmission overheads do not perform well, as they result in optimizing local greedy solutions. To improve the system performance, novel method called MA-DDQN was proposed to reduce training times, improve learning stability and improve the reward received on convergence. It was showed that the method performs approximately 50% better than greedy algorithms and 22% better the DDQN algorithm, when deployed in the same environment. The algorithms were simulated in a network with a small number of antennas and a comparably large number of UEs, a varying number of cached objects and by altering the frequency of target network updates to measure the stability of the proposed algorithm. The simulation results have validated that the proposed method performs better than the DDQN algorithm. The reward at the end of training was improved and following the results of training losses it was presented that MADDQN is significantly more stable during training. MADDQN outperforms DDQN in all metrics, i.e. by serving $25\% - 50\%$ more users, consuming 2 times less average power and reducing user experienced delay by $10\% - 30\%$.

Chapter 5

Distributed Cache Access in Fog Radio Access Networks

5.1 Introduction

With autonomous driving, smart cities, cloud-based gaming and beyond 4K video coming into fruition, network requirements are increasing at a rapid rate, which is proving challenging for the current generation of mobile network architecture. With every household increasing the number of connected devices, the amount of data being generated is vast and the amount of data consumed has increased by 300 times compared to 2011. In an Ericsson Mobility Report, released in 2021, it is forecast that global mobile network traffic will further increase from 59 exabytes per month, consumed currently, up to 370 exabytes **emob**. With video-rich application traffic being the highest compared to voice calls and text traffic, it is clear that the network architecture has to be sufficiently equipped to provide for the increasing user demand. With this in mind, a cell-free hybrid distributed and centralised architecture, fog radio access network (F-RAN), with caches and computing being available at the edge has been introduced [44]. With a large number of communications needed to make F-RANs work, a latency constraint must be satisfied before they can be brought into operation, which makes F-RAN an interesting and necessary subject to research.

F-RAN is proposed as a network evolution to a centralised architecture, aiming to reduce the overhead and load on front-haul links and provide low latency [12] by evolving the RRHs in C-RAN with a novel component called a fog access point (FAP). FAPs have baseband processing capabilities and are equipped with caches, with this evolution F-RANs allow for network control and storage to be distributively performed. With distributed baseband processing, cooperative radio resource management and cooperative radio signal processing, similar to [36], the architecture enables low-latency content access at the edge. Cloud layer is also present in F-RAN, and can be used when content is unavailable to be accessed in fog layer. As most of the processing and caching is done within the fog layer, channel information does not need to be transmitted over long front-haul links, which would introduce latency. Therefore, F-RAN architecture can improve channel accuracy, resulting in high transmission rates and low latency. However, the unique challenges caused by cell-free resource management and content acquisition capabilities enabled at the edge, cooperative methods have to be developed among FAPs. Without cooperative algorithms, strong interference can be introduced due to the overlapping coverage areas of different FAPs. Therefore, joint cache allocation and cache content control is essential to manage available resources efficiently.

Cache management was addressed in [14] using a classical tabular Q-learning reinforcement learning method. The paper used content popularity to populate the caches with the most popular contents and improved the cache hit probability for a random association and random requests of users. Cache-focused transmission, otherwise known as cache selection was studied in [28]. Using a hard-coded two-step optimisation algorithm the authors target a low-latency requirement and prioritises FAPs that have cached contents available and uses joint transmission to maximise the possible file delivery rate. While optimizing these problems individually has shown good results, it is difficult to judge the optimality of the solutions, as both solutions' optimality depends on the performance of

the other algorithm. A joint solution has been attempted in [29] using a tabular reinforcement learning method to select which services to cache at the edge by evaluating possible rewards using an upper confidence bound function and using a low complexity method to guide requests to adjacent edge computing nodes or the cloud. A few shortcomings exist in previous literature, such as not taking into account that memory is utilized during run-time, which could lead to edge nodes being unable to execute a task if too many users request the same service. Additionally, the reinforcement learning work [29] does not consider the state of the adjacent edge nodes prior to off-loading, which could result in reduced quality of service. Considering the extra information leads to a complex problem, for which joint optimisation of cache content delivery and cache content management could be solved with a deep reinforcement learning method.

Since cache optimisation involves key performance metrics such as power consumption, fronthaul load and total data delivery rate, it becomes a high dimensionality problem. Since high dimensionality problems have sparse data sets it can be difficult for low-complexity algorithms to solve the problems through generalisation. Deep learning has already been used for complex, high dimensionality tasks. Sharing information between agents has been investigated in some work. In one work agents agents had different roles in the same environment, using communication to increase the number of observations available to each other. Experience sharing was possible in A3C [20]. It was an algorithm that simultaneously performs policy optimisation as well as policy evaluation, through the actor and critic respectively. In A3C, experiences are shared among homogeneous agents interacting with an instance of the same environment where they do not interact with each other, which is identical to single-agent learning. Previous reserach studied how multiple agents can learn cooperatively to improve learning speeds. However the results from the mentioned papers are only applicable to fully observable or complete Markov decision process (MDP) environments, where agents are deployed and interact with separate instances of

the same environment. If agents exist within each others' environment, they require a lot of communication prior to making a decision. However, in F-RAN architecture, FAPs can only partially observe the environment, leading to a partially observable MDP (POMDP), as exchanging observation information to form a joint observation and joint action could lead to significant overheads. Some work has been done in the area of MARL in POMDP, mainly however the paper used a method similar to hyper-Q-Learning, in which agents learn about the policies of the other agents and does not focus on the area of experience sharing or exchange.

This chapter describes a solution to a joint cache selection and cache management problem in F-RAN using a focused batch distributed deep reinforcement learning (DRL) method. The method optimises the utilisation of caches by acquiring content from the cloud and allocates users to FAPs to maximize the cache hit ratio. The main contributions of this chapter are as follows:

1. For F-RAN, which is a partially observable MDP a distributed DRL method is developed with focused experience exchange. Using a focused experience exchange method it is shown that in a multi-agent DRL, convergence can be guaranteed with a decrease in the size of information overhead consumed when learning. It is shown that the method outperforms conventional DRL methods that use gradient exchange or naive experience exchange in terms of communications efficiency.

2. The proposed method is applied as a novel two-step optimisation method which manages the cache placement in the first step and allocates users to F-APs providing the desired content in the second step. The performance of the proposed algorithm is compared to distributed methods that provide near optimal performance without consideration of the joint optimisation.

The further structure of the chapter is as follows, in Section 5.2, the system model is introduced. In the system model, both the transmission model and

caching model are explained and the problem to be solved is formulated. In Section 5.3 the problem solving algorithm, focused batch double deep Q-network (FB-DDQN), is presented. The problem solving algorithm must be deployed in an MDP, which is described in Section 5.4. The results from simulations are presented in 5.5. Finally, the chapter is summarised in 5.6.

5.2 System Model

5.2.1 Ricean Fading Transmission

Start with an F-RAN environment, with a set of F-APs $\mathcal{J}$, of size $|\mathcal{J}| = J$. Each FAP is equipped with M antennas, all of which are used for transmitting and F-APs are connected to the cloud network via fronthaul. The FAPs are also interconnected with each other with backhaul [36] links that have a lower capacity than the fronthaul. In F-RAN the network access layer needs connectivity between FAPs to be able to exchange information required for CRRM. With this in mind, backhaul links are utilized to share network information with low latency. Thanks to CRRM, the FAPs can connect to multiple users by using coordinated beamforming (CB) and joint transmission (JT) techniques [36]. As more resource intensive functionality is handled by the cloud network, it can be assumed that the cloud is responsible for FAP synchronisation and as such, no additional interference is incurred from phase inaccuracies at the FAPs. A downlink scenario is studied with a set of users $\mathcal{K}$, waiting to be allocated resources, of size $|\mathcal{K}| = K$. Each user is equipped with a single antenna. Since each FAP can associate with any number of users through multiple beams, and user $k \in \mathcal{K}$ to FAP $j \in \mathcal{J}$ association factor is given as a variable $x_{j,k}$:

$$x_{j,k} = \begin{cases} 1, & \text{if } k \text{ is associated,} \\ 0, & \text{if not.} \end{cases} \tag{5.1}$$

the sum of associations $\Sigma_k x_{j,k}, \forall j \in \mathcal{J}$ is the cardinality of set $\mathcal{K}^{(j)}$ that includes all the users associated with FAP j. The union of all association sets produces a set of users associated with the fog and is shown as $\bigcup_{j=1}^{J} \mathcal{K}^{(j)} = \mathcal{K}^{(f)}$.

