Artificial Intelligence-Based Radio Resource Management in Sliced Radio Access Networks

Apoorva Arora

TU Delft

Artificial Intelligence-Based Radio Resource Management in Sliced Radio Access Networks

by

Apoorva Arora

in partial fulfilment of the requirements for the degree of

Master of Science
in Embedded Systems
Specialisation in Software and Wireless Networking

at the Delft University of Technology,
to be defended publicly on Thursday October 29, 2020 at 10:00 AM.

Thesis committee:	Dr. Remco Litjens, MSc.	TU Delft, TNO
	Dr. Haibin Zhang,	TNO
	Dr. Ir. Jos Weber,	TU Delft

This thesis is confidential and cannot be made public until further notice.

An electronic version of this thesis is available at
http://repository.tudelft.nl/

.

TU Delft TNO innovation for life

Preface

This thesis marks the completion of my studies at the Delft University of Technology (TU Delft) for the Masters of Science degree in Embedded Systems with specialisation in Software and Networking. This work was done in collaboration with the Networks department at TNO, the Hague as a part of the European H2020-ICT-19-2019 project 5G-HEART (Grant agreement no. 857034) and the Network Architecture and Services (NAS) group at the faculty of Electrical Engineering, Mathematics, and Computer Science (EEMCS) at TU Delft.

Working on this thesis has been a very enriching and knowledgeable experience for me. I have many people to express my gratitude to, who helped me to make the best out of this challenging yet exciting journey. I want to thank Remco Litjens, my supervisor from TU Delft for his constant guidance on various technical aspects of the study. Moreover, his feedback and critical analysis of my work motivated me to stretch my limits and challenge myself. I want to thank Haibin Zhang, my supervisor from TNO for his guidance in various technical and business aspects of this study. Moreover, I thank him for welcoming me in the 5G-HEART project. Participation in the 5G-HEART project enabled me to broaden my experience of working on big projects that involve multiple partners and stakeholders. I want to thank both of my supervisors for their constructive feedback on the writings of this report and very valuable weekly brainstorming sessions. I also thank Toni Dimitrovski, my colleague at TNO for his contributions in the brainstorming sessions. His ideas helped me gain greater clarity and contributed in the progress of my work. I want to thank my colleagues at TNO for extending their help whenever needed. I want to thank Kallol Das for providing me the opportunity to contribute in projects besides my thesis. This allowed me to broaden my experience within TNO and expand my knowledge. I want to thank my colleagues from the 'radio group' at TNO for organising the 'Let's Talk Radio' discussion sessions. These fun sessions provided a great knowledge sharing platform. I want to thank Dr. Ir. Jos Weber for being a part of my thesis committee. I also want to thank Edgar van Boven, from the networks group at TU Delft for providing me guidance, whenever I needed.

Last but not least, I am grateful to my parents and brother for their love, blessings and constant motivation. I want to thank my friend Rahul for supporting me in difficult times and remaining a constant source of motivation. I want to thank my friends in Delft for having fruitful discussions with me which helped me get new ideas. Finally, I am sincerely grateful to Delft Global for providing me a scholarship to study at TU Delft.

Apoorva Arora
Delft, October 2020

Abstract

5G networks will support different industry verticals with very diverse service requirements. These diverse services in a likely scenario are handled in isolation from each other on virtually separate networks (called network slices). While the concept of network slicing is valid for an end-to-end network, we focus on Radio Access Network (RAN) Slicing. Slicing requires (dynamic) reservation of resources for each slice in advance. The reservation of resources for each slice from a common pool of resources requires prediction of their resource demand for a future time interval which depends on the timescale of resource re-allocation. Moreover, the resource allocation problem in a sliced RAN is an optimisation problem that has been motivated to be addressed by artificial intelligence techniques.

In this thesis, we design and assess a multi-slice resource allocation framework that is based on machine learning techniques (subset of artificial intelligence techniques). The proposed framework employs two machine learning techniques namely, artificial neural networks and reinforcement learning for resource management in sliced RAN. Alternative multi-slice resource allocation methods that involve only artificial neural networks but not reinforcement learning are also defined.

A scenario-based analysis is used to evaluate the proposed framework. We consider the specific use case of an ambulance that represents an emergency scenario. The use case involves so-called priority applications that need network assistance for their operation. The examples of such applications include road-traffic coordination to enable quick transportation of an ambulance to and from the place of incidence and remote connection of the ambulance with the specialists at a hospital. The applications are mapped to two so-called priority slices, namely medical and transport. The resource allocation problem is then defined for the above-mentioned priority slices and a third so-called background slice.

For the evaluation of the proposed framework, a system-level simulator is developed incorporating realistic characteristics of the considered scenarios, suitable propagation aspects and traffic models. The proposed framework is compared with the alternative methods. Moreover, a non-slicing-based resource management method known as priority scheduling is evaluated as a baseline. For the purpose of performance evaluation, various performance metrics are defined which capture both performance (w.r.t. QoS satisfaction of the priority applications) and the cost of resource reservation/assignment.

Based on the simulation assessment of the considered resource management methods, it is found that the proposed framework which uses both artificial neural network and reinforcement learning performs the best in all the considered scenarios in terms of the bottom-line metric, namely the weighted average utility of the priority classes. In essence, the proposed framework which employs reinforcement learning makes the

resource allocation decisions robust against errors introduced by various sources such as inaccuracy in demand predictions, change in the policies of the slice owners and/or network operator or variation in the load and traffic characteristics.

Contents

List of Tables

List of Figures

List of Abbreviations

AI	Artificial intelligence
ANN	Artificial neural network
ARE	Absolute relative error
ARIMA	Autoregressive integrated moving average
AWGN	Additive white Gaussian noise
BLER	Block error rate
BS	Base station
BSA	Best server area
CI	Confidence interval
CLPC	Closed-loop power control
CMAB	Contextual multi-armed bandit
CNN	Convolutional neural network
CQI	Channel quality indicator
DL	Deep learning
DNN	Deep neural network
DQL	Deep Q-learning
DQN	Deep Q-network
DRL	Deep reinforcement learning
DS	Delay-sensitive
DT	Delay-tolerant
eMBB	Enhanced mobile broadband
FA	Function approximation
gNB	gNodeB
HARQ	Hybrid automatic repeat request
IC	Intersection controller
KPI	Key performance indicator
LA	Link adaptation
LOS	Line-of-sight
LSTM	Long short-term memory
LTE	Long term evolution
MAC	Medium access control
MARE	Mean absolute relative error
MCS	Modulation and coding scheme
MDP	Markov decision process
MIMO	Multiple-input and multiple-output
ML	Machine learning

MLP	Multilayer perceptron
MLR	Multiple linear regression
NFV	Network function virtualisation
NLOS	Non-line-of-sight
NR	New radio
OFDMA	Orthogonal frequency division multiple access
OLPC	Open-loop power control
OLS	Ordinary least squares
PRB	Physical resource block
PDR	Packet drop-rate
QAM	Quadrature amplitude modulation
QoE	Quality of experience
QoS	Quality of service
RAN	Radio access network
ReLU	Rectified linear unit
RF	Random forests
RL	Reinforcement learning
RNN	Recurrent neural network
RRM	Radio resource management
SCC	Shadowing cross-correlation
SCS	Sub-carrier spacing
SDN	Software-defined networking
SE	Spectral efficiency
SINR	Signal-to-interference-plus-noise ratio
SISO	Single-input single-output
SNR	Signal-to-noise ratio
SRS	Sounding reference signal
TD	Temporal difference
TPC	Transmission power control
TSB	Truncated Shannon bound
TTI	Transmission time interval
UE	User equipment
URLLC	Ultra-reliable low-latency communication
vMNOs	Virtual mobile network operators
VoIP	Voice over Internet Protocol

1

Introduction

The 5th generation of mobile networks (5G) will facilitate very diverse services from different industry verticals such as healthcare, smart-automotive, smart city and entertainment, each having very diverse and often conflicting requirements from a telecom network. These services can be classified into three broad categories, namely enhanced mobile broadband (eMBB), massive machine-type communication (mMTC) and ultra-reliable low-latency communication (URLLC). The present telecommunication network is largely a “one-size-fits-all” type of network, where the traffic of all applications follow the same pipeline. In a diverse service environment, deployment of a separate network infrastructure for each service class appears as a more attractive option because of its potential benefits in service performance isolation and data security [59]. However, deploying separate infrastructure is infeasible due to huge capital investments. To aid resource sharing in a diverse service environment, network slicing has been proposed as a key technology for future mobile networks [59]. Network slicing allows the mobile network infrastructure to be logically separated into various virtual networks called network slices. A network slice is an end-to-end (including radio access network (RAN), transport and core network) logically self-contained network over the infrastructure of a telecommunication network [54]. Each network slice may be customized to support specific services and contains a set of network functions and resources abstracted from a common pool of physical network resources. Network slicing enables efficient infrastructure and spectrum sharing among entities demanding very diverse requirements from the network. These entities called tenants or slice owners own a slice. Tenants can be virtual mobile network operators (vMNOs) or over-the-top service providers or users in vertical industries such as smart-automotive, healthcare, e-agriculture, smart-city and entertainment.

Network slicing can be seen as infrastructure as a service and involves the creation and provision of fully functional virtual networks to tenants. These virtual networks share the physical computing, storage and networking resources. The concept of network slicing could be envisioned due to the development in the fields of software-defined networking (SDN) and network function virtualization (NFV) [59]. SDN and NFV are

in fact considered as the key technology enablers for network slicing in 5G networks. SDN is a technology that enables dynamic and programmable network configuration to improve network performance and monitoring. In SDNs, the network intelligence is centralised in a control plane which is in charge of traffic routing decisions. The control plane is separated from the underlying data plane, that controls the traffic forwarding process. NFV is the softwarization of network functions such as traffic analysis (quality of experience (QoE) measurements), switching and security functions (firewalls, spam detection). In NFV, virtual network functions are created that can run on general-purpose hardware or clouds. Both SDN and NFV have provided flexibility in network deployment and customization and enabled automation of network resource allocation.

1.1. Motivation

End-to-end network slicing includes slicing of the core network, transport network and the radio access network. In this thesis, we focus on RAN slicing. Two major objectives have been identified for a RAN slicing solution [16]. First, to achieve slice independence by ensuring performance isolation. Performance isolation means that the performance of the services in a slice is unaffected by traffic dynamics in other slices. The second objective is to allow adaptive sharing of RAN resources among slices, to enable multiplexing gains [16]. It is noted that RAN resources include radio processing hardware, power, antennas and the wireless frequency spectrum. However, the frequency spectrum is considered as the most important RAN resource due to its high cost and scarcity in nature. In this study, like most related studies, we focus only on the frequency spectrum as a shared resource.

Many recent works have focused on RAN slicing architecture development, implementation [16] and resource allocation schemes [57]. However, in these studies, the authors have assumed that the slice resource requirement is known at each resource allocation interval based on a pre-defined traffic model. This assumption is not always valid in real-world and time-variant wireless networks [57]. In order to enable adaptive spectrum sharing in a sliced RAN such that service performance is guaranteed and system resource utilization is maximised, it is essential to forecast the resource demand of the slices dynamically. In this thesis, we focus on the research question of slice resource demand prediction and multi-slice resource allocation in RAN slicing. The resource demand prediction methods, proposed in related works address the scenario where network slices have network traffic that is virtually active at all times. Consequently, these methods are not suitable for slices which are only occasionally active in a cell and additionally for short duration. An example would be a slice dedicated to an ambulance provider, handling mobile traffic generated by the ambulances during their journey. We address such a use case in this study. Chapter 5 details the use case.

1.2. Contributions

The slice resource demand forecasting is an essential element for a practical RAN slicing framework [57]. Use of artificial intelligence (AI) has been motivated in literature to address the slice resource allocation problem based on slice resource demand forecasting. The AI-based methods can be divided into two classes based on their respective approaches to solve the slice resource allocation problem. The first class employs time series forecasting using, for example statistical methods like autoregressive integrated moving average (ARIMA) [63], or more sophisticated machine learning methods like long short-term memory (LSTM) [11] to get slice resource demand and an allocation based on the demand. The second class of methods is based on reinforcement learning (RL) techniques that adaptively estimate the resource requirement for each slice and tune or learn the resource allocation strategy online based on observed system performance. Both classes use substantial amount of relevant slice historical data to make decisions on resource allocation. Furthermore, reinforcement learning techniques proposed in the literature have the drawback of slow convergence to an optimal policy due to complex and large state/action spaces (details in Section 4.2). Hence, the proposed methods are not suitable for many practical scenarios (e.g. emergency ambulance network slice), where the availability of slice historical data is limited and hence optimal decision rules cannot be learnt.

In this thesis, we overcome the above-mentioned limitations of the state of the art methods by proposing a novel AI-based RAN slicing resource allocation framework. More specifically, the major contributions of this thesis include:

1. We develop a neural network-based architecture for the prediction of resource demand of the slices. The proposed architecture uses the available mobile traffic statistics to *learn* to predict the resource demand for a user equipment (UE) call with given quality of service (QoS) requirements (details in Chapter 8). The sum over the predicted resource demands of the UE calls in a specific slice gives the slice resource demand. It is noted that in general, the resource demand prediction for aggregate slice traffic (as proposed in the literature) is a more accurate estimate for the actual slice resource demand than the sum over the resource demand predictions for the calls in a slice. However, the resource demand prediction for the aggregate slice traffic is based on the historical data of the aggregate resource usage in a slice. Therefore, it is not suitable in the case when enough historical data is not available. Information about the expected resource demand of a UE call with given QoS requirements may be obtained from various network factors (details in Chapter 8), independently from historical resource usage information. Therefore, in the proposed method we make use of resource demand predictions on a per call basis. This overcomes the limitation of lack of historical data per network slice.

2. Related to the neural network-based resource demand prediction, we develop a methodology to enhance the training data for optimal neural network learning

(details in Chapter 8). The enhancement includes, for example making the training data more diverse covering aspects of scenarios where the model may be applied. The proposed methodology is not limited to RAN slicing and can be applied in other applications employing neural networks or supervised learning AI techniques for resource management or dimensioning in mobile networks.

3. Furthermore, we propose a closed-loop architecture that includes the above-described neural network architecture in cascade with a reinforcement learning block. The RL block acts as an additional intelligent layer (in addition to the neural network architecture for resource demand prediction) and is responsible for resource allocation decision-making among all active slices. The resource allocation is done by adding a suitably learned resource margin for each slice to the demand prediction of the corresponding slice to compensate for the uncertainty of the demand prediction (details in Chapter 8). The RL block makes the resource allocation adaptive to the dynamic wireless and cellular system environment. The complete architecture behaves as a closed-loop system, since the RL block tunes its behaviour online based on observed system performance (including all active slices). The proposed RL block is fast in convergence and hence practical thanks to a simplified definition of state and action space.

4. We carry out an extensive quantitative scenario-based assessment of the proposed AI-based multi-slice resource allocation method, via simulations.

1.3. Thesis outline

In Chapter 2, we firstly present some background information on RAN slicing and multi-slice resource allocation. A brief discussion on artificial intelligence techniques, in particular neural networks and reinforcement learning techniques is also provided. In Chapter 3, we clearly define the scope of this study. In Chapter 4, we discuss the related state-of-the-art works in resource allocation for RAN slicing, presenting their methodologies, key contributions and drawbacks. We conclude the chapter by highlighting the limitations of the discussed state-of-the-art methods. In Chapter 5, we describe in detail the use case considered in this thesis to address the research problem of resource demand prediction and multi-slice resource allocation. Chapter 6 presents the details of the modelling aspects considered in this study. In Chapter 7, we define the evaluation and key performance metrics used for the assessment of the proposed method. Chapter 8 describes the proposed architecture and methodology to address the resource allocation problem in RAN slicing. In Chapter 9, we outline the considered scenarios for the evaluation of the proposed method and analyse the simulation results for each scenario. The performance of the proposed method is compared with a baseline and also other alternative methods for multi-slice resource allocation to illustrate the performance improvement achieved by our proposed method. In Chapter 10, we conclude our study and propose directions for further research.

2

Background

This chapter is organized as follows. Section 2.1 provides some background information of RAN slicing with a focus on the RAN sharing concept, slice requirements and isolation principles. Section 2.2 describes the radio resource management problem in a sliced RAN Some background information on AI-techniques, more specifically supervised machine learning and reinforcement learning is presented in Sections 2.3 and 2.4, respectively. In Section 2.4.1, a framework for reinforcement learning called contextual bandits or contextual multi-armed bandits is also discussed.

2.1. Radio access network slicing

A radio access network is the part of a telecommunications system that provides access to user devices to other parts of the system via wireless or radio connections. RAN slicing involves sharing of RAN resources among multiple RAN slices. Often, the wireless frequency spectrum is considered as the most important RAN resource. From the resource allocation and scheduling perspective, the time-frequency resources are divided into time slots and physical resource blocks (PRBs). A PRB is the smallest resource unit in the frequency domain that can be allocated to a user device for wireless data transmission. In an orthogonal frequency division multiple access (OFDMA) system, the frequency spectrum is divided into subcarrier units and a PRB has 12 subcarriers. Time is divided into slots. In 5G new radio (NR), a slot is the duration to transmit fourteen OFDMA symbols. It is the smallest unit in the time domain for scheduling the PRB allocation. In the fourth generation mobile communication standard (long term evolution (LTE)), the smallest unit in the time domain consisting of 14 OFDMA symbols is called a transmission time interval (TTI). In LTE, the spacing between the subcarriers i.e. the subcarrier spacing (SCS) is fixed and is equal to 15 kHz and a TTI is equal to 1 ms. In 5G NR, there are different variants of SCS viz. 15 kHz, 30 kHz, 60 kHz, 120 kHz, 240 kHz and 480 kHz and the corresponding slot duration viz. 1 ms, 0.5 ms, 0.25 ms, 0.125 ms, 0.0625 ms and 0.03125 ms. This is called the 5G NR numerology. A shorter

slot (or a larger SCS) allows for faster scheduling decisions required for low-latency and high-reliability services. The downside of using a larger SCS is higher phase noise and consequently decreased robustness and less frequency diversity in scheduling. For the sub 6 GHz band, 15, 30, 60 kHz SCS are allowed and 60, 120, 240 and 480 kHz can be used for mmWaves [1]. In this thesis, in the context of RAN slicing, we focus on sharing of resource blocks.

RAN slicing enables service-specific customization of the resources allocated to a slice to support diverse service characteristics. An example for resource customization is the use of different sub carrier spacings and correspondingly different slot lengths. 5G NR provides support for very low latency applications such as ultra-reliable low-latency communications. A larger subcarrier spacing or a lower transmission interval allow for flexible and fast scheduling required for URLLC-type communications. Furthermore, 5G NR supports the so-called mini-slots. A mini slot is defined as the duration to transmit two, four or seven OFDMA symbols. The support for mini-slots further increases the flexibility in scheduling PRB allocation.

Slice customization is closely related to the level of isolation between slices and is affected by how the common physical resources are shared among the slices [55]. Different levels of isolation that can be achieved between slices are discussed in Section 2.1.2

Slicing in the RAN has evolved from the well-known principles of RAN sharing. However, the objective is not to expand capacity but to enable wider business opportunities via customization. RAN sharing can be considered as a precursor for RAN Slicing. RAN sharing is widely adopted by mobile networks and involves sharing of RAN hardware elements, e.g. antennas and radio heads as well as radio spectrum among different operators.

2.1.1. RAN slicing requirements

RAN slicing enables handling of spectrum from capacity sharing perspective and also from customization perspective to meet diverse service requirements in future mobile networks (5G and beyond). Customization in RAN slices is facilitated by the enhancements in radio resource management to support isolation of resources and flexibility in resource allocation. Future mobile networks are envisioned to possess the following high-level characteristics to support RAN slicing:

- **Application-specific intra-slice resource allocation policies**: intra-slice resource allocation polices should be re-configurable, enabling the slice owners to allocate resources according to the service characteristics and requirements in the slice. For instance, resource scheduling policy in an eMBB slice should allocate radio resources to the users providing high data rate and/or fairness among users, while in an URLLC slice, scheduling policy must guarantee the required low latency and high reliability.

- **Multi-slice radio resource management**: multi-slice radio resource manage-

ment policies must allow operators to concurrently maintain various RAN slices with the required level of radio resource isolation, customization and quality of service requirements. For instance, URLLC services have stringent latency and reliability requirements, that often can be satisfied only with static multi-slice radio resource allocation or hard spectrum slicing. This comes at the cost of effectively lower spectrum efficiency and hence a waste of scarce resources. On the other hand, for services with relaxed performance requirements dynamic spectrum sharing or soft slicing may prove to be cost-efficient. Hence, the choice of the type of slicing depends on the trade-off between more deterministic QoS provisioning through isolation and a higher resource efficiency through sharing.

2.1.2. Slice resource isolation

The choice of a slicing model depends on the trade-off between the desired level of isolation between the slices and the efficiency of the utilisation of the underlying physical resources. The key aspect which controls this trade-off is the timescale of resource sharing among multiple slices. This can vary from infinite which corresponds to static resource sharing to extremely dynamic, i.e. no slicing and just scheduling. Anything in between would be slicing with dynamic assignments with two hierarchical allocation layers: allocation of resources to slices and intra-slice scheduling.

- **RAN slicing model with static resource sharing**

  Static resource sharing between RAN slices is achieved by allocating a dedicated portion of spectrum to each slice for the lifetime of the slice. This approach provides functional isolation among RAN slices. Functional isolation means that a slice can be independently customized for functionality in terms of radio access technology, radio frame structure and medium access control (MAC) scheduling policies [55]. However, static resource sharing model results in inefficient use of the radio resources (when the resource of one slice is not fully utilized, it cannot be used by other slices) and offers no statistical multiplexing gains.

- **RAN slicing model based on dynamic resource sharing**

  In this model all slices employ the same radio access technology, have common physical layer, RAN protocols, control plane and share the same radio hardware and spectrum [55].

  In this model, an infrastructure provider controls the operations of all slices, such as radio resource allocation through a common scheduler. As can be concluded, this model enables statistical multiplexing of physical resources but lacks the support of strict QoS guarantees and functional isolation [55].

2.2. Resource management problem in RAN slicing

The primary objective of radio resource management in RAN slicing is to efficiently allocate RAN resources to various slices while meeting the user QoS requirements [57]. Prior works in efficient resource management in sliced networks have addressed two distinct problems: *slice admission control* and *multi-slice resource allocation*. Slice admission control refers to the mechanism adopted by an infrastructure provider to accept slice creation requests by tenants such that the resource efficiency is maximised and slice service level agreements (SLAs) are guaranteed. In contrast, multi-slice resource allocation includes approaches to efficiently share network resources among *active* slices dynamically to maximize resource efficiency and revenue for both infrastructure provider and slice owners or tenants. This is achieved by minimizing the consumed resources while guaranteeing the traffic SLAs within a network slice [56].

The existing approaches of resource allocation in RAN slicing can be broadly classified into two categories, namely model-based methods and artificial intelligence-based methods. The model-based methods define the resource allocation problem as a constrained optimization problem with the objective of maximizing a predefined overall network utility, while satisfying the QoS constraints of admitted slices [57]. The optimal solution for such problems is identified to be NP-hard [24]. Sub-optimal solutions have been proposed in prior works for these problems which use heuristics or iterative optimization algorithms.

The use of artificial intelligence techniques has gained recent attention for radio resource management (RRM) in mobile networks, especially in network slicing. This is because AI-based RRM methods offer reduced computational complexity as compared to the model-based methods [24]. Moreover, AI-based methods can provide accurate service- or slice-specific traffic predictions [57]. Only with an accurate prediction of slice traffic or slice resource requirement, effective and efficient resource allocation can be done to accommodate slice demands for future time periods [57].

AI techniques such as reinforcement learning (RL) and supervised machine learning provide opportunities to solve the RRM problem in sliced RAN. In the following sections we present some background on supervised machine learning and reinforcement learning.

2.3. Supervised machine learning

Supervised machine learning refers to the set of algorithms that learn a function that maps a set of input variables to an output value based on example input-output pairs. The example input-output pairs form a *labelled training data*. The supervised ML infers a mapping function from the input variables to an output by finding patterns in the training data and is expected to generalise to unseen situations. Supervised ML is used in two problem domains namely, classification and regression. In classification, the input vectors are mapped or categorised into a set of classes. Classifying a picture as that of a dog or a cat is an example of the classification ML problem. On the other hand, in

regression the algorithms maps a set of inputs variables to continuous-valued output. An example for regression ML problem is the prediction of temperature based on input variables such as region, date, humidity and wind speed.

Various algorithms exist under the class of supervised machine learning. We discuss three such algorithms in the following sections. These are namely, multiple linear regression (MLR), random forests (RF) and supervised deep learning or artificial neural networks (ANN).

2.3.1. Multiple linear regression

Multiple linear regression (MLR) is a supervised ML technique in which a linear relationship is determined between the (multiple) input variables and the output of the defined ML model in an underlying physical system. A multiple linear regression is expressed by the following:

$$y = b_0 + b_1 i_i + b_2 i_2 + b_3 i_3 + ... + b_n i_n \tag{2.1}$$

where y is the output variable, $i_1, i_2, ...i_n$ are the input variables with $b_1, b_2, ...b_n$ as their respective coefficients.

MLR commonly utilises the ordinary least-squares error regression that minimises the sum of squared errors between the predictions of the output variable and its true values to find the values of the coefficients that define the linear relationship. However when the objective is to minimize relative errors between the predictions and true values, least-squares relative errors which minimizes the sum of squared relative error is relevant.

2.3.2. Random forests

Random forests is another supervised machine learning algorithm that can be used for both classification and regression problems. RF is based on the concept of ensemble learning. In ensemble learning, multiple learning algorithms are combined to solve a particular problem. Random forests algorithm constructs multiple so-called decision trees. The training data-set is divided into various subsets. Each subset is assigned to a decision tree that predicts the output for each subset. The mean of the outputs by the different trees is the prediction output for the RF. This technique is known as bootstrap aggregation or *bagging*.

As decision tree is a predictive model that uses so-called decision making based on the values of the different inputs in the training data set to make conclusions about the output. A decision tree consists of so-called nodes at which decisions are taken to move in a particular direction. The so-called leaf nodes of the tree give the the final output. In the case of regression, the leaf node represents numeric values and in the case of classification represent the different classes. A decision tree with five nodes and three leaf nodes is illustrated in Figure 2.1. When the value of the output is to be predicted given a single input , the following steps are taken: at each non-leaf node, a decision is made based on the value of the input until a leaf node is reached which gives the final

output. For instance, if the value of the input is less than 3, then the output is 10. If it lies between 3 and 15, then the output is 20 and if the value of the feature is greater than 15, then the output is 30.

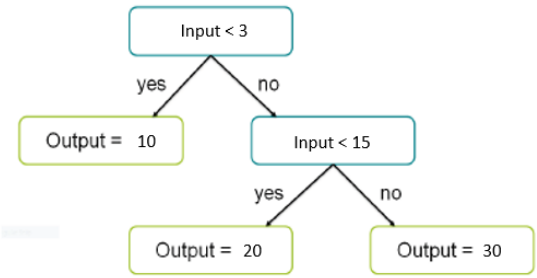

Figure 2.1: An example of a decision tree with five tree nodes.

2.3.3. Supervised deep learning (or artificial neural networks)

Supervised deep learning or deep learning (DL) has emerged as a popular machine learning technique. It enables to make predictions, classifications, decisions based on raw data, without explicit feature definition and programming by domain experts [71].

DL allows to approximate complex mathematical functions through a composition of sum and product operations via units called neurons. The mathematical functions represent the functioning of real-world systems and are learnt from the data generated by these systems. DL is employed to perform tasks like classification into multiple classes, linear and non-linear regression for prediction. A deep learning architecture consists of layers of neurons. A set of input data travels through these layers to output some information depending on the model. Hence, there is an input layer, an output layer and more than one hidden layers, for the architecture to be *deep*. In a fully-connected architecture, each neuron of a layer is connected to every other neuron of the adjacent layer. A deep and fully-connected neural network architecture is illustrated in Figure 2.3. The architecture has two hidden layers, four inputs and is used for classification into two classes as given by two neurons in the output layer. The connection between two neurons or equivalently input connection to a neuron has a weight. As data passes through a neuron, it undergoes some operations as illustrated in Figure 2.2.

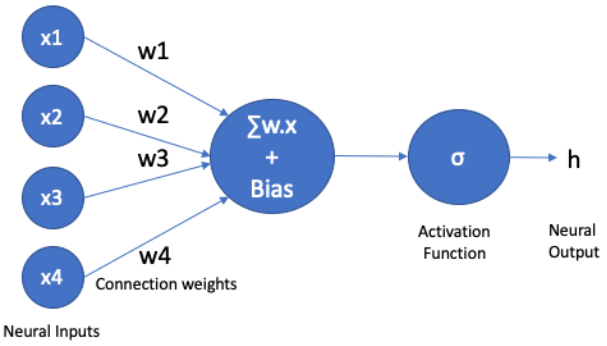

Figure 2.2: Basic artificial neuron structure.

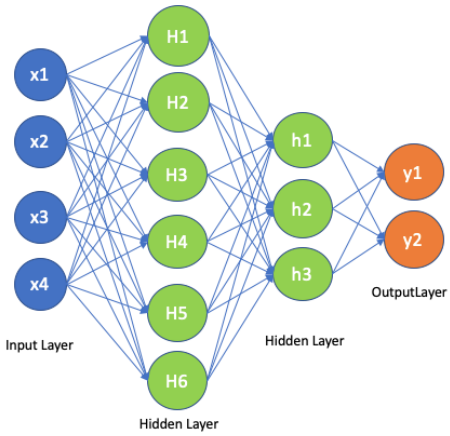

Figure 2.3: Architecture of a deep and fully-connected neural network with two hidden layers.

w_i is the weight of the i^{th} connection and σ is the activation function of a hidden layer. Activation functions allow to learn non-linear relations between inputs and the model output. Popular activation functions are the sigmoid given by: $sigmoid(x) = \frac{1}{1+e^{-x}}$ and Rectified Linear Unit (ReLU) defined as $ReLU(x) = max(x, 0)$. The output of a neuron n with M inputs is given by :

$$h_n = \sigma(\sum_{i=1}^{i=M} w_i x_i + b_n) \tag{2.2}$$

b_n is the bias term of a neuron and x_i is the i^{th} input.

Forward and backward propagation

The process of function approximation using DL is actually learning the correct weights of the neural connections. The biases of the neurons remain constant during the learning process. The weights are *learned* based on a set of *training data*. The learning

and inference processes in a deep neural network are illustrated in Figure 2.4 [71]. In forward propagation, the input moves through the hidden layers to the output layer. The input data is processed at each neuron according to Equation 2.2. At the core of the learning process is the loss function $\mathcal{L}(\mathbf{w})$. $\mathbf{w}$ is the vector of w_is i.e. the weights of the neural connections. It measures the error between the output of the network and the *true* output. The objective of learning or training is to find the best weights $\mathbf{w}$, which minimize the loss function $\mathcal{L}(\mathbf{w})$. This is achieved by the back propagation through gradient descent [71].

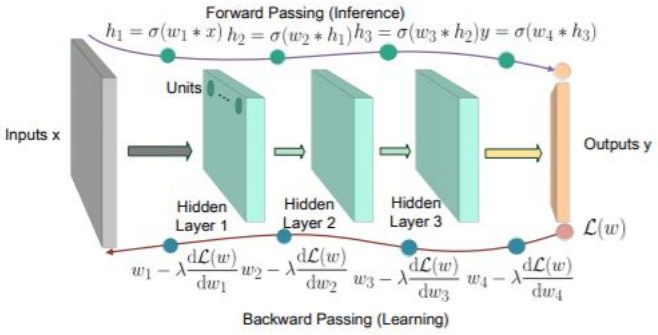

Figure 2.4: Learning and inference in deep learning [71].

In gradient descent, the weights are updated using the gradient of the loss function $\mathcal{L}(\mathbf{w})$. The update equation is given as:

$$\mathbf{w} = \mathbf{w} - \theta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \tag{2.3}$$

Here θ is the learning rate, which controls the step size of moving in the direction indicated by the gradient. In stochastic gradient descent (SGD), the above update is made for each training data sample. The process is repeated until the algorithm converges.

Three architectures exist under the class of supervised deep learning or ANNs: multilayer perceptron (MLP), convolutional neural network (CNN) and recurrent neural network (RNN).

Multilayer perceptron (MLP)

Deep MLP is a basic ANN with at least two hidden layers and a fully-connected structure. The structure of a deep MLP is shown in Figure 2.3.

The MLP design is easy and straightforward and performs well in feature extraction problems. Hence, MLPs can be used in models built to solve specific problems in mobile network applications [71]. In this thesis, we propose a MLP architecture to aid resource allocation in RAN slicing (details in section 8).

Convolutional neural network (CNN)

Unlike MLP, CNNs do not have fully-connected layers. Instead they use a set of locally connected filters to capture correlations between different data regions [71]. This decreases the number of model parameters to be learnt. CNNs are good in finding spatial-patterns and hence are commonly used in image processing/recognition applications.

Recurrent neural network (RNN)

While CNNs are good in finding spatial correlations in data, RNNs can model sequential data well, where sequential or temporal correlations exist between samples. In a RNN, the output of a layer not only depends on the input vector but also on the internal state (memory) of the network. The internal state is updated with every new input. It is a function of the current input and the previous state. Hence, in this way, a RNN takes into account the prior inputs to generate an output. Standard RNNs suffer from issues of gradient vanishing and exploding. As discussed in section 2.3.3, in gradient-based learning methods (e.g. SGD) and backpropagation, weights of a neural network are updated in proportion to the partial derivative of the loss function with respect to the current weight at each iteration of training. In some cases, the gradient might become very small, preventing the weight from changing its value and possibly stopping the neural network training. This is called the gradient vanishing problem. On the other hand, in the exploding gradient problem, large error gradients are encountered with some activation functions whose derivatives can take large values. This results in very large updates to the neural network's weights which prevents the model from learning. These problems makes it difficult to train RNNs. A variant of RNNs, called long short term memory mitigates these issues [20]. LSTM RNNs have been proved to perform well in applications like speech recognition, weather prediction and text categorization. For more details, readers are referred to [71].

Over-fitting and Under-fitting

A trained neural network model is required to adequately learn the rules of the underlying system from the training data and also to generalise well from the training data to new and unseen sets of input variables (i.e. the sets of input variables that are not in the training data set). A model that doesn't meet the above requirements is said to either *over-fit* or *under-fit* the training data. An over-fit model is one that models the training data so well that it learns the detail and noise in the training data resulting in less accurate predictions for new data as compared to the training data. On the other hand, an under-fit model is unable to adequately capture the underlying structure of the training data and hence it neither models well the training data nor generalises well to the unseen data. Under-fitting is easily addressed by increasing the model capacity i.e. using more layers and neurons in the model structure. However, preventing over-fitting

is a challenging problem. The primary reason for over-fitting is that the model has higher capacity than required. This means that the information contained in the training set is not enough to train all the neurons so that the model can actually *memorize* the training data. The reason for limited information and hence over-fitting can also be a biased data set. With proper choice of the hyper-parameters and the training data, over-fitting can be prevented.

Early stopping and Drop-out method

Early stopping and drop-out are techniques to prevent over-fitting in neural networks. Dropout involves leaving out some nodes in the neural network during training. Early stopping prevents the model to assume the weights learnt during the last epoch which might not be the best weights and might lead to higher errors. Instead, it stops the training earlier than finishing all the epochs. This is done based on a patience value. For example, a patience value of 50 means that if during training the errors do not decrease for consecutive 50 epochs, then the training is stopped and the weights thus obtained are used.

2.4. Reinforcement learning

Reinforcement learning is the area of machine learning where an agent learns the optimal actions to be taken in various states of an environment with the objective of maximizing some notion of a *cumulative reward or return*. It learns the optimal state-action *policy* through repetitious trial and error.

On interaction with the environment, the agent receives a reward $r \in R$. The reward directs the agent towards optimal actions in each environment state $s \in S$. The policy of an agent is determined by the maximization of total reward over time called *return*. The return is computed as $\sum \gamma^{t} r_t$. γ is the discount factor that is used to reduce the weight of future rewards, compared to immediate ones when calculating the return. t is the time instant at which a reward is obtained. The reward function definition determines the performance of a RL method. Hence, domain knowledge is important to define the goal of the agent, capturing all relevant aspects of the problem.

In RL, the agent has to take actions given an environment state that maximizes the total reward. A RL problem is commonly modeled as a decision process that obeys the Markov property. The Markov property states that the conditional transition probability (conditional on present and past states) from a state s_t to a future state s_{t+1} depends only upon the present state s_t, not on the sequence of events that preceded it [18]. A process with this property is called a Markov decision process. An MDP is characterised by a known Markovian state transition model. However in RL, the underlying state transition model is unknown in advance but instead is learnt on-line via agent-environment interactions. The RL problem in an MDP framework is defined using the tuple: (S, A, P, r, γ), where S and A are the state and action spaces respectively, $P(s_{t+1}|s_t, a_t)$ defines the transition probability from a state s_t to state s_{t+1} after taking an

action a_t, r is the reward function and γ is the discount factor. The transition probability distribution (or transition model) and the reward function are often collectively called the 'model' of the environment.