Note, that even if the interest lies in associations within fog, with global synchronisation between FAPs assumed some information may not be known by the rest of the FAPs. Assuming CoMP and imperfect channel estimation, the data stream would be jointly transmitted to the user in multiple streams of message signals s_k, the main source of interference comes from the inter-stream interference of F-APs themselves. As such the received symbol at user k is:

$$y_k = \sum_j x_{j,k} \hat{\mathbf{H}}_{j,k}^{\mathrm{H}} \mathbf{F}_{j,k} s_k + \sum_j \sum_{i \neq k} x_{j,i} \hat{\mathbf{H}}_{j,k}^{\mathrm{H}} \mathbf{F}_{j,i} s_i + n, \tag{5.2}$$

where n is the additive white gaussian noise (AWGN), satisfying $\mathcal{N}(0, \sigma_n^2)$. $\hat{\mathbf{H}}_{j,k}$ is the $\mathbb{C}^{M \times 1}$ estimated channel response that follows a Rician channel model with U paths between an FAP and user. Since users are mobile, signals transmitted by the users undergo Doppler fading, and have coherence time t_d. $\hat{\mathbf{H}}_{j,k}$ is composed of M channel responses, and is denoted as

$$\hat{\mathbf{H}}_{j,k} = \begin{bmatrix} \hat{h}_{j,k,1} & \hat{h}_{j,k,2} & \cdots & \hat{h}_{j,k,M} \end{bmatrix}^{\mathrm{T}} \tag{5.3}$$

where each entry $\hat{h}_{j,k,m}$ represents a single user's k channel response to the antenna m of AP j. With imperfect channel knowledge, the channel response that is known at the FAP j is also affected by error and is split into the true channel response $\mathbf{H}_{j,k} = \begin{bmatrix} h_{j,k,1} & h_{j,k,2} & \cdots & h_{j,k,M} \end{bmatrix}$, where each $h_{j,k,m} = \mathrm{Rice}(a, \sqrt{\sigma^2/2})$, where $n \geq 0$ is the magnitude of direct path, and $\tilde{\mathbf{H}}_{j,k}$, the associated channel error, which has a certainty property $\rho = \mathrm{J}_0(2\pi f_d \tau)$, where τ is the time passed since channel estimation $\tau << \tau_d$ and f_d is the respective Doppler spread. The channel certainty tells us how much of the estimated channel is true and how

much is affected by the channel error, as:

$$\hat{\mathbf{H}}_{j,k} = \rho \mathbf{H}_{j,k} + \tilde{\mathbf{H}}_{j,k} + \mathbf{J} \tag{5.4}$$

the channel error $\tilde{\mathbf{H}}_{j,k}$ is of the same dimension as $\mathbf{H}_{j,k}$ and follows a zero mean complex Gaussian distribution $\mathcal{CN}(0, \sigma_\epsilon^2)$ and each element has variance given by $\sigma_\epsilon^2 = \sigma_{\mathbf{H}}^2(1 - |\rho|^2)$, $\eta \in \mathcal{CN}(0, \sigma_n^2)$ is additional complex Gaussian noise during estimation. Since the channel information is not perfect or fully accurate, Zero Forcing beamforming weights are calculated for the estimated channel as:

$$\mathbf{F}_{j,k}(\rho) = (\hat{\mathbf{H}}_{j,k}^{\mathrm{H}} \hat{\mathbf{H}}_{j,k})^{-1} \hat{\mathbf{H}}_{j,k} \tag{5.5}$$

FAPs have limited instantaneous transmit power, denoted P_j^{TRI}, which is calculated as a sum of all beamforming vectors allocated to the users:

$$P_j^{TRI} = \sum_k P_{j,k}^{TRI} = \sum_k x_{j,k} |\mathbf{F}_{j,k}(\rho)^2| \tag{5.6}$$

The resulting SINR experienced by the user is calculated and fed back as follows:

$$\gamma_k(\rho) = \frac{\sum_j x_{j,k} |\hat{\mathbf{H}}_{j,k}^{\mathrm{H}} \mathbf{F}_{j,k}(\rho)|}{\sum_j \sum_{i \neq k} x_{j,i} |\hat{\mathbf{H}}_{j,k}^{\mathrm{H}} \mathbf{F}_{j,i}(\rho)|^2 + \sigma_n^2}, \tag{5.7}$$

from this the resulting rate that can be experienced by the user is denoted R_k and is derived as:

$$R_k(\rho) = B \log_2(1 + \gamma_k(\rho)). \tag{5.8}$$

Where B is the communications bandwidth.

5.2.2 Dynamic Service Caching Model

Service applications with an index l from a set of available applications $\mathcal{L} = \{1, 2, \cdots, L\}$ can be cached within an FAP. $z_{j,l}$ is the service placement indicator that is decided as

$$z_{j,l} = \begin{cases} 1, & \text{if } l \text{ is present in cache j,} \\ 0, & \text{if not.} \end{cases} \tag{5.9}$$

Each service has an application size, I/O size, memory and computing requirements associated with it, having units of gigabytes (GB) for application, I/O and memory sizes, and giga-instructions (Gi) for computing, respectively. Requirements can be represented as a requirement tuple $r_l = \{D_l^{app}, D_l^{io}, M_l, C_l\}$. In an FAP with limited cache size D_j services can be cached so long as $\sum_{l,j} z_{j,l} D_l^{app} < D_j$. Service applications are assigned popularity p_l which represents the probability of the service being requested, which follows a Zipf distribution [119] that is unknown to the FAP, an example request distribution is shown in Figure 5.1. Based on caching decisions made by the FAP, cache hit-rate χ_l is a metric that tracks how well can the fog radio layer accommodate service requirements, particularly for low-latency services.

When user k makes a request for a service l, the request is received by the closest FAP j. Based on factors like M_l, C_l, S_j^l and $\left|K^{(j)}\right|$, the FAP can make a decision to either process the requested service locally at the FAP j, off-load it to the other FAPs $v \neq j$ or off-load the request to the cloud. If the request off-loading is successful, the user can be appended to the list of users served by another FAP, which reduces the resources that can be used in another FAP.

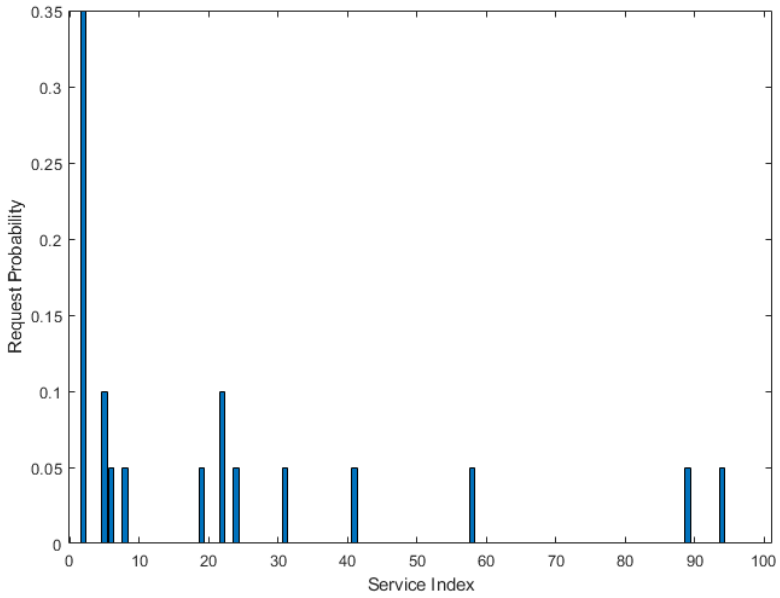


FIGURE 5.1: Recorded probabilities of each service request

5.2.3 Problem Formulation

Given the system model, it is possible to split the problem into two parts which can be individually optimized. First, optimizing the placement of service applications within FAP caches after initially having a random placement of applications. Second, optimizing the service orchestration given random requests from users. To solve these problems, the objective is to maximize cache hit rate χ within the fog, by populating caches with popular contents. The objective is shown in (5.10).

$$\max \chi = \frac{\sum_{l=0}^{L} \chi_l}{K}. \tag{5.10}$$

This problem is most commonly seen as a capacitated facility location problem (CFLP), due to a set of candidate FAPs and a set of clients for any service. The capacity can be perceived as limited resources at each FAP. Establishing facilities are similar to caching services, the cost of serving a client is the service response time. The objective being caching services (opening facilities), given their constraints and minimize serving costs. Since CFLP is proven NP-hard, this applies

to this problem as well.

5.3 Focused Batch Double Deep Q-Network

Algorithm 2 Focused Batch multi-agent DDQN

Initialise environment with $T = K$ UEs, J FAPs with M antennas, J agents $\beta_J = [\beta_1, \beta_2, \cdots, \beta_J]$, deviation σ_e, and J empty experience replay buffers $\mathcal{E}_j = \emptyset \quad \forall j$
for t < T **do**
  for all agent j **do**
    Agent receives s_j^t and performs action a_j^t
    Agent receives reward r^t, stores result in $E_j^t = \left\{s_j^t, a_j^t, r^t, s_j^{t+1}\right\}$ and updates $\beta_j \leftarrow$ (4.14)
    Agent appends $\mathcal{E}_j \cup E_j^t$
    if t mod $T_{\text{com}} = 0$ **then**
      Choose and transmit experiences 5.12
      if Received $\mathcal{E}_s \neq \emptyset$ **then**
        Collect and transmit experiences $\mathcal{E}_e = \left(\mathcal{E}_j : |s_j - s_s| > s_j \cdot \sigma_e\right)$
      end if
      if Received $\mathcal{E}_e \neq \emptyset$ **then**
        Update $\beta_j \leftarrow \mathbb{E}_{\mathcal{E}_e}\left[\left(r^t + \rho Q\left(s_j^{t+1}, \arg\max_a Q(s_j^{t+1}, a|\beta)|\beta'\right) - Q(s_j^t, a_j^t|\beta)\right)^2\right]$
      end if
    end if
  end for
end for

In this section, the algorithm that makes distributed DRL experience exchange communication efficient is described. When learning in DDQN, agents use experience replay buffers to reduce temporal correlation of samples when learning and thus enforce reduced variance when learning about possible solutions. When an experience replay buffer is used the loss function is modified to include the mini-batches that are sampled every time step following any probability distribution. If a uniform distribution is used, the loss function that is used to update the neural networks are calculated by

$$\mathcal{L}(\beta) = \mathbb{E}_{\mathcal{U}\sim(\mathcal{E})}\left[\left(r^t + \rho Q\left(s_j^{t+1}, \arg\max_a Q(s_j^{t+1}, a|\beta)|\beta'\right) - Q(s_j^t, a_j^t|\beta)\right)^2\right]. \tag{5.11}$$

This function is passed into an Adam optimizer to minimize the loss and the resulting gradient update is applied to the neural network. As agents are learning, in the same time step, the temporal difference (TD) error $e^t = Q_t(s_j^{t+1}, a_j^{t+1}|\beta') - Q(s_1^t, a_1^t|\beta)$ of the experiences can be extracted and stored alongside the sampled experiences, as $E_t = \left(s_t, a_t, r, s_{t+1}, e^t\right)$. The fact that the system is always aware of how much knowledge the neural network has about certain states can be used here. In algorithm 2 the FBDDQN algorithm is presented. The algorithm uses current maximum TD error, and this is used together with a globally known given deviation σ_e to collect a number of experiences from the local experience buffer. This algorithm is made to be communication efficient, by reducing the amount of communication data. This is done by introducing a hyperparameter - a number of time steps T_{com} after which to exchange experiences. The experiences are collected by comparing all of the experience TD errors to the current maximum TD error, this is expressed by

$$\mathcal{E}_s = \left(\mathcal{E}_j : (\max e_{\text{TD}} - e_{\text{TD}}) \,/\, \max e_{\text{TD}} > \sigma_e\right) \tag{5.12}$$

If the TD errors are within a sufficient range they get selected as sample experiences within a set $\mathcal{E}_s$. This set is sent out as a request to nearby agents in the fog and using the same σ_e other agents collect experiences in a set $\mathcal{E}_e$ that would be similar to the the received samples and transmit them back to the requesting agent. Upon receipt of the experiences, rather than storing them in the experience

buffer, they get immediately sampled and used to generate an additional loss gradient to update the neural networks and then discarded to prevent agents from confusing external environment experiences with their own when using classical experience replay. If the deviation σ_e stays the same while learning, the issue of oversampling becomes significant. As agents learn, TD errors decrease as well as more similar experiences are present, which results in more experiences within the same range from the max TD error. It is proposed to reduce the range σ_e with the value of *epsilon* that is used to determine the greediness of the system, which will result in sample communication converging.

5.4 Environment Design

The environment in which the algorithm is deployed is presented as a process split into two stages, cache population and service orchestration. In the first stage, that runs every T timesteps, agents are responsible for populating caches. In this stage, agents receive information about L most popular service applications and choose which to download into caches. Service popularity p_l, storage requirement D_l, and cache hitrate χ_l are the most important values for this decision. The state in this scenario is $s_t^c = \{p_1, \chi_1, D_1, \cdots, p_L, \chi_L, D_L\}$. Processed by a neural network, this agent produces an action $a_t^c = \{\emptyset, \{1\}, \{2\}, \{1,2\} \cdots, \{1,2,\cdots, L\}\}$, which decides which service applications should be cached in the current agent's FAP.

In the second stage, which is the service orchestration stage, agents receive a number of requests from K users. Each request comes in the shape of a service index l and a payload of D_l^{io}GB. Assuming that each FAP stores information on all possible services, information such as M_l and C_l are also available to the agent. Additionally, the agent wishes to prevent congestion at downlink transmission, as such, it has to monitor the number of currently associated users $|\mathcal{K}^{(j)}|$. Through network monitoring, and periodic updates, the core network can provide other

information as well, such as the memory utilisation of other fog nodes which store this service, $M_v, v \neq j$ as well as the CPU utilisation, C_v. Using this information, a state for a service request is given as $s_t^s = \{|\mathcal{K}^{(j)}|, M_l, C_l, M_j, C_j, M_v, C_v\}$. The processed action that this agent produces is $a_t^s = \{0, 1, 2\}$, where the indicators are

- 0 - Process locally at the current FAP.
- 1 - Process within other FAPs.
- 2 - Process in the cloud.

Where cloud processing would only be used as a fallback for very demanding tasks due to the additional delay that is introduced when performing cloud computing.

The reward function is formulated following the objective with an addition of rewarding the agent for also serving more users in the fog layer, i.e.

$$r = \begin{cases} \chi + |\mathcal{K}^{(f)}|, & \text{if } |\mathcal{K}^{(f)}| > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.13}$$

This function rewards the agent for correctly predicting which services to cache as well as how to orchestrate the delivery of these services. The second case is used to ensure that the agents attempt to allocate some users to the fog layer regardless if they have services that satisfy the requirements, as they can still be served through the connection to the cloud layer. Rewards are granted to the agents after every step of the second, user service, stage, while rewards for the cache population stage are granted once the user service episode is over. Using this information, agents can be trained in a simulated environment.

5.5 Simulation Results

The purpose of this section is to show the performance of the proposed FB-DDQN algorithm in comparison to other methods of learning, namely naive exchange that would exchange every experience as it is being experienced by the agents and gradient data exchange algorithm which would exchange update gradients in order to reach convergence. The environment that the algorithms are deployed in are identical. That is, the numbers of FAPs, sizes of caches and UEs have not changed, the simulation parameters and hyperparameters are the same. These parameters are shown in table 5.2. Services that can be requested are split into 4 different categories and the categories and their normalised requirements are shown in Table 5.1. Type 1 would be representative of a lightweight tool, such as calculators or QR Code readers or text-based applications, such as news applications. Type 2 application is most close to productivity tools, such as calendar apps, while storing a calendar on a network cache would not be safe or private, the network could cache data relevant to personalisation or important events in the area. More resource intensive software, like online or cloud-based gaming is a type 3 application. Finally the type 4 application is a standard application with medium requirements all around, media players, browsers or file compression tools that could be deployed in a network.

TABLE 5.1: Normalized application requirements

	Type 1	Type 2	Type 3	Type 4
Application Size, D_l^{app}	0.2	0.4	0.3	0.2
I/O size, D_l^{io}	0.1	0.2	0.3	0.3
Memory, M_l	1.0	1.0	2.0	0.4
Compute, C_l	0.4	0.14	2.0	0.6

Simulation results of the amount of data sent over the network are shown in Figure 5.2. Data is presented for ranges of training steps. It can be seen in the figure that a naive experience exchange algorithm (MADDQN) shares the same amount of data during the training period, as the amount of data only depends on

TABLE 5.2: Simulation parameters and hyperparameters

	Value
Discount rate ρ	0.99
Learning rate μ	0.0001
Batch Size $S_{\mathcal{E}}$	512
Experience Replay Buffer size	25000
Pre-Train Period T_{pretrain}	512
Target network update period τ	1500
Greedy action probability ϵ	[0.99:0.05]
Deviation σ_e	$[0.80\epsilon, 0.95\epsilon]$
Communication Period T_{com}	500
Number of UEs K	20
Number of FAPs J	3
Number of antennas M	16

how many agents are learning with MADDQN. For the proposed algorithm, the amount of data being sent depends on four factors: what states the requesting agent is unfamiliar with, as shown by the TD-error, how close the worst TD-errors are, what other states are similar inside its own experience replay buffer, determined by the σ parameter and finally what experiences are available in the other agents' experience replay buffers. For this reason, during the later stages of training as all agents fill up their experience replay buffers, and as agents get more familiar with the environment, leading to TD-errors being closer together, the amount of experiences exchanged increases by a large amount. It is expected that agents learning under $\sigma = 0.80$ will exchange more experiences due to a wider spread of errors. At the end of training, the same agents exchange less experiences, which is caused by either of the 4 factors being less satisfied.