The objective of RL is to learn an optimal policy $\pi^*(s) \rightarrow a \; \forall s \in S$ and $a \in A$ that maximizes the return (expected discounted reward over time). That is,

$$\pi^* = \arg\max_{\pi} \mathbb{E} \left[\sum_{t=0}^{t=T} r_t \mid \pi \right] \tag{2.4}$$

There are two classes of RL algorithms, namely model-based and model-free. In model-based RL, the agent learns the transition probability distribution and the reward function associated with the Markov decision process to learn the policy. In contrast, model-free RL algorithms do not use the model of the environment to learn an optimal policy, hence the name 'model-free'.

Three fundamental challenges exist in solving RL problems. Policy learning in RL is based on reward observations for various state-action pairs. However, in an MDP setting, the reward obtained at any time-step is the contribution of a sequence of actions taken at previous time-steps. Hence, it is challenging to assign an appropriate credit for the reward to a state-action pair. This is the **credit assignment problem** in RL.

Second challenge is to address the **exploration vs. exploitation trade-off**. Choosing actions known to provide maximum reward in a state based on what the agent has learnt so far is called exploitation. On the other hand, randomly choosing actions is known as exploration. Exploration allows to expand the agent's knowledge about its environment and often leads to a higher return [61]. Various methods like the ϵ-greedy algorithm [18] and Thompson sampling [13] are used to handle the exploration problem in RL. However, exploring all possible state-actions is computationally demanding, especially in complex environments. Moreover, in real world learning settings, exploration may lead to drastic outcomes. As an example, consider the case of autonomous driving where an RL agent has to learn to drive a car. During exploration in the real world, the agent may take actions leading to outcomes such as crashing into a tree or hitting a pedestrian. Although the agent will eventually learn not take such actions, these are drastic outcomes that absolutely cannot occur.

The third challenge is to learn an optimal policy in a reasonable amount of time. Often RL problems have large state-action space. As the number of states and/or the number of action increase, the time required to converge to an optimal policy increases substantially.

Readers are referred to [61] and [18] for a detailed introduction to RL methods.

An important breakthrough in RL was the development of an off-policy temporal difference (TD) RL algorithm called Q-learning. Q-learning falls under the class of model-free RL algorithms. Central to Q-learning is a Bellman equation as a value iteration update [18]. The equation can be viewed as the weighted average of the old value and some new information obtained after taking an action. Hence, one-step Q-learning is defined as:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \theta \times [(r_{t+1} + \gamma \times \arg\max_a Q(s_{t+1}, a)) - Q(s_t, a_t)] \tag{2.5}$$

where θ is the learning rate and $Q(s, a)$ is the Q-value of a state-action pair.

Q-values or the Q-function $Q(s, a)$ reflect the importance of taking an action a in a given state s and thereafter following an optimal policy in terms of the expected return. Q-learning is called a TD algorithm because the Q-values of a state-action pair can be updated after every time step or after taking an action. Q-learning is an off-policy algorithm because the Q-values updates are done using the Q-value of the next state s_{t+1} and the *greedy action* ${a^*}_{t+1}$. Moreover, Q-learning is an iterative algorithm.

Q-learning can identify an optimal action-selection policy for an MDP if infinite exploration time and a partly-random policy is used [18]. Hence, it needs a lot of iterations or experience for convergence. In the simplest form, Q-learning involves storing the Q-values in tables. This approach cannot be used in problems with a huge state/action space or a continuous state/action space. This is because, in case of a huge state/action space, the probability that the agent visits a particular state and performs a particular action gets increasingly small [18]. Q-learning can be combined with function approximation (FA) to solve problems with huge state/action space [25]. One way to apply FA is to use an artificial neural network. ANNs are great function approximators [25]. They can speed up learning by generalizing the agent's earlier experiences to previously unseen states. This is called deep Q-learning (DQL) or deep reinforcement learning (DRL). In DQL, a neural network is trained such that it approximates, given a state, the Q-values for each possible state-action pair. For training, the neural network weights, given by weight vector $\mathbf{w}$ are updated to minimize a loss function at each step k. The loss function in DQL is described as follows:

$$L_k(\mathbf{w}_k) = \mathbb{E}_{s,a,r,s' \ \rho(.)} [(y_k - Q(s, a; \mathbf{w}_k))^2] \tag{2.6}$$

where y_k is called the TD (temporal difference) target, and $y_k - Q$ is the TD error. ρ is the distribution over transitions (s, a, r, s') collected from the environment. The target, y_k is given by the following equation:

$$y_k = r(s, a, s') + \gamma \times \arg\max_{a'} Q(s', a'; \mathbf{w}_{k-1}) \tag{2.7}$$

The weight update can be done via stochastic gradient descent. Hence, the weight update rule is given by:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \theta \times \nabla_{\mathbf{w}_k} L_k(\mathbf{w}_k) \tag{2.8}$$

where θ is the learning rate that determines the step size of the weight update. $\nabla_{\mathbf{w}_k} L_k(\mathbf{w}_k)$ is the gradient of the loss function computed at step k.

It is important to note that, in target equation 2.7, the term $(\gamma \times \arg\max_{a'} Q(s', a'; \mathbf{w}_{k-1}))$ is variable as a result of the learning process itself. Hence, the target in a DQL problem is variable unlike typical deep learning problems with a stationary target. The problem of a variable target is addressed by having two neural networks- a primary network and a target network. Both have the same architecture. The weights of the primary network are adjusted according to equation 2.8, while the target network has frozen weights. The target network is used to compute the target. After some iterations in the primary network, the weights are copied to the target network.

2.4.1. Contextual bandits

We have discussed RL in an MDP setting or framework in Section 2.4. Here, another framework of RL called contextual multi-armed bandits (CMABs) is discussed. In CMABs, a set of contexts or states are specified and the expected reward of an action depends on the state and action, $r(s, a)$ [70]. However, unlike MDPs, the next state is independent of the current state and the action. In fact, CMABs are a simplification of MDPs [70].

The MDP-RL problems face many challenges such as credit assignment, learning in real environments from limited samples and no convergence guarantees. These challenges hinder the practical applications of MDP-RL in many problems. Alternately, CMAB is a simplified reinforcement learning paradigm. CMABs allow to create algorithms of a reinforcing nature with convergence guarantees [35]. They require exploration but unlike MDP-RL do not require dealing with credit assignment [35]. Moreover, CMABs perform better than MDPs in terms of computational complexity and statistical efficiency [70].

As discussed above, CMAB-RL is a simplification of MDP-RL. The difference between the two is illustrated in figure 2.5. In CMAB, the agent observes a context/state, makes a decision of selecting an action from a number of alternative actions, and observes the reward for the outcome of its decision. The objective is to learn an optimal decision-making policy that maximizes the expected reward at each state.

From figure 2.5, it can be observed that in CMAB, the RL problem is a one-step learning process, where the reward at time t depends only on the state-action pair at time t. However, in MDP RL, the reward at time t additionally depends on the prior state-action pairs or prior decisions of the agent.

CMABs are widely employed in recommendation systems and user-aware online advertising [67]. Existing algorithms for CMABs can be classified into two categories- *online linear bandits* such as contextual Thompson sampling [6], LinUCB [39] and *online non-linear bandits* such as Neural Bandit [52] and Banditron [34]. In the linear bandits, the expected reward, $\mathbb{E}\ [r_t|s_t, a_t]$ is modelled as a linear function of the state s_t and action a_t. The non-linear bandits show superior performance than the linear bandits, especially when the underlying system is non-linear [10].

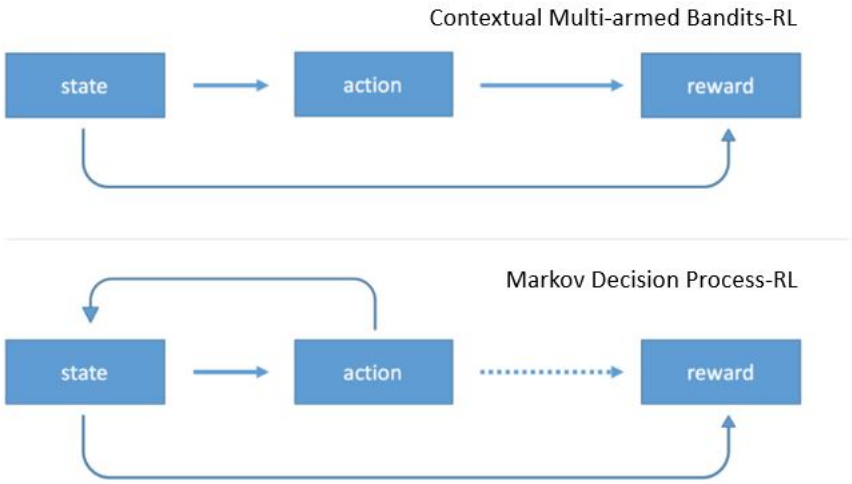

Figure 2.5: CMAB-RL vs MDP-RL [33]

In this algorithm, K MLP neural networks, one for each action $k \in [K]$ are used to predict the probability that the action results in reward 1 (binary rewards case), given a state s_t at time t. The ϵ-greedy algorithm is used as the exploration strategy. The probability of exploring or playing a random action is ϵ. Based on the prediction of the best action (action with the highest predicted probability) and the exploration strategy, an action k_t is taken at t and the reward $y_{s_t,k_t} \in \{0, 1\}$ is obtained. Backpropagation with gradient descent is used to update the weights of the neural networks. Note that a similar process is done in DQL, where a neural network, given a state, predicts the Q-values of all actions. However, in DQL the target is not stationary (refer to equation 2.7). This is because in DQL, the Q-value of a state depends on the Q-value of the next state. This adds the complexity of having a separate target neural network in DQL.

3

Scope of the Study

In this thesis, we focus on multi-slice resource allocation in radio access networks. Specifically, we focus on sharing of physical resource blocks among multiple RAN slices.

The limitations of the model-based optimization methods for resource management in a sliced RAN motivate the use of artificial intelligence techniques for resource allocation problems. Hence, we focus on two such techniques, namely supervised machine learning and reinforcement learning.

We propose a novel supervised ML based architecture for resource demand prediction, that is independent of historical resource usage data (details in Chapter 8).

In addition to the resource demand prediction using supervised ML, we evaluate the use of reinforcement learning to aid resource allocation in a sliced RAN. The proposed model has a simplified state and action definition (details in Chapter 8), which allows for faster policy convergence even with limited agent experience. To achieve this we formulate the RL problem for multi-slice resource allocation in a CMAB setting (details in Chapter 8) which is a simplified RL paradigm.

4

Related Work

In this chapter, we discuss the related works done in the field of resource management in RAN slicing. As already discussed in Chapter 2, the existing approaches of resource allocation in RAN slicing are broadly classified into two categories, namely the model-based methods and the artificial intelligence-based methods. In Section 4.1, we discuss prior works in model-based methods for RAN slicing and point out their limitations. Then we shift our focus to AI-based methods in Section 4.2. Some conclusions are presented in Section 4.3

4.1. Model-based multi-slice resource allocation

Recall that the model-based methods define the resource allocation problem as a constrained optimization problem. The optimal solution for such problems is identified to be NP-hard. In most studies, sub-optimal solutions for these problems are found by heuristics or iterative optimization algorithms. These are discussed as follows.

The architectural aspects of RAN slicing based on resource abstraction are discussed in [37]. The authors propose a two-level MAC scheduler. First a slice-specific scheduler assigns virtual resources to the UEs in a slice based on a slice-specific scheduling policy. Then, a multi-slice scheduler maps the virtual resources to physical resources based on business requirements of the slice and accounting for slice resource limits. The authors remark that such a scheduling process is defined by a multi-objective function where the optimal solution is usually NP-hard.

Network virtualization substrate (NVS) [36] is one of the earliest works in virtualization of wireless resources in cellular networks. The NVS design includes a two-level scheduler, decoupling the flow scheduling from the slice scheduling. The notion of utility functions is used to reflect the business models between MNOs and slice owners. A utility-optimization problem is formulated and an optimal resource allocation algorithm is proposed.

Caballero et al. [12] propose a weighted proportionally fair algorithm for dynamic re-

source allocation between slices. In a dynamic and time-varying setting, the solution for multi-slice resource allocation problem is identified to be NP-hard. Hence, the authors design a distributed semi-online algorithm for near-optimal allocations.

In [38], a dynamic network slicing approach for multi tenant heterogeneous cloud radio access networks (H-CRANs) is presented. Network resources are shared between users associated with different tenants, taking into account tenants' priorities, system resources, quality of service (QoS) requirements and interference. The objective is to maximize the weighted network throughput of the multi tenant system, subject to capacity and QoS constraints. The capacity allocation problem is solved by linear programming.

Shrivastava et al. [58] propose a policy-based dynamic slicing algorithm for time division duplex (TDD) systems. The proposed scheme takes into account the heterogeneity in terms of the required UL/DL ratios in a frame per slice. Initial resource allocation and UL/DL configuration are based on the traffic demand of the slice. A soft resource allocation algorithm is then used to allow borrowing of resources from other slices to accommodate sudden traffic peaks.

Some works in the literature have proposed solutions for RAN/spectrum sharing. However, the proposed solutions are applicable in multi-slice resource allocation problem with some necessary changes. Hence, we discuss them briefly in this sub-section.

A spectrum sharing framework is proposed in [69] to share radio resources among heterogeneous base stations. To obtain an optimal bandwidth sharing solution, a network utility maximization problem is formulated. The method accounts for traffic statistics, user spatial distribution, multi-cell interference, cell load and QoS requirements of services. The problem is defined as a biconcave maximization problem and an alternative concave search algorithm is used to obtain partial optimal solutions. A similar approach is taken in [50] for dynamic spectrum sharing in autonomous vehicular networks.

In [26], the authors present an application-oriented framework for RAN sharing among mobile (virtual) network operators called 'AppRAN'. It computes application level resource allocation for each base station at a central controller based on traffic demands and average channel condition. The resource allocation problem is formulated as an optimization problem and a heuristic algorithm is proposed to solve it.

The above discussed model-based multi-slice resource allocation methods have the following limitations:

- The requirement for a mobile traffic resource demand model: the discussed methods assume that a mobile traffic resource demand model is known in advance. This however is not a practical assumption given highly time-varying wireless networks.

- The policy-based multi-slice resource allocation methods are integer programming problems, where the amount of resources allocated to a slice are positive integers. These problems are known to be NP-hard, requiring high computational effort to optimize the policy through exhaustive search [24].

The above discussed limitations undermine the practicality of the model-based optimization methods and motivate the use of artificial intelligence techniques for resource allocation problems.

4.2. Artificial intelligence-based multi-slice resource allocation

Artificial neural networks (ANNs) are highly efficient in modeling non-linear systems. Hence, they are commonly used in model estimation and prediction of complex non-linear processes [24]. Mobile environments are characterised as highly complex, heterogeneous and dynamic, making various network management tasks extremely difficult to model via conventional methods [71]. For this reason, many recent works have explored the use of ANNs in areas like network state prediction, traffic classification, user mobility analysis, mobile data analysis and network resource management [71].

The RAN slicing approaches that employ neural networks or supervised deep learning can be categorised into two classes. One class focuses on mobile traffic prediction while the other focuses directly on resource demand or capacity forecast to aid resource allocation.

Furthermore, a RAN slicing method must flexibly allocate radio resources according to the slice status and the wireless mobile environment. Deep reinforcement learning (see Chapter 2) has been proposed in the literature as a control mechanism to achieve optimal resource allocation.

RAN slicing based on mobile traffic prediction

Mobile traffic prediction, employing both AI and non-AI methods, has been explored in mobile networks not just for RAN slicing but also for other network management applications such resource management, load balancing and energy savings. The proposed methods are inspired by the observations from real networks that traffic arrival rates and *aggregate* traffic possess strong temporal and spatial correlations [68]. The aggregate traffic at a BS fluctuates during a day. However, the fluctuations show a repetitive pattern over a day and do so everyday, indicating strong correlation over time [68]. Hence, most studies on mobile traffic or resource usage prediction employ time-series prediction.

Recent studies have proposed the use of deep neural network (DNN) and long short-term memory (LSTM) to forecast slice traffic load. For instance, in [68], a DNN is employed to predict traffic arrival rates. The proposed method exploits the historical information of the traffic arrival rate data to predict the traffic arrivals at all BSs at each time slot with a DNN.

In [41], the entropy theory is utilised to predict the mobile traffic based on temporal, spatial and multi-service relationships over time and across cells. The authors analyse the role of traffic prediction for efficient resource management in cellular radio access networks empowered by SDN.

The authors in [56] propose a model that performs an aggregate traffic forecast for

a slice per cell utilizing the Holt-Winters theory. The traffic forecast is coupled with the forecast of the slice users' spatial distribution in the network to make slice admission control decisions and slice resource allocation decisions at cell level. A heuristic algorithm is proposed to solve a minimization problem with the aim to minimize consumed resources by a slice while guaranteeing the SLAs within a slice. Here a key assumption is that the slice traffic follows a periodic pattern. The Holt-Winters theory is only applicable when the underlying system is periodic [56].

A slice capacity broker model is proposed in [63] that takes resource allocation decisions on a coarse time scale based on slice traffic volume forecast. The authors examine four short-term time series forecasting methods, namely ARIMA, compressive sensing-based method, Kalman filter and Holt-Winters. Holt-Winters and Kalman Filter are shown to perform the best due to their ability to capture seasonality patterns in the input traffic data [63].

Forecast of slice traffic volumes facilitates the prediction of cell load for different network slices. This aids in proactive and accurate dimensioning of network slices and hence allows to maximize the overall system resource utilisation. The existing studies illustrate the benefits of AI-based prediction methods to accurately capture mobile traffic patterns [57].

RAN slicing based on resource demand prediction

The second category of neural network-based prediction methods steps away from mobile traffic prediction and focus on predicting the resource demand/usage/capacity of a slice. For instance, in [9], the authors identify that conventional mobile traffic prediction based resource allocation methods have limitations. For example, in these methods, slice resource overdimensioning to minimize SLA violations is not taken into account. Hence, these models lead to frequent capacity violations. To overcome these limitations, they propose a deep learning architecture for capacity forecast. The model consists of an encoder, followed by a decoder. The encoder is a three layered three-dimensional convolutional neural network (3D-CNN) and the decoder is a multi-layer perceptron. A data driven model acts as a slice orchestrator and makes slice admission control and multi-slice resource allocation decisions. The proposed model overcomes the limitations of conventional traffic predictors by explicitly providing a *capacity* forecast that mitigates the effect of underprovisioning. To this aim, they propose a loss function which drives the system towards an optimal trade-off between slice overprovisioning and SLA violations.

In [11], the authors propose a modified LSTM neural network (X-LSTM) architecture for average slice resource demand prediction. To this aid, they define a new metric called REVA which captures resource usage of a slice. The X-LSTM model uses historical values of REVA to predict slice resource usage for multiple time steps in future.

Both traffic and capacity prediction models use historical observations to predict future traffic volumes/capacity i.e. they utilize time series prediction. Hence, the underlying key-assumption in these models is that slice traffic volumes/resource usage follow a periodic pattern, which is needed to apply time-series forecasting algorithms

[56]. Therefore, these models cannot be applied for slice capacity initialization, in cases where limited or even no historical data is available and when underlying system/input data is not periodic.

RAN slicing using reinforcement learning

Applications of RL, specifically DRL have been studied in mobile networks, specifically for resource allocation problems. Prior works have defined the concerned RL problem in a MDP setting (see Chapter 2). Below we discuss some of the prior works and point out their limitations.

In [42], the author proposed a resource allocation method using so-called deep Q-networks (DQN). They use the number of active slices and a vector of demands for each slice as the state. The action is a bandwidth sharing solution i.e. a vector of bandwidth allocated to each slice and the reward is calculated using spectral efficiency and a service quality of experience satisfaction ratio. The authors showed that the proposed DQN method implicitly incorporates the relationship between demand and resource allocation. However, the proposed definition of the DQN model makes the state and action space infinite and consequently makes the model convergence slow. Furthermore, the model assumes that the demand for each slice is known at every resource allocation interval. This assumption is not practical since the future resource demands of the slices are not known beforehand. Finally, the number of slices is fixed when learning the DQN model, thus if the number of slices changes, model re-training is needed.

In [60], the authors propose a DQN model that is independent of the number of slices. However, the state space is still continuous, requiring a large number of trials/experience to converge. Another interesting observation is that the state definition and reward definition both include slice QoS utility value and resource utilization, making these two variables in the state definition redundant. The proposed method first makes an initial reservation based on initial slice requirements and applies DQN thereafter to adjust the slice resources based on the QoS utility and resource utilization feedback. This way, the slices occupy only the required fraction of the system resources. However, a limitation of the method is the slow adaptation of slice resources in case new user flows start in the slice. This is because the DQN agent does not know the resource requirement of the new user flows directly.

In [4] also, the authors propose a DRL-based radio resource allocation method, that is independent of the number of slices. The state is defined using three variables- the traffic volume, bandwidth allocation and the user satisfaction of each slice. The resource allocation decision is taken separately for each slice based on the user satisfaction of the slice. Hence, the problem is defined as a multi-agent RL problem, where one agent is assigned to each slice. However, learning in a multi-agent setting is complex due to issues like multi-agent credit assignment and global exploration in a joint action environment [30]. Furthermore, the resource reservation of slices must account for the *future* traffic volumes, while the proposed method with traffic volume in the state definition considers only the current traffic volume. In [5] the authors also propose a multi-agent DRL

method to make the multi-slice resource allocation method independent of the number of slices. However, the method suffers from the same limitations.

Hap-SliceR [7] utilizes deep Q-learning to make multi-slice resource allocation decisions. It is a network-wide radio resource slicing framework, especially designed to support haptics applications over 5G networks. Haptics applications require ultra-low latency communications. The proposed model accounts for slice dynamics in terms of resource utilization and resource allocation as well as the QoS performance in the state definition, similar to [60]. Again, the defined state space and action space are huge, making policy convergence slow.

Learning in RL problems is predominantly online, that means, real-world interactions are required to learn optimal actions. Because of this, RL needs multiple trials to produce an effective model. This is feasible in situations when the task is simple and states and actions are discrete. If the state/action space is too large, it will take “infinite” trials to learn an effective model. The above discussed RL-based resource allocation methods suffer from a common limitation of a huge state/action space. This limits the practicality of these solutions.

4.3. Conclusion

A multi-slice resource allocation mechanism should meet two major objectives: first, minimize slice SLA violations. This is related to the slice performance and functional isolation. Second, maximize the system resource utilization efficiency. In the above discussion we point out that in order to achieve these objectives, prediction or estimation of slice resource requirement is needed to reserve the resources beforehand. To this aid, the use of AI has been motivated in the literature. The state-of-the-art AI-based resource allocation methods fall into two groups, viz. time series prediction based models using ANNs and reinforcement learning based models.

The discussed prior works proposing resource or traffic prediction methods employ time-series prediction based on historical data. Hence, the implicit assumption in these models is that traffic or capacity prediction problem involves time dependency. The dependence of the *aggregate* amount of traffic in a network on time has been shown in [68]. It is observed that networks typically exhibit time patterns due to seasonal, weekly, daily, and hourly traffic variations. However, in sliced networks, forecasts must occur at the slice level, for specific mobile services in isolation. While time series forecasting can be employed for slices that are active for long enough duration, so as to generate sufficient and relevant resource usage data, it fails otherwise. Moreover, time series can also not be used for slice capacity initialization, when no historical data is present.

The discussed prior works proposing reinforcement learning-based resource allocation, motivate the use of deep Q-learning in a MDP setting. All the proposed models have continuous state-action space. In such formulation of the RL problem, a lot of experience in terms of states visits and actions in each state is required in order to converge to an optimal policy. The convergence time or the required amount of experience for convergence increases with the increase in the number of states and actions. This

hinders the practical applications of the proposed models in the literature. But at the same time, this motivates the use of CMAB which is a much simpler paradigm of RL as compared to RL in an MDP setting (see Chapter 2). To the best of our knowledge, no prior work has employed the CMAB-RL framework to address resource allocation in mobile network (or sliced RANs).

5

5G-HEART Ambulance Use Case

This study is done under the umbrella of the European 5G-HEART project. As introduced in Chapter 1, a 5G network is envisioned to support multiple vertical industries on a common infrastructure, leveraging the concept of network slicing. 5G-HEART addresses three vertical industries namely, healthcare, aquaculture and transport to support the applications for each on a common 5G infrastructure. The use case considered in this study addresses two vertical industries, namely healthcare/medical and transport. In this chapter, we describe the use case.

5.1. Overview

The use case considered in this study is that of an ambulance. Whenever an incident occurs that requires medical attention, an ambulance is sent to the place of incidence. The paramedic de-boards the ambulance and looks for the patient. The patient is then brought to the hospital. The journey of the ambulance as described can be divided into three phases. Phase 1 is when the ambulance is sent to the place of incidence, Phase 2 is when the paramedic finds the patient and provides any possible immediate care and Phase 3 is when the patient is brought to the hospital. The use case represents an emergency scenario, where the ambulance needs to reach the place of incidence and to the hospital as quickly as possible. This is done via so-called road traffic coordination, which is explained in Section 5.2. Furthermore, in situations when the patient needs immediate help, remote support from the doctors/specialists must be provided via the on-spot paramedic. In the considered use case, the ambulance supports various applications to fulfil the above-mentioned requirements, which are discussed in more detail in the following sections. These applications require support from a communications network for their operation and hence generate network traffic. The applications involved in road traffic coordination are classified as belonging to the transport vertical industry and are referred to as transport applications. The applications involved in providing remote medical assistance to the on-spot paramedic are classified as belonging

to the medical vertical industry and are referred to as medical applications. The network traffic generated by the transport and medical applications is classified into distinct priority classes namely, transport class and medical class respectively. The network traffic from the two priority classes is handled on separate RAN slices namely, transport slice and medical slice. These slices are configured as so-called *priority slices*. The remaining network traffic is handled on another RAN slice called the background slice. The background slice is configured as so-called *best-effort slice*. The priority slices are guaranteed some minimum resource availability, while a best-effort slice gets no such guarantees from the network. This will be compared with a single-slice alternative, using priority-based scheduling. In this study, the transport class is considered to have a higher priority than the medical class. This choice is made because the transport class involves road safety applications that must achieve high performance levels or otherwise may pose threat to life and property. On the other hand, the medical class involves video and haptics applications. Reduction in the performance levels of these applications may be handled by the on-spot paramedic and hence is not harmful to life or property.

In the following sections, we describe the applications supported by the ambulance from both the application and network point of view. The applications are categorised as belonging to three different phases in the ambulance's journey. We describe the data exchanged between the UEs and the network for each application. The application characteristics and the QoS requirements of the applications are discussed in Chapter 6 along with other modelling aspects of the study.

5.2. Phase 1: Drive to the site of incidence

In this phase, the ambulance moves from the ambulance centre to the site of incidence. The objective in this phase is to minimize the time of arrival of the ambulance at the place of incidence. This is achieved via *road traffic coordination*. Road traffic coordination includes clearing the route for the ambulance, avoiding collisions and navigational directives to the ambulance and providing green signal priority to the ambulance. The applications involved in Phase 1 belong to the transport vertical and hence are handled by the transport slice.

5.2.1. Application description

Collision avoidance includes coordination among vehicles to ensure safe and efficient navigation through intersections, lane changing, overtaking and entering/exiting highways. This is achieved by providing the drivers, an *overall view of the scene*. This in contrast to the current situation where in order to navigate, the drivers rely only on the behaviour of other vehicles, brake lights and traffic light signals. These only provide a partial view of the scene which might make it impossible to anticipate risky situations. With only a partial view, the source of a potential collision or lane blockage may not be known to the driver. For instance, the ambulance may be at a blind crossing or the

source of lane blockages may be hidden by other vehicles.

Collision avoidance as described above is realised with the assistance of a 5G cellular network. All vehicles near an intersection or on a highway send the information of their local view (gathered from their on-board sensors) to the network. This information, complemented by the information from static sensors on the road and traffic light status information is aggregated by so-called transportation controller placed in a cloud to create an overall view of the scene. Thus, the transportation controller is aware of the entire traffic situation near an intersection or on a highway, that is, the position of the vehicles, pedestrians, cyclists, status of traffic lights, presence of emergency vehicles such as ambulance, fire brigades and police cars. This information is stored in the form of so-called global map and transmitted to the vehicles. The global map shows the overall occupancy of the scene. In addition, vehicles on a highway or at an intersection may upload a real-time video of the scene in front of them in the case an ambulance is present on that highway/intersection. The transportation controller aggregates the information from the videos uploaded by the vehicles to create so-called 'beyond-vision video' of the scene. The beyond-vision video is transmitted to the ambulance for increased vigilance.

5.2.2. Radio communications aspects

The uplink and downlink data exchanged between the network (gNodeBs) and the UEs (vehicles, static road sensors, ambulance and intersection controller) is as follows:

- **gNodeBs:**
  The gNodeBs (gNBs) receive the road status information from various UEs (vehicles, static road sensors and intersection controller) in the uplink and transmits the 'global map' message to the vehicles and the ambulance in the downlink.
- **UEs installed in regular vehicles:**
  The vehicles are equipped with sensors like radar, accelerometer, GPS and a camera. Through these sensors the speed, location, lane intention, heading and obstacles in view of the vehicles can be detected. This information is aggregated in the form so-called local map by the vehicles themselves. A 5G-enabled UE is installed in the vehicles that communicates with the gNBs. The vehicles send uplink 'local map' message to the gNB. In addition, the vehicles may send in uplink, a real-time video of the scene in front of them in case an ambulance is present on the highway or intersection.
- **UEs installed in an ambulance:**
  The ambulance, in addition to sending the local map message in the uplink, can request green signal priority for the intersections if needed through a 'signal request' message. In response to the signal request message, the ambulance receives a downlink 'priority status' message which tells the ambulance whether it can pass through an intersection or needs to slow down in case priority could not be granted. In addition, the ambulance receives in the downlink beyond vision-video described in Section 5.2.1.

- **UE connected to the static road sensors:**
  Road radars, underground induction loops[1] and cameras are placed at the intersections and highways. These sensors are connected to a 5G-enabled UE that locally aggregates the sensor measurements in a 'road status' message and sends the message to the serving gNodeB.

- **UE installed at the intersection controller:**

  An intersection controller (IC) is placed at each intersection that is equipped with traffic lights. The IC knows the status of all traffic lights at the intersection and can control them. A 5G-enabled UE is installed at the IC. An ambulance approaching an intersection sends the signal request message to the gNB to request green light priority. This triggers the transportation controller in the cloud to assess the scene based on the global map and decide the new status of the traffic lights at the intersection. The status information is packed in a 'traffic light' message and transmitted by the gNB in the downlink to the IC. The IC responds to this message by updating the traffic lights and sending an acknowledgement in an uplink 'light status' message to the gNB. The gNB then transmits a downlink 'priority status message' to the ambulance, informing it about the updated status of the traffic lights.

Figure 5.1 illustrates an example of road traffic coordination at an intersection during Phase 1 based on [27]. Table 5.1 summarizes the data exchanged between the UEs and the gNBs in Phase 1.

[1]The induction loops are commonly found in The Netherlands, laid underground near traffic lights and on highways. They are used to sense metallic objects such as cars, bikes passing over them. The measurement from the induction loops can be used to estimate traffic density and speed of the vehicles.

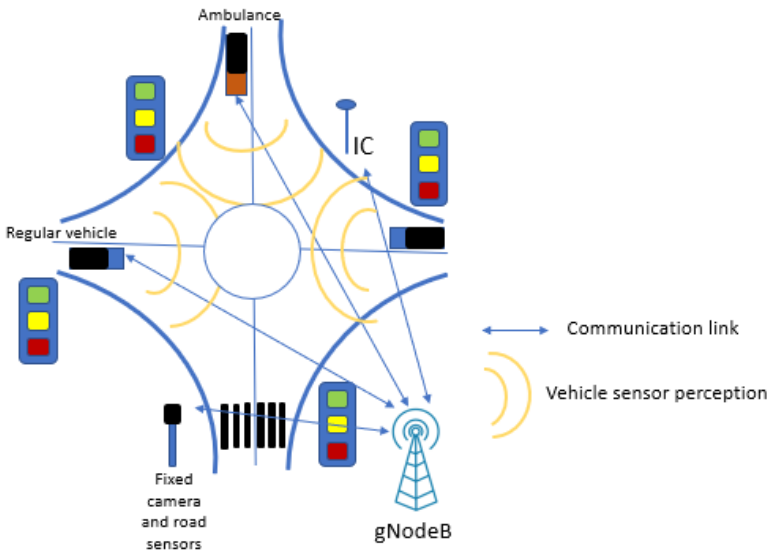

Figure 5.1: . Road traffic coordination at an intersection assisted by a 5G network.

Table 5.1: Data exchanged between the UEs and the gNB in Phase 1.

UE	Data	Direc-tion	Traffic Class
UE installed in regular vehicles	Global map message	Downlink	Transport
	Local map message	Uplink	Transport
	Real-time video	Uplink	Transport
UE installed in ambulance	Global map message	Downlink	Transport
	Local map message	Uplink	Transport
	Signal request message	Uplink	Transport
	Priority status message	Downlink	Transport
	Beyond-vision video	Downlink	Transport
UE connected to the static Road Sensors	Road status message	Uplink	Transport
UE installed at the IC	Traffic light message	Downlink	Transport
	Light status message	Uplink	Transport

In Phase 1, the network traffic generated by the ambulance and other ‘transport’ UEs (IC, regular vehicles and static road sensors) is present only in the transport slice. The medical slice does not carry any network traffic in Phase 1.

5.3. Phase 2: Paramedic de-boards the ambulance to attend to the patient

Phase 2 is when the ambulance has reached the place of incidence and the paramedic de-boards the ambulance to attend to the patient. The ambulance-specific applications involved in Phase 2 belong to the medical vertical and hence are handled by the medical slice.

5.3.1. Application description

In many cases, the status of a patient is not critical and can be easily handled by an on-site paramedic, possibly remotely guided by one or more specialists. This saves medical resources needed when a patient has to be brought to the hospital. On the other hand, in some life-critical cases, immediate action may be needed. In such cases, remote and precise monitoring of the patient by the specialists is required so that appropriate actions can be taken timely. This is achieved by multiple applications:

- **Streaming a high resolution (8K or 4K) video of the patient to the specialists at the hospital:** the video is recorded using a 360-degree camera mounted on the head or body of the paramedic. 360-degree[2] video streaming refers to the transmission of multi-angle video streams through the network. Using a 360-degree video, the specialists at the hospital can select the point of interest (camera angle) to be watched. This facilitates precise diagnosis of the patient and quick directions to the on-site paramedic.

- **Streaming ultrasound images and electrocardiogram (ECG) data**: ultrasound images and ECG data provide information of vital body functions of the patient for proper analysis of the patient's health by the specialist.

- **Using haptics for a proper ultrasound of the patient's body:** using the ultrasound machine for a proper body scan requires a specialist. The paramedic wears a haptic glove which is remotely controlled by the specialist. The specialist sees the ultrasound images. He adjusts his hand positions according to the feedback received from the ultrasound images. The hand movements of the specialist are communicated to the glove worn by the paramedic via *'glove position'* messages. These messages trigger the generation of corresponding electric signals in the glove which makes the paramedic *replicate* (with high accuracy) the hand movements of the specialist.

- **Audio-video connection with the remote specialist:** the on-spot paramedic may need consultation from the remote specialist as he attends to the patient.

[2] a 360-degree video camera is equipped with multiple cameras pointing to different directions from a single ball-shaped device and therefore forming 360-degree coverage from the operation environment.

5.3.2. Radio communication aspects

The 360-degree video camera, the portable ultrasound and ECG machine and the haptic glove are assumed to be connected to 5G-enabled UEs. The UEs communicate the respective information of the devices to the network (gNB) and vice versa. In addition, the paramedic consults the specialist via an audio/video connection (e.g. video calling). Figure 5.2 illustrates an example of communication between the UEs at the patient side (controlled by the on-spot paramedic) and UEs at the remote specialist side based on [28]. The data exchanged between the UEs and the network (gNB) in Phase 2 is described in Table 5.2.