The proposed algorithm has to learn the reward function of the environment designed in Subsection 5.4. This is done by training the different variations of the algorithm for an environment with the same seed, which ensures repeatability. How experience exchange impacts learning is shown in Figure 5.3. A comparison between MADDQN, FBDDQN as well as conventional reinforcement learning DCC-MAB algorithm is shown. The DCC-MAB algorithm uses a function approximation method to learn about the environment, by assigning values to

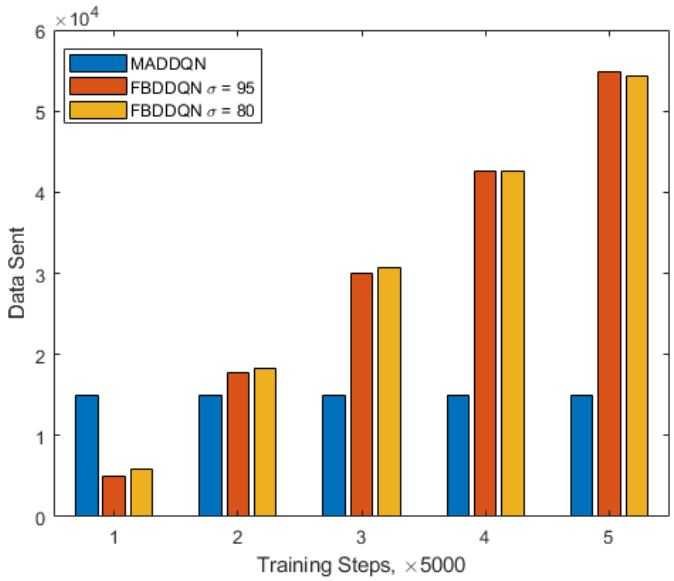


FIGURE 5.2: Number of batches sent over the network

different services to be allocated as well as users to be served. The goal of the algorithm is to improve the cache hit-rate by using a two step method, so it is valuable for comparison. The proposed algorithm performs the best when it learns from experiences that are more focused around the unfamiliar areas of the state space, i.e. when $\sigma = 0.95$. When the experiences used for learning are spread out over a larger deviation $\sigma = 0.80$ the performance degrades to 60% of the best performance. It can also be seen that in this environment exchanging experiences naively leads to a significant loss in performance, as MADDQN performs the worst out of all tested algorithms. From the figure, a conclusion can be drawn that out of all algorithms under test, FBDDQN with a narrow focus on the unfamiliar state space learns how to satisfy the reward function the best.

To provide a more practical look at the performance of the algorithms the cache hit rate and number of users served are shown in Figures 5.4 and 5.5 respectively. The sum of both statistics make up the reward function. However, since cache hit rate can be significantly larger than the number of users in the environment, the algorithms might prefer maximising cache hit rate instead of the number of users. This is clearly visible in the performance of MADDQN algorithm, which is outperforming FB-DDQN $\sigma = .80$ in terms of rewards, as can

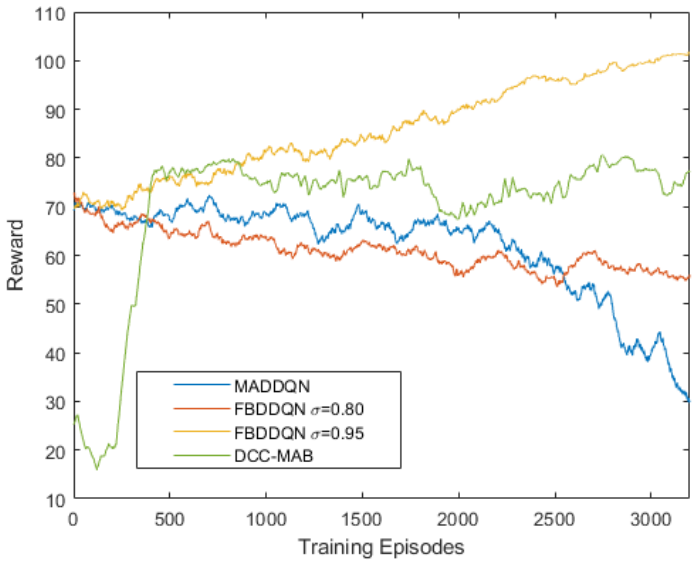


FIGURE 5.3: Reward comparison between FB-DDQN, MA-DDQN and DCC-MAB algorithms

be seen from Figure 5.3. This is mostly due to its higher cache hit rate shown in Figure 5.4. However, the latter algorithm outperforms the former by the number of users served, as can be seen from Figure 5.5. The DCC-MAB algorithm outperforms both in both objectives at the end of training. However, it is equal in cache hit rate performance to MADDQN algorithm between 500 and 2000 episodes. At the end of training when all algorithms rely on the learned solution instead of the exploratory actions, the proposed algorithm with a narrow spread of training experiences outperforms all algorithms under test. The approach manages to serve as many users as the DCC-MAB algorithm and finds a solution for service placement in the caches to maximise the cache hit rate. Other approaches have to locate compromises, which is the case for DCC-MAB, as it has a high number of users served, but does not have an equally high cache hit rate. On the worse end of the spectrum, approaches can also fail to converge to a solution, such as the MADDQN algorithm.

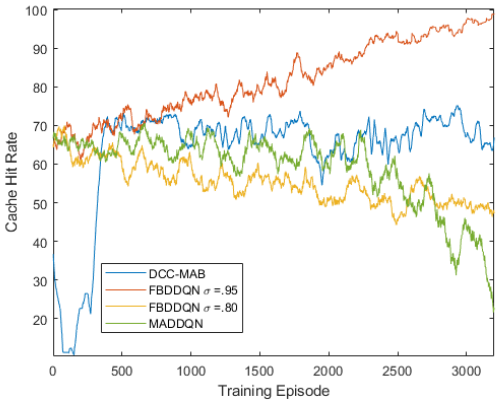


FIGURE 5.4: Cache hit-rate for algorithms under test

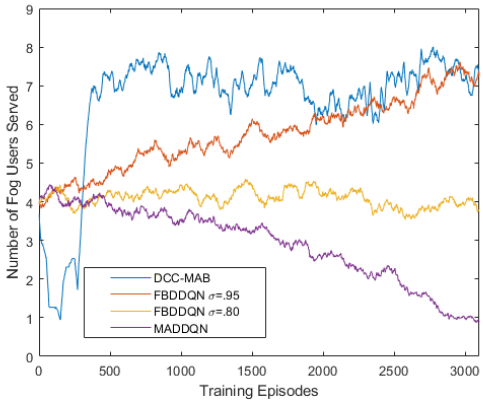


FIGURE 5.5: Number of users served by the fog layer

5.6 Summary

In this chapter, an approach that aims to improve the performance of a multi agent DDQN algorithm was proposed. The proposed solution selects states that the agent considers to be the worst performers and makes requests for similar states from other agents in the network. The performance of this method is demonstrated in a simulated F-RAN service orchestration environment. The simulation considered a Zipf distribution for user service requests that the algorithms had to learn to accurately predict which services should be cached in the fog layer. Experimental results showed that focusing on a smaller range of unfamiliar states leads to significantly better outcomes, the rewards, which are a combination of the cache hit rates and number of users served, were improved from 55 at the end of training to slightly above 100. The contributions of each metric have been showcased. The algorithm with a broader focus on the states was predicting cache placements with 50% accuracy and had a stable number of around 4 users served. The algorithm focused on a smaller range of states has learned how to accurately place the services and how to choose users to serve. Compared to DCC-MAB, a RL algorithm that is developed specifically for this task, the proposed algorithm receives better rewards at the end of training. Practical results showed that this is due to the agent choosing to serve an equal number of users

that the DCC-MAB algorithm chooses, but the cache placement strategy that the proposed algorithm converges to is superior, as it leads to a 40% higher cache hit rate.

Chapter 6

Conclusions and Future Work

6.1 Introduction

In this chapter, the conclusions and key findings of previous chapters are summarised. The key findings are discussed in relation to the research aims. Finally the limitations of the current studies are presented and future avenues for research are proposed.

6.2 Conclusions

The main motivation of the thesis was the evolving specification of the mobile network infrastructure. While networks are perfectly capable of processing voice and audio/video media, they will not be able to keep up with the bandwidth requirements of virtual, augmented or mixed reality applications. Data consumption is expected to increase from 146EB to 452EB by the end of 2028, although an even larger increase is likely with more novel services. Primarily, a centralised RAN was proposed to deal with the exponential increase in data and latency requirements, however with virtualisation cloud RAN can be used. However, both are limited by the fronthaul links that are used to communicate with radio units in the field. If the core network is expected to provide both high speed signal processing as well as support some of the computing for user applications, the fronthaul load is a significant issue. To tackle novel low-latency applications F-RAN

is proposed, as it can provide low-latency radio signal processing and cooperative radio resource management, as well as support services and cache data at the edge, close to the user, while the core network can perform synchronisation and high intensity tasks. F-RAN has many cooperating nodes and this raises a novel challenge to tackle before this network architecture can show its full benefits.