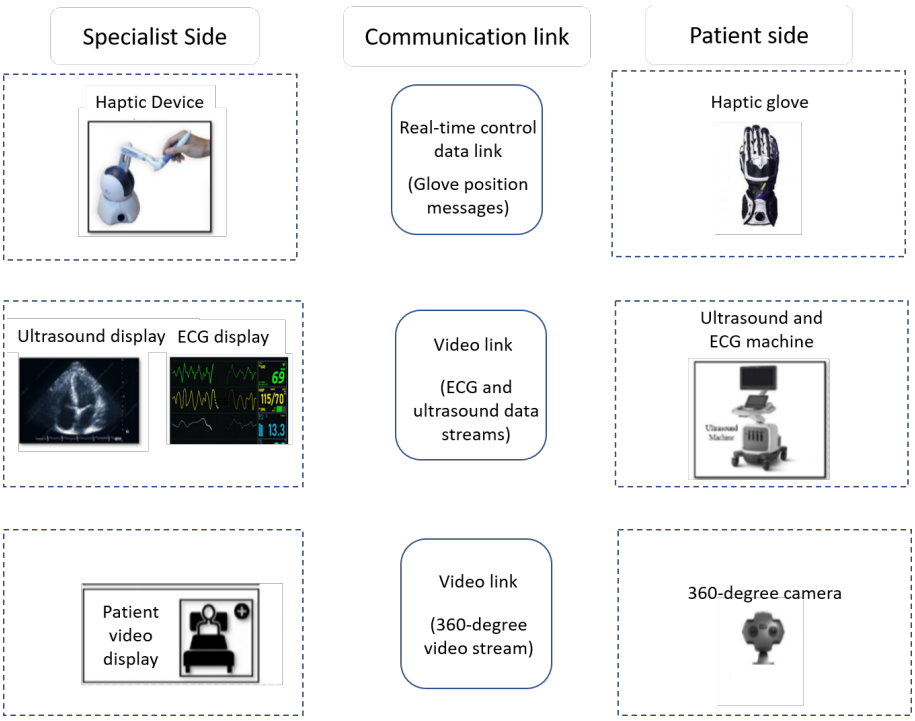

Figure 5.2: . Communication between UEs at the patient side and UEs at the remote specialist side for remote medical assistance.

Table 5.2: Data exchanged between the UEs and the gNB in Phase 2.

UE	Data	Direc-tion	Traffic Class
UE connected to 360-degree video camera	360-degree video stream	Uplink	Medical
UE connected to ultrasound and ECG machines	Ultrasound and ECG data streams	Uplink	Medical
UE connected to haptic glove	Haptic glove position messages	Down-link	Medical
Mobile phone	Paramedic's audio/video stream	Uplink	Medical
	Specialist's audio/audio stream	Down-link	Medical

In Phase 2, the network traffic generated by the ambulance is present only in the medical slice. However, the transport slice carries the network traffic generated by the regular vehicles and static road sensors.

5.4. Phase 3: Drive back to the hospital with the patient

In Phase 3, the ambulance carrying the patient needs to reach the hospital as quickly as possible. The ambulance is equipped with a *patient monitoring panel*. The patient is monitored by the paramedic with the remote guidance of the specialist on the way to the hospital.

5.4.1. Application description

The patient monitoring panel in the ambulance contains a 360-degree video camera, an ultrasound machine, a haptics glove for the paramedic, an ECG machine and a audio/video device (e.g screen with speakers). Similar to Phase 2, as described in Section 5.3, the paramedic controls the 360-degree video camera, the ultrasound and ECG machine to send real-time updates of the patient's health status to the specialist at the hospital. The specialist remotely controls the haptics glove worn by the paramedic for a proper ultrasound. Furthermore, the paramedic in the ambulance consults the specialist via an audio-video connection.

Also, same as in Phase 1, road traffic coordination is required in Phase 3 to minimize the time of arrival of the ambulance at the hospital. Therefore, all the applications in Phase 1 are also active in Phase 3.

5.4.2. Radio communication aspects

The radio communication aspects of both Phase 1 (Section 5.2.2) and Phase 2 (Section 5.3.2) are applicable for Phase 3.

In Phase 3, the ambulance generates the maximum network traffic. Unlike in Phase 1 and Phase 2 where the ambulance generates network traffic either in the transport slice or in the medical slice respectively, in Phase 3 the ambulance generates network traffic in both the medical slice and transport slice.

Furthermore, it is noted that all the aspects of Phase 1 are covered by Phase 3 completely. As compared to Phase 1, Phase 3 additionally involves network traffic from the applications in the medical priority class. There is a higher demand for resources in Phase 3 from both the medical and transport priority classes as compared to Phase 1 where resource demand only comes from the transport priority class. Therefore, Phase 1 is not considered for the scenario-based assessment.

5.5. Background slice

The background slice, in this study represents the mobile traffic that is commonly present in real-world mobile networks. The applications that generate network traffic in the background slice include video streaming, web applications (social networking, browsing, navigation), audio streaming, messaging, emails and file downloads. It is assumed that some network traffic is always present in the background slice.

6

Simulation Set-up and Modelling

We have developed a system-level simulator using SimPy based on the Python programming language [51] to evaluate the performance of the proposed AI-based multi-slice resource allocation model. In this chapter we detail the simulation set-up, the modelling aspects and configuration parameters.

6.1. Network layout

The network layout is a hexagonal grid, with a wrap-around configuration of four sites, each configured with three sectors/cells. Users are distributed uniformly over the whole considered area. Figure 6.1 illustrates the network layout used for the simulations for both the urban environment and the rural environment. Different inter-site distances (ISDs) are used for different environments (see Section 6.3.1 and Section 6.3.2). The cell range, cell radius and the ISD are also indicated in Figure 6.1. The cell radius, $R = \frac{ISD}{3}$.

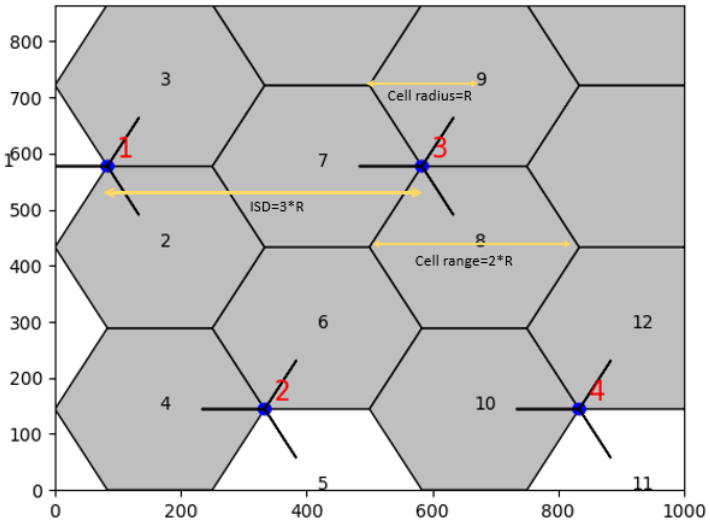

Figure 6.1: Hexagonal grid network layout used in simulations for both the urban environment and the rural environment. Different cell radii are applied for different environments.

6.2. Antenna characteristics

The antenna characteristics, namely antenna patterns and gains, of the base station (BS) and the user equipment (UE) that are applied in the simulations are described below. In the urban environment, the antennas of the base stations are at a height of 25 metres and in the rural environment the BS antennas are at a height of 35 meters. The UE antennas are at a height of 1.5 metres in both urban and rural environments. The base stations have four antennas and the UEs have two antennas.

6.2.1. Base station antenna

BS antenna pattern

The BS antenna pattern (horizontal and vertical directional characteristics) is taken from [23]. The horizontal antenna pattern used for each cell is given by (in dB):

$$G_h(\theta) = -min[12(\frac{\theta}{\theta_{3dB}})^2, FBR_h] + G_m \tag{6.1}$$

where $G_h(\theta)$ is the horizontal antenna gain (dB) in the direction θ ($-180° \leq \theta \leq 180°$), that is the horizontal angle relative to the main beam direction. θ_{3dB} is the $3dB$ beamwidth or *Horizontal Half Power Beamwidth*. The Half Power Beamwidth (HPBW) is the angular separation in which the magnitude of the antenna radiation pattern decrease by 50% (or -3 dB) from the peak of the main beam. FBR_h is the Front Back Ratio and G_m is the

maximum gain. In our simulations we set $\theta_{3dB} = 65°$, $FBR_h = 30dB$ and $G_m = 18dBi$ [23]. Figure 6.2 shows the horizontal antenna pattern.

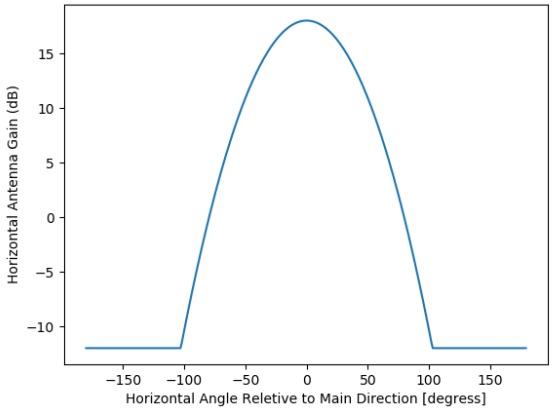

Figure 6.2: Horizontal antenna pattern for BS

The antenna pattern used for elevation in simulations is given by (in dB):

$$G_v(\phi) = max(-12(\frac{\phi - \phi_{etilt}}{\phi_{3dB}})^2, SLL_v) \tag{6.2}$$

where $G_v(\phi)$ is the vertical antenna gain (dB) in the direction ϕ (-90° $\leq \phi \leq$ 90°), that is the negative elevation angle relative to the horizontal plane at the antenna height (i.e. $\phi = -90°$ is upwards, $\phi = 0°$ is along the horizontal plane, and $\phi = 90°$ is downwards). ϕ_{3dB} is the *Vertical Half Power Beamwidth*, SLL_v is the side lobe level relative to the maximum gain of the main beam and ϕ_{etilt} is the electrical downtilt angle. In our simulations we set $\phi_{3dB} = 6.2°$, $\phi_{etilt} = 8°$ for the urban environment and $\phi_{etilt} = 4°$ for the rural environment (as suggested in [53]) and $SLL_v = -18dB$. Figure 6.3 shows the BS vertical antenna pattern with $\phi_{etilt} = 8°$.

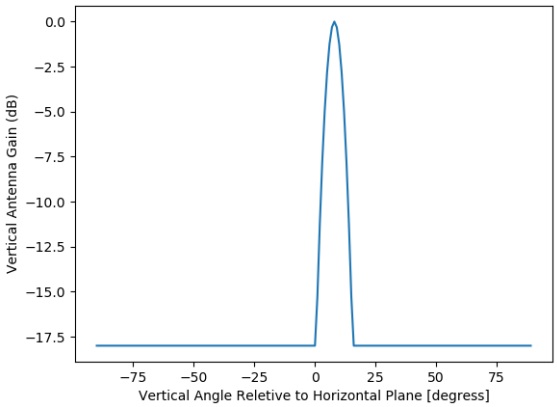

Figure 6.3: Vertical antenna pattern for BS with $\phi_{etilt} = 8^{\circ}$

The combined antenna pattern or antenna gain at angles (θ, ϕ) is computed as:

$$G(\theta, \phi) = G_h(\theta) + G_v(\phi) \tag{6.3}$$

6.2.2. UE antenna

The UE antenna is taken to be omni-directional [62] with an antanna gain of 0 dBi.

6.2.3. Antenna configurations

As mentioned above, the base station is considered to have four antennas and the UE two antennas. Multiple antennas at the base station and UE allow the use MIMO (multiple-input and multiple-output) antenna configurations in both the uplink and downlink. In this study, in the downlink we consider single user 4x2 MIMO. 4x2 MIMO antenna configuration allows to make use of spatial multiplexing and diversity coding (transmit/receive diversity) techniques to increase the capacity of a radio link by exploiting multipath propagation. In spatial multiplexing, a high-rate signal is divided into two or more lower-rate data streams. Each stream is transmitted from a different transmit antenna in the same frequency channel [45]. In transmit diversity, a single data stream is transmitted from each of the transmit antennas with orthogonal coding. The independent fading in the multiple antenna links enhances signal diversity. In receive diversity the independent fading paths associated with multiple receive antennas are combined to obtain a resultant signal. In order to optimize the link capacity, there is a switching between the multiplexing and diversity modes of transmission based on the channel conditions. In the uplink, we consider 1x4 MIMO antenna configuration which allows to make use of only receiver diversity. For simplicity, in this study we model the transmissions with 4x2 MIMO antenna configuration and 1x4 MIMO antenna configuration using

the modified and truncated Shannon formula discussed in Section 6.5.3.

6.3. Test environments and channel models

In the context of the 5G-Heart project as described in Chapter 5, we consider two different cellular network deployment scenarios, namely the urban-macro environment and the rural-macro environment. The urban-macro environment is used for modelling Phase 2 of the use case and the rural-macro environment is used to model Phase 1 and Phase 3 of the use case. The simulation configuration parameters for each environment are summarised in Table 6.1.

In this study, we consider time division duplex (TDD) mode of transmission. In TDD, the uplink and downlink traffic is time multiplexed in the same frequency band. Recall (see Section 2.1) that time is divided into frames of 10 ms, sub-frames of 1 ms and time slots whose duration depend on the numerology or sub-carrier spacing. In this study we consider 20 MHz BW in the 3.5 GHz band and 60 KHz sub-carrier-spacing. Hence, in each time slot 25 PRBs are available to be assigned to the UEs. Each time slot is 0.25 ms long in duration and contains 14 OFDM symbols. 60 kHz SCS or 0.25 ms time slot is considered to support low-latency requirements for some of the services modelled in this study. A sub-frame consists of a fixed pattern of UL and DL time slots. Different flavours of UL/DL pattern exist to choose from depending on the expected mobile traffic. We consider a slot configuration of DSUU where D stands for downlink time slot, consisting of only downlink symbols, U stands for uplink time slot, consisting of only uplink symbols and S stands for special time slot that can contain a mix of uplink and downlink symbols. In addition, the special time slot contains guard symbols or Guard Period (GP) between downlink and uplink transmission. The purpose of the GP is to avoid interference within a cell by compensating for propagation delays [22]. The GP is not required between the uplink and downlink slots, as there is less chance of collision because of the base station timing advance feature [22]. The special slot “S” format used in the SCS 60 kHz and DSUU frame configuration is configured as 11D+3G+0U that stands for eleven downlink symbols, followed by a guard period for the duration of three symbols and no uplink symbols.

Network slicing allows to have different slot configurations and SCSs for different slices, however, in this thesis we assume that a common slot configuration and a common SCS are used in all the slices for simplicity.

For the purpose of our study, we take the following radio wave propagation effects into account: path loss, shadow fading, vehicle and car penetration loss. Multipath fading effect of radio wave propagation is not considered for the sake of simplification. In the following sub-sections, we describe the modelling of each of the considered environments.

6.3.1. Urban-macro propagation environment

The urban-macro propagation environment is taken to evaluate Phase 2 of the ambulance use case (see Chapter 5). Propagation is usually non-line of sight (NLOS). The channel model for the urban-macro propagation environment is called 'UMa' [2]. The following equations describe the UMa channel model taken from [2].

The UMa channel model defines the probability of having line of sight (LOS) communication, the LOS path loss and the NLOS path loss as follows:

- **Line of sight probability**

$$P_{LOS} = min(\frac{18}{d}, 1)(1-exp(-\frac{d}{63})) + exp(-\frac{d}{63}) \tag{6.4}$$

- **LOS path loss (dB)**

$$PL = 22log_{10}(d) + 28 + 20log_{10}(f_c)\ ;\ 10m < d < d_{BP} \tag{6.5}$$

$$PL = 40log_{10}(d)+7.8-18log_{10}(h'_{BS})-18log_{10}(h'_{UT})+2log_{10}(f_c)\ ;\ d_{BP} < d < 5000m \tag{6.6}$$

- **NLOS path loss (dB)**

$$PL = 161.04-7.1log_{10}(W) + 7.5log_{10}(h)-(24.37-3.7(\frac{h}{h_{BS}})^2)log_{10}(h_{BS}) \tag{6.7}$$

$$+(43.42-3.1log_{10}(h_{BS}))(log_{10}(d) - 3)$$

$$+20log_{10}(f_c)-(3.2(log_{10}(11.75h_{UT}))^2 - 4.97)\ ;\ 10m < d < 5000m$$

where P_{LOS} is the probability of having a line of sight communication, d is the 3-D distance between the BS and the UE in meters, PL is path loss in dB, f_c is the centre frequency in GHz. d_{BP} is the break point distance. For the urban-macro propagation environment, $d_{BP} = 560$ m. Readers are referred to [2] for details. h_{UT} is the height of the UE. h_{BS} is the height of the BS. For the urban-macro propagation environment, $h_{BS} = 25$ m, W is the street width, taken as 20 m. h is the average building height. In this study, $h = 20$ m [2] is considered for the urban-macro propagation environment.

Table 6.1: General configuration parameters for the rural-macro and urban-macro propagation environments

Parameters	Urban-macro propagation environment	Rural-macro propagation environment
Layout	Hexagonal grid, 4 sites, 3 sectorized sites, wrap around	
Numerology	60 kHz SCS, 0.25 ms slot	
Carrier frequency	3.5 GHz	
Bandwidth	20 MHz per cell (25 PRBs per slot)	
BS antenna height	25 m	35 m
No. of BS antennas	4	
UE antenna height	1.5 m	
No. of UE antennas	2	
BS total transmit power	46 dBm	
UE maximum transmit power	23 dBm	
Inter-site distance	500 m	1732 m
Channel model (path loss, correlated shadowing, car penetration loss)	3GPP urban-macro (UMa) [2]	3GPP rural-macro (RMa) [2]
BS noise figure	5 dB	
UE noise figure	7 dB	
BS antenna gain	8 dBi	
UE antenna gain	0 dBi	
Thermal noise level	-174 dBm/Hz	

6.3.2. Rural-macro propagation environment

The rural-macro propagation environment is used to evaluate Phase 1 and Phase 3 of the ambulance use case (see Chapter 5). The propagation environment for the rural-macro scenario is represented by large areas with low building density. The base station is fixed and the height of the base station is in the range 20 m to 70 m, much higher than average building height [32]. As a result, the line of sight propagation conditions exist in the most of the coverage area. The UE speed is in the range from 0 to 350 kmph [32]. Hence, the rural-macro scenario is also used for highway test environments. The channel model for the rural-macro propagation environment is called ‘RMa’ [2]. The following equations describe the RMa channel model taken from [2].

The RMa channel model defines the probability of having line of sight communication, the LOS path loss and the NLOS path loss as follows:

- **Line of sight probability**

$$P_{LOS} = exp(-\frac{d-10}{1000}) \; ; \; d > 10m \tag{6.8}$$

- **LOS path loss (dB)**

$$PL_1 = 20log_{10}(\frac{40\pi df_c}{3}) + min(0.03h, 10)log_{10}(d) - min(0.044h, 14.77) \quad (6.9)$$
$$+0.002log_{10}(h)d \ ; \ 10m < d < d_{BP}$$

$$PL_2 = PL_1(d_{BP}) + 40log_{10}(\frac{d}{d_{BP}}) \ ; \ d_{BP} < d < 10,000m \quad (6.10)$$

- **NLOS path loss (dB)**

$$PL = 161.04 - 7.1log_{10}(W) + 7.5log_{10}(h) - 24.37 \quad (6.11)$$
$$-3.7(\frac{h}{h_{BS}})^2)log_{10}(h_{BS}) + (43.42 - 3.1log_{10}(h_{BS}))(log_{10}(d) - 3)$$
$$+20log_{10}(f_c) - (3.2(log_{10}(11.75h_{UT}))^2 - 4.97) \ ; \ 10m < d < 5000m$$

The symbols defined for the UMa model also hold for the RMa model. For the RMa model, $d_{BP} = 3847.6$ m, $h_{BS} = 35$ m and $h = 5$ m is considered.

6.3.3. Shadow fading

Shadow fading observed in cellular networks is approximated using a normal distribution (in the dB scale) with zero mean and an environment-specific standard deviation [46]. Furthermore, shadowing exhibits spatial autocorrelation, where shadowing components between a transmitter and receivers in neighbouring positions are correlated. Another aspect is the shadowing cross correlation (SCC), where the shadowing components from two transmitters at a common receiver position are correlated. In our simulator we consider both the spatial autocorrelation and SCC to model shadow fading in both the urban-macro propagation environment and the rural-macro propagation environment. For implementation, a shadowing map of the considered network area is created for each BS site. One set of the shadowing maps containing one map per BS site constitutes one specific realisation of the propagation environment. In our simulations, we consider one realisation of each of the propagation environments (rural-macro and urban-macro). A resolution of 1 meter x 1 meter for the urban-macro propagation environment and 2.5 meters x 2.5 meters for the rural-macro propagation environment is used. Each pixel of 1 meter x 1 meter (urban-macro) or 2.5 meters x 2.5 meters (rural-macro) in the shadowing map gives the value of shadow fading (in dB) at the corresponding receiver position. The same generated shadowing maps are applied for both uplink and downlink. The standard deviation for shadow fading σ is taken as 6 dB for the urban-macro propagation environment and 8 dB for the rural-macro propagation environment [2]. The cross-correlation coefficient is kept fixed and equal to 0.5, among all transmitters, as proposed in [17]. The decorrelation distance for computing spatial autocorrelation is assumed to be 20 m, as proposed in [46]. The implementation of the shadowing maps is taken from [17]. Figure 6.4 and Figure 6.5 show the shadowing

maps of all four sites for the urban-macro and rural-macro propagation environment respectively. The horizontal and the vertical axes represent the x and y coordinates (in meters) respectively of a receiver in the considered network area. The colour represent the magnitude of shadow fading (in dB) at a receiver position in the network area.

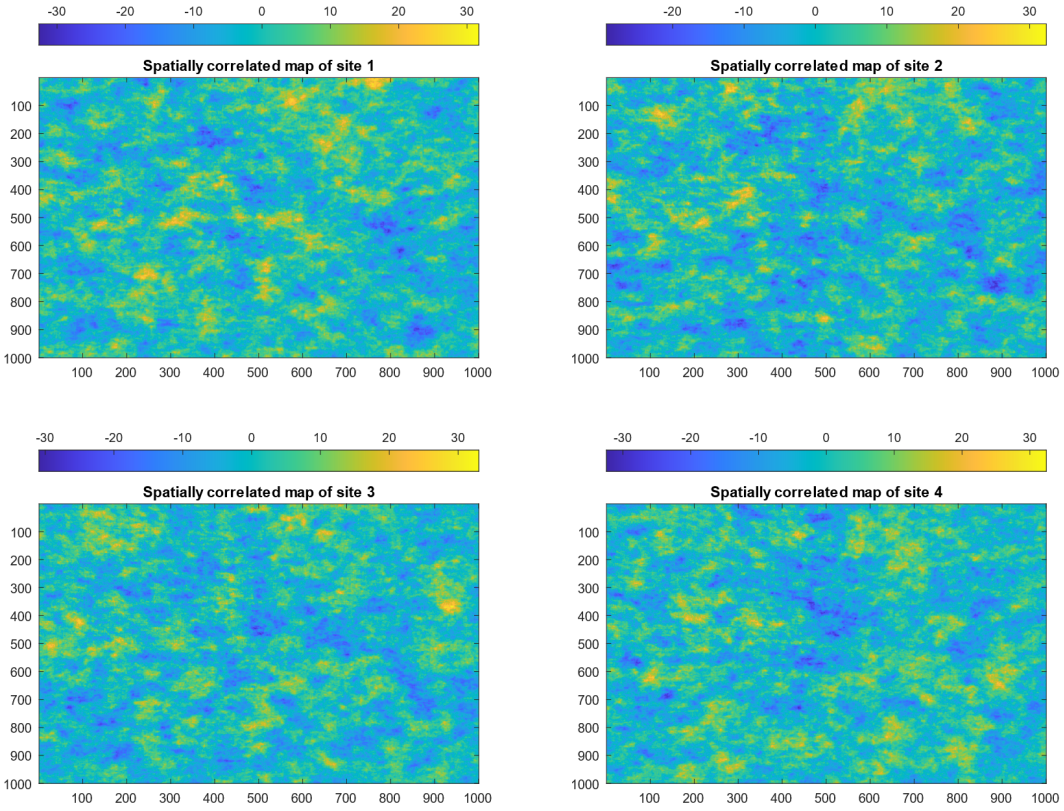

Figure 6.4: Spatially correlated shadowing maps of four sites for the urban-macro propagation environment.

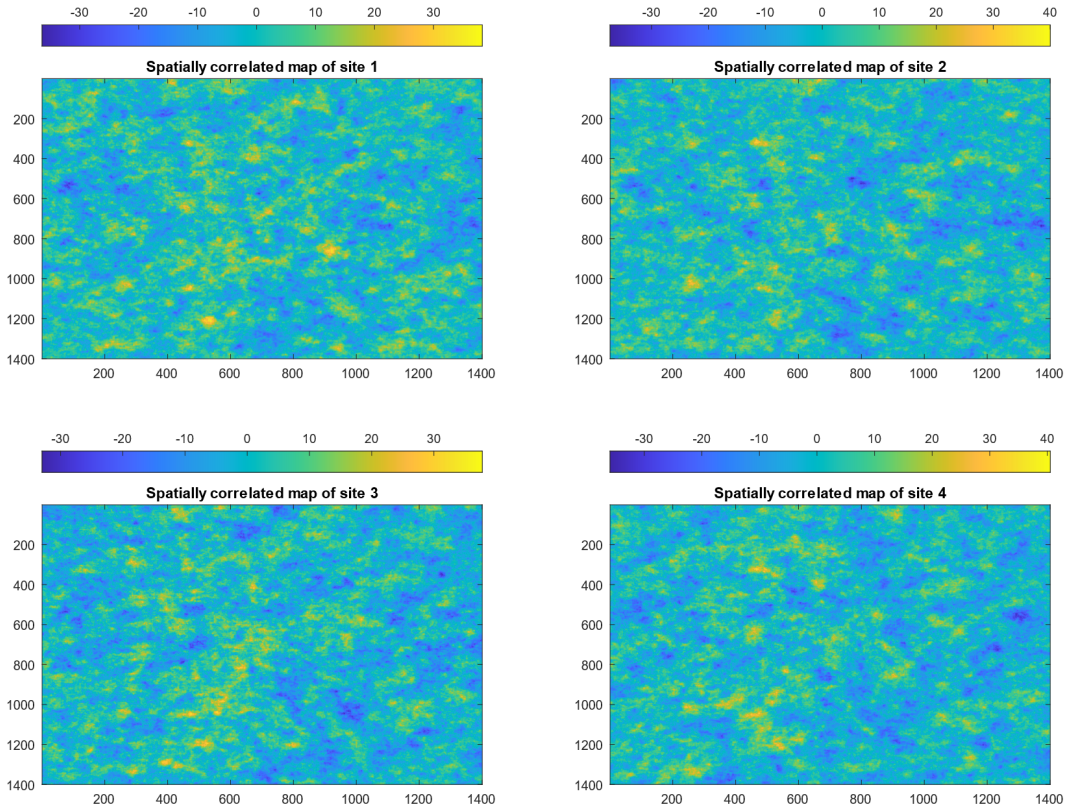

Figure 6.5: Spatially correlated shadowing maps of four sites for the rural-macro propagation environment.

6.3.4. Best server area (BSA)

The BSA plot illustrates the coverage of various cells in the network. As also mentioned above, the considered network area comprises of pixels. For a UE in the centre of each pixel, the total coupling loss is determined between the UE and the BS of each cell. The coupling loss includes the antenna gain, shadowing and distance-based path loss. The cell having the minimum coupling loss towards the UE is referred to as the 'best server'. Figure 6.6(a) shows the BSA plot for the urban-macro propagation environment and 6.6(b) shows the BSA plot for the rural-macro propagation environments. The horizontal and the vertical axes represent the x and y coordinates (in meters) respectively of the center of a pixel in the considered network area. Each colour in the BSA plots corresponds to a unique cell in the network and represents the cell's coverage.

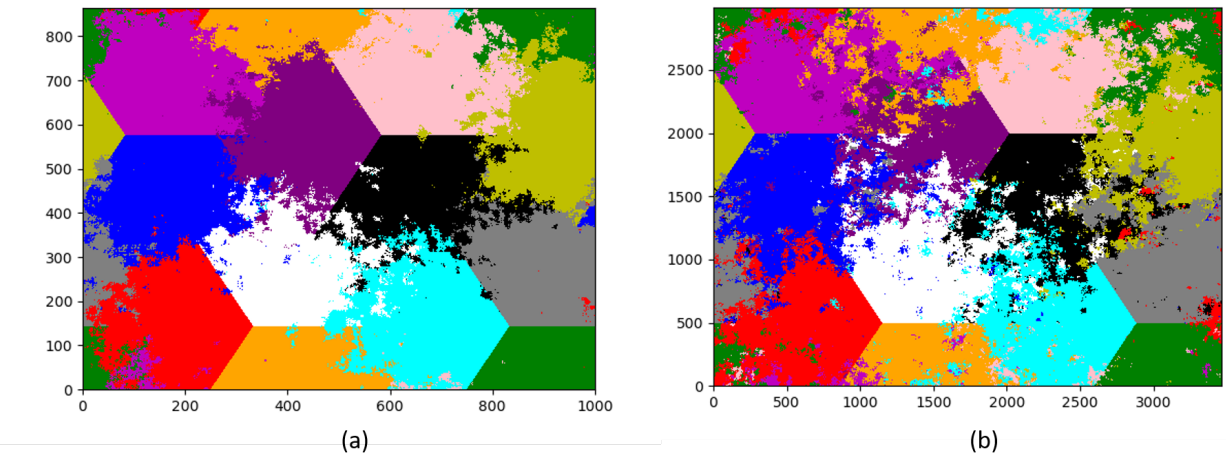

Figure 6.6: (a) BSA plot for the urban-macro propagation environment. (b) BSA plot for the rural-macro propagation environment

6.3.5. Car penetration loss

The simulation model also considers UEs inside vehicles. An additional penetration loss is experienced in case of an outdoor BS and a vehicle UE. This outdoor to in-car penetration loss is modelled to be normally distributed (in the dB scale or log normally distributed in the linear scale) with a mean of 9 dB and a standard deviation of 5 dB, as suggested in [32].

6.4. Network traffic modelling

In this section, we describe the different aspects related to the network traffic in the medical, transport and background classes, e.g. user deployment in a cell, application characteristics, call arrival, call duration and traffic mix (type and direction(UL/DL)). Two different traffic models are used in the simulator to model the traffic of the different considered applications in all the classes. These are, as follows:

1. **Traffic Model 1: finite buffer traffic model**
   The so-called finite buffer traffic model is used to model, e.g. non-real time video and file transfers. In this model, the data to be transmitted during a call is available in the data buffer of the call at the start of the call. The buffer of the call is at the UE in the case of an uplink call and at the BS in the case of a downlink call. The data is commonly referred to as a 'file'. The file size is sampled from a lognormal distribution for each type of files (e.g. video or file).

2. **Traffic Model 2: real-time traffic model**
   The so-called real-time traffic model is used to model, e.g. real-time video call and voice over Internet Protocol (VoIP) calls. In this model, it is assumed that packets arrive in the data buffer of a call, with a constant packet size and a constant packet inter-arrival time (IAT). The packet size and the packet inter-arrival time are determined based on the type of the application.

.

6.4.1. Background class traffic

The background class UEs are uniformly distributed in the network, where some UEs are outdoors and others are in vehicles. Different user deployments (outdoor vs in-vehicles) are considered for urban-macro and rural-macro propagation environments, as described in Table 6.2.

The call arrival process in a cell is modelled as a Poisson arrival process with three different settings of mean arrival rate (λ), corresponding to high load, medium load and low load scenarios, respectively. For high load, $\lambda = 20$ calls per second is used which corresponds to a cell resource utilisation of 70% on average, for medium load, $\lambda = 12$ calls per second is used which corresponds to a cell resource utilisation of 40% on average and $\lambda = 6$ calls per second is used for low load scenarios, which corresponds to a cell resource utilisation of 20% on average. It is noted that the above-mentioned cell resource utilisation values correspond to the scenarios with only background class traffic (and no medical and transport class traffic).

Each background UE moves with a constant speed which is determined at the start of the simulation and in a direction which is randomly and uniformly selected from the range 0 to 360-degrees. Different values of UE speeds are considered for urban-macro and rural-macro propagation environments, as described in Table 6.2.

80% of the total background traffic is modelled to be downlink, while the remaining 20% is uplink [48]. The traffic is expressed in terms of number of bits per second. In the simulations, 40% of the total background traffic is due to non-real time video streaming (e.g. YouTube) (35% video downloads and 5% video uploads), 40% is due to file transfers (35% file downloads and 5% file uploads) and the remaining 20% is modelled to be real-time video traffic (e.g. Skype) and VoIP (10% real-time video and 10% VoIP). The traffic mix is inspired by [48].

Traffic Model 1 is used to model the traffic generated by non-real time video and file transfers. If a user makes a video file request, a file size is sampled from a lognormal distribution with an average file size of 29 MB and median file size of 5.6 MB [21]. In the case of file transfers, the average file size of 1.3 MB and the median file size of 136 kB is used [21].

Traffic Model 2 is used to model the traffic generated by Skype and VoIP. For a Skype call, with a sending bitrate of 600 kbps and a packet size of 800 B [66], the IAT is taken as 10.7 ms. For a VoIP call the packet size is taken as 150 B and the IAT is taken to as 40 ms [31].

Since, in this study, the background application are considered to be best-effort, the QoS requirements for these application are not relevant in this study.

Table 6.2: Network traffic aspects related to the background users.

Traffic aspects of background UEs	Urban-macro propagation environment	Rural-macro propagation environment
UE mobility model	Fixed speed per UE, randomly and uniformly distributed direction	
UE speeds of interest	3 kmph for pedestrians, 30 kmph for outdoor vehicles	Speed randomly selected from a uniform distribution in range of 90 to 120 kmph
Device deployment	80% users outdoors in vehicles, 20% outdoor pedestrians, randomly and uniformly distributed over the considered network area.	100% users outdoors in high speed vehicles, randomly and uniformly distributed over the considered network area.
Call arrival	Poisson process per cell	
Call duration	Sampled from log-normal distribution with mean = 84 seconds and standard deviation = 124 seconds for real-time calls	

6.4.2. Medical class traffic

In Chapter 5 we discussed the different medical applications considered in this study and their respective radio communication aspects (data exchanged between the UEs and network). In this section we discuss the application characteristics of the medical applications. Aspects related to the UE deployment and the UE speeds are described in Chapter 9 along with the evaluation of the different scenarios.

Recall that the medical application considered in this study include 360-degree video streaming, audio/video calling, ultrasound and ECG data streaming and the messages conveying the haptic glove positions. All these applications are modelled using the above-described Traffic Model 2. Table 6.3 summarizes the parameters, namely the packet size and the packet inter-arrival time, selected for modelling the traffic generated by the medical applications. Table 6.5 shows the maximum allowed packet drop rate (PDR), packet latency budget and bit rate considered for the different medical class applications. The values are selected based on recommendations in [27], [49].

Table 6.3: Mobile traffic parameters (packet size and IAT) for medical applications [27].

Application	Packet size	Packet inter-arrival time
360-degree video	1500	5 ms
Ultrasound and ECG data	1000 B	10 ms
Haptic glove position messages	64 B	5 ms
Audio/video call	800 B	11 ms

6.4.3. Transport slice traffic

As detailed in Chapter 5, the transport network slice handles the mobile traffic generated by vehicles, intersection controller (IC), road sensors and the ambulance's transport unit. These UEs and their respective radio communication aspects are defined in Chapter 5. In this section, we describe the application characteristics of the transport applications. The aspects related to the UE deployment and the UE speeds are described in Chapter 9 along with the evaluation the different scenarios.

The real-time video traffic generated by the regular vehicles and for the ambulance's transport unit (beyond-vision) is modelled using Traffic Model 2, described in Section 6.4.1. A packet size of 500 B and IAT of 10 ms is considered. The traffic signal priority messages (URLLC-type traffic) of the IC and the ambulance's transport unit is modelled as a deterministic event-triggered traffic. The event is triggered in the simulations when the ambulance is 50 meters away from the IC. A message is sent from the ambulance to the BS asking for traffic green signal priority. The BS sends in the downlink, a message to the IC asking the same. The IC then responds in the uplink, to the BS regarding the status of the request (accept or reject). The BS then sends a message in the downlink to the ambulance's transport unit informing about the status. All the messages are taken to be of equal size and equal to 150 B. In addition, the road sensors and the vehicles generate small messages at a fixed frequency. This is also modelled using Traffic Model 2, described in Section 6.4.1.Table 6.4 summarizes the considered packet sizes and packet inter-arrival times for the transport applications modelled using Traffic Model 2. Table 6.5 shows the maximum allowed packet drop rate, packet latency budget and bit rate considered for the different transport class applications. The values are selected based on recommendations in [27], [49].

Table 6.4: Mobile traffic parameters (packet size and IAT) for transport applications [27].

Application	Packet size	Packet inter-arrival time
Global map and local map messages	600 B	50 ms
Beyond-vision video	500 B	10 ms
Road sensor messages	300 B	50 ms

Table 6.5: Considered values of maximum allowed PDR, packet latency budget and bit rate for the different priority applications [27], [49].