With this motivation in mind the thesis looked at machine learning methods that could be applied to support cloud RAN and F-RAN applications. In literature review it was established that supervised learning is limited by what it can achieve, based on the base solution algorithm it samples from. Unsupervised learning is promising, but the applications for mobile networks have been shown to be of narrow scope, as one of the best applications is tracking anomalies of when a network is under-performing to signal to experts that further analysis may be needed. Reinforcement learning was chosen as the area of research that could prove to be the most useful in F-RAN, as these algorithms can learn from and adapt to the environment they are deployed in. With this learning paradigm in mind, three chapters of the thesis focus on the applications of reinforcement learning and they are summarised in the following paragraphs.

The thesis looked at the problem of latency management in RAN Xhaul networks. CoMP is proposed as a way for RUs to mitigate interference caused by beamforming and improving the signal quality, which in return improves the data throughput. To perform CoMP the network has to collect the necessary channel information from multiple RUs to coordinate them. Channel information is latency dependent, high latency leads to low channel accuracy, resulting in reduced throughput of the network. The thesis proposes to solve the problem in two ways – FlexE slot allocation and user offloading. By accurately allocating FlexE slots, the network's latency effects caused by serialising and buffering delays can be reduced. By performing user offloading between RU groups, traffic on heavily congested network paths can be alleviated.

DRL was used to predict traffic with the objective to perform slot allocation

for FlexE links in the centralised RAN Xhaul. In simulation results it was seen that DRL outperformed statistics based ARIMA method with 11.6% less over-allocations. The data was consistent in under-allocations as well, as the DRL method performed 25% less under-allocations. It was also shown that when deployed simultaneously with another DRL algorithm latency effects have been avoided by controlling which RU groups handled which users when RU groups were affected by varying loads. Specifically, when considering the same background traffic, the offloading agent learned a policy which helped it determine the correct RU-group allocations to improve the data rate by 7.3%.

As the network requirements evolve, so does the network architecture. F-RAN has been proposed as an evolution of C-RAN architecture, which reduces the latency and improves fronthaul link efficiency by replacing RRHs with network components capable of caching and processing, called FAPs. In this architecture time-sensitive operations, such as CRRM and cell-free MIMO processing are performed at the network edge, while the cloud layer is kept for resource-intensive tasks. With caching and processing pushed to the edge, transmission of channel information to the cloud via fronthaul links is reduced. Evidently, content and network can be coordinated at the fog network access layer, improving channel accuracy and lowering latency. However, due to the mixed distribution and centralisation, F-RAN evolution faces unique challenges for implementation.

The thesis aims to solve the problem of user association to FAPs to satisfy users' content requirements. F-RAN is studied as a POMDP environment due to latency constraints which would lead to a lack of global state information being available to the agents. The networks collected information such as the number of hops to acquire content requested by the user, the index of UE being considered, the channel strength between the UE and FAP, the expected data rate for the user and the predicted delay. Since the environment is partially observable, the states and outcomes perceived by the agents are stochastic. The common method for handling partially observable data is to introduce memory into the

agents. Instead the thesis proposed a novel MA-DDQN algorithm, which uses experience exchange to increase the amount of knowledge agents collect every training step. The proposed method performs better than greedy algorithms by achieving 50% better rewards. The performance is better than the conventional DDQN algorithm as well, by both serving a $25\% - 50\%$ larger number of users and improving the stability of learning as much as three times under unstable learning conditions.

Since FAPs are capable of executing services to reduce network latency for applications or web-requests, the thesis looked into a joint optimisation of cache content delivery and cache content management problem. This problem can be formulated as a CFLP, which is NP-hard. The thesis identified a few shortcomings in previous literature, such as not considering the memory or processing being utilized during run-time, which could lead to FAPs being unable to process requests from users. In addition to this, reinforcement learning research did not consider the state of adjacent edge nodes prior to off-loading users for service, which could also result in reduced quality of service. It was acknowledged that cache optimisation involves performance metrics such as power consumption, fronthaul load and total data delivery rate, which makes it a high dimensionality problem. It can be difficult for low-complexity algorithms to solve the problems through generalisation.

To solve the joint optimisation problem, a focused batch DDQN (FB-DDQN) algorithm is proposed. The solution used TD-error as a statistic to determine which states the agent is the least familiar with and focuses on learning about similar states. Users in the simulated environment made requests based on a Zipf distribution that algorithms had to learn, which would help them make accurate decisions. The proposed algorithm was shown to outperform an already known RL solution for this problem, DCC-MAB. At the end of training, the proposed algorithm reached a solution that was approximately 30% better. It has also been shown that the parameter responsible for choosing the states to learn about and

from can significantly impact the performance of FB-DDQN, focusing on a larger number of states reduced the performance by as much as 50%.

6.3 Future Work

In chapter 3 the thesis introduces DRL as a possible solution to mitigate latency effects by incorporating DRL as a decision maker for FlexE slot allocation and user off-loading. The thesis showed that network performance improves as the agents learn about the environment, however the environment could be reformulated as a POMDP for FlexE agents. By assuming that other agents' decisions impact the decision of a single FlexE agent, it would introduce additional complexity into the environment. Research of using memory-based RL algorithms has a high chance of improving the network performance further. Recurrent neural networks (RNNs) have previously been used for solving POMDPs. This is due to the fact that RNNs consider both the current state and action as well as the historical data.

Due to the software and compute limitations it was difficult to scale the model, especially the number of UEs or RUs and certain limitations led to some artifacts being introduced into simulations. By abstracting the model it would be possible to extract more accurate results for the off-loading controller agent. In addition to this, it would make the research more easy to scale. It is known that the number of RUs can easily change in a C-RAN solution, as more RUs are installed by the network provider or activated by the BB pool. Further research into how deep learning agents can support decision making for a scaling number of RUs by using a more scaleable DRL method, such as policy gradient could produce significant results.

In the following chapter, the thesis introduces the F-RAN environment as a POMDP and uses MA-DDQN to solve the problem of user association with FAPs. The chapter considers that the network is capable of making decisions based on

user requests. While that theory still holds, a more traditional approach is to look at user-centric approaches for such decisions. Indeed, UEs are capable of processing deep learning algorithms. Equally, from the UEs' perspective the problem is a POMDP, a single UE is not aware of the decisions made by the nearby UEs. In F-RAN, due to the capabilities present in the FAP protocol stack, UEs could also switch modes into supporting the network. Future work could consider this mode selection by UEs to support traffic or content requirements. Due to the POMDP criteria, an algorithm could consider a belief vector to approximate the state of the network and possible best solution.

In the same chapter, the method uses a uniform probability distribution to sample all experiences that an agent receives. It can be argued that this is not the most efficient way of handling experiences received from other agents. A probability distribution function that creates an optimal strategy for sampling experiences in such environments could be developed. To study the impact of such a strategy, the environment should be repeatable and could take into consideration that some agents are in an environment that is different to the requesting agents'. In a real world environment it would be more likely, than the idealised model used in the research so far.

Chapter 4 also looks at the impact of network update frequency and compares two different scenarios, acknowledging that a lower frequency in target network updates produces better results. However, a lower frequency in target network updates also requires more simulation run time to get a good view of training results. Due to time constraints for experimentation, the impact understanding is limited. In future work, this could be addressed, producing optimisation curves for the number of time steps required to train versus the target network update frequency. This would aid in solving the fundamental problem of optimising simulations and experiments for DRL.

The thesis explored using experience exchange to improve the performance of DRL algorithms, however it did not find an efficient way of optimising the

number of experiences being exchanged. Due to a large amount of information needed to be transferred over the network, significant overheads can be introduced with a large number of agents communicating, as shown in chapter 5. The thesis already established that mobile network applications require low latency to enforce decisions being made with accurate information, as such reducing the overhead from DRL requirements would benefit the network. A question of how to do this still stands. There are multiple criteria that make this a complex problem. One criteria in particular is that as agents spend more time learning, they are more familiar with possible states, increasing the sample size for future requests. Improving the algorithm for sample selection on both the requester and the server sides could improve the data overhead problem that chapter 5 introduces.

The thesis explores state spaces that are continuous, one user at a time. There are many ways to encode normalised state information into a single bitmap image that could then be processed by a convolutional neural network to make decisions for multiple users at once. This would simultaneously improve the processing capabilities of the algorithms, reduce the computational complexity, and produce solutions that could be easily scalable for any number of users. In addition to this, convolutional neural networks are better at processing continuous data compared to fully connected neural networks. Future work could address this problem. Both encoding the feature data into a bitmap file and processing of such data to study how scalable such solutions could be.

Bibliography

[1] *Ten years of gsm in australia*, 2003. [Online]. Available: https://web.archive.org/web/20080720110058/http://www.amta.org.au/default.asp?Page=142.

[2] *Ericsson mobility report*. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/key-figures.

[3] [Online]. Available: https://www.grandviewresearch.com/industry-analysis/virtual-reality-vr-market.