Priority applications	Maximum allowed PDR	Packet latency budget	Bit rate
360-degree video streaming	1%	100 ms	2400 kbps
Ultrasound and ECG data streaming	1%	100 ms	800 kbps
Haptic glove position message transmission	0.1%	1 ms	102. 4 kbps
Audio/video calling (bi-directional)	1%	130 ms	581.82 kbps
Global map and local map message transmissions	0.1%	10 ms	96 kbps
Beyond-vision video	1%	100 ms	400 kbps
Road sensor messages	0.1%	10 ms	48 kbps

6.5. Radio resource management mechanisms

Radio resource management (RRM) involves strategies and algorithms for controlling parameters such as transmit power, resource allocation to users, data rates and modulation and coding schemes. The objective of RRM mechanism is to utilize the limited radio-frequency spectrum resources and radio network infrastructure as efficiently as possible. In the following sections we discuss the RRM techniques considered in this study.

6.5.1. Scheduling policy

A scheduling policy dictates in which order the UEs get access to the resources and the number of PRBs each UE is assigned for data transmission. We use a delay-based priority scheduling policy for selecting the UEs that get access to the resources in each time slot and deciding how many PRBs each UE gets in each time slot. The same scheduling policy is used for both uplink and downlink. In each time slot, the scheduler calculates the priority for each active UE, based on the head-of-the-line packet delay and the corresponding latency budget. The calls modelled in the simulator can be categorised into two classes, viz. **delay-sensitive (DS) calls** and **delay-tolerant (DT) calls**. The delay-sensitive calls are characterised by a finite packet-latency budget. This means, if a packet is not completely transmitted within the latency budget, it is dropped. Alternately, the delay-tolerant calls have no such requirement. Hence, the packet-latency budget can be considered to be infinite for the delay-tolerant calls. The scheduling priority ($\mathcal{P}$) calculation is done as follows:
If $B < W$, the packet is dropped, otherwise the scheduling priority is calculated as:

$$\mathcal{P} = \begin{cases} \frac{W}{B-W}, & \text{for delay-sensitive calls} \\ 0, & \text{for delay-tolerant calls} \end{cases} \tag{6.12}$$

where W is the packet wait time in the call buffer, B is the packet-latency budget. Hence $B - W$ is the time left until the packet's deadline. In the case when W exceeds B, the packet is dropped. It is noted that in the case of downlink, the call buffer is at the BS and hence, the wait time of the HOL packet for each active DL call is known to the BS-based scheduler. However, in the case of uplink, since the call buffers are at the UE, this information is not directly available to the scheduler. In this study, we assume that the HOL packet wait times for each active UL call is made available at the BS via means of uplink control signals. The PRBs are allocated to the calls in the order of priority, starting from the highest priority. In case there are multiple calls that have the same priority, e.g. if only multiple delay-tolerant calls are active in a time slot, all having the same priority equal to zero, Round Robin scheme is used where one PRB is given to a call at a time.

6.5.2. Uplink power control

Uplink power control is a mechanism to control the transmission power of the UEs. It is used to provide adequate transmit power to the UEs for attaining the required signal-to-interference-plus-noise (SINR) and meanwhile minimizing the interference caused to other users in the neighbouring cells. Like LTE, the 5G-NR uplink power control mechanism is based on the combination of open-loop power control (OLPC) and closed-loop power control (CLPC). In OLPC, the UE sets its transmit power based on an estimated uplink path loss (estimated by the UE based on downlink measurements), a target received power and a path loss compensation factor. While, in the case of closed-loop power control, the UE adjusts its transmit power based on the feedback from the BS (transmission power control (TPC) commands in the downlink) [64]. OLPC has been shown (in LTE) to work just as well as or even better than CLPC [47]. Hence, in this study, we only consider OLPC. The basic idea of OLPC in uplink is to compensate for (generally a fraction of) the path loss between the BS and the UE by controlling the transmit power of the UE. This is called fractional path loss compensation. Fractional path loss compensation makes cell-edge users with a higher path loss operate at a lower SINR as compared to other users in the cell so that less interference is caused to the neighbouring cells [40]. The UE's transmit power per assigned PRB (dBm) with OLPC with fractional path loss compensation is given by:

$$P_{PRB} = P_0 + \rho PL \tag{6.13}$$

where P_0 is the target received power per assigned PRB at the UE's serving cell (dBm), ρ is the path loss compensation factor for fractional path loss compensation. PL is the downlink path loss (dB) estimated by the UE based on the downlink reference symbols measurements in the serving cell. ρ is taken as 0.6 and P_0 as -56 dBm based on the recommendation of [64].

The UE's total transmit power cannot exceed the maximum allowed UE transmit power P_{max}. Therefore, the maximum number of PRBs that can be assigned to an uplink call is given by:

$$M_{max} = \frac{P_{max}}{P_{PRB}} \tag{6.14}$$

where symbols have the meaning as mentioned above. P_{max} and P_{PRB} are in mW.

An uplink call may get PRBs that are less than or equal to M_{max} depending on the cell load and the scheduling policy. That is,

$$M \leq M_{max} \tag{6.15}$$

where M is the number of assigned PRBs to the UE.

The UE's total transmit power (in dBm) with OLPC with fractional path loss compensation is given by:

$$P = min\{ P_{max}, P_0 + 10log_{10}M + \rho PL \} \tag{6.16}$$

where symbols have the meaning as mentioned above.

6.5.3. Link adaptation

The capacity of a radio link depends on the SINR at the receiver. Link capacity is measured using spectral efficiency (SE), expressed in bits/sec/Hz. The maximum SE in a single-input-single-output (SISO) system on an additive white Gaussian noise (AWGN) channel is derived from the Shannon capacity formula given by Equation 6.17. The Shannon capacity formula gives the theoretical limit for the achievable SE for a given signal-to-noise ratio (SNR).

$$SE = log_2(1 + SNR) \tag{6.17}$$

where SNR is the signal-to-noise ratio.

Link adaptation (LA) is the process of dynamically changing the modulation and coding scheme (MCS) used for radio transmission based on the radio link conditions between the UE and the BS. LA is done by the BS. LA for downlink is done based on feedback from the UE. The UE estimates the downlink channel quality based on the experienced SINR of the downlink reference signals and reports it to the BS using a channel quality indicator (CQI) report. The BS uses the CQI report to determine the MCS to be used for downlink transmissions so that the block error rate (BLER) does not exceed a threshold value. In good radio link conditions, a high CQI is reported and the BS selects a high MCS value (more bits per symbol and less redundancy), resulting in a higher SE. On the hand, in bad radio link conditions, a low CQI is reported, and a low MCS value is used (less bits per symbol and more redundancy) to achieve robust wireless transmissions, resulting in lower SE. Hence, LA is a way to approach the theoretical bound on SE as given by the Shannon formula. LA for uplink is based on a sounding reference signal (SRS) transmitted by the UEs. UL channel measurements are done at the BS based on the experienced SINR and accordingly the BS determines the MCS to be used for the uplink. The process of LA is better illustrated in Figure 6.7.

It can be observed from the figure that as the SNR improves, higher MCS can be used resulting in increased spectral efficiency. By selecting higher-order MCS with improving SNR, the bound given by the Shannon formula can be approached.

The SE in practical systems is below the theoretical limit and is better approximated by the truncated Shannon bound (TSB) formula [3]:

$$SE(SINR) = \begin{cases} 0, & \text{if } SINR < SINR_{min} \\ \kappa log_2(1 + SINR), & \text{if } SINR_{min} \leq SINR < SINR_{max} \\ SE_{max}, & \text{if } SINR > SINR_{max} \end{cases} \tag{6.18}$$

where $SINR$ is the SINR of the link, $SINR_{min}$ is the lower limit on the SINR below which the SE is zero. $SINR_{min}$ is set based on the required SINR for the lowest MCS. $SINR_{max}$ is the upper limit on the SINR. $SINR_{max}$ is the SINR at which the highest implemented MCS is used, resulting in the maximum SE i.e. SE_{max}. The approximation of the Shannon curve thus obtained is truncated at points $SINR_{min}$ and $SINR_{max}$. κ is the attenuation factor that accounts for implementation losses and system overhead. The values

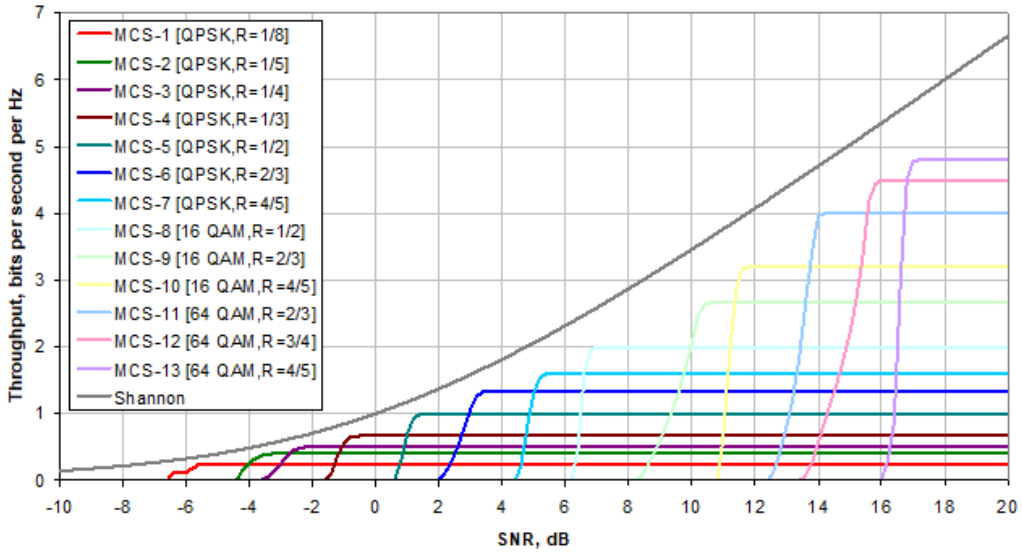

Figure 6.7: SE of a set of modulation and coding combinations for AWGN channels [3]

of κ, $SINR_{min}$ and SE_{max} can be chosen to represent different modem implementations and link conditions [3]. The TSB formula for a baseline case has been proposed in [3]. Table 6.6 describes the parameters for the TSB formula proposed for the baseline case. The baseline assumes 1x2 antenna configuration (receive diversity), typical urban fast fading channel model. UE speed of 10 kmph is considered in the case of DL and 3 kmph in the case of UL. In addition, link adaptation, channel prediction and an error-control plus error correction method called hybrid automatic repeat request (HARQ) [19] are also considered in the baseline.

Table 6.6: Parameters describing the link-level performance for the baseline scenario [3]

Parameter	DL	UL	Comments
κ, attenuation	0.6	0.4	Represents implementation losses
$SINR_{min}$ (dB)	-10	-10	Based on QPSK, 1/8 rate (DL) and 1/5 rate (UL)
SE_{max} (bps/Hz)	4.4	2.2	Based on 64QAM 4/5 (DL) and 16QAM 3/4 (UL)

In addition to the SINR, multipath propagation also impacts the achievable SE. Multipath propagation leads to frequency selective fading across the wideband radio channel. UE movement causes variation in the fading levels over time. As mentioned earlier, the network uses the channel estimates in the form of CQI reports to compensate and exploit the variations through LA. However, high UE speeds cause rapid variations in the channel, reducing the effectiveness of channel estimation and consequently adversely affecting the link capacity. The impact of UE speed on link capacity is illustrated in Figure 6.8. It can be observed from the figure that for a given SNR, achievable throughput

decreases with an increase in the UE speed.

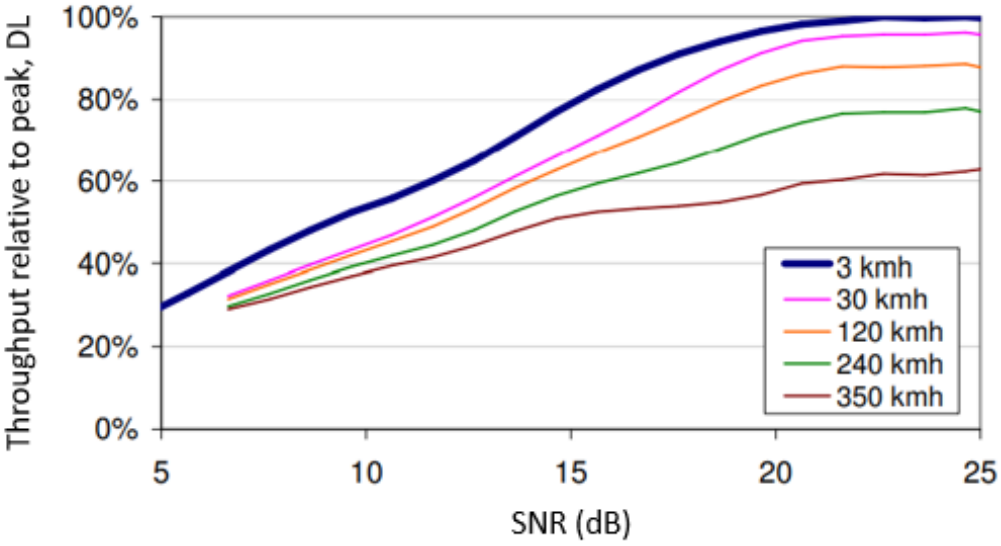

Figure 6.8: Impact of UE speed on link capacity [3]

Moreover, in general, the link capacity also increases with increase in the number of MIMO streams [65]. Hence, for instance, the SE for a given SINR and UE speed would be higher for 4x4 MIMO than for 1x2 receive diversity.

In this study, we derive modified forms of the baseline TSB formula separately for UL and DL to obtain an approximation of SE for a given SINR. The modified TSB is used to model the LA mechanism in the simulations. Using the modified TSB, the average SE that is achievable for a call, given the SINR and the UE speed is derived. The SE estimate is used for packet scheduling and also for the calculation of actual throughput achieved for the call, given the number of PRBs allocated to the UE.

The modification to the baseline TSB formula is required because in this study, we consider 4x2 MIMO in DL and 1x4 receive diversity in UL unlike 1x2 receive diversity for both UL and DL in the baseline and a high UE speeds (see Table 6.2). Hence, we apply two correction factors to the TSB formula for the baseline case, one for the change in antenna configuration and another for high UE speeds. The correction factor for 1x2 to 4x2 antenna configuration conversion is noted as c_{DL} and for 1x2 to 1x4 antenna configuration conversion is noted as c_{UL}. The values of c_{DL} and c_{UL} are set based on the recommendations of [65].

$$c_{DL} = 1.636 \text{ and } c_{UL} = 1.205 \tag{6.19}$$

c_{speed} is the correction factor to account for ineffective channel estimation due to UE speed. It is the same for both UL and DL TSB formulas. The values of c_{speed} are set

based on the findings in [65].

$$c_{speed} = \begin{cases} 1, & \text{if } v < 30kmph \\ 0.875, & \text{if } 30 \geq v < 120kmph \\ 0.75, & \text{if } v \geq 120kmph \end{cases} \tag{6.20}$$

For 5G NR, the maximum modulation order is 256 QAM and the maximum code rate is 948/1024. However, for simplicity we keep the same codeset as used in the baseline case. Hence, the modified TSB formula used in this study is given by Equation 6.21.

$$SE(SINR) = \begin{cases} 0, & \text{if } SINR < SINR_{min} \\ \kappa' log_2(1 + SINR), & \text{if } SINR_{min} \leq SINR < SINR_{max} \\ SE'_{max}, & \text{if } SINR > SINR_{max} \end{cases} \tag{6.21}$$

where $\kappa' = \kappa c_{DL} c_{speed}$ and $SE'_{max} = SE_{max} c_{DL} c_{speed}$ for DL. For UL, $\kappa' = \kappa c_{UL} c_{speed}$ and $SE'_{max} = SE_{max} c_{UL} c_{speed}$.

7

Key Performance Indicators

7.1. Performance metrics

The main focus of this study is to design and assess a resource management method for sliced radio access networks that is able to *efficiently* distribute the limited amount of resources among multiple slices in a cell such that the QoS requirements of the applications in the priority classes are satisfied and the highest attainable performance is achieved for the best-effort applications in the background class. In a sliced RAN, resources are reserved for the priority classes that incurs cost for the slice owners. An efficient resource management method would provide high performance to the applications in the priority classes while minimizing the amount of reserved resources and hence the cost of resources for the slice owners. Furthermore, in the situations of resource scarcity, the resource management method must be able to efficiently balance the performance of multiple priority classes, in accordance with the relative priorities of the slices. Another indicator for the performance of a resource management method is cell resource utilisation. In general, for a given performance level, the resource management method that leads to a lower cell resource utilisation is said to have a better performance. But, at the same time, for a given cell load, a higher cell resource utilisation means that a lower number of packets were dropped which indicates towards a more efficient multi-slice resource allocation.

Therefore, the considered key performance indicators (KPIs) must capture the performance level of the priority applications, the performance level of the background applications, the cost of resources incurred by the slice owners and the cell resource utilisation. In the following sections we define the KPIs that quantify the above mentioned aspects.

Slice utility function, average performance and average cost of resource reservation

We define three different KPIs to capture three things, namely the performance level of the priority applications, the average cost of resources incurred by the slice owners and the balance between the cost and performance. These are respectively, the average performance of the applications in a priority class, the average cost of resource reservation for the priority class and the slice utility of the slice to which the priority class is mapped.

The performance of a call is quantified by a *service utility* function. The service utility function for delay-sensitive services is given by Equation 7.1 and for delay-tolerant services is given by Equation 7.2.

$$U_{DS} = \begin{cases} \alpha_{DS}, & \text{if } r_{DS} \leq 1 \\ \exp(-\beta_{DS} \times r_{DS}) - \alpha_{DS}, & \text{if } r_{DS} > 1 \end{cases} \tag{7.1}$$

where U_{DS} is the utility for a delay-sensitive call computed over a time interval equal to the lifetime of the call or T_0 (whichever is less). $r_{DS} = \frac{PDR_{obs}}{PDR_{max}}$. PDR_{obs} is the packet drop-rate of the call measured in the same interval and PDR_{max} is the maximum allowed PDR for the call. $r_{DS} > 1$ means that the observed PDR exceeds the maximum allowed PDR and hence the PDR requirement of the call is not satisfied.

$$U_{DT} = \begin{cases} \alpha_{DT}, & \text{if } r_{DT} \leq 1 \\ \exp(-\beta_{DT} \times r_{DT}) - \alpha_{DT}, & \text{if } r_{DT} > 1 \end{cases} \tag{7.2}$$

where U_{DT} is the utility for a delay-tolerant call computed over a time interval equal to the lifetime of the call or evaluation period T_0 (whichever is less). $r_{DT} = \frac{Th_{req}}{Th_{obs}}$. Th_{obs} is the throughput of the call measured in the same interval and Th_{req} is the minimum required throughput by the call. $r_{DT} > 1$ means that the observed throughput is less than the minimum required throughput and hence the throughput requirement of the call is not satisfied.

Figure 7.1 shows the service utility function for delay-sensitive (and delay-tolerant) services with shape-parameters (α, β). In the case of delay-sensitive services, $\alpha = \alpha_{DS}$, $\beta = \beta_{DS}$ and in the case of delay-tolerant services, $\alpha = \alpha_{DT}, \beta = \beta_{DT}$. It can be observed from the figure that in the case of DS services, if the PDR requirement is not satisfied (i.e. r_{DS}>1), the value of the service utility function falls exponentially with the increase in the value of r_{DS}. Similarly, in the case of DT services, if the throughput requirement is not satisfied (i.e. $r_{DT} > 1$), the value of the service utility function falls exponentially with the increase in the value of r_{DT}. This means, worse the performance (in terms of PDR or throughput), higher is the penalty. As long the performance requirement is satisfied, the service utility is positive and equal to α. The value of β controls the steepness of the exponential curve. This means, higher the value of β, the service utility drops to the worst possible value (given by $-\alpha$) for a smaller drop in the performance level below the required performance level. Hence, the choice of the values of α and β determine

the stringency imposed by a slice owner in meeting the required performance levels for the services on the slice. Higher values of α and β mean more stringent requirement for meeting a certain level of performance. It is noted that the relative importance of meeting certain performance levels for DS and DT services in a slice is controlled via the choice of α_{DS}, α_{DT}, β_{DS} and β_{DT}.

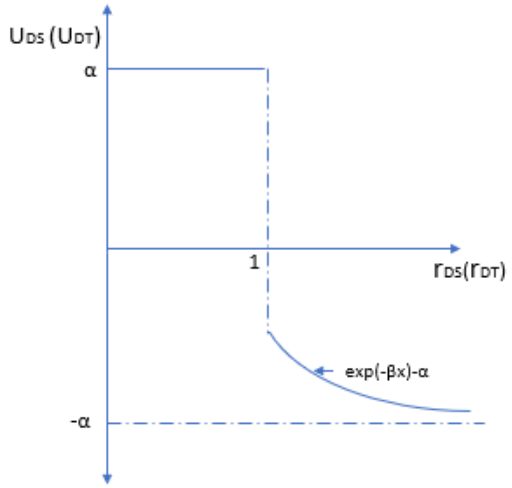

Figure 7.1: Service utility function for delay-sensitive and delay-tolerant services with shape parameters (α, β)

The average performance of the priority applications in a class (including both DS and DT applications) is given as follows:

$$\frac{\sum_{n=1}^{n=N_{DS}} U_{DS}}{N_{DS}} + \frac{\sum_{m=1}^{m=N_{DT}} U_{DT}}{N_{DT}} \tag{7.3}$$

where N_{DS} and N_{DT} are respectively the total number of delay-sensitive calls and delay-tolerant calls activated in the slice in the interval T_0. U_{DS} and U_{DT} are respectively the service utility values for the delay-sensitive calls and delay-tolerant calls. The relative importance of DS services and DT services in the average performance of priority applications is implicitly determined by the values of the parameters: α_{DS}, α_{DT}, β_{DS} and β_{DT} used in calculating the service utility.

The average cost of resource reservation for a slice is given as follows:

$$average\ cost = \gamma F_{slice} \tag{7.4}$$

where γ is a unit-less quantity and F_{slice} is the fraction of the total PRBs allocated to the slice for the interval T_0. It is noted that the slice owner pays for the number of PRBs

reserved for the slice and not for the fraction of the total PRBs. However, in the slice utility function we use the fraction of the total PRBs allocated to the slice to represent the average cost of reserved resources. In this way, γ can be expressed as a unit-less quantity that is independent of the total available PRBs in the interval T_0.

As mentioned above, the *slice utility* function quantifies the balance achieved by a resource management method, between the cost of resources and the performance of applications in a slice in a cell. The slice utility over a certain time interval T_0 takes into account the average cost of resources allocated to the slice and the average performance of the calls activated in the slice in the same time interval T_0. The slice utility function includes the average performance of the delay-sensitive services and delay-tolerant services activated in a slice and the average cost of the PRBs allocated to the slice in the interval T_0. The slice utility function is given by:

$$U_{slice} = \frac{\sum_{n=1}^{n=N_{DS}} U_{DS}}{N_{DS}} + \frac{\sum_{m=1}^{m=N_{DT}} U_{DT}}{N_{DT}} - \gamma F_{slice} \tag{7.5}$$

where U_{slice} is the slice utility.

α_{DS},α_{DT},β_{DS},β_{DT} and γ are configurable parameters that can be selected by the network operator to reflect its policies with different slice owners. For instance, if the operator wants to charge highly for the reserved resources for a slice, it can choose a high value of γ.

In addition to determining the level of stringency imposed by a slice owner for meeting the required application performance levels (see above), the choice of α (α_{DS} or α_{DT}) also determines the relative importance of service satisfaction w.r.t the resource reservation cost for the slice owner. This means, for example, if the ratio $\frac{\alpha}{\gamma}$ is taken to be higher, the slice utility value is affected more by the performance of the services and less by the cost of reserved PRBs.

In this study we analyse the performance of the proposed AI-based multi-slice resource allocation framework considering different settings of the above-discussed configurable parameters. Details are in Chapter 9.

Weighted average utility of the priority slices

As the name suggests, the weighted average utility of the priority slices for a time interval T_0 is the (weighted) average of the slice utility values of the priority slices calculated over the interval T_0.

Thus, it is defined as follows:

$$U_{avg} = w_{med} U_{medical} + w_{tran} U_{transport} \tag{7.6}$$

The choice of w_{med} and w_{tran} also reflect the operator's policy for favouring one slice over the other. Recall that in this study, the transport slice is considered to have a higher priority than the medical slice. The values of w_{med} and w_{tran} considered in this study are specified in Chapter 9. This KPI is considered as the bottom-line KPI for the priority classes.

Average cell resource utilisation

The average cell resource utilisation indicates the amount of resources used to yield certain performance levels for the different classes. It is defined as the ratio of the number of PRBs used in a cell in a certain time interval T_0 and the total available number of PRBs in the cell in the same interval. As mentioned above, for a given performance level a lower resource utilisation is better. At the same time, for a given traffic load in all the classes, a higher resource utilisation indicates towards a more efficient resource allocations in all the classes. This is because, for a certain traffic load, an efficient resource allocation results in a lower number of packet drops in the priority classes. Consequently, more traffic (in total) can be handled by the cell leading to a higher resource utilisation.

10^{th} throughput percentile of the background class applications

This KPI is considered as the bottom-line KPI for the background class. The background class is taken as the best-effort class, which only gets the resources which are left-over after allocation to the priority classes. Hence, the other KPIs, namely, the average performance of applications, the slice utility (and hence, the weighted average), average cost of resource reservation are not relevant for this class.

8

AI Model

In this chapter we discuss the details of the artificial intelligence model designed for resource management in a sliced RAN. The proposed framework has two modules, namely resource prediction module and resource allocation module. The proposed framework takes the resource management decisions at the cell-level. In the following sections we elaborate the design and functioning of each module and discuss how these modules work together for the purpose of resource management in a sliced RAN.

8.1. Overview

According to the use case considered in this study, there are three slices that can be active in a cell, namely medical slice, transport slice and background slice (see Chapter 5 for details). As mentioned in Chapter 5, the transport slice and the medical slice are priority slices, while the background slice is a best-effort slice. As discussed in Chapter 5, in this study, we assume the transport slice has a higher priority than the medical slice.

For resource allocation among the slices in a cell, first the resource prediction module predicts the resource demand for the priority slices in the cell. The collective resource demand is represented by the tuple $\{\mathcal{D}_1, \mathcal{D}_1\}$. This resource demand tuple is then input to the resource allocation module that takes the final resource allocation decision for all the slices in the cell. The resource management (resource demand prediction followed by resource allocation) is done periodically and additionally may be triggered when a new call connection is established in the cell or when a slice changes its status from inactive to active and vice-versa. The time interval at which the resource management decisions are taken periodically is referred to as the re-allocation interval. The above is illustrated in Figure 8.1. For all the active calls in the medical and transport classes, the resource demand prediction module predicts the resource demand based on various call specific statistics, e.g. throughput requirement, RSRP of the UE (see Section 8.2.1 for the considered call specific statistics). The resource demand for each call in a class

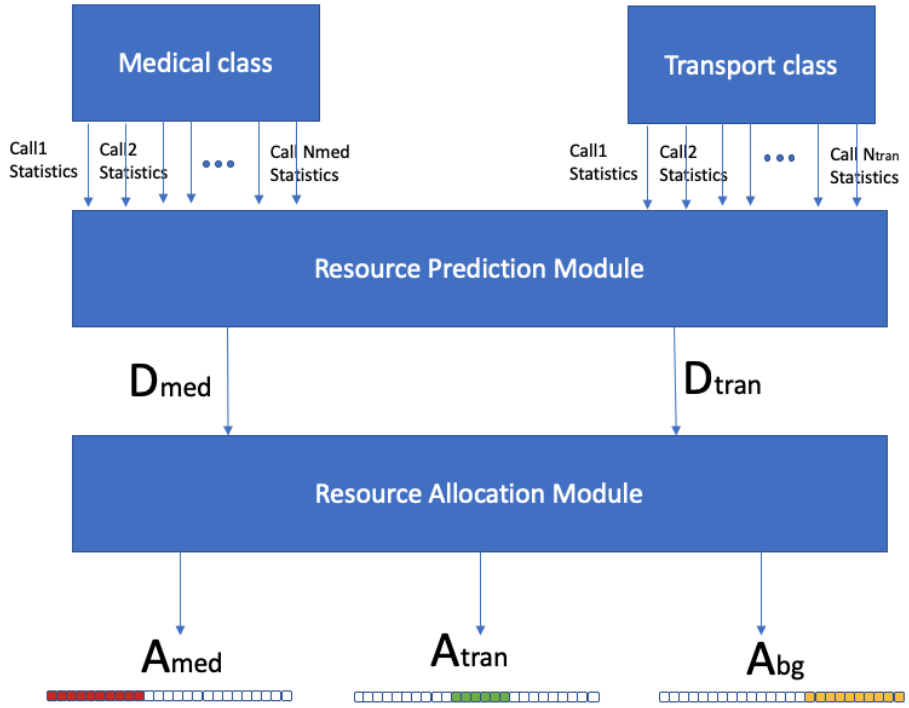

Figure 8.1: Overview of the complete AI-model for multi-slice resource allocation.

is aggregated to get the resource demand of the slice to which the class is mapped (denoted in the figure by D_{med} and D_{tran}). The resource demands of the priority classes, viz. medical and transport as predicted by the resource prediction module are input to the resource allocation module which then based on the collective demand, allocates certain amount of resources to each priority class (denoted in the figure by A_{med} for the medical class, A_{tran} for the transport class and A_{bg} for the background class). It is noted that the resource allocation module only gives the number of PRBs to be allocated to a slice and not *which* PRBs. Recall (see Chapter 6) that in this study, we do not explicitly model frequency-selective fading. Hence, all the PRBs are identical. The background class gets whatever is remaining after allocation to the priority classes.

8.2. Resource prediction module

The task of the resource prediction module is to predict the resource demand for the slices in a cell for an upcoming re-allocation interval. The resource demand for all the slices as predicted by the resource prediction module is then used by the resource allocation module to obtain the final resource allocation for all the slices in the cell for the upcoming interval (see Section 4.2 for details on the resource allocation module).

The resource prediction module predicts the *expected* resource demand for each active call in a slice in the cell and sums the predicted demands over all active calls in the slice to get an estimate of the expected demand of the slice. It is noted that the calls that will be active in a slice during the upcoming interval may vary and cannot be known in advance. To tackle this issue, the resource demand prediction module, in addition to predicting the resource demand periodically, updates resource demand prediction for a slice whenever a new call is established in the slice or when a slice changes its status from inactive to active and vice-versa. A new resource demand prediction in turn triggers the resource allocation module to update the resource allocation among the different slices. This, however, also depends on the length of the re-allocation interval. In general, if the re-allocation time-interval is small (e.g. 100 ms), the number of active calls in a slice during the interval can be taken constant and equal to the number of active calls in the slice at the start of the re-allocation interval.

The design of the resource prediction model is such that it takes some inputs that represent some call-specific statistics (discussed in Section 8.2.1) and based on the values of the inputs, outputs the expected resource demand for a call as a single continuous-valued variable. For this purpose, the resource prediction module employs a supervised machine learning model in the regression setting (refer to Chapter 2). Figure 8.2 illustrates the overview of such a machine learning model. The inputs are the considered call-specific statics that are discussed in detail in Section 8.2.1.

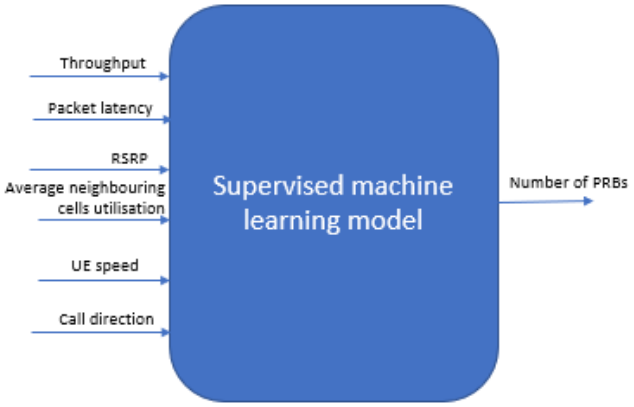

Figure 8.2: Overview of the machine learning model used in the resource demand prediction module.

The aim of supervised machine learning in the regression setting is to estimate so-called *true relationship* between the inputs and the output of a defined model in an underlying physical system. The physical system in this study is a cell in a cellular wireless system with active calls in the defined three slices, viz. medical, transport and

background slices. The performance of a ML model depends on the application domain that is represented by the training data. In this study we analyse the performance of three machine learning models for the resource prediction problem, namely multiple linear regression, random forests regression and artificial neural networks (also see Chapter 2). Multiple linear regression is the simplest ML model for regression problems with two or more model inputs and one model output. We use MLR to find whether a linear relationship between the inputs and the output exists. However, in many cases the inputs are non-linearly related to the output and therefore, MLR cannot be used to find the true relation. In such cases, non-linear ML models are used to get more accurate estimate of the true relation between the inputs and the output. Most commonly used supervised machine learning models for non-linear regression are RF and ANN. In this study, we analyse and compare the performance of MLR, RF and ANN for the defined resource prediction problem. The training data and the testing data is taken to be the same for the three ML models. The ML model that minimizes the defined loss function (see Section 8.2.10) is considered to achieve the most accurate estimation of the true relation between the defined inputs and the output and hence is selected for the resource prediction module.

In the following sections, we describe the inputs and the output for the machine learning models and the process of creating the training data. We then discuss the design of each of the machine learning models and analyse their performance.

8.2.1. Machine learning model inputs and output

A supervised machine learning model essentially approximates an unknown function called the target function that represents the true relationship between the model inputs and output. In this study, the ML model takes some inputs and outputs the average resource demand of a call. **We define the resource demand of a call as the number of PRBs per time slot that would be required by the call to satisfy its throughput and/or packet latency requirements**. In a cellular wireless environment the demand as defined above depends on a number of factors. Hence, the inputs of the ML model must be selected carefully such that all such factors are taken into account. We identified that the following factors affect the resource demand of a call. Hence, we select these factors as the inputs for the ML model. Below we briefly argue our choice for the inputs.

- **Throughput requirement**
  If the throughput requirement for a call is higher, a given number of packets of the call must be transmitted in a smaller time interval, consuming or requiring more PRBs per time slot *on average*, compared to the case of a lower throughput requirement.

- **Packet latency requirement**
  A packet must be given the required amount of resources for transmission within the latency budget. If the packet latency budget of a call is smaller, the required amount of resources must be given in a smaller interval, consuming or requiring

more PRBs per time slot *on average*, compared to the case of a higher packet latency budget.

- **Reference signal received power (RSRP)**
  The RSRP is the average reference signal power received at the UE, averaged over the resource elements carrying the reference signal in a cell. The RSRP is computed at the UE and reported back to the BS and indicates the strength of the received wireless signal. Moreover, the RSRP indicates the proximity of the UE to the serving cell BS and hence is positively correlated with the experienced SINR and hence the modulation and coding scheme (MCS). The MCS used determines the number of bits transmitted *successfully* per spectrum unit and time unit, i.e. the spectrum efficiency. A higher RSRP and accordingly a higher MCS result in a higher spectrum efficiency or equivalently result in a lower PRB requirement to transmit a given number of bits.

- **Average neighbouring cells PRB utilisation**
  The cell PRB utilisation is defined as the ratio of the number of PRBs allocated to users in a cell per time slot and the total number of PRBs available in a cell per time slot. The average of the PRB utilisation values (as defined above) of the cells that neighbour the serving cell is defined as the average neighbouring cells PRB utilisation. The PRB utilisation values of the neighbouring cells give an indication of the mobile traffic load in the neighbouring cells. The load in the neighbouring cells affects the inter-cell interference level at the serving cell. Hence, average neighbouring cells PRB utilisation can be used to indicate the average inter-cell interference level at a receiver in the serving cell. A higher inter-cell interference at a receiver decreases the SINR at the receiver that in turn adversely affects the achievable bit rate. Hence, the number of PRBs required to transmit a given number of bits also depends on the average PRB utilisation of the neighbouring cells.

- **UE speed**
  The UE speed determines how fast the wireless channel conditions for a UE are varying. We discussed in Section 6.5.3 how higher UE speeds may result in reducing the effectiveness of channel estimation and consequently adversely affecting the link capacity. To achieve a certain throughput (or to satisfy a given packet latency budget), while taking into account the effect of higher UE speed on the link capacity, more PRBs are required per time unit.

- **Call direction, i.e. uplink (UL) or downlink (DL)**
  If more PRBs are assigned to a call, bit rate increases in general on account of more available resources. However, unlike in DL, the relationship between the number of PRBs and the bit rate is not linear in UL. In UL, an increase in the number of assigned PRBs to a UE means that the limited UE power is distributed over a high number of PRBs. This results in a decrease in the per-PRB SINR and hence the attainable MCS and consequently the bit rate. Therefore, the increase in

the number of PRBs leads to two opposing effects in UL, which makes the bit rate generally increases much less with an increase in the number of PRBs in UL as compared to DL. Therefore, in general, UL transmissions have a lower spectrum efficiency than DL transmissions and hence the PRB requirement to transmit a given number of bits is higher for UL than DL.