[4] M. A. Habibi, M. Nasimi, B. Han, and H. D. Schotten, "A comprehensive survey of ran architectures toward 5g mobile communication system," *IEEE Access*, vol. 7, pp. 70 371–70 421, 2019. DOI: 10.1109/ACCESS.2019.2919657.

[5] C. Pan, M. Elkashlan, J. Wang, J. Yuan, and L. Hanzo, "User-centric c-ran architecture for ultra-dense 5g networks: Challenges and methodologies," *IEEE Communications Magazine*, vol. 56, no. 6, pp. 14–20, 2018. DOI: 10.1109/MCOM.2018.1700483.

[6] H. Ren, N. Liu, C. Pan, *et al.*, "Low-latency c-ran: An next-generation wireless approach," *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 48–56, 2018. DOI: 10.1109/MVT.2018.2811244.

[7] J. G. Andrews, S. Buzzi, W. Choi, *et al.*, "What will 5g be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, 2014. DOI: 10.1109/JSAC.2014.2328098.

[8] I. Laurinavicius, H. Zhu, J. Wang, and Y. Pan, "Beam squint exploitation for linear phased arrays in a mmwave multi-carrier system," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6. DOI: 10.1109/GLOBECOM38437.2019.9013598.

[9] [Online]. Available: https://www.o-ran.org/who-we-are.

[10] J. Opadere, Q. Liu, and T. Han, "Energy-efficient rrh sleep mode for virtual radio access networks," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6. DOI: 10.1109/GLOCOM.2017.8254729.

[11] G. Interdonato, E. Björnson, H. Quoc Ngo, P. Frenger, and E. G. Larsson, "Ubiquitous cell-free massive mimo communications," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 197, 2019, ISSN: 1687-1499. DOI: 10.1186/s13638-019-1507-0. [Online]. Available: https://doi.org/10.1186/s13638-019-1507-0.

[12] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio access networks: Issues and challenges," *IEEE Network*, vol. 30, no. 4, pp. 46–53, 2016. DOI: 10.1109/MNET.2016.7513863.

[13] Y. Shih, W. Chung, A. Pang, T. Chiu, and H. Wei, "Enabling low-latency applications in fog-radio access networks," *IEEE Network*, vol. 31, no. 1, pp. 52–58, 2017. DOI: 10.1109/MNET.2016.1500279NM.

[14] L. Lu, Y. Jiang, M. Bennis, Z. Ding, F.-C. Zheng, and X. You, "Distributed edge caching via reinforcement learning in fog radio access networks," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019, pp. 1–6. DOI: 10.1109/VTCSpring.2019.8746321.

[15] H. Zhang, Y. Qiu, X. Chu, K. Long, and V. C. M. Leung, "Fog radio access networks: Mobility management, interference mitigation, and resource optimization," *IEEE Wireless Communications*, vol. 24, no. 6, pp. 120–127, 2017. DOI: 10.1109/MWC.2017.1700007.

[16] A. Haleem, M. Javaid, and R. P. Singh, "An era of chatgpt as a significant futuristic support tool: A study on features, abilities, and challenges," *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, vol. 2, no. 4, p. 100 089, 2022, ISSN: 2772-4859. DOI: https://doi.org/10.1016/j.tbench.2023.100089. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772485923000066.

[17] E.-w. Bai, "Big data: The curse of dimensionality in modeling," in *Proceedings of the 33rd Chinese Control Conference*, 2014, pp. 6–13. DOI: 10.1109/ChiCC.2014.6896586.

[18] R. Bellman, R. Corporation, and K. M. R. Collection, *Dynamic Programming* (Rand Corporation research study). Princeton University Press, 1957, ISBN: 9780691079516. [Online]. Available: https://books.google.co.uk/books?id=wdtoPwAACAAJ.

[19] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018, ISBN: 0262039249.

[20] V. Mnih, A. P. Badia, M. Mirza, *et al.*, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, PMLR, 2016, pp. 1928–1937.

[21] H. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., vol. 23, Curran Associates, Inc., 2010. [Online]. Available: https://proceedings.neurips.cc/paper/2010/file/091d584fced301b442654dd8c23b3fPaper.pdf.

[22] H. van Hasselt, A. Guez, and D. Silver, *Deep reinforcement learning with double q-learning*, 2015. arXiv: 1509.06461 [cs.LG].

[23] S. Ahilan and P. Dayan, "Correcting experience replay for multi-agent communication," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=xvxPuCkCNPO.

[24] Z. Gao, J. Zhang, S. Yan, Y. Xiao, D. Simeonidou, and Y. Ji, "Deep reinforcement learning for bbu placement and routing in c-ran," in *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, 2019, pp. 1–3.

[25] E. H. Bouzidi, A. Outtagarts, and R. Langar, "Deep reinforcement learning application for network latency management in software defined networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6. DOI: 10.1109/GLOBECOM38437.2019.9013221.

[26] T. H. L. Dinh, M. Kaneko, and L. Boukhatem, "Energy-efficient user association and beamforming for 5g fog radio access networks," in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2019, pp. 1–6. DOI: 10.1109/CCNC.2019.8651884.

[27] M. Sana, A. De Domenico, and E. Calvanese Strinati, "Multi-agent deep reinforcement learning based user association for dense mmwave networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6. DOI: 10.1109/GLOBECOM38437.2019.9013751.

[28] S. He, C. Qi, Y. Huang, Q. Hou, and A. Nallanathan, "Two-level transmission scheme for cache-enabled fog radio access networks," *IEEE Transactions on Communications*, vol. 67, no. 1, pp. 445–456, 2019. DOI: 10.1109/TCOMM.2018.2869562.

[29] H. T. Malazi and S. Clarke, "Distributed service placement and workload orchestration in a multi-access edge computing environment," in *2021 IEEE International Conference on Services Computing (SCC)*, 2021, pp. 241–251. DOI: 10.1109/SCC53864.2021.00037.

[30] T. Rappaport, *Wireless Communications: Principles and Practice* (Prentice Hall communications engineering and emerging technologies series). Prentice Hall PTR, 2002, ISBN: 9780130422323. [Online]. Available: https://books.google.se/books?id=TbgQAQAAMAAJ.

[31] J. L. Frauendorf and É. Almeida de Souza, *The Evolution of RAN (Radio Access Network), D-RAN, C-RAN, V-RAN, and O-RAN*. Cham: Springer International Publishing, 2023, pp. 139–154, ISBN: 978-3-031-10650-7. DOI: 10.1007/978-3-031-10650-7_10. [Online]. Available: https://doi.org/10.1007/978-3-031-10650-7_10.

[32] S. Shibata, K. Kinoshita, K. Kawano, and T. Watanabe, "A fair spectrum sharing method in consideration of comp," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 1137–1142. DOI: 10.1109/NOMS.2016.7502976.

[33] P. Zetterberg, "Interference alignment (ia) and coordinated multi-point (comp) overheads and rf impairments: Testbed results," in *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, 2014, pp. 1–7. DOI: 10.1109/VTCFall.2014.6966014.

[34] J. Zhao, T. Q. S. Quek, and Z. Lei, "Coordinated Multipoint Transmission with Limited Backhaul Data Transfer," *IEEE Transactions on Wireless Communications*, vol. 12, no. 6, pp. 2762–2775, 2013, ISSN: 1558-2248. DOI: 10.1109/TWC.2013.050613.120825.

[35] M. Vaezi and Y. Zhang, *Cloud Mobile Networks*. Jan. 2017, ISBN: 978-3-319-54495-3. DOI: 10.1007/978-3-319-54496-0.

[36] A. Nagate, S. Nabatame, D. Ogata, K. Hoshino, and T. Fujii, "Field experiment of comp joint transmission over x2 interface for lte-advanced," in *2013 IEEE 77th Vehicular Technology Conference (VTC Spring)*, 2013, pp. 1–5. DOI: 10.1109/VTCSpring.2013.6692564.

[37] 3GPP, *Etsi ts 137 340 v17.1.0*, 2022.

[38] R. Irmer, H. Droste, P. Marsch, *et al.*, "Coordinated multipoint: Concepts, performance, and field trial results," *IEEE Communications Magazine*, vol. 49, no. 2, pp. 102–111, 2011. DOI: 10.1109/MCOM.2011.5706317.

[39] S. Chen, J. Zhang, J. Zhang, E. Björnson, and b. ai, "A survey on user-centric cell-free massive mimo systems," *Digital Communications and Networks*, vol. 8, Dec. 2021. DOI: 10.1016/j.dcan.2021.12.005.

[40] H. Q. Ngo, A. Ashikhmin, H. Yang, E. G. Larsson, and T. L. Marzetta, "Cell-free massive mimo versus small cells," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1834–1850, 2017. DOI: 10.1109/TWC.2017.2655515.

[41] S. Zhou, X. Liu, F. Effenberger, and J. Chao, "Low-latency high-efficiency mobile fronthaul with tdm-pon (mobile-pon)," *Journal of Optical Communications and Networking*, vol. 10, no. 1, A20–A26, 2018. DOI: 10.1364/JOCN.10.000A20.

[42] G. Mountaser, T. Mahmoodi, and O. Simeone, "Reliable and low-latency fronthaul for tactile internet applications," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 11, pp. 2455–2463, 2018. DOI: 10.1109/JSAC.2018.2872299.

[43] S. Hung, H. Hsu, S. Lien, and K. Chen, "Architecture harmonization between cloud radio access networks and fog networks," *IEEE Access*, vol. 3, pp. 3019–3034, 2015. DOI: 10.1109/ACCESS.2015.2509638.

[44] T. Zhang, X. Fang, Y. Liu, and A. Nallanathan, "Content-centric mobile edge caching," *IEEE Access*, vol. 8, pp. 11 722–11 731, 2020. DOI: 10.1109/ACCESS.2019.2962856.

[45] W. Hao, G. Sun, O. Muta, J. Zhang, and S. Yang, "Coordinated hybrid precoding design in millimeter wave fog-ran," *IEEE Systems Journal*, vol. 14, no. 1, pp. 673–676, 2020. DOI: 10.1109/JSYST.2019.2921557.

[46] S. Park, O. Simeone, and S. Shamai, "Joint optimization of cloud and edge processing for fog radio access networks," in *2016 IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 315–319. DOI: 10.1109/ISIT.2016.7541312.

[47] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943, ISSN: 1522-9602. DOI: 10.1007/BF02478259. [Online]. Available: https://doi.org/10.1007/BF02478259.

[48] D. O. Hebb, *The organization of behavior*. New York: Wiley, 1949.

[49] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992, ISSN: 1573-0565. DOI: 10.1007/BF00992698. [Online]. Available: https://doi.org/10.1007/BF00992698.

[50] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach* (Pearson series in artificial intelligence). Pearson, 2021, ISBN: 9781292401133. [Online]. Available: https://books.google.se/books?id=B4xczgEACAAJ.

[51] F.-J. Yang, "An implementation of naive bayes classifier," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2018, pp. 301–306. DOI: 10.1109/CSCI46756.2018.00065.

[52] M. Ramon, N. Xu, and C. Christodoulou, "Beamforming using support vector machines," *IEEE Antennas and Wireless Propagation Letters*, vol. 4, pp. 439–442, 2005. DOI: 10.1109/LAWP.2005.860196.

[53] M. Martinez-Ramon, N. Xu, and C. C. Christodoulou, "Support vector minimum variance distortionless response doa detector," in *2007 IEEE Antennas and Propagation Society International Symposium*, 2007, pp. 5291–5294. DOI: 10.1109/APS.2007.4396741.

[54] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, pp. 276–280, 1986. DOI: 10.1109/TAP.1986.1143830.

[55] A. Pan, T. Zhang, and X. Han, "Location information aided beam allocation algorithm in mmwave massive mimo systems," in *2017 IEEE/CIC*

International Conference on Communications in China (ICCC), 2017, pp. 1–6. DOI: 10.1109/ICCChina.2017.8330410.

[56] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982. DOI: 10.1109/TIT.1982.1056489.

[57] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[58] R. Nikbakht, A. Jonsson, and A. Lozano, "Unsupervised learning for c-ran power control and power allocation," *IEEE Communications Letters*, vol. 25, no. 3, pp. 687–691, 2021. DOI: 10.1109/LCOMM.2020.3027991.

[59] M. Nour, M. Awad, M. Kamel, M. Essa, and N. Abdelbaki, "Anomaly detection using unsupervised learning in lte mobile network," in *2021 Tenth International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2021, pp. 195–199. DOI: 10.1109/ICICIS52592.2021.9694214.

[60] G. A. Rummery and M Niranjan, "A connectionist approach to learning," *Learning in Natural and Connectionist Systems*, 1–25, 1994. DOI: 10.1007/978-94-011-0840-9_2.

[61] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012. DOI: 10.1109/TSMCC.2012.2218595.

[62] S. Varges, G. Riccardi, S. Quarteroni, and A. V. Ivanov, "The exploration/exploitation trade-off in reinforcement learning for dialogue management," in *2009 IEEE Workshop on Automatic Speech Recognition Understanding*, 2009, pp. 479–484. DOI: 10.1109/ASRU.2009.5373260.

[63] Q. Liu, A. Chung, C. Szepesvari, and C. Jin, "When is partially observable reinforcement learning not scary?" In *Proceedings of Thirty Fifth Conference on Learning Theory*, P.-L. Loh and M. Raginsky, Eds., ser. Proceedings of Machine Learning Research, vol. 178, PMLR, 2022, pp. 5175–5220. [Online]. Available: https://proceedings.mlr.press/v178/liu22f.html.

[64] P. Stone, G. A. Kaminka, S. Kraus, and J. S. Rosenschein, "Ad hoc autonomous agent teams: Collaboration without pre-coordination," in *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence*, 2010.

[65] S. Sritharan, H. Weligampola, and H. Gacanin, "A study on deep learning for latency constraint applications in beyond 5g wireless systems," *IEEE Access*, vol. 8, pp. 218 037–218 061, 2020. DOI: 10.1109/ACCESS.2020.3040133.

[66] R. Ding, Y. Xu, F. Gao, X. Shen, and W. Wu, "Deep reinforcement learning for router selection in network with heavy traffic," *IEEE Access*, vol. 7, pp. 37 109–37 120, 2019. DOI: 10.1109/ACCESS.2019.2904539.

[67] M. N. Chughtai, S. Noor, I. Laurinavicius, *et al.*, "User and resource allocation in latency constrained xhaul via reinforcement learning," *Journal of Optical Communications and Networking*, vol. 15, no. 4, pp. 219–228, 2023. DOI: 10.1364/JOCN.485029.

[68] A. Giannopoulos, S. Spantideas, N. Capsalis, *et al.*, "Wip: Demand-driven power allocation in wireless networks with deep q-learning," in *2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2021, pp. 248–251. DOI: 10.1109/WoWMoM51794.2021.00045.

[69] A. Nassar and Y. Yilmaz, "Reinforcement learning for adaptive resource allocation in fog ran for iot with heterogeneous latency requirements," *IEEE Access*, vol. 7, pp. 128 014–128 025, 2019. DOI: 10.1109/ACCESS.2019.2939735.

[70] X. Zhang, Y. Shen, B. Yang, W. Zang, and S. Wang, "Drl based data offloading for intelligent reflecting surface aided mobile edge computing," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, 2021, pp. 1–7. DOI: 10.1109/WCNC49053.2021.9417469.

[71] B. Akgun, M. Krunz, and D. Manzi, "Impact of beamforming on delay spread in wideband millimeter-wave systems," in *2020 International Conference on Computing, Networking and Communications (ICNC)*, 2020, pp. 890–896. DOI: 10.1109/ICNC47757.2020.9049739.

[72] S. Sirotkin, *5G Radio Access Network Architecture: The dark side of 5G*. Wiley-IEEE Press, 2021.

[73] B. E. Godana and D. Gesbert, "Coordinated beamforming in multicell networks with channel state information exchange delays," in *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2013, pp. 713–718. DOI: 10.1109/PIMRC.2013.6666229.

[74] A. Eira, A. Pereira, J. Pires, and J. Pedro, "On the efficiency of flexible ethernet client architectures in optical transport networks [invited]," *Journal of Optical Communications and Networking*, vol. 10, no. 1, A133–A143, 2018. DOI: 10.1364/JOCN.10.00A133.

[75] F. Zhou, G. Luo, L. Fan, and J. Tang, "Coordinated beamforming scheme for heterogeneous networks with band-limited backhaul constraint," in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2017, pp. 1–5. DOI: 10.1109/WCSP.2017.8171041.

[76] M. Klinkowski and D. Mroziński, "Latency-aware flow allocation in 5g ngfi networks," in *2020 22nd International Conference on Transparent Optical Networks (ICTON)*, 2020, pp. 1–4. DOI: 10.1109/ICTON51198.2020.9203440.

[77] A. Tzanakaki, M. P. Anastasopoulos, and D. Simeonidou, "Converged optical, wireless, and data center network infrastructures for 5g services," *Journal of Optical Communications and Networking*, vol. 11, no. 2, A111–A122, 2019. DOI: 10.1364/JOCN.11.00A111.

[78] N. Molner, A. de la Oliva, I. Stavrakakis, and A. Azcorra, "Optimization of an integrated fronthaul/backhaul network under path and delay constraints," *Ad Hoc Networks*, vol. 83, pp. 41–54, 2019, ISSN: 1570-8705. DOI: https://doi.org/10.1016/j.adhoc.2018.08.025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870518306206.

[79] M. Elsayed and M. Erol-Kantarci, "Reinforcement learning-based joint power and resource allocation for urllc in 5g," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6. DOI: 10.1109/GLOBECOM38437.2019.9014032.