8.2.2. Mutual information and correlation between model inputs and output

Figure 8.3 shows the mutual information between each of the above-described model inputs and the model output. The model inputs and output are random variables. Mutual information (MI) of two random variables quantifies the mutual dependence between the two variables. In other words, MI is a measure of the 'amount of information' obtained about one variable by observing the other. For two discrete variables X and Y whose joint probability distribution is $P_{XY}(x, y)$, the mutual information between them, denoted $I(X;Y)$ is given by:

$$I(X:Y) = \sum_{x,y} P_{XY}(x,y) \log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)} \tag{8.1}$$

where $P_X(x)$ and $P_Y(y)$ are the marginal probability distributions of X and Y respectively. MI is also linked to the entropy of a random variable. Entropy is a measure of the uncertainty in the outcome of a random variable [8]. In fact, MI is the reduction of the uncertainty of a variable after observing the other variable. Readers are referred to [8] for more details.

Mutual Information is commonly used in machine learning problems for selection of model inputs. Mutual information between a model input and the output gives information about how important the model input is for predicting the output. Input selection is an important step in creating a ML model that achieves high prediction accuracy. The inputs that have high inter-dependence with the output, indicated by high MI (above a certain threshold) are selected. In fact, the inputs exhibiting low inter-dependence with the output, indicated by low MI may add noise to the data and hence result in a decrease in the model prediction accuracy.

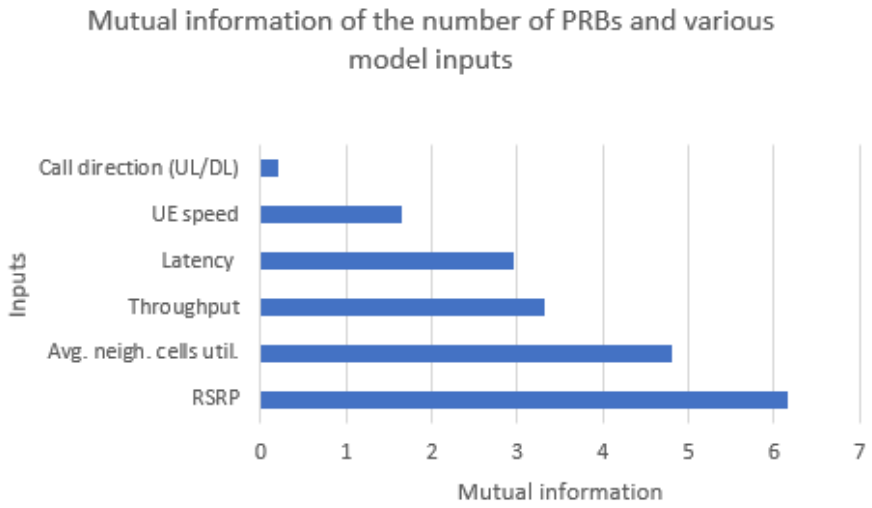

Figure 8.3: Mutual information between the model inputs and output

It can be observed from the figure that the MI between the output, i.e. the average number of PRBs required by a call and the RSRP is the highest. This indicates that the model predictions are highly dependant on the UE RSRP. The MI between the output and the call direction is the lowest, indicating that the model predictions are least affected by the call direction.

MI is a reliable measure to find the mutual dependence between a model input and the output. However, MI does not give any information about the direction of association of between a model input and the output. That is, MI cannot tell whether an increase in the value of an input leads to an increase or decrease in the corresponding value of the output. Hence, to analyse the direction of association between the different model inputs and output, we use Spearman's rank correlation coefficient. Spearman's correlation coefficient assesses whether the two variables being compared are monotonically related. The sign of the Spearman correlation indicates the direction of association between a model input and the output. If the output value tends to increase when the input value increases, the Spearman correlation coefficient is positive. If the output value tends to decrease when the input value increases, the Spearman correlation coefficient is negative. Readers are referred to [14] for more information on Spearman correlation.

Table 8.1 shows the values of the Spearman correlation for the different model inputs and the output. It can be observed that for the inputs, namely the throughput requirement, the average neighbouring cells PRB utilisation and the UE speed, the Spearman correlation coefficient is positive, indicating that as the values of these inputs increase, the output, i.e. the average number of PRBs required by a call per time slot increases. This is as expected based on the arguments given in Section 8.2.1. On the other hand, the Spearman correlation coefficient is negative for the input namely, packet latency requirement, indicating that as the packet latency requirement increases, the PRB re-

Table 8.1: The Spearman correlation between the different model inputs and the output.

Model input	Spearman correlation coefficient
Throughput	0.714
Packet latency	-0.342
RSRP	-0.136
Avg. neighboring cells PRB utilisation	0.021
UE Speed	0.058

quirement decreases, as expected.

8.2.3. Training data

A training data set consists of tuples of the input variables and the output called data samples. A training data sample in this study is given by the tuple $\{\mathcal{I}_1, \mathcal{I}_2 ... \mathcal{I}_6, \mathcal{O}\}$, where $\mathcal{I}_1$ to $\mathcal{I}_6$ are the input variables and $\mathcal{O}$ is the output label described in Section 8.2.1. The training data must represent the relation between the input variables and the output in a given underlying system. As mentioned above, in this study the underlying system is a cell in a cellular wireless system with active calls in the different slices. More specifically, the system comprises of a combination of the wireless channel conditions, behaviour of the scheduler, network layout, available system bandwidth, BS antenna settings, TDD frame structure and numerology. The training data samples are collected using network measurements for various active calls in the network. For this study, network simulations (see Chapter 6) are used to collect the training data samples. In the following text we discuss how the required network measurements are taken and which calls are used in this study for collecting the training data samples.

As mentioned, the machine learning model predicts the average number of PRBs required by a call per time slot in order to meet its throughput and packet latency requirements. Since the output is an average number per time slot, the input variables are also represented as averages per time slot to create a training data sample. As mentioned in Section 6.5.1, all the services in a cell are categorised into two classes, viz. delay-sensitive services and delay-tolerant services. The network measurements are taken in different ways for the DS and DT calls. For a DS call, the network measurements for the input variables are taken in every time slot for the duration of the transmission of a packet and are averaged over the same duration to obtain the respective average values per time slot. Not all packets are used for this purpose, instead every N^{th} *successfully transmitted* packet of the call is used. This is to avoid taking similar measurements due to similar radio conditions experienced in transmission of consecutive packets. N may be selected per call based on the speed of the UE. For example, for a UE moving at a high speed, measurements may be taken for every other packet, since the radio conditions for the UE vary rapidly, while for a stationary UE, a high value of N should be used since the radio conditions around such a UE change slowly. It is noted that the choice of N also depends on the packet inter-arrival time of the call and

the prorogation environment. For simplicity, we keep $N = 40$ for all calls in the urban environment (with stationary UEs or UEs moving at low speeds) and $N = 10$ for calls in the rural environment (with UEs moving at high speeds). On the other hand, for DT calls, the necessary measurements are taken in every time slot for the duration of the call and then averaged over the call duration to obtain the respective average values per time slot. As mentioned in Chapter 6, in the case of DT calls, the call duration is the time required to transmit all the data of the call from the transmitter to the receiver.

Out of the six input variables described above, the call direction needs no network measurement. The UE speed is assumed to be either reported by the UE to the BS or estimated by the BS based on e.g. the channel measurement reports of the UE. The UE speed is assumed to be constant for the whole call duration. The RSRP and average neighbouring cells utilisation are recorded in the manner described above for DS and DT calls. Throughput and packet latency requirements are QoS requirements that are specified by the customers of the services. While creating the training data set, instead of recording the throughput and packet latency requirements of the call, the measured values of throughput and packet latency are recorded. Also, the average number of PRBs used by the call per time slot that resulted in the measured throughput and packet latency is recorded. The throughput and packet latency experienced by a call depend on the PRBs assigned to the call. Therefore, by using the experienced throughput and packet latency and the corresponding average number of used PRBs in the training data samples, the relationship between these variables is captured and later will be used to predict the PRBs required by a call to result in a certain throughput and packet latency.

It is noted that unlike a DS call, a DT call does not have a maximum packet latency requirement and hence the experienced packet latency values for DT calls are irrelevant for the training data. Thus, for a DT call, the network measurements are taken on a call level as described above and not the packet level as in the case of DS calls. For this reason, in the case of DT calls, the call duration is recorded instead of packet latency.

This way of defining the training data samples captures the behaviour of the underlying system that is required in a supervised machine learning problem. More specifically, the training data captures the following:

*Given the channel quality between the UE and the BS as represented by the RSRP, the inter-cell interference levels as represented by the average neighbouring cells PRB utilisation, the effectiveness of channel estimation for LA as represented by the UE speed and the call direction, how many PRBs **were** used on average per time slot to result in the **observed** throughput and packet latency*.

8.2.4. Test data

A test data sample is a tuple of the above described inputs variables and output label, i.e. $\{\mathcal{I}_1, \mathcal{I}_2 ... \mathcal{I}_6, \mathcal{O}\}$, defined for the call for which the resource prediction is needed. $\mathcal{O}$ is so-called *true value* of the output for the given input variables. However, unlike a training data sample, a test data sample includes the throughput and packet latency **requirement** for the considered call and not their experienced values. At the time of prediction, a test data sample is input in the ML model which predicts the average PRB requirement per time slot for the considered call. More specifically, the following is answered by the ML-model:

*Given the channel quality between the UE and the BS as represented by the RSRP, the inter-cell interference levels as represented by the average neighbouring cells PRB utilisation, the effectiveness of channel estimation for LA as represented by the UE speed and the call direction, how many PRBs **will be** required on an average per time slot, **to achieve the required throughput and packet latency**.*

8.2.5. Training data quality

The training data dictates to a great extent how accurately the model is able to predict. In general, the training data must be abundant and well representative of the scenarios or applications for which the model is expected to make predictions. A poor training data set is characterised by small size (insufficient training samples for the given model/problem complexity), bias in data or non-representative data. Insufficient data leads to overfitting. An overfit model learns the training data perfectly but may not work on new unseen data. Biased training data, on the other hand, does not cover the complete input range expected at the time of model implementation. This leads to poor generalization to unseen data and hence prediction errors.

In this study, the resource prediction module is required to predict the resource demand of active calls in the medical and the transport slices. Hence, for the training data to be representative of the target applications, the necessary measurements must come from active calls in these slices. However, it is noted that in the considered use case the medical slice is only temporarily active (i.e. having one or more active calls) in a cell. This prevents from obtaining adequate data samples required for training in a reasonable time duration. On the other hand, the background slice is always active and hence can provide abundant data samples. Hence, to meet the requirement of having sufficient training data, we propose to use the background services for training data creation. The challenge in this case is, of course to ensure that the training data is still well-representative of the target sample distribution. The background slice may or may not have calls that resemble the calls (in the sense of the QoS requirements and traffic characteristics) in the medical and transport slices for which the predictions have to be made. This may lead to a bias in the training data towards the background services, resulting in poor model generalisation to unseen data. This means, while such a

model will perform well when used for predicting resource demand for background slice services, it will give poor predictions when applied for medical and/or transport slice services. We propose a novel solution to overcome the bias in the training data. The solution is discussed in detail in Section 8.2.6.

8.2.6. Removing the bias in the training data

As reasoned in Section 8.2.5, the training data samples are created using network measurements of the active calls in the background slice. As also mentioned in Chapter 5, the background slice, in this study represents the mobile traffic that is commonly present in real-world mobile networks. An example of the traffic mix in the real-world mobile networks includes video streaming (63%), web applications (social networking, browsing, navigation), audio streaming, messaging, emails and file downloads (around 22%). These services belong to the class of delay-tolerant services. Only a minority of the background traffic is composed of the delay-sensitive calls (including VoIP, video-conferencing/video-chats and online gaming) [15]. The training data developed using the background traffic does not contain enough training samples from DS services e.g. services in medical and transport classes. As a result the training data contains a bias towards the delay-tolerant traffic and is non-representative of the scenario where the model is expected to make predictions (i.e for the medical and the transport slice traffic that consists of only delay-sensitive services). Therefore, the model learnt using the background traffic will not generalise well to the medical and transport traffic, resulting in poor prediction performance.

To address the problem of bias in the training data created using the background slice traffic, it needs to be ensured that the training data set includes sufficient data samples for services which may not be present or present only in a small minority in the background slice mobile traffic. These are the services for which the resource prediction may be required at the time of model implementation. To achieve this, we propose a method to handle the traffic from services that make the majority of the background slice mobile traffic, e.g. delay-tolerant traffic as if it were generated by the services for which the data samples are needed for the training purpose, e.g. delay-sensitive services in the medical and transport classes. This *differentiated handling* of so-called 'majority traffic' is proposed to be done at the base station at the scheduler level. In this study we use a delay-based (QoS aware) scheduler. This scheduler makes the resource assignment schedule based on the packet latency budget of the active services (refer to Section 6.5.1). In the proposed method of differentiated handling, the network selects some of the delay-tolerant services and overrides their latency requirement with the latency requirement of the delay-sensitive services. A packet latency budget is (artificially) associated with the packets of the delay-tolerant calls for a certain duration. The delay-based (QoS aware) scheduler creates the resource assignment schedule based on the updated latency budgets for the active calls. The schedule based on the artificially enforced packet latency budgets is used for actual data packet transmissions, resulting in measurements and hence desired training data samples corresponding to

the delay-sensitive calls. An entity in or outside the network may decide when and for which delay-sensitive service this differentiated handling is necessary based on the prediction model requirements, the traffic characteristics and/or the observed service traffic mix in the cell and which delay-tolerant service is to be differently treated, e.g. based on which of the candidate services have the most similar characteristics and/or QoS requirements.

The training data set created using the above-described method of differentiated handling is referred to as 'modified' training data set and the training data set that has not been modified via means of differentiated handling is referred to as the 'original' training data set in the context below.

8.2.7. Evaluation of supervised machine learning models

We analyse the performance of three supervised ML models, namely multiple linear regression, random forests regression and artificial neural networks based on their prediction performance on common training and test data sets. The training data set and the test data set that are considered for the evaluation are obtained from the rural-macro propagation environment (see Section 8.2.3). The medical and transport services (described in detail in Section 6.4) are used to create the test data samples.

For each supervised ML algorithm, two training data sets, namely the modified training data set and the original training data set are used to train the models. The two cases are compared to analyse the improvement in prediction performance achieved by using the modified training data set (see Section 8.2.6) as opposed to the original training data set. The prediction by the model for the average PRB requirement per time slot for a call is compared with the observed average PRB usage per time slot by the call to compute an error metric. Commonly used error metrics include absolute error and squared error. However, we identify **absolute relative error (ARE)** as a more suitable error metric for the case of resource demand prediction as defined in this study. This is because, an absolute error of one PRB per time slot when the actual demand is for example two PRBs per time slot is worse than the case with the same absolute error but a higher actual demand of, for example, twenty-two PRBs per time slot. This difference is adequately captured by ARE. The ARE is defined as:

$$ARE(y_{pred}, y_{true}) = |1 - (\frac{y_{pred}}{y_{true}})| \tag{8.2}$$

where y_{pred} is the predicted value of the required output by a supervised ML model, given an input tuple and y_{true} is the true value of the output given the same input tuple. The ARE is computed for all test data samples. It is noted that average-based error metrics are commonly used in the evaluation of ML models as opposed to variance-based metrics. Hence, we select mean absolute relative error (MARE) as the metric to select the best performing ML model. The model that minimizes the MARE computed over all test data samples is selected for the resource prediction module.

8.2.8. Multiple linear regression

The multiple linear regression (MLR) model tries to find a linear relation between multiple input variables and the output using the training data. Unlike most MLR problems that use ordinary least squares (OLS) approach to fit the model (see chapter 2), in this study, the model is fit using the least square relative error approach where the sum of squared relative prediction errors of the training-data are minimized (see Chapter 2). We use the modified and original training data sets to learn the linear mapping between the inputs, $\{\mathcal{I}_1, \mathcal{I}_2 ... \mathcal{I}_6\}$ and the output, $\mathcal{O}$. MLR outcome visulaisation The MARE on the test data set for the model fit using the modified training data set was found to be equal to 2.073 and for the model fit using the original training data set to be equal to 7.039.

The MARE in both the cases of modified and original training data sets is very high. The poor prediction performance of MLR as reflected by a very high MARE value is because the defined model inputs and the output may have a non-linear relation. Hence, we expect more complex non-linear models such as random forests and artificial neural networks to show better prediction performance.

8.2.9. Random forests regression

The configurable parameter in a random forests model is the number of so-called trees or number of estimators (see Chapter 2). To select the suitable number of trees, we calculate the MARE on the test data using models with different number of trees. Table 8.2 shows the number of trees along with the MARE on test data using the respective models. A model with a particular number of trees is trained on two different training data sets namely, modified and original. Figure 8.4 illustrates the same. As can be observed from the table and figure, the minimum MARE on test data is obtained with 64 trees with the model trained on the modified training data which is equal to 0.876. This is a decrease of 95.77% (2.073 to 0.876) in the MARE as compared to the multiple linear regression model. The drastic improvement in prediction performance as expressed by the MARE on using a non-linear regression model such as RF, as opposed to MLR shows that the relationship between the defined model inputs and output is more accurately expressed as non-linear than linear. It is observed that the MARE on the test data using the model trained on the modified training data is always lower as compared to when using the model trained on the original data. This shows, the the model trained on the modified training data is able to generalise better to the unseen test data.

Furthermore, the MARE on the test data gets worse for the model trained on the original training data on increasing the number of tress. This shows that the model starts to overfit (see Section 2.3.3) even with a fewer number of trees. Where as, in the case of the model trained on the modified training data, the MARE decreases up to 64 tress and then starts to increase from there on. This shows, that the performance of the model improves on adding more trees, as expected. However, with too many trees (e.g. 128 or 256), the model trained on the modified training data starts to overfit as well

Table 8.2: MARE for different number of tress in the RF regression model.

No. of trees	MARE for modified training data	MARE for original training data
2	1.092	1.402
4	0.937	1.405
8	0.903	1.443
16	0.894	1.473
32	0.89	1.461
64	0.876	1.465
128	0.898	1.478
256	0.901	1.481

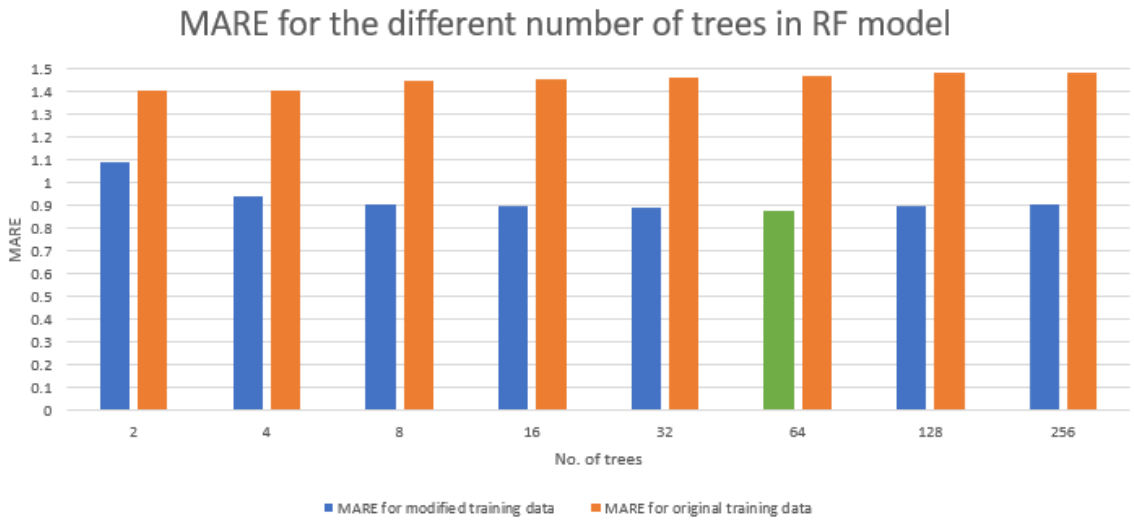

Figure 8.4: Mean absolute relative error for different number of trees in the RF regression model. Blue bars represent the MARE for the model that is trained using the 'modified' training data. Orange bars represent the MARE for the model trained using the 'original' training data. The model with 64 trees has the minimum MARE.

8.2.10. Artificial neural network

A feed-forward and fully-connected ANN (also called MLP (refer to Chapter 2) is used for the resource prediction problem. The ANN is used in the regression setting (see Chapter 2). The architecture of a ANN includes an input layer, one or more hidden layers and an output layer. The number of neurons in the input layer is equal to the number of inputs. For regression, the output layer has a single neuron. The choice of the number of hidden layers and the number of neurons per hidden layer depends on the problem complexity. The number of hidden layers and neurons are so-called ANN

hyper-parameters. In addition, the ANN hyper-parameters also include the loss function, the activation function, the optimizer and the learning rate. The hyper-parameters play an important role in the prediction performance of a ANN. We discuss in detail the choice of the various hyper-parameters in the following text.

Loss Function

A ANN approximates the target function between the model inputs and output by minimizing a pre-defined loss function calculated for the training data samples during the training phase (refer to Chapter 2). The loss function dictates how the weights of the connections between the neurons in the ANN are adjusted or equivalently how the target function is approximated. Hence, the loss function must be defined in a way that best represents the (prediction) goal of the ANN. The most commonly defined goal for ANN models is to get an outcome that is as close as possible to the desired value. Hence, MAE and root mean squared error are used as loss functions in these models. However, as discussed in Section 8.2.7, MARE is more relevant for the resource demand prediction problem as defined in this study. Hence, in this study, MARE is selected as the loss function.

ANN Size

The size of a ANN is defined by the number of hidden layers and their respective number of neurons. The choice of the size depends on the complexity of the prediction task. Highly complex problems such as computer vision tasks require a large number of hidden layers and more neurons per layer as opposed to simpler problems that work well with two or fewer layers and fewer neurons per layer [29]. To determine the best size, different size options must be evaluated based on the prediction error that is derived from a pre-defined metric (MARE in this study). The size that minimizes the prediction error calculated on a test dataset is selected. This is done using a trial and error approach. We start with two hidden layers with three neurons each based on the recommendation in [29] and subsequently change the size by either adding layers and/or neurons. Figure 8.5 shows the MARE for the different sizes of the ANN model. A model with a particular size is trained on both the modified and original training data. As can be observed from the figure, the MARE on the test data is lower for all the sizes when using the modified training data as compared to when using the original training data. This shows that the ANN model trained using the modified training data generalises better to the unseen test data, as expected. The minimum MARE on test data is obtained for the model with two hidden layers, with twelve neurons in the first hidden layer and four neurons in the next. The MARE in this case is 0.546. This is a decrease of 37.67% (0.876 to 0.546) as compared to the MARE obtained for the best performing RF regression model. Recall that since the model has two hidden layers, it is a deep ANN model.

Furthermore, it is observed that with very high model capacity, e.g. in the case of four hidden layers with twelve neurons each, the MARE on test data increases for both

the cases of modified and original data. This indicates overfitting (see Section 2.3.3) of the training data which prevents in generalising to new data and hence results in higher prediction errors.

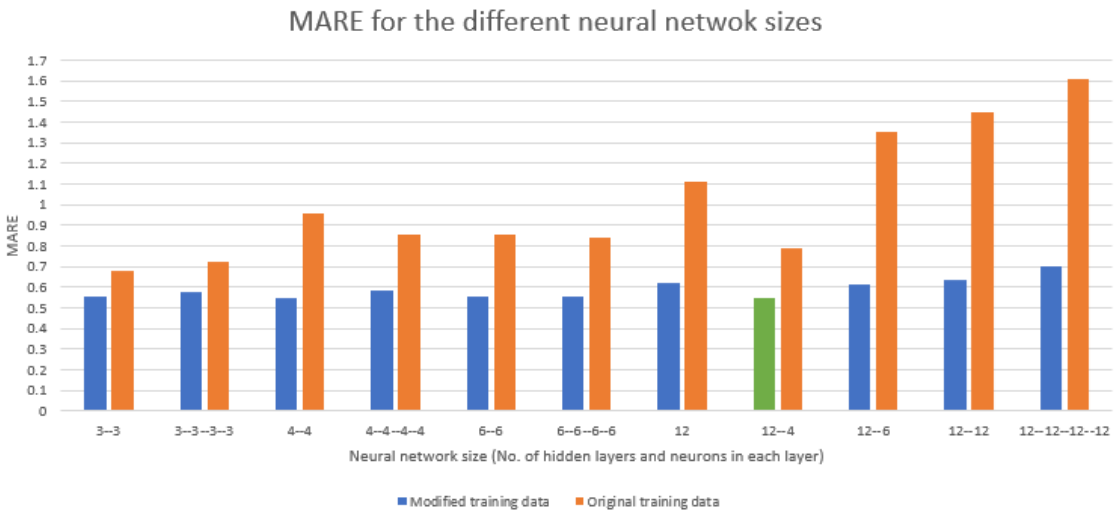

Figure 8.5: Mean absolute relative error for different sizes of the ANN model (number of hidden layers and neurons).

Other Hyper-parameters

In addition to the loss function, the number of hidden layers and their respective number of neurons, hyper-parameters for a ANN model also include the drop-out rate in the case the drop-out method is enabled, patience value in the case early-stopping is used, the number of epochs, the batch size, the optimizer, the learning rate and the activation function. These are introduced in Chapter 2. Here we mention the assumed settings for each. Dropout method is enabled with a drop-out rate of 20%. Early-stopping is used with a patience value of 10. The number of epochs is set to 500 and the batch size is taken as 64. Root-Mean-Square propagation (RMSprop) is selected as the optimiser with the learning rate set to 0.001. The values of these parameters were selected from a trial and error experiment and based on the minimum MARE on test data. ReLU is selected as the activation function for the input and the hidden layers. Rectified linear unit (ReLU) is the most preferred activation function since it allows for learning non-linear functions and is computationally efficient [44]. Table 8.3 summarizes the choice of hyper-parameters used for the final evaluation of the ANN model.

Table 8.3: The hyper-parameters selected for the ANN model.

Hyper-parameter	Choice
No. of hidden layers	2
No. of hidden neurons per layer	12 (first hidden layer), 4 (second hidden layer)
Optimizer	RMSprop
Learning rate	0.001
Activation function	ReLU
Loss function	MARE
Early stopping	Yes
Patience value	10
No. of epochs	500
Batch size	64
Restore best weights	Yes
Dropout rate	20%

8.2.11. Comparison of the different supervised machine learning models

It is observed from the above analysis that the minimum MARE on the test data is obtained for the ANN model with two hidden layers and twelve and four neurons that is trained using the modified training data set. In this section, we further do a comparative analysis of the best performing models of each of the above-discussed supervised machine learning algorithms. This means, for the ANN, we select a model with two hidden layers, with twelve and four neurons and for RF, we select the model with 64 trees. For all three, the models trained with the modified training data are used.

Figure 8.6 shows the distribution of the true values of the number PRBs in a test data

set and the distribution of the predicted values of the number of PRBs by the different supervised learning models. It can be observed from the figure that in the case of RF regression, the distribution of true and predicted values of PRBs is the most similar. However, this does not mean that the predictions are most accurate in the case of RF. This is also observed from the fact that MARE in the case of RF is actually larger than that in NN. However, it is evident from the figure that the predictions in the case of MLR are way off than the true values. This further strengthens the claim made earlier based on the MARE for MLR that the relationship between the defined model inputs and model output is not linear.

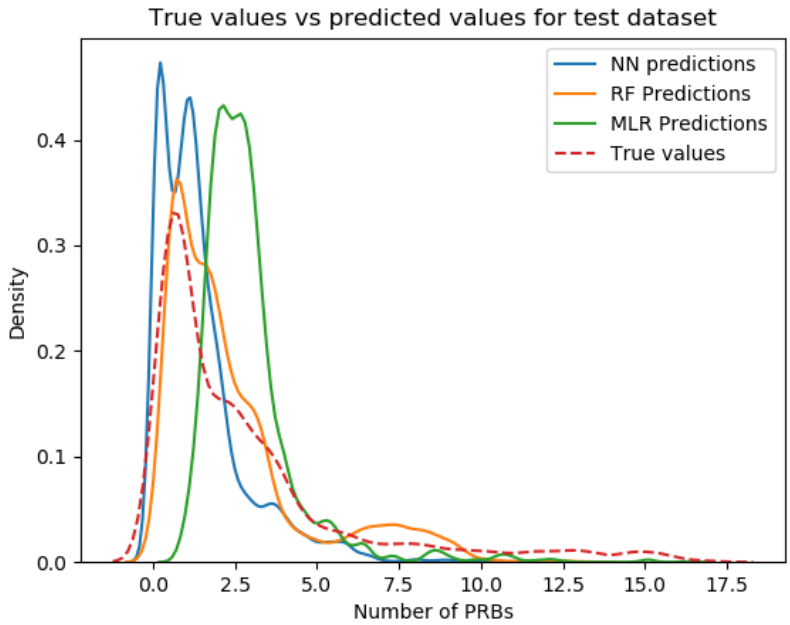

Figure 8.6: Comparison of the distribution of the predicted number of PRBs and the actual (true) number of PRBs for the considered supervised learning algorithms.

Figure 8.7 illustrates the distribution of the relative error for ANN, RF and MLR. It can be observed from the figure that for ANN, the relative error distribution has the least variance and a peak at the relative error around 0.5. The distribution of relative error for the RF model has the peak at a smaller relative error value (equal to 0.3) than ANN, but has a higher variance, resulting in a higher MARE as compared to ANN. The distribution of the relative errors in the case of MLR also has a higher variance with the least number of samples having relative errors closer to zero. As a result, the mean of relative errors is the highest for MLR.

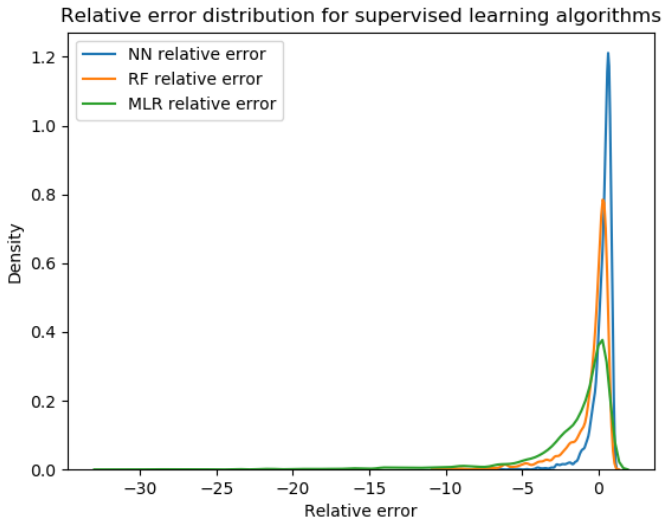

Figure 8.7: Comparison of the distribution of the relative error for the considered supervised learning algorithms.

Figure 8.8 shows the MARE for the best performing models for the three supervised machine learning algorithms. As also observed from the above analysis, MARE is the least for ANN.

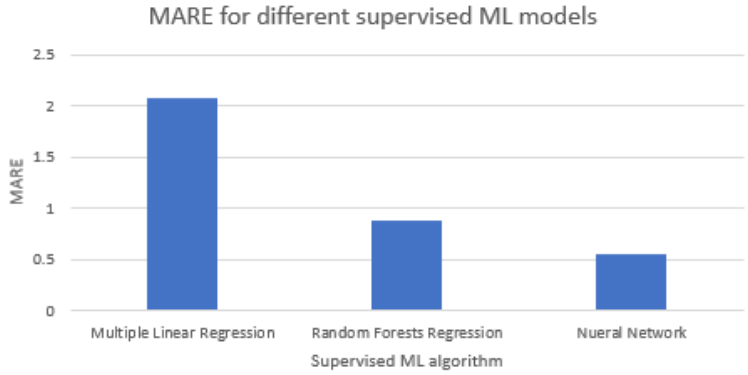

Figure 8.8: Comparison of the MARE computed on the test data set for the best performing models for the three supervised machine learning algorithms.

8.2.12. Conclusion

We observe from the above analysis that both the random forests regression model and the artificial neural network model outperform the multiple linear regression model in terms of prediction performance. This shows that the model inputs and output have a non-linear relation. The ANN model performs better than the RF regression model in terms of MARE on the test data. In fact, in general ANN models are able to generalise better to new data than RF models. This is because of the way learning algorithm works in each. The ANN allows to explicitly define the training objective as the minimization of MARE via the choice of loss function. Where as, the RF regression involves building the so-called decision trees based on the training data points (see Chapter 2). Since the goal of the resource demand prediction as defined in this study is to minimize the MARE, we select the ANN model for the resource demand prediction module.

8.2.13. Prediction performance of the ANN model

From the above analysis, it is concluded that the ANN is used for the resource demand prediction module of the proposed multi-slice resource allocation framework. Hence, in this section we analyse in greater detail, the predictions made by the defined ANN model. We discussed in Section 8.2.2, the correlation between the different model inputs and output. The correlation between the model output and the different model inputs is also observed by using the trained ANN to predict the model output, i.e. the average number of PRBs required by a call per time slot for different values of a particular input and fixed values of all other inputs. For instance, Figure 8.9 shows the predicted values of the average number of PRBs for different values of RSRP, keeping the values of all other inputs fixed. It is observed that as the RSRP increases, the predicted value of the average number of PRBs required by a call per time slot decreases. This shows a negative correlation between the predicted values of the average number of PRBs and the RSRP, as expected.

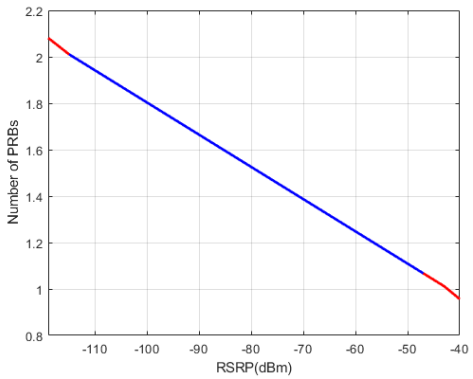

Figure 8.9: The average number of PRBs predicted by the trained ANN for different values of RSRP.

Recall that a so-called good fit ANN model is able to extrapolate the training data samples to achieve good prediction performance for unseen data samples. This is referred to as model generalisation. In order to illustrate the generalisation capacity of the trained ANN, in the above-described correlation analysis per model input, we consider values of the model inputs that lie outside the respective ranges of values of the inputs covered by the training data. For these values, the predicted outputs are shown in red. The predicted outputs for the values of the inputs that lie in the ranges of the respective inputs covered in the training data are shown in blue.

Figure 8.10 (a) shows the predicted values of the average number of PRBs for different values of latency, keeping all other inputs fixed. It is observed that as the latency (requirement) increases, the predicted value of the average number of PRBs required by a call per time slot decreases. This shows a negative correlation between the average number of PRBs and latency, as expected. Similarly, Figure 8.10 (b), 8.10 (c) and 8.10 (d) show a positive correlation between the predicted values of the average number of PRBs and the average neighbouring cells utilisation, the UE speed and the throughput (requirement), respectively.

For the above-described correlation analysis per model input, we use the ANN that is trained using the modified training data collected from the rural-macro propagation environment. Table 8.4 shows the fixed values used for the different inputs and the considered range for each input when that input is varied.

Table 8.4: Fixed values, considered range of values and range of values in the training data of the different model inputs.

Model input	Considered fixed value	Considered range for analysis	Range covered in the training data
RSRP	-73.2546 dBm	-120 to -40 dBm	-115 to -47.5 dBm
Latency (requirement)	13 ms	0 to 150 ms	0.5 to 135 ms
Avg. neighbouring cells utilisation	0.75	0 to 1	0.2 to 0.75
UE speed	100 kmph	80 to 160 kmph	90 to 120 kmph
Throughput (requirement)	5.187 Mbps	0 to 40 Mbps	0.45 to 24 Mbps
Call direction	DL	-	-

8.3. Resource allocation module

The resource allocation module takes as inputs, the resource demand of the priority slices as predicted by the resource demand prediction module. Based on the (collective) demand of the priority slices, it computes a margin (positive or negative) over the predicted demand for each slice to obtain the final resource allocation. The resource

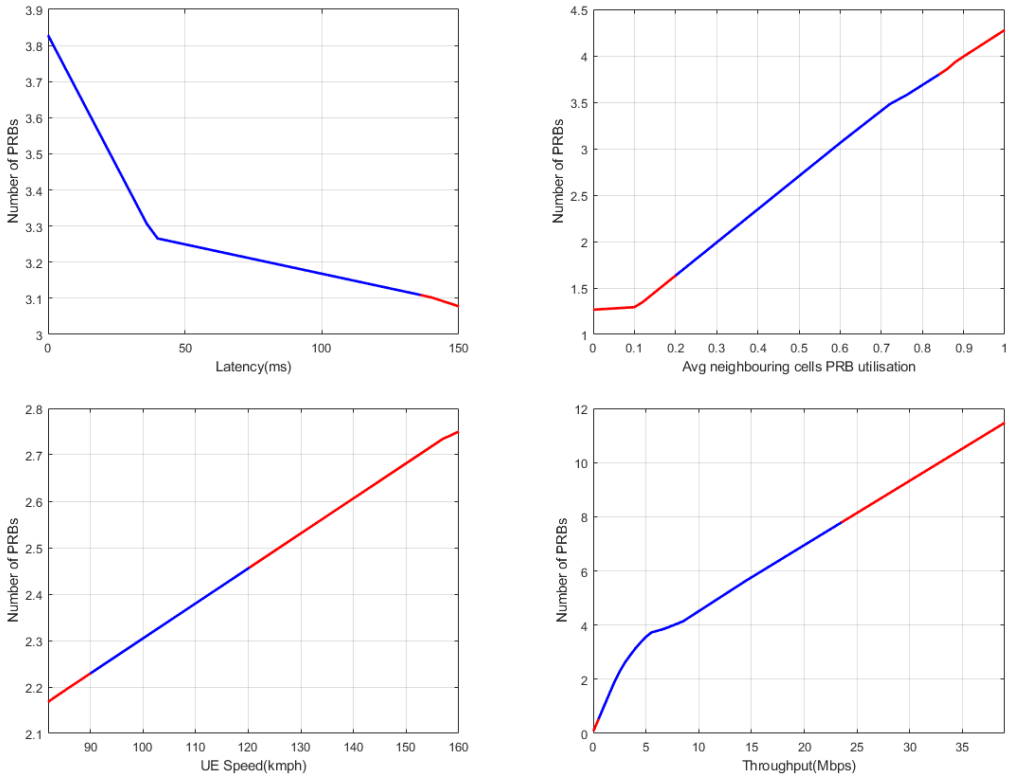

Figure 8.10: (a) The average number of PRBs predicted by the trained ANN for the different values of the latency (requirement). (b) The average number of PRBs predicted by the trained ANN for the different values of the average neighbouring cells utilisation. (c) The average number of PRBs predicted by the trained ANN for the different values of the UE speed. (d) The average number of PRBs predicted by the trained ANN for the different values of the throughput (requirement)

allocation module may be designed in multiple ways, e.g. using heuristics or using artificial intelligence, e.g. via reinforcement learning. The deign of the resource allocation module employing artificial intelligence (reinforcement learning) is discussed in Section 8.3.1, while other, non-intelligent options for the design of the resource allocation module are discussed in Section 8.3.2. Furthermore, we also describe a non-slicing option of resource scheduling, namely priority scheduling in Section 8.3.3. Priority scheduling is used as the baseline when evaluating the proposed multi-slice resource allocation framework (see Chapter 9).

8.3.1. Resource allocation module based on reinforcement learning

In this study, the resource allocation problem in sliced RAN is defined as a reinforcement learning problem in a contextual multi-armed bandit setting (refer to Section 2.4.1). Consequently, the resource allocation module is a CMAB agent specific for a cell. As a recap of Section 2.4.1, a CMAB agent receives a state from the environment and based on the state and previously gathered experiences takes an action. Upon taking the action it receives a reward from the environment. The goal of the agent is to learn the relationship between state, actions and reward so that it can predict the best action for a given state. In the following section, we define state, actions, reward and agent-learning.

State, Actions and Reward

State

A state is defined by the tuple $\{\mathcal{D}_1, \mathcal{D}_2\}$ where $\mathcal{D}_1$ and $\mathcal{D}_2$ are integers in the range $[0, N]$. N is the total number of available PRBs in a cell per time slot. $\mathcal{D}_1$ and $\mathcal{D}_2$ describe the average resource demands of the medical slice and the transport slice per time slot in a cell respectively. $\mathcal{D}_1$ and $\mathcal{D}_2$ are expressed as the number of PRBs. These resource demands come from the resource prediction module discussed in Section 8.2.

Actions

An action of the agent is defined by the tuple $\{\Delta_1, \Delta_2\}$, where Δ_1 and Δ_2 are integers in the range $[-N, N]$. $\Delta_1 > 0$ and $\Delta_1 < 0$ represent the increase and decrease respectively in the number of PRBs, $\mathcal{D}_1$, describing the predicted average resource demand for the medical slice. Similarly, $\Delta_2 > 0$ and $\Delta_2 < 0$ represent the increase and decrease respectively in the number of PRBs, $\mathcal{D}_2$, describing the predicted average resource demand for the transport slice. Given a state $\{\mathcal{D}_1, \mathcal{D}_2\}$, an action $\{\Delta_1, \Delta_2\}$ of the agent results in resource allocation for the medical slice and the transport slice in a cell, described by the tuple $\{\mathcal{A}_1, \mathcal{A}_2\}$ that is equal to $\{\mathcal{D}_1 + \Delta_1, \mathcal{D}_2 + \Delta_2\}$. $\mathcal{A}_1$ is the number of PRBs allocated to the medical slice in a cell and $\mathcal{A}_2$ is the number of PRBs allocated to the transport slice in a cell. The remaining PRBs i.e. $N - (\mathcal{A}_1 + \mathcal{A}_2)$ are allocated to the background slice.

In this way an action of the agent results in a resource allocation for all the slices in a cell. However, it is noted that, there may be actions that should not be permissible in a state. In the defined problem of resource allocation, the actions that lead to a negative or zero resource allocation for an active slice and the actions that lead to the sum of the allocated PRBs to both the slices, i.e. $\mathcal{A}_1 + \mathcal{A}_2$ to exceed the total number of available PRBs are not permissible. Consider for an example, a state {1,18}. With $N = 25$, the actions with values of Δ_1 in the range $\{-25, -24.., -1\}$ and actions with values of Δ_2 in the range $\{-25, -24.., -18\}$ should not be permissible as they result in PRB allocations of less than or equal to zero. Ideally, given an appropriate reward function and enough exploration (also refer to Section 4.2), the agent itself learns which actions need not be taken in a given state. However, practical systems may be limited by the time available for exploration. Moreover, in case of reinforcement learning in real-world environments, *risky* actions or the actions resulting in adverse consequences during the exploration phase may not be acceptable. In these cases, state-action exploration by an agent must be guided to speed-up exploration and avoid risky actions. Guided-exploration is applicable in problems that have the so-called *state-action permissibility (SAP)* property [43]. An RL problem is said to have SAP, if the agent can decide whether an action is permissible in a state or not. An action is not permissible in a state if the action can never lead to an optimal solution and hence should not be explored at all [43]. The resource allocation problem defined in this study has the SAP property since, given a state $\{\mathcal{D}_1, \mathcal{D}_2\}$, it can be inferred which actions should not be taken at all, for example the actions resulting in a negative PRB allocation for a slice. Hence, we make use of the guided-exploration. Details are discussed in Section 8.3.1.

The slice resource allocation resulting from an action of the agent gives the number of PRBs available to each slice per time slot. A particular resource allocation is used in the cell before a next action is taken by the agent. Recall that, the re-allocation is done periodically at a re-allocation interval T (refer to Section 8.1) and additionally may be triggered when a new call is established in a slice or whe a slice changes its status from inactive to active and vice-versa. A slice is active if at least one call is associated with it.

The actions of the agent determine a positive or negative *margin* over the predicted resource demand of the slices (given by the state $\{\mathcal{D}_1, \mathcal{D}_2\}$). The idea behind such a definition of the actions is to correct for any errors in the resource demand prediction by the resource prediction module and be prepared for unpredictable changes in the traffic, load and channel conditions in the time until the next decision moment. m and n can take integer values from $-N$ to N, where N is the total number of available PRBs in a cell per time slot. In this study, $N = 25$, hence the size of the action-space is equal to $(2N + 1)^2 = 2601$.

Reward

The reward function quantifies the performance of an agent's actions in an environment. Recall that in this study, the environment is a cell in a cellular wireless system with

active calls in the defined three slices, viz. medical, transport and background slices. The agent's actions result in a PRB allocation decision for the slices in the cell. The objective of the agent is to find an optimal resource allocation for the slices in a given environment state such that the QoS requirements of the services in the priority slices (medical and transport) are satisfied and the highest attainable performance is provided to the services in the background (best-effort) slice. As discussed in Section 8.3.1, the agent's actions give resource allocation for the priority slices (the remaining resources go to the background slice). Hence, we define the reward for the agent's actions in the environment as the weighted average of the rewards for the individual priority slices. The reward for a priority slice is expressed via means of the *slice utility* function described in Chapter 7. Hence, the reward for an agent's action is given by the same function as the KPI: weighted average utility of the priority classes described in Chapter 7. The reward for an action of the agent is obtained after duration T (the resource re-allocation interval) of taking an action.

Agent-learning

The goal of the agent is to learn to take optimal actions in each environment state. The state-to-action mapping governing the actions taken by the agent in different states is called a policy and denoted as $\pi(a|s)$ (refer to Section 4.2). The agent learns an optimal policy through experiences of interacting with the environment. In a CMAB problem, the experiences are described by the tuple {state, action, reward}. Here, s is a state provided by the environment, a is the action taken by the agent based on its policy, given the state s, and r is the reward obtained from the environment upon taking the action a. An optimal agent policy in CMAB-RL is the one that maximises the reward at each state. The reward may be presented immediately after the agent's action or after a certain duration, as is the case for the resource allocation problem defined in this study (refer to Section 8.3.1).

Q-learning for CMAB-RL

Recall (refer to Section 4.2) that for an MDP-RL problem, Q-value of a state-action pair $Q(s,a)$ gives the expected long-term reward of taking the action a in the state s and thereafter following the optimal policy π^*. Hence, in MDP-RL, $Q(s,a)$ for a state-action pair $\{s,a\}$ includes the expected (discounted) rewards of subsequent state-actions until a final state. This is unlike the case of CMAB-RL. CMAB-RL is a one-step learning process where the expected reward of an action in a given state does not depend on the future steps of the agent (refer to Section 2.4.1). Hence, for a CMAB-RL problem, $Q(s,a)$ gives the expected reward of taking the action a in the state s.

An agent uses an iterative process of exploration and exploitation to collect experiences denoted as the tuple {state, action, reward} in the environment. Given sufficient exploration time, the agent learns the optimal Q-values, $Q^*(s,a)$, for all state-action pairs in the environment. The optimal policy of the agent is to take the actions that maximise

the Q-values for each state.

In this study, we use the ϵ-greedy exploration strategy for the CMAB-RL agent. Recall (refer to Chapter 2) that in the ϵ-greedy strategy, the agent explores the environment by taking random actions with a probability ϵ and exploits by following the optimal policy learnt so far with probability $1 - \epsilon$. After each experience {state, action, reward}, the agent updates the Q-value of the state-action pair (s, a) using the rule given by (also see Section 4.2)

$$Q(s, a) \longleftarrow Q(s, a) + \theta \times r \tag{8.3}$$

where θ is the learning rate, taken to be equal to 0.01.

After sufficient exploration of a state-action pair (s, a) by the agent, $Q(s, a)$ for the state-action pair converges to its true value $Q^*(s, a)$. The Q-values of all the state-action pairs need to be stored and updated after every state-action visit by the agent. This can be done in a *Q-table*. The size of the Q-table is equal to the number of possible state-action pairs, i.e. $N_{states} \times N_{actions}$. Here, N_{states} is the total number of states in the defined environment and $N_{actions}$ is the total number of possible actions in each state. In the case of resource allocation reinforcement learning (in the CMAB setting) problem as defined in this study, $N_{states} = 26^2 = 676$ and $N_{actions} = 51^2 = 2601$ (including non-permissible actions). Hence, the size of the Q-table becomes huge (here it is equal to 1.76 million values). Moreover, if the agent has to visit each state-action pair for example, at least five times to get accurate estimates of Q-values for all state-action pairs, it will take around a total of nine million state-action visits to derive the optimal policy. As can be seen from the numbers, using a Q-table poses two main challenges. First, the memory required to save a Q-table increases as the number of states and/or actions increases. Second, the amount of time required to explore all state-action pairs in order to create an optimal Q-table is unrealistic.

To overcome these challenges, deep Q-learning (refer to Section 4.2) is used. In deep Q-learning, the agent's experiences are used to learn a function rather than a table. Given a state, the function approximates the Q-values of all possible actions in that state. The function may be approximated using a neural network, known as a deep-Q network (DQN). With DQN, the agent may not have to visit each and every state because DQN allows generalising to unseen states or seen states with unexplored actions to predict the Q-values of actions in these states. However, it still requires a lot of experience to learn an effective function. To further reduce the exploration time, guided exploration (introduced in Section 8.3.1) is used. Guided exploration involves constraining the exploration-space by using the domain knowledge of the underlying task. This means, the agent is prevented from randomly/blindly trying all possibilities but instead is made to choose only permissible actions [43]. In this study, we use guided exploration by prohibiting the agent from taking actions that lead to a negative or zero resource allocation for a slice and the actions that lead to the sum of the allocated PRBs to both the slices to exceed the total number of available PRBs.

Generally, in RL problems employing the ϵ-greedy exploration strategy, the agent starts with $\epsilon = 1$. This means at the start of the training, the agent is only exploring the environment. The value of ϵ is gradually decreased to a very small value, e.g. 0.01. By this point, the agent is assumed to have sufficiently learnt about the environment and hence can exploit its knowledge 99% of the time and explore very little. Another way is to divide the exploration and the exploitation phases. In this case, the agent first only explores ($\epsilon = 1$) and once sufficient experiences have been obtained, the agent only exploits ($\epsilon = 0$). We use the latter strategy. The experiences obtained during the exploration phase are used to update the DQN. The learnt model is then used during the exploitation phase to get optimal actions for the environment-states.

Re-training the agent

It is noted that in general, a reinforcement learning problem is defined for a given scenario. Hence, the agent learns optimal actions that are optimal for that scenario. A scenario is different from another if the network layout/setting differs (Phase 2 vs Phase 1 and Phase 3) or the the definition of the reward function changes (e.g. via the choice of β, γ, w_{med} and w_{tran}). Therefore, in these cases the CMAB agent needs to be re-trained.

8.3.2. Non-intelligent design options for the resource allocation module

Demand-only (no margin-estimation)

In this method, the resource allocation for the slices is based on the resource demand prediction only, that is, no margin is added to cope with e.g. demand prediction errors or unexpected variations in e.g. traffic load or channel conditions. The resource demand prediction module (refer to Section 8.2) predicts the number of PRBs that is required by the priority slices (medical and transport) per time slot, that is $\{\mathcal{D}_1, \mathcal{D}_2\}$. $\mathcal{D}_1$ and $\mathcal{D}_2$ number of PRBs are reserved for the respective slices for an interval T (called re-allocation interval). The remaining PRBs are allocated to the background slice. In case the total demand of the medical slice and the transport slice exceeds the total number of available PRBs, a weighted proportional distribution of the total number of PRBs between the medical slice and the transport slice is used, as described by the following:
Given the following:

$$\mathcal{D}_1 + \mathcal{D}2 > N \tag{8.4}$$

The allocations $\mathcal{A}_1$ and $\mathcal{A}_2$ for the medical slice and the transport slice respectively are given by Equations 8.5 and 8.6.

$$\mathcal{A}_1 = N(1 - \frac{1}{1 + \frac{w_1}{w_2}\frac{\mathcal{D}_1}{\mathcal{D}_2}}) \tag{8.5}$$

$$\mathcal{A}_2 = \frac{N}{1 + \frac{w_1}{w_2}\frac{\mathcal{D}_1}{\mathcal{D}_2}} \tag{8.6}$$

where w_1, w_2 are the weights for the medical slice and the transport slice respectively. If a higher share is required for a slice than its proportional share, the corresponding weight is kept higher than the weight for the other slice. N is the total number of PRBs.

Demand-plus-margin

In this method, a fixed margin is used with the predicted resource demands of the priority slices, e.g. +/- 10% of the slice resource demand to get a resource allocation for the slices. This means, the allocation is given by $\{\mathcal{A}_1, \mathcal{A}_2\} = \{\mathcal{D}_1 + \mathcal{M}_1, \mathcal{D}_2 + \mathcal{M}_2\}$. Here, $\mathcal{M}_1, \mathcal{M}_2$ are the respective margins for the medical slice and the transport slice. If $\mathcal{D}_1 + \mathcal{M}_1 + \mathcal{D}_2 + \mathcal{M}_2 > N$, a weighted proportional distribution of the total number of PRBs is used to get the allocations as described in Equations 8.5 and 8.6 except $\mathcal{D}_1$ is replaced by $\mathcal{D}_1 + \mathcal{M}_1$ and $\mathcal{D}_2$ is replaced by $\mathcal{D}_2 + \mathcal{M}_2$. The choice of margins considered in the evaluation are described in Chapter 9.

8.3.3. Priority-based scheduling

We choose a single-slice priority-based scheduling algorithm as a baseline method. In the context of RAN, it is a conventional (non-slicing) spectrum sharing method based on the priorities of the active services. In this method, the different classes of services are identified with their respective fixed priorities. For instance, in this study, the transport service class is given the highest priority (priority one), followed by the medical service class with priority two and lastly the background (best-effort) service class with priority three. A higher priority indicates a higher preference in the scheduler for PRB allocation. This means as long as there is data to be transmitted by an active call in a higher priority service class e.g. service class with priority one, a call in a lower priority service class, i.e. service class with priority two is not allocated resources. Within a certain service class, the scheduling priority for a particular call is computed based on the delay-based rule as discussed in Section 6.5.1.

8.3.4. Resource spill-over

Recall that the number of PRBs allocated to the background slice is simply what is left after allocating PRBs to the priority slices. But the unused resources in the medical slice and the transport slice in each time slot can be also made available to the background slice. We refer to this as spill-over of resources. Resource spill-over leads to a higher PRB utilization and an improved performance for the services in the background class. But at the same time allowing the calls in the background slice to use the resources reserved for the priority slices may result in a higher inter-cell interference caused to

the priority UEs in the neighbouring cells. In this study we consider that the spill-over of resources is allowed.

9

Experiments and Results

In this chapter we describe the scenario-based comparative assessment that was done to analyse the performance of the proposed AI-based multi-slice resource allocation method. In each considered scenario, the proposed method is compared with alternative resource management methods described in Chapter 8. Priority scheduling, also described in Chapter 8 is used as a baseline in the comparative assessment. In the following sections, we describe in detail, various aspects (simulation settings and parameters) related to the experiments.

9.1. Aspects related to multi-slice resource allocation methods

As described in detail in Chapter 8, the proposed multi-slice resource allocation method uses two modules, namely the resource demand prediction module and the resource allocation module to take resource allocation decisions in a cell. The resource demand prediction module employs an ANN to predict the average resource demand of the considered medical and transport classes. The resource allocation module employs a so-called CMAB agent that takes intelligent resource allocation decisions based on resource demand prediction of the medical and transport classes. We also discussed two alternatives for the resource allocation module in order to define alternative multi-slice resource allocation methods, namely the 'demand-only' method and the 'demand-plus-margin' method (see Chapter 8). In the demand-only method, the resource allocation module applies the resource demand prediction for the medical and transport classes and simply assigns the predicted number of PRBs to the respective slices. In the demand-plus-margin method, the resource allocation module uses pre-defined percentual margins (or mark-ups) with the predicted resource demands of the medical and transport classes and assigns the number of PRBs obtained after correction to the respective slices. Unlike in the case of CMAB-based resource allocation, the allocation obtained by the demand-only method and the demand-plus-margin method may have

the sum of the PRBs allocated to the medical and transport classes exceeding the total available PRBs. In such cases, the resource allocation module uses the so-called weighted proportional distribution method (see Chapter 8) to obtain the final allocation.

For the demand-plus-margin method, three different pairs of correction factors are used in the comparative study. Each pair of correction factors give the corrections used for the medical and transport classes. The considered pairs are (1.1, 0.9), (1.2,0.8) and (1.3,0.7), where 1.1, 1.2, 1.3 are the correction factors used for the medical class and indicate that the predicted resource demand for the medical class must be increased by 10%, 20% and 30% respectively. Similarly, 0.9, 0.8 and 0.7 are the correction factors used for the transport class and indicate that the predicted resource demand for the transport class must be decreased by 10%, 20% and 30% respectively to obtain the resource allocation. The choice of positive correction factors for the medical class and negative correction factors for the transport class is based on the observation from the scenario-based analysis. It was observed that the amount of resource demand prediction for the transport class was always more than the required amount of resources (over-provisioning) and lower for the medical class (under-provisioning).

The proposed multi-slice resource allocation method (with CMAB-based resource allocation module) is referred to as ‘agent’. The demand-only method is referred to as ‘demand’. The demand-plus-margin method with (1.1, 0.9) correction factors is referred to as ‘+/-10%margin’, with (1.2, 0.8) correction factors as ‘+/-20%margin’ and (1.3,0.7) correction factors as ‘+/-30%margin’. The baseline, i.e. priority scheduling is referred to as ‘priority’.

9.2. Aspects related to the use case

The comparative assessment of the proposed multi-slice resource allocation method with the above described alternative methods and the baseline is carried for Phase 2 and Phase 3 of the considered use case (see Chapter 5). As discussed in Chapter 5, Phase 3 covers all aspects of Phase 1, and even more beyond that. Consequently, Phase 1 does not impose any additional challenges beyond those of Phase 3, and is therefore not considered for the scenario-based assessment.

9.2.1. Phase 2

In Phase 2, the ambulance is considered to have arrived at the site of the incidence and is stationary. The paramedic attends to the patient and establishes remote contact with the specialist. Phase 2 is modelled in the urban-macro propagation environment (see Chapter 6).

Medical class

The paramedic uses so-called medical class applications that includes 360-degree video streaming, remote ultrasound and ECG monitoring, haptics communication and au-

dio/video telephony with the specialist. The radio communication aspects of these applications are listed in Table 5.2. The application modelling aspects are described in Chapter 6. As said above, the medical UEs are stationary in Phase 2.

Cell 9 in the considered network (see Chapter 6) is selected for the evaluation of the different multi-slice resource allocation methods. Therefore, the the incident is modelled to have occurred at a randomly selected position in the region marked as Cell 9 in the network layout, as shown in Figure 9.1. The position of the ambulance/paramedic/medical UEs is taken to be the same as the selected position of the incident.

Transport class

The transport class applications that are active in Phase 2 include the transmission of the global maps and the local maps to and from the regular vehicles, respectively and the transmission of the road sensor data. The applications of the intersection controller and the transport applications of the ambulance are not active in Phase 2, since the ambulance is at the site of the incidence (see Chapter 5). The radio communication aspects of the transport applications are listed in Table 5.1. The application modelling aspects are described in Chapter 6.

A straight road segment stretching from one edge of the region marked as Cell 9 to the other is considered as illustrated in Figure 9.1. The vehicles (using the considered transport applications) start from the left edge of the road and move in the right direction in a straight line with speeds randomly selected from a uniform distribution in the range of 30 to 60 kmph. The speed of a vehicle is constant during the simulation time. The intersection controller is placed randomly along the road at the start of the simulation and the road sensors are placed around the IC in a radius of 20 meters.

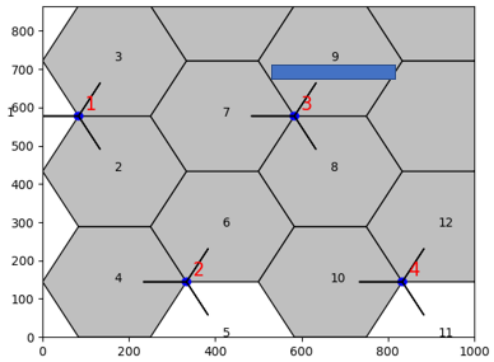

Figure 9.1: Illustration of the considered road and cell for the evaluation of various multi-slice resource allocation methods in Phase 2. The Thick blue line in cell 9 represents the considered road.

Load settings for the medical and transport classes

For the evaluation in Phase 2, we carry out a sensitivity analysis of the considered multi-slice re-allocation methods with varying traffic loads in the medical and transport classes. Nine different load settings are considered and expressed in terms of the qualitative level (low (L), medium (M), high (H)) of the load in the medical and transport classes. These are (L,L), (L,M), (L,H), (M,L), (M,M), (M,H), (H,L), (H,M) and (H,H), where the first index is the level of medical class load and the second index is the level of transport class load. A low medical class load corresponds to an offered traffic load of 4.466 Mbps and that in the transport class corresponds to an offered traffic load of 0.432 Mbps. A medium medical class load corresponds to an offered traffic load of 8.932 Mbps and that in the transport class corresponds to an offered traffic load of 1.008 Mbps. A high medical class load corresponds to an offered load of 13.398 Mbps and that in the transport class corresponds to 1.968 Mbps.

A low medical class load corresponds to the traffic load offered by one ambulance, medium load corresponds to the traffic load offered by two ambulances and high load corresponds to the traffic load offered by three ambulances. Each ambulance involves four calls in the medical class, viz. one 360-degree video streaming in the uplink, one ultrasound and ECG data streaming in the uplink, one haptics control data streaming in the downlink and one bi-directional audio/video streaming. In each load level in the transport class, the traffic offered by the road sensors is fixed. In order to change the load level in the transport class, the total traffic offered by the regular vehicles is varied by varying the number of regular vehicles. Low load corresponds to the traffic load offered by two regular vehicles, medium load corresponds to the traffic load offered by five regular vehicles and high load corresponds to the traffic load offered by ten regular vehicles. Each regular vehicle involves two calls in the transport class, viz. transmission of the global maps in the downlink and transmission of the local maps in the uplink.

9.2.2. Phase 3

In Phase 3, the ambulance carrying the patient drives towards the hospital. Phase 3 is modelled in the rural-macro propagation environment (see Chapter 6).

Medical class

In Phase 3, the paramedic uses the same medical class applications as in Phase 2 but now from an ambulance that is moving at a constant speed of 160 kmph. Hence, the medical UEs are moving at 160 kmph.

Transport class

In addition to the transport class applications active in Phase 2, as mentioned above, the applications of the intersection controller and the transport applications of the ambulance are also active in Phase 3. Furthermore, the regular vehicles also start transmitting real-time videos (see Chapter 5).

Same as in Phase 2, in Phase 3 cell 9 in the considered network is again selected for the assessment of the different multi-slice resource allocation methods. Therefore, a straight road segment stretching from one edge of the cell to the other is considered as illustrated in Figure 9.2. It is noted that the road segment is taken to be representative for the full trajectory of the ambulance. Figure 9.2 looks similar to Figure 9.1, but with the difference in the inter-site distance. The ambulance and the regular vehicles start from the left edge of the road segment and move in the right direction along a straight line. The vehicles move with different speeds randomly selected from a uniform distribution in the range of 90 to 120 kmph. The IC is placed randomly along the road at the start of the simulation and the road sensors are placed around the IC in a radius of 20 meters.

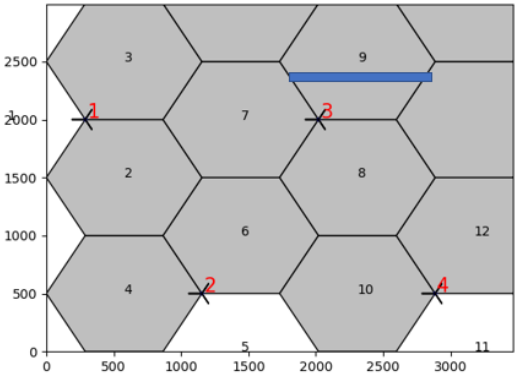

Figure 9.2: Illustration of the considered road and cell for the evaluation of various multi-slice resource allocation methods in Phase 3. The blue rectangle in cell 9 represents the considered road. This figure looks similar to Figure 9.1, but with the difference in the inter-site distance

Load settings for the medical and transport classes

Same as Phase 2, a sensitivity analysis of the considered multi-slice re-allocation methods with varying traffic loads in the medical and transport classes is carried for Phase 3. Again, nine different load settings, viz. (L,L), (L,M), (L,H), (M,L), (M,M), (M,H), (H,L), (H,M) and (H,H) are considered. The offered traffic loads corresponding to the different load levels (L, M, H) in the medical class in Phase 3 are the same as in Phase 2. However, the offered traffic loads corresponding to the different load levels in the transport

class in Phase 3 differs from that in Phase 2. This is because as discussed above, in Phase 3 additional transport applications are active, viz. the transport applications of the ambulance, intersection controller and real-time videos by regular vehicles.

A low transport class load corresponds to the traffic load offered by two regular vehicles and the transport applications of one ambulance, medium load corresponds to the traffic load offered by five regular vehicles and the transport applications of two ambulances and high load corresponds to the traffic load offered by ten regular vehicles and the transport applications of three ambulances. In each load level in the transport class, the traffic offered by the intersection controller and the road sensors is fixed. In order to change the load level in the transport class, the total traffic offered by the regular vehicles and the transport applications of the ambulance is varied by varying the number of regular vehicles and the number of ambulances, respectively. Each regular vehicle involves three calls in the transport class, viz. transmission of the global maps in the downlink, transmission of the local maps in the uplink and real-time video streaming in the uplink. Each ambulance involves five calls in the transport class, viz. transmission of the global maps in the downlink, transmission of the local maps in the uplink, beyond-vision video (see Chapter 5) in the downlink, signal request messaging in the uplink and priority status messaging in the downlink. In Phase 3, a low transport class load corresponds to an offered traffic load of 1.234 Mbps, medium transport class load corresponds to an offered traffic load of 2.81 Mbps and high transport class load corresponds to an offered load of 5.17 Mbps.

9.2.3. Background class

The spatial distribution of the background UEs in the network and the characteristics of the applications of the background UEs are common in both Phase 2 and Phase 3. These are described in Chapter 6. The considered speeds of the background UEs in Phase 2 and Phase 3 are described in Table 6.2. The load setting for the background class is kept constant for all cells in the considered network and throughout the simulation time. The load setting is common in both Phase 2 and Phase 3. As described in Chapter 6, the calls in the background class are generated according to a Poisson process. For the evaluation, we consider a mean arrival rate of 20 calls per second per cell which corresponds to a high load in the background class.

9.3. Evaluation-related aspects

The key performance indicators (KPIs) used in this study are described in detail in Chapter 7. The considered KPIs include the so-called medical class utility, the transport class utility, the average medical class performance and transport class performance, the average cost associated with reservation of resources for medical class and transport class, the weighted average utility of priority classes, i.e the medical and transport classes, the 10^{th} throughput percentile of background class applications and the cell resource utilisation.

As described in Chapter 7, the definition of the KPIs involve various parameters. These parameters are summarized in Table 9.1. The parameters can take different values which affect the values of the KPIs involving these parameters. The choice of the parameters depends on the policies of a certain slice as defined by slice owner and/or network operator. In addition, the re-allocation interval T is another parameter that can be tuned based on slice requirements and traffic characteristics. In this study we choose $T = 100ms$ (see Chapter 8).

Table 9.1: The considered values for the different evaluation parameters

Parameter	Values	Description
α	1	Determines the penalty for service performance below the required level and indicates relative importance of satisfying the required QoS as compared to the average cost associated with resources.
β	0.01, 1	Determines the penalty for service performance degradation. A higher value means higher penalty for a given performance below the required level.
γ	0.025, 9	Determines the average cost associated with resource reservation.
Medical class weight (w_{med})	0.2, 0.5	Determines the relative importance of the medical class as compared to the transport class in the weighted average utility calculation. A higher weight also indicates a higher priority for resource allocation in the case of priority scheduling and the agent methods.
Transport class weight (w_{tran})	0.8, 0.5	Determines the relative importance of the transport class as compared to the medical class in the weighted average utility calculation. A higher weight indicates a higher priority for resource allocation in the case of priority scheduling and the agent method.
Re-allocation interval	100 ms	Determines the periodicity of slice resource re-allocation in case of agent, demand, +/-10% margin, +/-20% margin and +/-30% margin methods.

9.4. Experiments

The key purpose the experiments performed in this study is to compare the different multi-slice resource allocation methods. We carry out two sets of sensitivity analyses for the considered multi-slice resource allocation methods. The first set with different loads in the priority classes and the second set with different values of various evaluation parameters discussed above. In the following sections, we describe each experiment and present and analyse their corresponding results.

We calculate the values of the different KPIs described in Chapter 7 for the different

Table 9.2: The parameters and their corresponding values considered for the sensitivity analysis w.r.t loads in the medical and transport classes.

Parameter	Considered values					
α	1					
β	1					
γ	0.025					
Medical class weight (w_{med})	0.2					
Transport class weight (w_{tran})	0.8					
Medical class load	Low	Medium			High	
Transport class load	Low	Medium			High	
Multi-slice resource allocation method	Agent	demand	+/- 10% margin	+/- 20% margin	+/- 30% margin	priority
Re-allocation interval T	100 ms					

analyses. In order to indicate the degree of statistical confidence in the calculated KPIs, the calculation of confidence intervals (CI) is needed. In this study, we chose a confidence level of 95% for all the considered KPIs. We illustrate the confidence intervals for the bottom-line metric for the priority classes, namely weighted average utility of the priority classes and the bottom-line metric for the background class, namely the 10^{th} throughput percentile of the background applications (and not elsewhere to keep the figures somewhat cleaner).

9.4.1. Sensitivity analysis with different loads in the medical and transport classes

The aim of the experiment is to demonstrate that the proposed multi-slice resource allocation method with the intelligent CMAB-based resource allocation module is able to adapt to variations in loads in the priority classes in order to achieve high performance of applications in the priority classes while minimizing the average cost associated with resource reservation. The above is demonstrated by comparing the proposed method with the alternative allocation methods and the baseline. In this experiment, different loads in the medical and transport classes are considered as described in Section 9.2.1 and Section 9.2.2. The values of the so-called evaluation parameters are fixed. The experiment is performed for both Phase 2 and Phase 3. The parameters and their corresponding values for this experiment are listed in Table 9.2. The values of all the considered KPIs are calculated and illustrated.

As mentioned in the table, unequal weights are chosen for the medical and transport classes with a higher weight given to the transport class. This is to aid a fair comparison with the baseline i.e priority scheduling which is configured to give the highest priority to the transport class, followed by the medical class.

Phase 2

The average performance of applications and the average cost associated with resource reservation for the medical and transport classes

Figure 9.3 illustrates the average performance of the medical class applications for the different resource-allocation methods and for different loads in the priority classes in Phase 2. It can be observed from the figure that for a given medical class load, the average performance of the medical applications decreases with increase in the transport class load in the case of *priority* and *agent*. In general, the degradation in the medical class performance is more in the case of *priority* as compared to that of *agent* because unlike *priority*, *agent* does not give strict priority to the transport class but tries to efficiently (in accordance with the priorities set by the slice weights) balance the performance in both classes in such a way that it maximizes the weighted average utility.

Agent adapts to the change in the transport class load and in case of a higher transport class load reserves more resources for the transport class by taking away some resources from the medical class. This can be observed from Figure 9.4. Figure 9.4 shows the average cost of resource reservation in the medical class for the different resource-allocation methods and for different loads in the priority classes in Phase 2. It is observed that with an increase in the transport class load, the average cost associated with resource reservation for the medical class decreases for a given medical class load. The reduction in the reserved resources in the medical class causes a performance drop. It is noted that *agent favours* taking resources away from the medical class to satisfy the transport class demand because the transport class is configured to have a higher weight in the calculation of the weighted average utility. This further shows that *agent* behaves according to the objectives set by the slice owner and/or network operator using the relative class weights.

On the other hand, in the case *of demand, +/-10% margin, +/-20% margin* and *+/-30% margin* methods, for a given medical class load, the average medical class performance is constant. In these methods, a higher amount of resources are allocated to the transport class with increase in the transport class load but not at the cost of a reduction of the amount of resources allocated to the medical class. This is because according to the allocation decisions made by these methods, there is no shortage of resources even as the transport load increases. Hence, the resources allocated to the medical class is constant with transport class loads and consequently the average medical class performance is also constant. The fact that the amount of resources reserved for the medical class for a given medical class load by these methods is constant can be observed from Figure 9.4 where the average cost associated with resource reservation in the medical class is constant for a given medical class load.

In the case of agent, however, a higher amount of resources are allocated to the medical class as compared to *demand, +/-10% margin, +/-20% margin* and *+/-30% margin* methods. Hence, for *agent*, there is a scarcity of resources and hence agent takes some resources from the medical class to satisfy the increased resource demand

of the transport class.