[80] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for ai-enabled wireless networks: A tutorial," *IEEE Communications Surveys Tutorials*, vol. 23, no. 2, pp. 1226–1252, 2021. DOI: 10.1109/COMST.2021.3063822.

[81] P. J. Brockwell and R. A. Davis, Eds., *ARMA Models*. New York, NY: Springer New York, 2002, pp. 83–110, ISBN: 978-0-387-21657-7. DOI: 10.1007/0-387-21657-X_3. [Online]. Available: https://doi.org/10.1007/0-387-21657-X_3.

[82] S. Zhang, Q. Zhong, M. Zha, and T. Zuo, "Hybrid multiplexing over flexe group," in *2018 23rd Opto-Electronics and Communications Conference (OECC)*, 2018, pp. 1–2. DOI: 10.1109/OECC.2018.8730124.

[83] P. Iovanna, G. Bottari, F. Ponzini, and L. M. Contreras, "Latency-driven transport for 5g," *Journal of Optical Communications and Networking*, vol. 10, no. 8, pp. 695–702, 2018. DOI: 10.1364/JOCN.10.000695.

[84] N.-S. Kim and Y. H. Lee, "Effect of channel estimation errors and feedback delay on the performance of closed-loop transmit diversity system," in *2003 4th IEEE Workshop on Signal Processing Advances in Wireless Communications - SPAWC 2003 (IEEE Cat. No.03EX689)*, 2003, pp. 542–545. DOI: 10.1109/SPAWC.2003.1319019.

[85] Q. Sun, D. Cox, H. Huang, and A. Lozano, "Estimation of continuous flat fading mimo channels," in *2002 IEEE Wireless Communications and Networking Conference Record. WCNC 2002 (Cat. No.02TH8609)*, vol. 1, 2002, 189–193 vol.1. DOI: 10.1109/WCNC.2002.993488.

[86] D. Tse and P. Viswanath, Eds., *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[87] 2017. [Online]. Available: https://itecspec.com/archive/3gpp-specification-tr-38-801/.

[88] V. Q. Rodriguez, F. Guillemin, A. Ferrieux, and L. Thomas, "Cloud-ran functional split for an efficient fronthaul network," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*, 2020, pp. 245–250. DOI: 10.1109/IWCMC48107.2020.9148093.

[89] J. F. C. Kingman, "The single server queue in heavy traffic," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 57, no. 4, 902–904, 1961. DOI: 10.1017/S0305004100036094.

[90] K. Cho, K. Mitsuya, and A. Kato, "Traffic data repository at the WIDE project," in *2000 USENIX Annual Technical Conference (USENIX ATC 00)*, San Diego, CA: USENIX Association, Jun. 2000. [Online]. Available: https://www.usenix.org/conference/2000-usenix-annual-technical-conference/traffic-data-repository-wide-project.

[91] W. Fedus, P. Ramachandran, R. Agarwal, *et al.*, "Revisiting fundamentals of experience replay," in *International Conference on Machine Learning*, PMLR, 2020, pp. 3061–3071.

[92] Y. Liao, S. A. Hashemi, H. ElBakoury, J. Cioffi, and A. Goldsmith, "Calendar allocation based on client traffic in the flexible ethernet standard," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6. DOI: 10.1109/ICC40277.2020.9149293.

[93] X. Huang, G. Xue, R. Yu, and S. Leng, "Joint scheduling and beamforming coordination in cloud radio access networks with qos guarantees," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 7, pp. 5449–5460, 2016.

[94] E. AB, H. T. C. Ltd, N. Corporation, A. Lucent, and N. Networks, *Cpri specification v7.0*, 2015.

[95] C. Pan, H. Zhu, N. J. Gomes, and J. Wang, "Joint precoding and rrh selection for user-centric green mimo c-ran," *IEEE Transactions on Wireless Communications*, vol. 16, no. 5, pp. 2891–2906, 2017. DOI: 10.1109/TWC.2017.2671358.

[96] X. Cao, C. Hu, and S. Yan, "A matching game approach for joint resource allocation and user association in fog radio access networks," in *2018 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, 2018, pp. 277–281. DOI: 10.1109/ICCChinaW.2018.8674520.

[97] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: Technical misconceptions and business barriers," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 16–22, 2016. DOI: 10.1109/MCOM.2016.7537172.

[98] D. Silver, J. Schrittwieser, K. Simonyan, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, 7676 2017. DOI: 10.1038/nature24270.

[99] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the Tenth International Conference on Machine Learning*, Morgan Kaufmann, 1993, pp. 330–337.

[100] L. Oliveira Souza, G. de Oliveira Ramos, and C. Ghedini Ralha, "Experience sharing between cooperative reinforcement learning agents," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, 2019, pp. 963–970. DOI: 10.1109/ICTAI.2019.00136.

[101] F. Christianos, L. Schäfer, and S. Albrecht, "Shared experience actor-critic for multi-agent reinforcement learning," English, in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, Thirty-fourth Conference on Neural Information Processing Systems, NeurIPS 2020 ; Conference date: 06-12-2020 Through 12-12-2020, Curran Associates Inc, Dec. 2020, pp. 10 707–10 717. [Online]. Available: https://nips.cc/Conferences/2020.

[102] J. N. Foerster, N. Nardelli, G. Farquhar, *et al.*, "Stabilising experience replay for deep multi-agent reinforcement learning," in *International Conference on Machine Learning*, 2017.

[103] T. Biermann, L. Scalia, C. Choi, W. Kellerer, and H. Karl, "How backhaul networks influence the feasibility of coordinated multipoint in cellular networks," *IEEE Communications Magazine*, vol. 51, no. 8, pp. 168–176, 2013, ISSN: 1558-1896. DOI: 10.1109/MCOM.2013.6576356.

[104] M. M. Do and H. J. Son, *CoMP (1): CoMP Types - CS, CB, JT and DPS*, 2014, Accessed 20 Feb, 2020. [Online]. Available: https://www.netmanias.com/en/post/blog/6558/comp-lte-lte-a/comp-1-comp-types-cs-cb-jt-and-dps.

[105] F. Qamar, K. Dimyati, M. Hindia, K. Noordin, and A. M. Al-samman, "A comprehensive review on coordinated multi-point operation for lte-a," *Computer Networks*, vol. 123, May 2017. DOI: 10.1016/j.comnet.2017.05.003.

[106] W. Wong and S. G. Chan, "Distributed joint ap grouping and user association for mu-mimo networks," in *IEEE INFOCOM 2018 - IEEE Conference*

on Computer Communications, 2018, pp. 252–260. DOI: 10.1109/INFOCOM.2018.8486284.

[107] S. Sun, T. S. Rappaport, S. Rangan, *et al.*, "Propagation path loss models for 5g urban micro- and macro-cellular scenarios," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, 2016, pp. 1–6. DOI: 10.1109/VTCSpring.2016.7504435.

[108] E. Björnson and E. Jorswieck, *Optimal Resource Allocation in Coordinated Multi-Cell Systems*. 2013, vol. 9, pp. 113–381.

[109] S. Tombaz, P. Monti, K. Wang, A. Vastberg, M. Forzati, and J. Zander, "Impact of backhauling power consumption on the deployment of heterogeneous mobile networks," in *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, 2011, pp. 1–5. DOI: 10.1109/GLOCOM.2011.6133999.

[110] D. Chen, H. Al-Shatri, T. Mahn, A. Klein, and V. Kuehn, "Energy efficient robust f-ran downlink design for hard and soft fronthauling," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, 2018, pp. 1–5. DOI: 10.1109/VTCSpring.2018.8417561.

[111] X. Jiang, H.-C. Chao, G.-M. Muntean, G. Ghinea, and C. Xu, "Green communication for mobile and wireless networks," *Mobile Information Systems*, vol. 24, no. 6, pp. 120–127, 2016. DOI: 10.1155/2016/8719763.

[112] J. Yu and M. Dong, "Distributed low-complexity multi-cell coordinated multicast beamforming with large-scale antennas," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018, pp. 1–5. DOI: 10.1109/SPAWC.2018.8446030.

[113] K. Bonawitz, H. Eichner, W. Grieskamp, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[114] J. B. Predd, S. B. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 56–69, 2006.

[115] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lactot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, vol. 48, pp. 1–15, 2016.

[116] T. P. Le, N. A. Vien, and T. Chung, "A deep hierarchical reinforcement learning algorithm in partially observable markov decision processes," *IEEE Access*, vol. 6, pp. 49 089–49 102, 2018. DOI: 10.1109/ACCESS.2018.2854283.

[117] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Hysteretic q-learning : An algorithm for decentralized reinforcement learning in cooperative multi-agent teams," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 64–69. DOI: 10.1109/IROS.2007.4399095.

[118] 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. DOI: 10.1109/JSAC.2019.2933973.

[119] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, vol. 1, 1999, 126–134 vol.1. DOI: 10.1109/INFCOM.1999.749260.