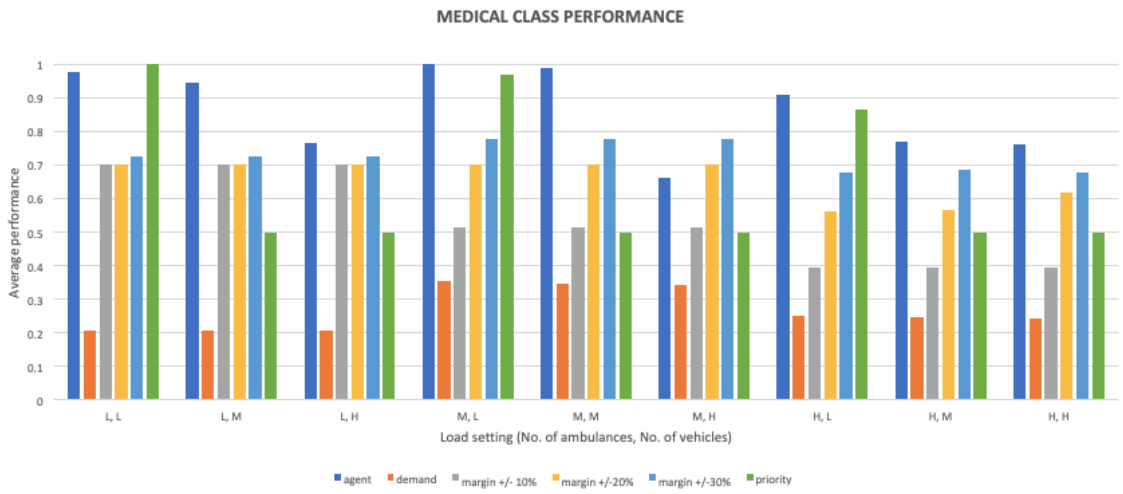

Figure 9.3: Average performance of applications in the medical class in Phase 2 for different load settings in the medical and transport classes.

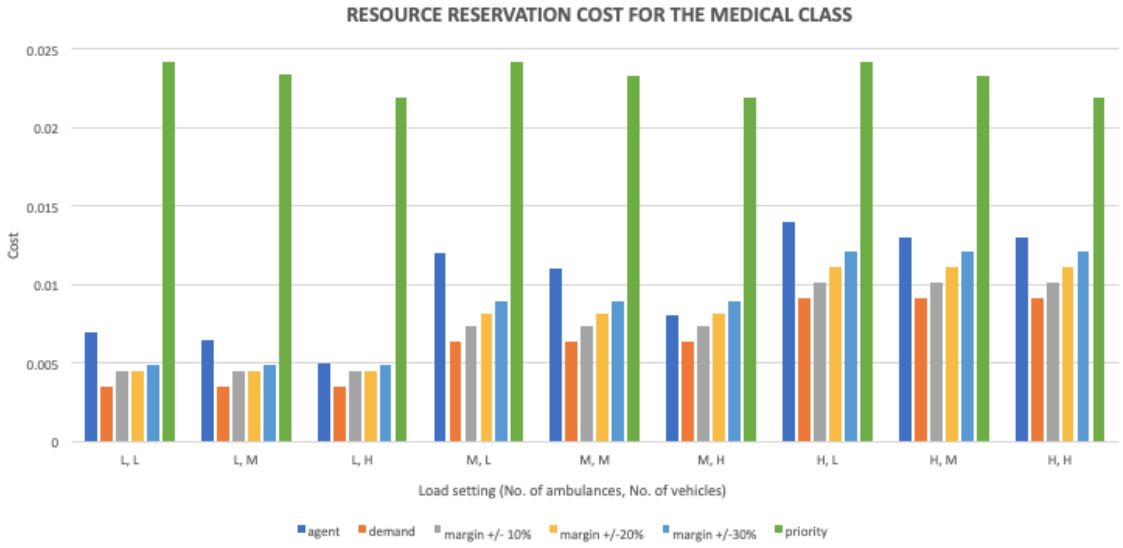

Figure 9.4: Average cost associated with resource allocation in the medical class in Phase 2 for different load settings in the medical and transport classes.

Figure 9.5 shows the average performance of the transport class applications for the different resource-allocation methods and for different loads in the priority classes in Phase 2. It is observed from the figure that the highest average transport class performance is achieved in the case of *priority* for all load scenarios. This is because it gives strict priority to the transport class. However as observed above, this comes at the cost of a reduced medical class performance. The fact that the highest priority is given to the transport class in the case of *priority* means that 100% of the available resources are made available to the transport class. This means that the cost associated with resource reservation for the transport class in the case of *priority* is equal to γ as also illustrated in Figure 9.6. Figure 9.6 shows the average cost of resource reservation in the transport class for the different resource-allocation methods and for different loads in the priority classes in Phase 2. Recall that since the medical class has the second highest priority, the resources that are left after allocating the required resources to the transport class are made available to the medical class.

Priority always performs better than any other method in terms of the average transport class performance because it gives strict priority to the transport class. However, it is observed that *agent* has either equal or slightly lower average transport class performance than *priority*. Also, recall that the average medical class performance is always higher in the case of *agent* than that of *priority*. This is because unlike *priority*, the *agent* reserves resources beforehand for both classes based on the collective resource demand such that high performance for both classes is achieved with minimum resources.

It is also observed from Figure 9.4 and Figure 9.6 that the cost associated with resource reservation for both medical and transport classes in the case of *priority* is very high. This is because in the case *priority* all the resources are made available to the transport class which is much more than what is required. Similarly for the medical class. Furthermore, it is observed that the cost of resource reservation in the case of *agent* is much lower than that in the case of *priority* for almost the same average performance for the transport class applications and even better performance for the medical class applications. This shows that *agent* is able to find an efficient resource allocation for the medical and transport classes. The cost associated with resource reservation for the medical class is the least in the case of *demand* for all the load settings because not enough resources are reserved by *demand* for the medical class. The effect of under-provisioning of the resources can be observed from the worst average medical class performance in the case of *demand*. The cost associated with resource reservation and hence the average medical class performance increases for *+/-10% margin, +/-20% margin* and *+/-30% margin* due to reservation of additional resources (10%, 20% and 30% respectively).

Figure 9.5: Average performance of applications in the transport class in Phase 2 for different load settings in the medical and transport classes.

Figure 9.6: Average cost associated with resource allocation in the transport class in Phase 2 for different load settings in the medical and transport classes.

Utility of the medical and transport classes and the weighted average utility of the medical and transport classes

As defined in Chapter 7, the utility of a priority class includes both the average application performance level and the average resource reservation cost for the class. A priority class is said to have a high utility if the trade-off between a high average performance of the applications in the class and a low resource reservation for the class is satisfied according to the policy defined by the slice owner and/or network operator via means of , β (penalty for performance degradation) and γ (cost associated with resource reservation). Figure 9.7 and 9.8 illustrate the utility of the medical and transport classes, respectively for the different resource-allocation methods and for different loads in the priority classes in Phase 2. It is observed from Figure 9.7 and Figure 9.8 that in each load scenario, *agent* always has the highest utility among all methods for either one or both of the priority classes. This is only true for *agent*. Additionally, when one of the utilities is not the highest for the case of *agent*, it is only slightly less than the highest. In the case of high/medium load in the transport class, *priority* achieves a high utility for the transport class but at the cost of a very low utility for the medical class. The above is observed from the bottom-line evaluation metric for the priority classes, namely the weighted average utility of the two priority classes. Figure 9.9 shows the weighted average utility of the two priority classes for the different resource allocation methods and for different loads in the priority classes in Phase 2. It can be observed from the figure that in each load scenario and for the given settings of β,γ and the medical and transport class utility weights, the highest weighted average utility is obtained for the case of *agent*. This shows that the *agent* is able to balance the resource requirements of two priority classes and decide on an efficient resource allocation that is based on the collective resource demand of the priority classes. This is in contrast with other methods, viz. *demand, +/-10% margin, +/-20% margin* and *+/-30% margin* that take resource allocation decisions only based on the demands of the individual classes and do not consider the overall picture of demands of both classes. The weighted average utility for the case of *priority* is lower than that of *agent* because *priority* gives all the resources to one single priority class, i.e. the transport class, with strict priority causing severe degradation for the utility of the medical class in the cases of high/medium transport class loads. The above conclusions are drawn with sufficient statistical relevance because the CI for the weighted average utility in the case of agent is narrow and in most cases non-overlapping with the CIs for other methods.

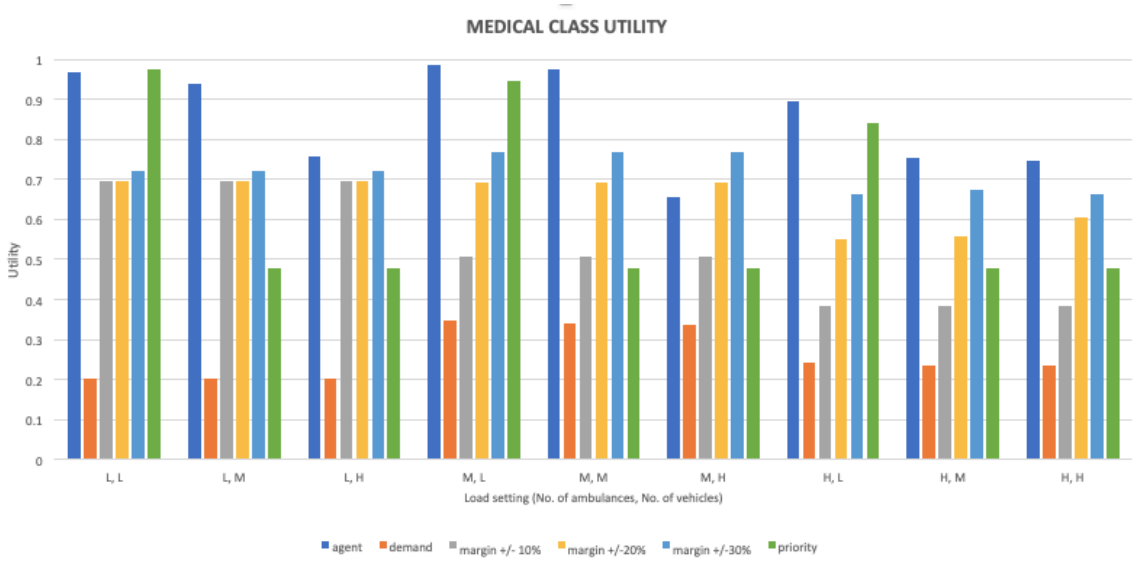

Figure 9.7: Utility of the medical class in Phase 2 for different load settings in the medical and transport classes.

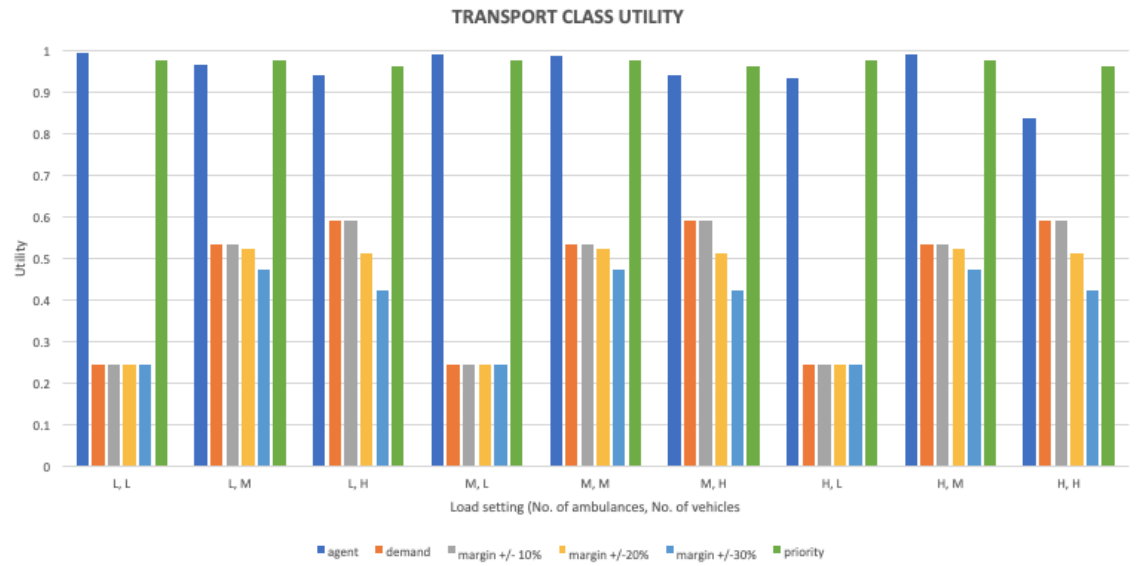

Figure 9.8: Utility of the transport class in Phase 2 for different load settings in the medical and transport classes.

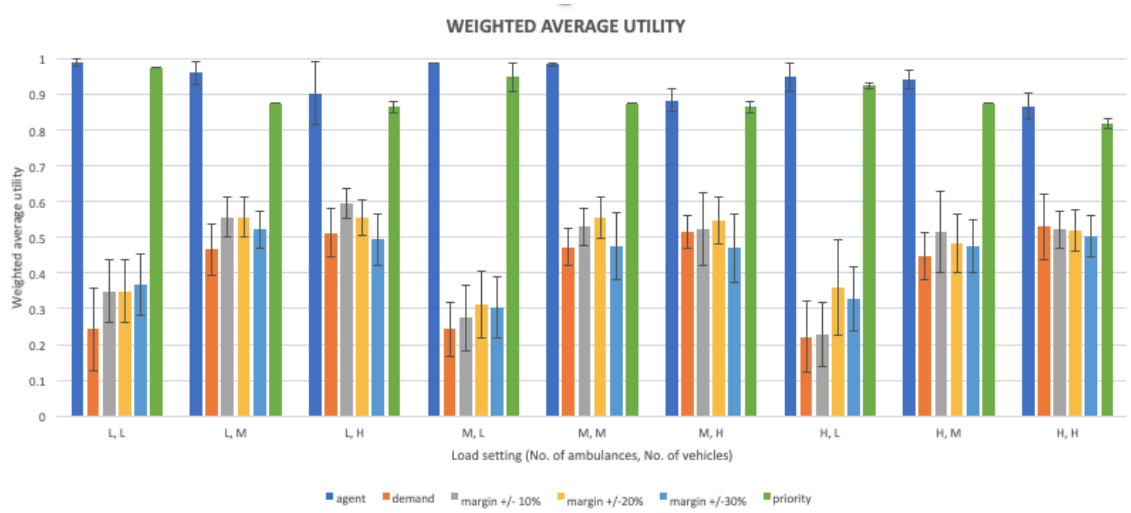

Figure 9.9: The weighted average utility of the medical and transport classes in Phase 2 for different load settings in the medical and transport classes.

10^{th} throughput percentile of applications in the background class

The performance of the background class is evaluated based on the 10^{th} throughput percentile of the applications in the background class. Figure 9.10 shows the 10^{th} throughput percentile of the background applications for various resource allocation methods and for different loads in the medical and transport classes. A trend can easily be observed that as the load in the priority classes increases (from (1,2) to (3,10)), the 10^{th} throughput percentile decreases under of all resource allocation methods. This is because with higher load and hence higher resource demand in the priority classes, fewer resources are available for the background class.

As mentioned in Chapter 8, in each time slot the resources that are reserved for the priority classes and were unused by the applications in these classes are made available to the background class. Therefore, the amount of resources that are relatively available to the background class is higher when fewer resources are reserved for the priority classes. It can be observed from the above analysis that this is potentially the case for *demand*, *+/-10% margin, +/-20% margin* and *+/-30% margin* methods. Hence, a relatively high 10^{th} throughput percentile is observed for these methods which comes at the cost of degradation in performance of the priority classes. Relatively low 10^{th} throughput percentile is observed for the *agent* and *priority* methods. This is because in these methods relatively high amount of resources are reserved for the priority classes and only the remaining (relatively few) resources are available for the background class.

It is observed from the figure that the confidence intervals for most resource allocation methods (in all load scenarios) are overlapping. Hence, no conclusions can be drawn (with sufficient statistical relevance) regarding the differences in the KPI across

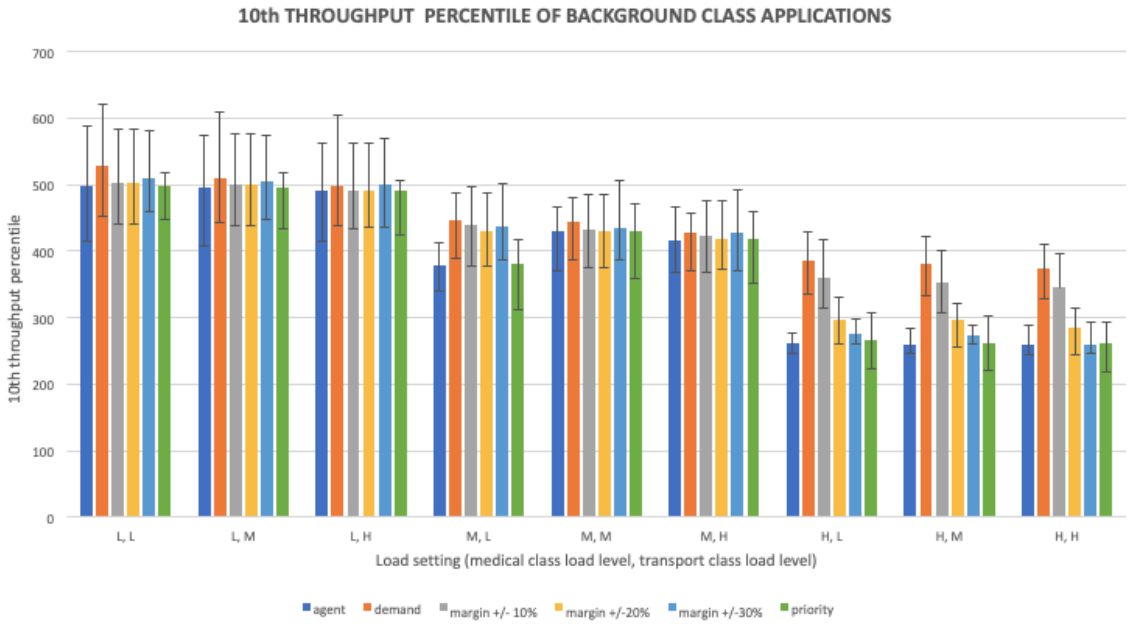

Figure 9.10: 10^{th} throughput percentile for background applications in Phase 2 for different load settings in the medical and transport classes.

different resource allocation methods. The results do hint at some conclusions as discussed above but due to closeness of the results (for a given load) and the width of the confidence intervals such conclusions cannot be made firm. Essentially, from the perspective of the background class, the performance difference between the different methods is not very significant.

Cell resource utilisation As discussed in Chapter 7, the cell resource utilisation indicates how many resources were used to yield the performance levels for the different classes shown above. For a certain network traffic in a cell, in general, the cell resource utilisation is lower for a higher number of packet drops. This is because for a certain offered traffic, if some amount of the traffic is dropped, the number of PRBs that will be used will drop. Therefore, for a certain load in the priority and background classes, cell resource utilisation would be the maximum for the most efficient resource distribution among all the classes. Figure 9.11 illustrates the cell resource utilisation for various resource allocation methods and for different loads in the medical and transport classes. A trend can easily be observed that as the load in the priority classes increases (from (1,2) to (3,10)), the cell resource utilisation increases for the cases of all resource allocation methods for a fixed load in the background class. This is because with a higher load

more traffic is handled by the cell. The cell resource utilisation is the highest in the case of *agent* or *priority* and least in the case of *demand* for all load scenarios. This is because in the cases of priority and agent, the resource allocation is most efficient, so that more traffic (in total) can be handled by the cell. This can be observed case of *demand*, not enough resources are allocated to the priority classes as also indicated by the relatively low average performance of applications in these classes. As a result of such under-provisioning, an undesirably high number of packets corresponding to these applications are dropped, effectively leading to a decrease in the traffic that is handled by the cell and hence a lower cell resource utilisation. On the other hand, in the cases of *priority* and *agent*, enough resources are allocated to the priority classes such that the number of packet drops is minimised for the applications in these classes. Hence, for a given load (approximately) the same high cell resource utilisation is obtained.

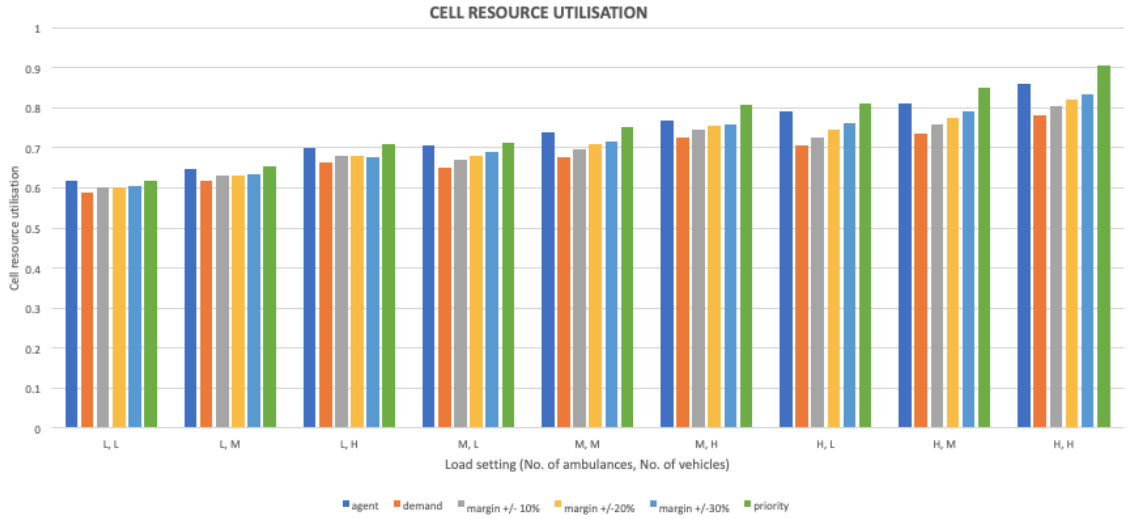

Figure 9.11: Cell resource utilisation in Phase 2 for different load settings in medical and transport classes.

Phase 3

The average performance of applications and average cost associated with resource reservation for medical and transport classes

Figure 9.12 illustrates the average performance of applications in the medical class for Phase 3. It can be observed from the figure that similar to Phase 2, for a given medical class load, the average performance of the medical applications falls with an increase in the transport class load in the case of *priority*. In the case of *agent*, however, the same phenomenon is observed only for medium and high medical class loads. For the

low medical class load, the average performance is the highest among the considered resource allocation methods and does not change with an increase in the transport class load. The reason is found in the average cost associated with resource reservation as illustrated in Figure 9.13. This figure shows the average cost of resource reservation for the medical class for the different resource allocation methods and different loads in the priority classes in Phase 3. It is observed that the amount of resources given to the medical class does not change with the increase in the transport class load. This is because for the case of low medical class load and according to *agent's* resource allocations decisions there are enough resources available to give more to the transport class as the transport class load increases without taking any resources away from the medical class. However, for higher medical class loads, there is a scarcity of resources. Hence, as the transport class load increases, *agent* takes away some resources from the medical class to satisfy the increased demand of the transport class. Therefore, the cost of resource reservation for the medical class and the average medical class performance decreases. This behaviour of *agent* can be explained better by observing the average transport class performance for different load scenarios. This is illustrated in Figure 9.14. It can be seen from the figure that in the case of *agent*, the average transport class performance still remains high for all load scenarios, although lower than other methods. But it is important to note that in the case of all other methods, the average medical class performance is worse than that in the case of *agent*. Hence, *agent* balances the average performance of the applications in both the priority classes in order to achieve the highest weighted average utility of priority classes in all load scenarios. This is illustrated in Figure 9.18. Figure 9.18 illustrates the weighted average utility of priority classes for the different resource allocation methods and for different loads in the priority classes in Phase 3.

The average medical class performance is the worst in the case of *demand*. This is even worse than the average performance obtained in the case of *demand* in Phase 2. This shows that in Phase 3, the prediction of resource demand of the medical class by the resource demand prediction module is highly inaccurate and substantially lower than the actual demand. As a consequence of very inaccurate resource demand predictions, the methods that rely on heuristics such as using correction factors with the predicted resource demand (*+/-10% margin*, *+/-20% margin* and *+/-30% margin* methods) also result in poor average performance. However, *agent* despite the fact that it is also fed with these inaccurate resource predictions, benefits from the additional CMAB layer that is trained to to some degree to deal with such poor predictions and hence is able to achieve high average performance both in Phase 2 and Phase 3. Hence, the performance of *agent* is less significantly affected by poor performance of the resource demand prediction module.

On the other hand, the average performance of applications in the transport class is good in the cases of *demand, +/-10% margin, +/-20% margin* and *+/-30% margin*. This is illustrated in Figure 9.14. This is unlike in Phase 2 where the average transport class performance is very low for these methods. The reason is because in Phase 3, the resource demand predicted for the transport class by the resource demand predic-

tion module is higher than required. Since *+/-10% margin, +/-20% margin* and *+/-30% margin* use correction factors smaller than one with the predicted demands, they lead to slight under-provisioning of the resources in the transport class and hence slightly worse performance as compared to *demand*.

Recall that for *demand*,*+/-10% margin, +/-20% margin* and *+/-30% margin* methods, the cost of resource reservation or, equivalently, the amount of reserved resources for a priority class is constant if the load in the same priority class is constant. However, it is observed in Figure 9.15, showing the average cost associated with resource reservation for the transport class, that for the above-mentioned methods, in the case of high load in the transport class, the cost of resource reservation or the amount of reserved resources decreases with decrease in the medical class load. This is because in these cases, the sum total of the resources allocated to the medical and transport classes exceeds the total available resources. Hence, the weighted proportional distribution (mentioned above) is used to obtain the final allocation which causes a decrease in the share of the transport class.

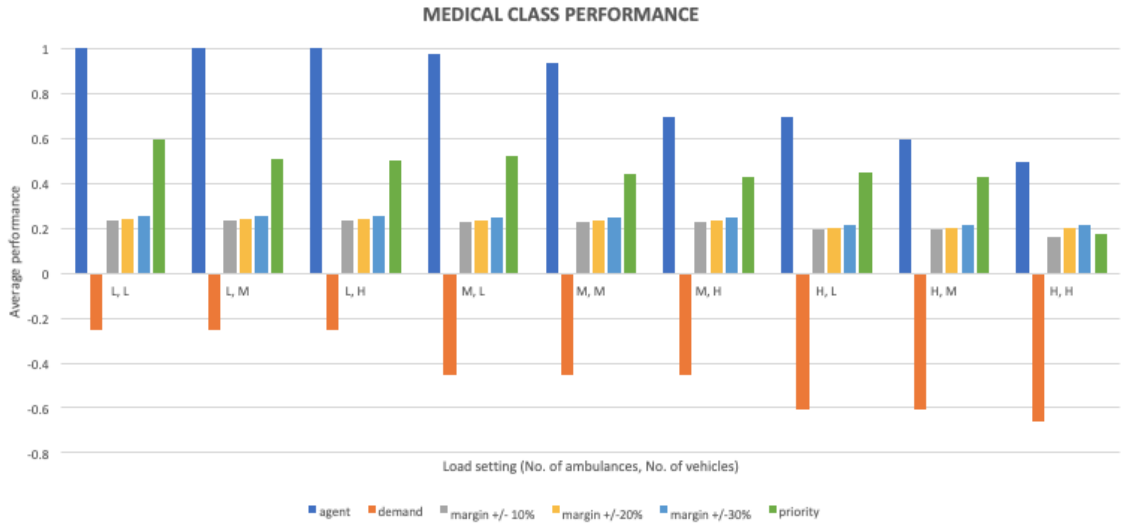

Figure 9.12: The average performance of applications in the medical class in Phase 3 for different load settings in medical and transport classes.

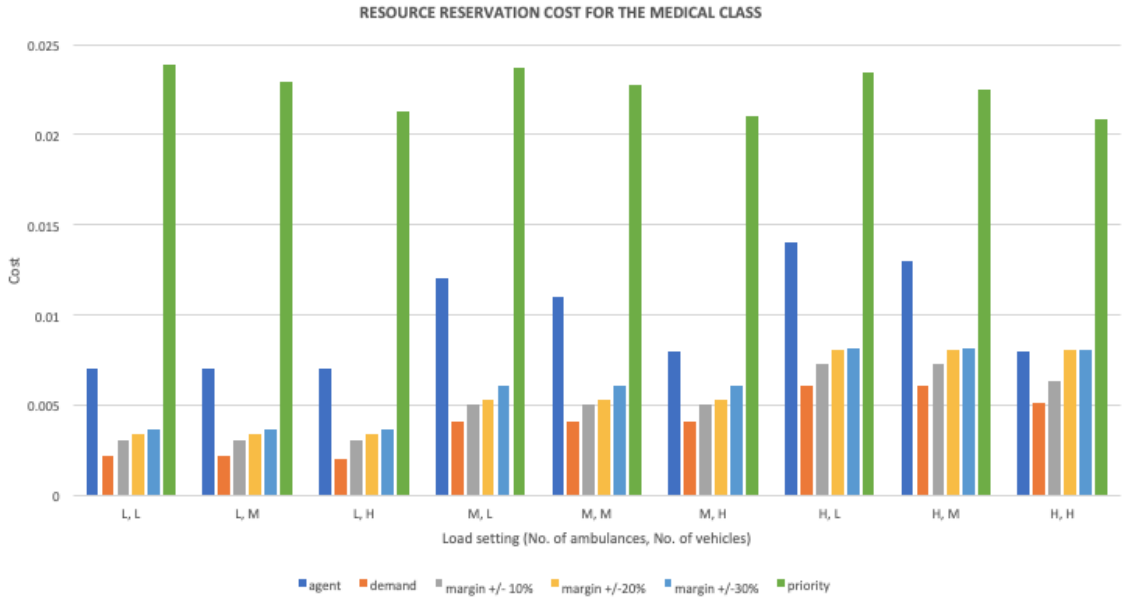

Figure 9.13: Average cost associated with resource allocation in medical class in Phase 3 for different load settings in medical and transport classes.

Figure 9.14: The average performance of applications in the transport class in Phase 3 for different load settings in medical and transport classes.

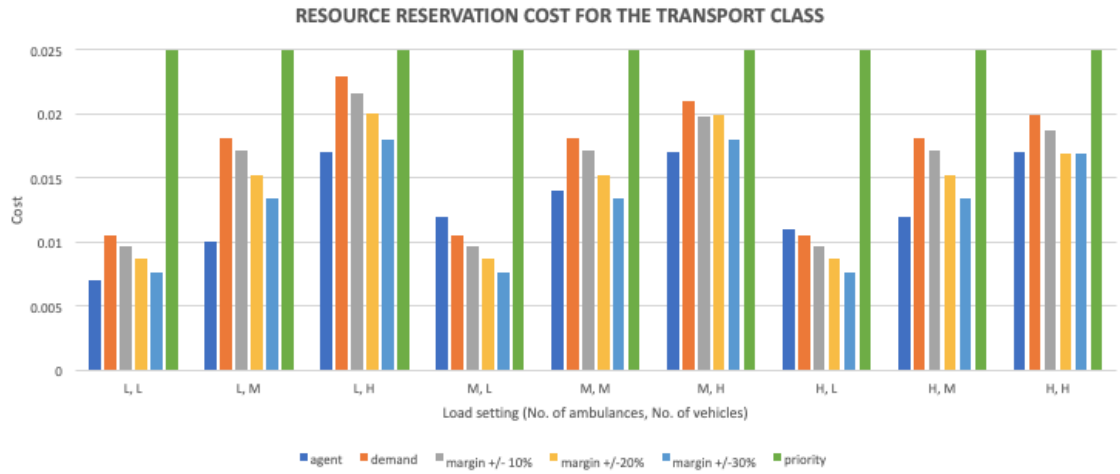

Figure 9.15: Average cost associated with resource allocation in transport class in Phase 3 for different load settings in medical and transport classes.

Utility of medical and transport classes and weighted average utility of medical and transport classes

Figure 9.16 and Figure 9.17 illustrate the utility of medical and transport classes respectively for the different resource allocation methods and different loads in the priority classes in Phase 3. It can be observed that *agent* outperforms all other methods in terms of medical class utility for all load scenarios. Also, *agent* performs either the best or very close to the best method in terms of transport class utility. As a result, the weighted average utility of priority classes is the maximum in the case of *agent* for all load scenarios as illustrated in Figure 9.18. It is observed from the figure that the confidence intervals for the weighted average utility are narrow for all resource allocation methods (for all load scenarios) and in almost all the cases non-overlapping. Hence, the above conclusions are drawn with sufficient statistical relevance.

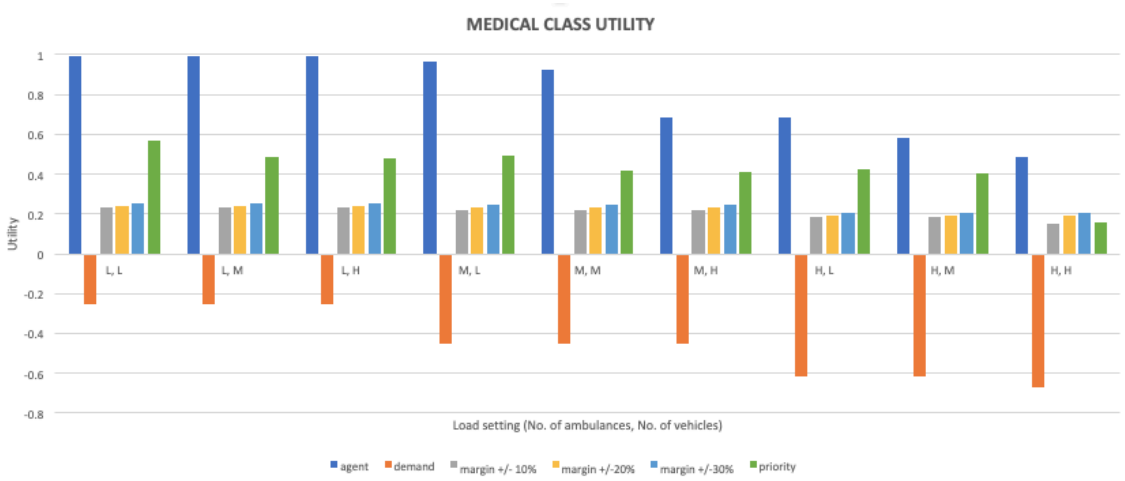

Figure 9.16: Utility of medical class in Phase 3 for different load settings in medical and transport classes.

Figure 9.17: Utility of transport class in Phase 3 for different load settings in medical and transport classes.

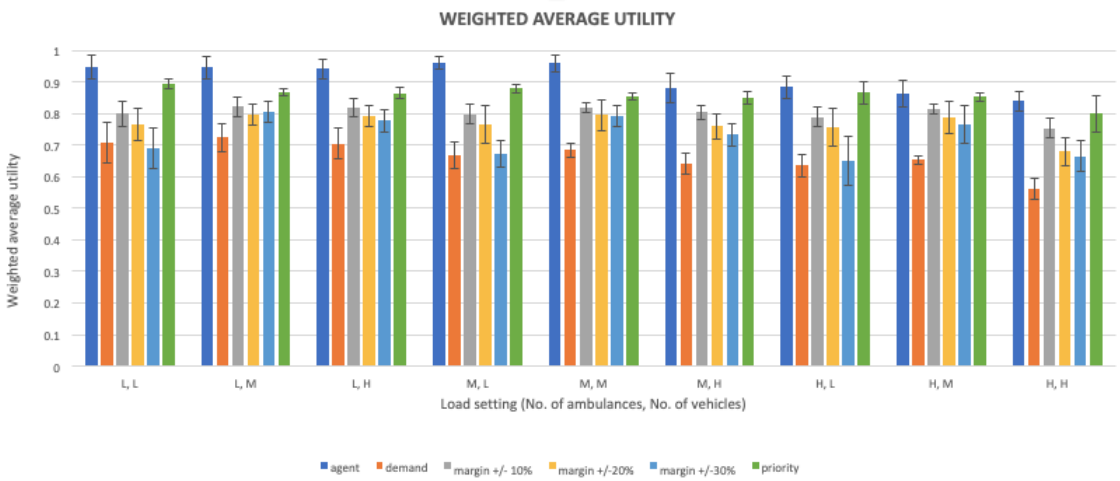

Figure 9.18: Weighted average utility of medical and transport classes in Phase 3 for different load settings in medical and transport classes.

10^{th} throughput percentile of applications in the background class

Figure 9.19 illustrates the 10^{th} throughput percentile of applications in the background class for different loads in the medical and transport classes. Same as in Phase 2, it is observed for Phase 3 as well that the confidence intervals for most resource allocation methods (in all load scenarios) are overlapping. Hence, no conclusions can be drawn

(with sufficient statistical relevance) regarding the differences in the KPI across different resource allocation methods. The results do hint at some conclusions as discussed below but cannot be made firm.

Same as Phase 2, the 10^{th} throughput percentile is the highest in the case of *demand*, with *+/-10% margin, +/-20% margin* and *+/-30% margin* following closely for all load scenarios. It is lower for priority and *agent*. The average performance of the applications in the priority classes shows the opposite relation, i.e it is high for *agent* and priority and lower for *demand*, *+/-10% margin, +/-20% margin* and *+/-30% margin*. Hence, a trade-off exists between the performance of the background applications and the performance of the priority applications. Since *agent* is designed to maximize the performance of the priority classes via the reward function (see Chapter 8), it achieves the maximum weighted average utility of priority classes because it ensures that all the offered priority class traffic is transmitted (no dropping) which maximises the resources utilised by the priority classes, thus minimising the resources available for the background class.

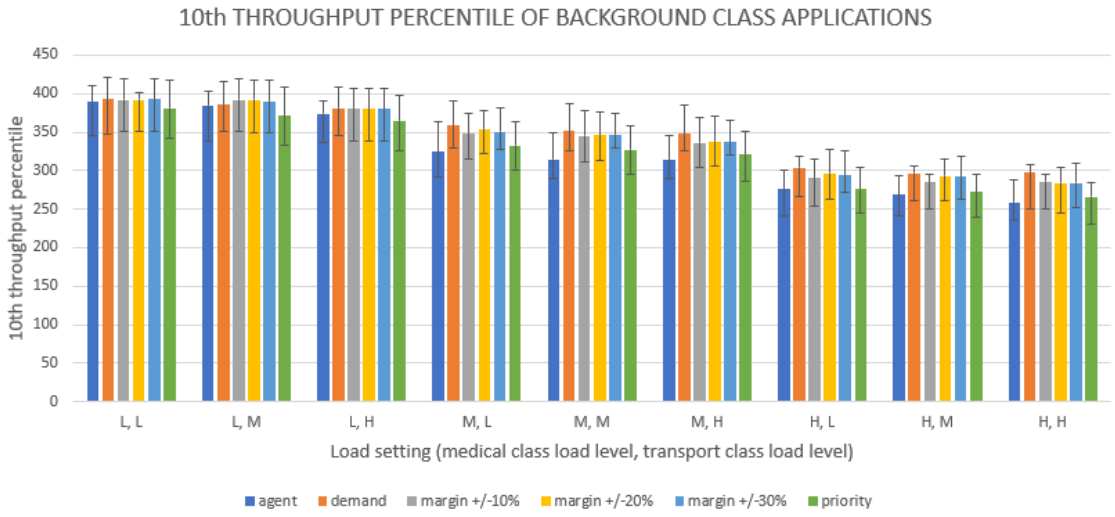

Figure 9.19: 10^{th} throughput percentile for background applications in Phase 3 for different load settings in medical and transport classes.

Cell resource utilisation

Figure 9.20 shows the cell resource utilisation for different resource allocation methods and different loads in the medical and transport classes. Same as in Phase 2, the highest resource utilisation is obtained for *priority* or *agent* for all load scenarios, indicating that minimum packet drops occur for these allocation methods and henc

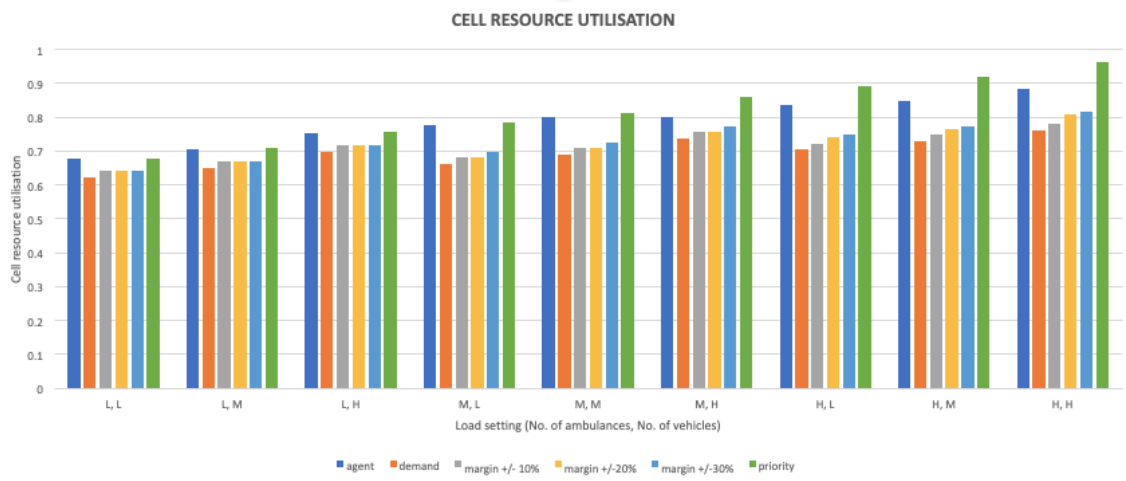

Figure 9.20: Cell resource utilisation in Phase 3 for different load settings in medical and transport classes.

9.4.2. Sensitivity analysis with the different evaluation parameters

The aim of this experiment is to demonstrate that the proposed multi-slice resource allocation method with intelligent CMAB-based resource allocation module is able to adapt to the objectives set by the slice owner and/or network operator in terms of the cost of resource reservation, relative class weights and performance penalty. This is done by comparing it with the alternative slice-oriented resource allocation methods and the baseline of priority scheduling in a common (or single) slice. In this experiment, the load in the medical and transport classes is kept fixed and set to (M,M), i.e. medium load in both medical and transport classes. A sensitivity analysis is done w.r.t the so-called evaluation parameters (β,γ, w_{med} and w_{tran}). The experiment is performed for both Phase 2 and Phase 3. The fixed and variable parameters for this experiment are listed in Table 9.3. The value of the bottom-line KPI, i.e. the weighted average utility of priority classes is calculated and shown in a separate chart for each phase..

Table 9.3: The parameters and their corresponding values for sensitivity analysis with the different evaluation parameters.

Parameter	Considered values					
α	1					
β	0.01			1		
γ	0.025			9		
Medical class weight (w_{med})	0.2			0.5		
Transport class weight (w_{tran})	0.8			0.5		
Medical class load	Medium					
Transport class load	Medium					
Multi-slice resource allocation method	Agent	demand	+/- 10% margin	+/- 20% margin	+/- 30% margin	priority
Re-allocation interval T	100 ms					

Phase 2 and Phase 3

Weighted average utility of priority classes

Figure 9.21 shows the values of the weighted average utility of priority classes for a fixed setting of load in the priority classes and different settings of the evaluation parameters, viz. β, γ, w_{med} and w_{tran} for Phase 2. Figure 9.22 illustrates the same for Phase 3. The horizontal axis in the figures shows the different sets values of the evaluation parameters, i.e. (β, γ, w_{med} and w_{tran}). It is observed for both the phases that for all settings, the highest weighted average utility is obtained by *agent*. This shows that *agent* adapts to different objectives of multi-slice resource allocation defined by slice owners and/or network operator. Furthermore, in the case of low cost ($\gamma = 0.025$), the performance of *priority* closely follows that of *agent*. However, for high cost associated with resource reservation, i.e $\gamma = 9$, *priority* performs the worst. This is because it reserves all the available resources for the priority classes and is consequently imposed a high cost, unlike in the case of *agent*, *demand, +/-10% margin, +/-20% margin* and *+/-30% margin* which only reserves a portion of the resources for the priority classes. Moreover, the performance of *priority* as indicated by the weighted average utility of the priority classes varies with different settings of weights of the priority classes. Since *priority* gives a higher priority for resource allocation to the transport class as compared to the medical class, if a higher weight is selected for the transport class as compared to the medical class, its allocation decisions align with the priority choice imposed by the network operator. However in the case of equal weights, that is not the case. As a result, for a positive utility of the transport class, the weighted average utility increases and for a negative utility of the transport class, the weighted average utility decreases if a higher weight is selected for the transport class as compared to if equal weights are selected. However, *agent* aligns allocation decisions with the priority choice imposed by the network operator via the class weights. Hence, in the case of *agent*, the variation in weights of the priority classes has (virtually) no effect on the weighted average utility of the priority classes. It is observed from the figures that the confidence intervals for the weighted average utility are narrow and non-overlapping for all resource allocation methods (for all load scenarios). Hence, the above conclusions are drawn with sufficient statistical relevance.

10^{th} throughput percentile of applications in the background class

Figure 9.23 shows the 10^{th} throughput percentile of the background class applications for a fixed setting of load in the priority classes and different settings of the evaluation parameters for Phase 2. Figure 9.24 illustrates the same for Phase 3. In both the phases, common observations are made. It is observed that in the case of *agent*, the

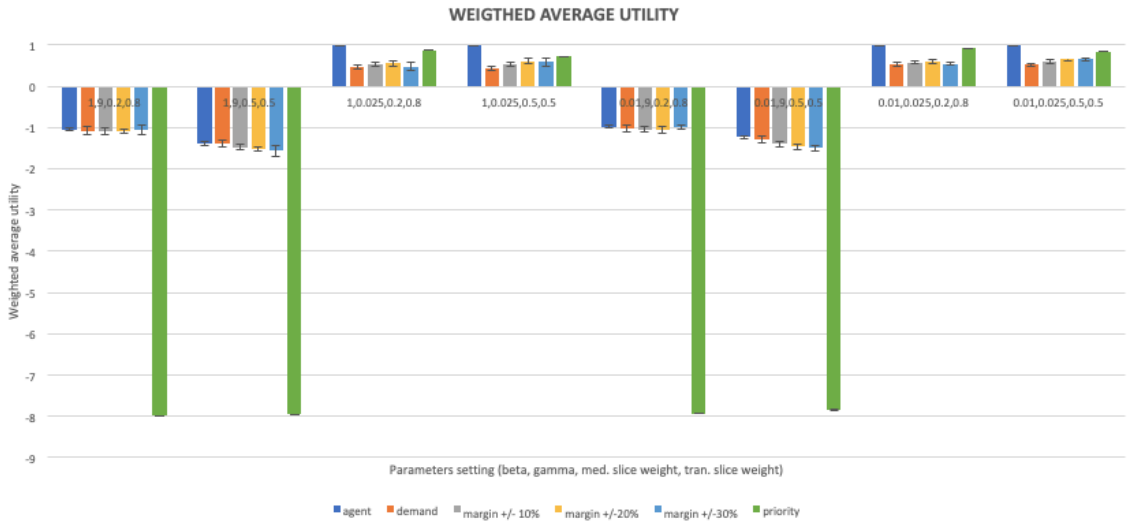

Figure 9.21: Weighted average utility of medical and transport classes in Phase 2 for different settings of evaluation parameters.

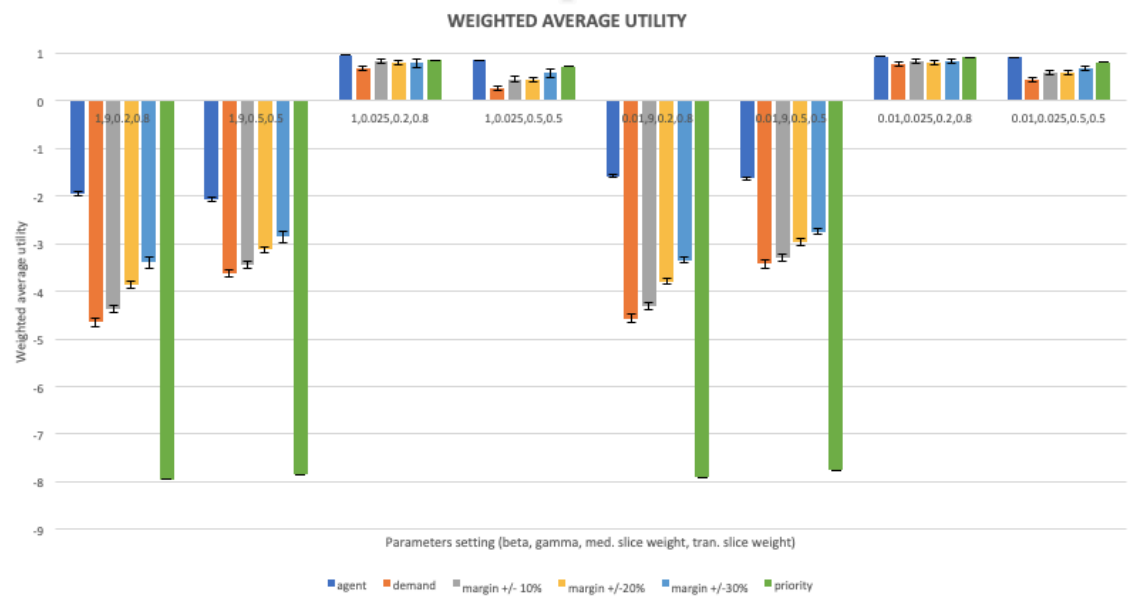

Figure 9.22: Weighted average utility of medical and transport classes in Phase 3 for different settings of evaluation parameters.

10^{th} throughput percentile of the background class applications is the highest for the case of $\gamma = 9$ and $\beta = 0.01$. This is because this evaluation setting corresponds to a higher cost of resource reservation and a lower penalty for performance degradation in the priority classes. Therefore, in this setting, *agent* reserves fewer resources for the priority classes in order to maximise the bottom-line metric, i.e the weighted average utility (which defines the reward for the CMAB agent) and hence leaves more resources for the background class.

Similarly, the 10^{th} throughput percentile of the background class applications is the minimum for $\gamma = 0.025$ and $\beta = 1$ because this setting for evaluation corresponds to a lower cost for resource reservation and a higher penalty for performance degradation in the priority classes. Therefore, in this setting, *agent* reserves a higher amount of resources for the priority classes and hence leaves fewer resources for the background class

It is observed for the considered load in the priority classes (medium, medium) in this experiment that on varying the slice weights with a fixed evaluation setting and load in all the classes, the 10^{th} throughput percentile of the background class applications does not change. This is because slice weights dictate how the resources are distributed among the priority classes. Hence, with a change in slice weights, there may not be a change in the amount of resources allocated to the background class and consequently, no change in the 10^{th} throughput percentile.

For all the other multi-slice resource allocation methods, no change in the 10^{th} throughput percentile of the background class applications is observed due to a change in the evaluation setting. This is because, the resource allocation decisions made by these methods only depend on the load in the priority classes (which is kept fixed in this experiment). On the other hand, as discussed above *agent* adapts to the variation in the evaluation parameters. Recall that the evaluation parameters define the business policies and objectives of slice owners and/or network operator. Hence, *agent* also considers the business policies of the slice owners and.or network operator when taking resource allocation decisions.

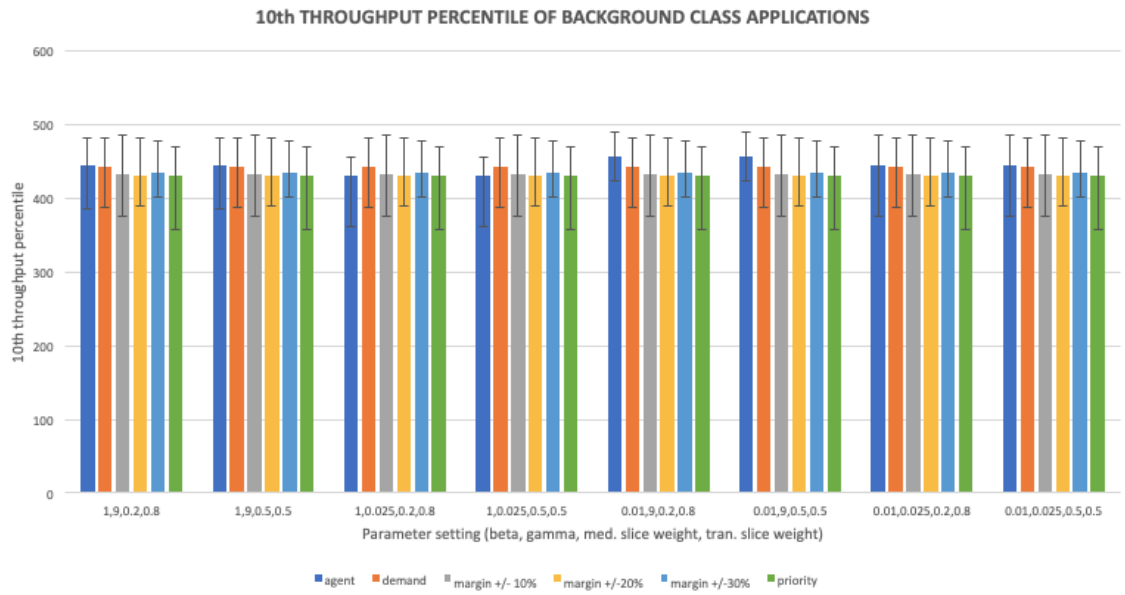

Figure 9.23: 10^{th} throughput percentile of background class applications in Phase 2 for different settings of evaluation parameters.

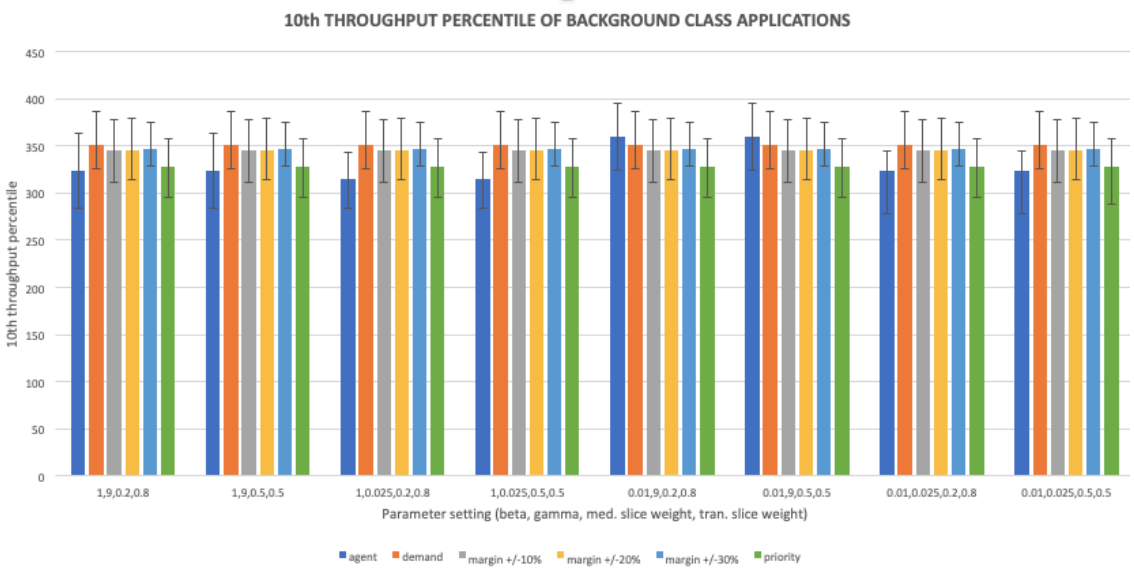

Figure 9.24: 10^{th} throughput percentile of background class applications in Phase 3 for different settings of evaluation parameters.

10

Conclusions and Future Work

In this chapter, we summarize the work done in this thesis and highlight the key findings. We further give recommendations for future research.

10.1. Conclusion

In this thesis we have addressed the problem of radio resource management in sliced radio access networks. More specifically, we have looked into the case of resource allocation among multiple priority slices. We have identified that the multi-slice resource allocation in RAN consists of two separate problem areas, namely prediction of resource demand of all the slices and efficient resource allocation based on the collective resource demand prediction. Hence, we have designed and proposed a framework for multi-slice resource allocation with two modules, namely the *resource demand prediction* module and the *resource allocation* module.

We have identified two different artificial intelligence techniques, namely supervised machine learning and contextual multi-armed bandit reinforcement learning as suitable candidates for solving the consecutive subproblems of multi-slice resource allocation as stated above. The problems of resource demand prediction and efficient resource allocation to achieve multi-slice resource management have been addressed separately in detail.

We have designed a novel model for predicting the resource demand of the different slices based on supervised machine learning. The proposed resource demand prediction model has been argued to work for the slices which have scarcity of information about their historical resource usage. Three different supervised ML algorithms, namely multiple linear regression, random forests regression and an artificial neural network have been studied for this purpose. Based on the prediction performance defined by the mean absolute relative error metric, the artificial neural network has been found to be the best among the three considered algorithms and hence has been used in the resource demand prediction module.

Supervised machine learning models, in order to make accurate predictions, must be trained on a training data set that is abundant, unbiased and well-representative of the scenarios where the model is expected to be used. In order to address this in the context of *resource demand prediction*, we have proposed a novel methodology to create a training data set for optimal machine learning particularly in scenarios where no or limited training data is available. We have proposed so-called *differentiated handling of network traffic* to 'enhance' the training data by making it more diverse and unbiased. The proposed methodology of *differentiated handling of network traffic* is not limited to RAN slicing and can be applied in other applications employing supervised learning AI techniques for resource management or dimensioning in mobile networks. It has been shown for the considered supervised ML techniques that the models learnt using the enhanced training data had better prediction performance as indicated by the mean absolute relative error (MARE) for the test data as opposed to the models learnt using the original training data.

The lowest MARE has been found to be equal to 0.546 in the case of ANN trained using the above-mentioned methodology of enhancing the training data. However, a MARE equal to 0.546 is still very high, indicting that the resource predictions are not very reliable. This further motivated the use of a second intelligent layer in the multi-slice resource allocation framework. For the design of the *resource allocation* module of the multi-slice resource allocation, we have employed another ML technique referred to as CMAB (reinforcement learning).

Based on the extensive scenario-based simulation analysis done in this thesis it has been shown that the proposed framework performed the best in terms of the bottom-line metric for the priority slices, namely the weighted average utility of the priority classes in all the considered scenarios as compared to both the other slice-oriented considered resource allocation methods and the priority scheduling scheme, which is a non-sliced resource allocation method.

The other slice-oriented resource allocation methods lacked the second layer of intelligence (CMAB) and hence were not able to compensate for the various sources of errors such as inaccurate slice resource demand prediction or changes in the policies of the network operator. On the other hand, the performance of priority scheduling as reflected by the bottom-line metric varied from very good to the worst when the cost of resource allocation was varied from a lower value to a higher value, since it reserved all the available resources for the highest priority class resulting in a very high cost and consequently a very low value of the bottom-line metric for the priority class. Hence, although priority scheduling is not affected by the inaccuracy of resource demand prediction, it also cannot adapt to the policies of the network operator and/or slice owners. In contrast, the proposed framework with two intelligent layers has been shown to adapt to the different load settings and different evaluation parameter settings to always give the highest performance.

Based on the simulation-based comparative analysis done in this thesis, it can be concluded that for shared resources such as the PRBs and in the case of multiple priority classes, efficient reservation of resources for *all* the classes is needed beforehand which

is the essence of network slicing in order to guarantee some level of performance in each irrespective of variation in the traffic load, traffic characteristics in the other classes.

AI techniques, specifically artificial neural networks and CMAB were found to be very promising in achieving efficient resource allocation among multiple priority slices.

10.2. Recommendation for further research

In this study, the problem of multi-slice resource allocation was shown to be efficiently solved by AI techniques. For the evaluation of the proposed methods, we made some simplifying modelling choices in relation to a.o. network layout, propagation aspects (e.g. neglecting the impact of multi-path fading) and technology modeling upon which improvements are possible to make the scenario analyses somewhat more realistic. A simplification made in the technology modelling was the use a common SCS and TDD frame structure across all the considered slices. However, one of the key features of RAN slicing is to be able to customise the resources (in terms of, e.g. SCS and TDD frame structure) of a slice based on, e.g. the QoS requirements and traffic characteristics of the traffic in the slice. Hence, it is recommended to evaluate the performance of the proposed method with different slices having different configurations of TDD frame structure and/or SCS.

It was found that the mean absolute relative error in the case of resource demand predictions was quite high even for the best performing model. Hence, there are still some improvements needed to deal with the high MARE. Moreover, it was found that the use CMAB along with resource demand prediction improved resource allocation performance as compared to the case when only resource demand prediction was used. Hence, it is also recommended to explore the application of only CMAB or reinforcement learning for the resource-allocation problem with necessary modification in the model.

In this study we carried out two sets of sensitivity analyses, viz. w.r.t. the load in the priority classes and w.r.t the different evaluation parameters. Another parameter to be considered is the time scale at which the resource allocation decisions are made. Smaller re-allocation intervals allow for more dynamic updates in the resource allocation with changing load and traffic characteristics of the users in a slice resulting in increased efficiency of resource allocation. On the other hand, longer re-allocation intervals are easier to control and handle, especially when resource customization is used. It is recommended to analyse the effect of the length of resource allocation interval on the efficiency of the proposed multi-slice resource allocation method

In this study we considered only two priority slices. With more priority slices and with similar priorities for each slice, the resource allocation problem becomes more challenging as it poses the challenge of resource scarcity. It is recommended to evaluate the performance of the proposed method with more priority slices and in the case when the total amount of the resource is not sufficient to serve all the slices.

Bibliography

[1] *3GPP Release 15 Overview*, 2020. URL <https://spectrum.ieee.org/telecom/wireless/3gpp-release-15-overview>.

[2] 3GPP. *Study on Channel Model for Frequencies from 0.5 to 100 GHz, TR 38.901 Release 16*, 2019.

[3] 3GPP. *Evolved Universal Terrestrial Radio Access (E-UTRA), Radio Frequency (RF) System Scenarios, TR 36.942 Release 16*, 2020.

[4] Y. Abiko, D. Mochizuki, T. Saito, D. Ikeda, T. Mizuno, and H. Mineno. Proposal of allocating radio resources to multiple slices in 5g using deep reinforcement learning. In *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, 2019.

[5] Y. Abiko, T. Saito, D. Ikeda, K. Ohta, T. Mizuno, and H. Mineno. Flexible resource block allocation to multiple slices for radio access network slicing using deep reinforcement learning. *IEEE Access*, 2020.

[6] S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. *30th International Conference on Machine Learning, ICML 2013*, 2012.

[7] A. Aijaz. Hap-slicer: A radio resource slicing framework for 5g networks with haptic communications. *IEEE Systems Journal*, 2018.

[8] P. Baudot, M. Tapia, D. Bennequin, and J. Goaillard. Topological information data analysis. *Entropy*, 2019.

[9] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez. Deepcog: Cognitive network management in sliced 5g networks with deep learning. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019.

[10] G. Burtini, J. Loeppky, and R. Lawrence. A survey of online experiment design with the stochastic multi-armed bandit. 2015.

[11] E. Grinshpun C. Gutterman. Ran resource usage prediction for a 5g slice broker. *Mobihoc '19*, 2019.

[12] P. Caballero, A. Banchs, G. de Veciana, and X. Costa-Pérez. Multi-tenant radio access network slicing: Statistical multiplexing of spatial loads. *IEEE/ACM Transactions on Networking*, 2017.

[13] O. Chapelle and L. Lihong. An empirical evaluation of thompson sampling. In *Advances in Neural Information Processing Systems 24*. 2011.

[14] W.W. Daniel. *Applied Nonparametric Statistics*. Duxbury advanced series in statistics and decision sciences. 1990.

[15] Ericsson. *Ericsson Mobility Calculator*. URL <https://www.ericsson.com/en/mobility-report/mobility-calculator?up=2&bp=2&v=0&c=2>.

[16] X. Foukas, M. Marina, and K. Kontovasilis. Orion: Ran slicing for a flexible and cost-effective multi-service mobile network architecture. 2017.

[17] R. Fraile, J.F. Monserrat, J. Gozalvez, and A. Cardona. Wireless systems mobile radio bi-dimensional large-scale fading modelling with site-to-site cross-correlation. *European Transactions on Telecommunications*, 2008.

[18] V. Francois-Lavet, P. Henderson, R. Islam, M.G. Bellemare, and J. Pineau. An introduction to deep reinforcement learning, 2018.

[19] P. Frenger, S. Parkvall, and E. Dahlman. Performance comparison of harq with chase combining and incremental redundancy for hsdpa. 2001.

[20] F. Gers and J. Schmidhuberand F. Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 2000.

[21] C. Gros, G. Kaczor, and D. Markovic. Neuropsychological constraints to human data production on a global scale. *The European Physical Journal B*, 2011.

[22] GSMA. *5G TDD Synchronisation Guidelines and Recommendations for the Coexistence of TDD Networks in the 3.5 GHz Range*, 2020. URL <https://www.gsma.com/spectrum/wp-content/uploads/2020/04/3.5-GHz-5G-TDD-Synchronisation.pdf>.

[23] F. Gunnarsson, M. N. Johansson, A. Furuskar, M. Lundevall, A. Simonsson, C. Tidestav, and M. Blomgren. Downtilted base station antennas - a simulation model proposal and impact on hspa and lte performance. In *2008 IEEE 68th Vehicular Technology Conference*, 2008.

[24] B. Han and H. D. Schotten. Machine learning for network slicing resource management: A comprehensive survey. *ArXiv*, 2020.

[25] H. Van Hasselt and M. Wiering. Reinforcement learning in continuous action spaces. 2007.

[26] J. He and W. Song. Appran: Application-oriented radio access network sharing in mobile networks. In *2015 IEEE International Conference on Communications (ICC)*, 2015.

[27] 5G HEART. *5G HEalth AquaculturRe and Transport Validation Trials. Use Case Description and Scenario Analysis.*, 2020.

[28] 5G HEART. *5G HEalth AquacultuRe and Transport Validation Trials. Initial Solution and Verification of Healthcare Use Case Trials*., 2020.

[29] J. Heaton. *Heaton Research, The Number of Hidden Layers*, 2017.

[30] P. Hernandez-Leal, B. Kartal, and M. Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 2019.

[31] Xcelerator IP. *Calculating Voice Bandwidth Requirements*, 2007. URL <http://www.vertical.com/media/support/an-xip07010_calculating-voice-bandwidth-requirements.pdf>.

[32] ITU-R. *Guidelines for Evaluation of Radio Interface Technologies for IMT-Advanced, Report ITU-R M.2135-1*,. 2009.

[33] A. Juliani. *Simple Reinforcement Learning with Tensorflow Part 1.5: Contextual Bandits*, 2016. URL <https://medium.com/emergent-future/simple-reinforcement-learning>.

[34] S. Kakade, S. Shalev-Shwartz, and A. Tewari. Efficient bandit algorithms for online multiclass prediction. 2008.

[35] N. Karampatziakis, S. Kochman, J. Huang, P. Mineiro, K. Osborne, and W. Chen. Lessons from contextual bandit learning in a customer support bot. *arXiv: Learning*, 2019.

[36] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan. Nvs: A substrate for virtualizing wireless resources in cellular networks. *IEEE/ACM Transactions on Networking*, 2012.

[37] A. Ksentini and N. Nikaein. Toward enforcing network slicing on ran: Flexibility and resources abstraction. *IEEE Communications Magazine*, 2017.

[38] Y. L. Lee, J. Loo, T. C. Chuah, and L. Wang. Dynamic network slicing for multitenant heterogeneous cloud radio access networks. *IEEE Transactions on Wireless Communications*, 2018.

[39] L. Li, W. Chu, J. Langford, and R. Schapire. A contextual-bandit approach to personalized news article recommendation. *Computing Research Repository*, 2010.

[40] Q. Li, M. Zhou, Y. Wu, S. Feng, and P. Zhang. Precise interference estimation for the uplink of lte heterogeneous networks. In *2014 IEEE Globecom Workshops (GC Wkshps)*, 2014.

[41] R. Li, Z. Zhao, X. Zhou, J. Palicot, and H. Zhang. The prediction analysis of cellular radio access network traffic: From entropy theory to networking practice. *IEEE Communications Magazine*, 2014.

[42] R. Li, Z. Zhao, Q. Sun, C. I, C. Yang, X. Chen, M. Zhao, and H. Zhang. Deep reinforcement learning for resource management in network slicing. *IEEE Access*, 2018.

[43] S. Mazumder, B. Liu, and S. Wang. Action permissibility in deep reinforcement learning and application to autonomous driving. 2018.

[44] missinglink.ai. *7 Types of Neural Network Activation Functions: How to Choose?* URL <https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/>.

[45] A. Mohammadi and F. Ghannouchi. *MIMO Wireless Communications*. 2012.

[46] J. F. Monserrat, R. Fraile, D. Calabuig, and N. Cardona. Complete shadowing modeling and its effect on system level performance evaluation. In *VTC Spring 2008 - IEEE Vehicular Technology Conference*, 2008.

[47] R. Müllner, C. F.Ball, K. Ivanov, J. Lienhart, and P. Hric. Performance comparison between open-loop and closed-loop uplink power control in utran lte networks. 2009.

[48] J. Navarro-Ortiz, P. Romero-Diaz, S. Sendra, P. Ameigeiras, J. J. Ramos-Munoz, and J. M. Lopez-Soler. A survey on 5g usage scenarios and traffic models. *IEEE Communications Surveys Tutorials*, 2020.

[49] I. Parvez, A. Rahmati, I. Guvenc, A. Sarwat, and H. Dai. A survey on low latency towards 5g: Ran, core network and caching solutions. *IEEE Communications Surveys Tutorials*, 2017.

[50] H. Peng, Q. Ye, and X. Shen. Spectrum management for multi-access edge computing in autonomous vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[51] Team SimPy Revision. *SimPy Overview*. URL <https://simpy.readthedocs.io/en/latest/index.html>.

[52] A. Robin and R. Feraudand D. Bouneffouf. A neural networks committee for the contextual bandit problem. 2014.

[53] A. C. Ruiz, C. Cavdar, and C. Beckman. Impact of base station antenna vertical beam width and tilt on the performance of lte networks. In *2016 IEEE International Symposium on Antennas and Propagation (APSURSI)*, 2016.

[54] K. Samdanis, X. Costa-Perez, and V. Sciancalepore. From network sharing to multi-tenancy: The 5g network slice broker. *IEEE Communications Magazine*, 2016.

[55] K. Samdanis, X. Foukas, E. Pateromichelakis, and A. Ksentini. *Slicing and Radio Resource Management*. 2019.

[56] V. Sciancalepore, X. Costa-Perez, and A. Banchs. Rl-nsb: Reinforcement learning-based 5g network slice broker. *IEEE/ACM Transactions on Networking*, 27(4): 1543–1557, 2019.

[57] X. Shen, J. Gao, W. Wu, K. Lyu, M. Li, W. Zhuang, X. Li, and J. Rao. Ai-assisted network-slicing based next-generation wireless networks. *IEEE Open Journal of Vehicular Technology*, 2020.

[58] R. Shrivastava, K. Samdanis, and A. Bakry. On policy based ran slicing for emerging 5g tdd networks. In *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018.

[59] R. Su, D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang, and Z. Zhu. Resource allocation for network slicing in 5g telecommunication networks: A survey of principles and models. *IEEE Network*, 2019.

[60] G. Sun, Z. T. Gebrekidan, G. O. Boateng, D. Ayepah-Mensah, and W. Jiang. Dynamic reservation and deep reinforcement learning based autonomous resource slicing for virtualized radio access networks. *IEEE Access*, 2019.

[61] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, 2018.

[62] R. M. Taylor. A broadband omnidirectional antenna. In *Proceedings of IEEE Antennas and Propagation Society International Symposium and URSI National Radio Science Meeting*, volume 2, pages 1294–1297 vol.2, 1994. doi: 10.1109/APS.1994.407852.

[63] G. Tseliou, K. Samdanis, F. Adelantado, X. C. Pérez, and C. Verikoukis. A capacity broker architecture and framework for multi-tenant support in lte-a networks. In *2016 IEEE International Conference on Communications (ICC)*, 2016.

[64] C. Ubeda Castellanos, D. L. Villa, C. Rosa, K. I. Pedersen, F. D. Calabrese, P. Michaelsen, and J. Michel. Performance of uplink fractional power control in utran lte. In *VTC Spring 2008 - IEEE Vehicular Technology Conference*, 2008.

[65] Real Wireless. *4G Capacity Gains V 1.4*, 2011. URL <https://www.ofcom.org.uk/__data/assets/pdf_file/0038/74999/4gcapacitygainsfinalreporta1.pdf>.

[66] X. Zhang, Y. Xu, H. Hu, Y. Liu, Z. Guo, and Y. Wang. Profiling skype video calls: Rate control and video quality. In *2012 Proceedings IEEE INFOCOM*, 2012.

[67] M. Yang, Q. Li, Z. Qin, and J. Ye. Hierarchical adaptive contextual bandits for resource constraint based recommendation. 2020.

[68] J. YE and Y. Zhang. Drag: Deep reinforcement learning based base station activation in heterogeneous networks. *IEEE Transactions on Mobile Computing*, 2019.

[69] Q. Ye, W. Zhuang, S. Zhang, A. Jin, X. Shen, and X. Li. Dynamic radio resource slicing for a two-tier heterogeneous wireless network. *IEEE Transactions on Vehicular Technology*, 2018.

[70] A. Zanette and E. Brunskill. Problem dependent reinforcement learning bounds which can identify bandit structure in mdps. *ArXiv*, 2018.

[71] C. Zhang, P. Patras, and H. Haddadi. Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys Tutorials*, 2019.