Machine Learning Approach to Slice Admission Control in 5G Wireless Network

BY

-MOURICE OTIENO OJIJO

A thesis submitted for the degree of
Doctor of Philosophy
in the Department of Electrical Engineering,
Faculty of Engineering and the Built Environment, University of Cape Town
JUNE 2024

Supervisor: Dr.Daniel Ramotsoela

As the candidate's supervisor, I have approved this dissertation for submission.

Supervisor: Daniel Rammotsoela

Sign: Signed by candidate

Date: _____

Declaration

I hereby declare that: (1) the above thesis is my own unaided work, both in conception and execution, and that apart from the normal guidance of my supervisor, I have received no assistance apart from that stated below; (2) except as stated below, neither the substance nor any part of the thesis, has been submitted in the past, or is being, or is to be submitted for a degree in the University, or any other University.

I am now presenting the thesis for examination for the Degree of PhD in Electrical Engineering. I also grant the University free license to reproduce the above thesis in whole or in part, for the purpose of research.

-MOURICE OTIENO OJIJO _____

NAME

_____

DATE

Dedication

To

Emily

My best friend and wife, who has endured some of the toughest times alongside me: first, during my extended absence in Cape Town for almost a year; second, while caring for our small baby alone during my absence; and third, through years of witnessing sacrifices as I worked late into the night without complaint. She has always provided me with wonderful support and encouragement when I felt like giving up. This thesis is also dedicated to my children, who may not have understood my constant travels and absence, but never complained. May God bless them abundantly.

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to the Almighty God for guiding me this far. His unwavering support has been the cornerstone of my journey, and without His blessings, I would not have reached this milestone.

I extend my sincere appreciation to my supervisor, Dr. Daniel Ramotosoela, for graciously agreeing to accompany me through the final phase of my PhD program. His guidance, expertise, and encouragement have been invaluable to me.

I also wish to acknowledge my former supervisor, Mr. Neco Ventura (May his soul rest in peace), who stepped in during a challenging period of stagnation and provided unwavering support. He mentored me through some of my lowest moments as a PhD student at UCT, and his guidance during the preparation for my proposal presentation was instrumental in shaping my research journey. I will always be grateful for his support and mentorship.

Additionally, I am indebted to Prof. Fred Nicols for his willingness to shoulder my burdens during times when continuing seemed nearly impossible. He graciously agreed to serve as my administrative supervisor until Dr. Daniel took over, and his support was instrumental in navigating administrative challenges.

I cannot overlook the pivotal role played by Prof. Olabisi Falowo. His support was instrumental in securing my admission to the PhD program at UCT, and his guidance in navigating my first quality publication kick started my research journey. I am truly grateful for his contributions.

I also want to extend my sincere gratitude to my brother Joseph Adiang' he has been praying for me ever since. I can not forget his financial support.

Finally, I extend my thanks to the CoE CRG for their support in attending conferences, where I had the opportunity to share and present my work. Their support and encouragement have been invaluable throughout my academic journey.

Contents

Abstract

The advent of fifth-generation (5G) wireless communication has introduced a paradigm shift in how cellular networks operate and how network resources are allocated. As data networks become increasingly dynamic and complex, resource allocation can not be uniformly applied to all network users; rather, it should be implemented through a user-centric construction of virtual functions , commonly known as network slices. However, network slicing can be complemented by a slice admission control (SAC) process that permits only those slice requests that satisfy quality-of-service (QoS) requirements. Furthermore, automated and intelligent networking approaches can be employed to enhance SAC and optimize key objectives such as revenue, fairness, scheduling efficiency, and network resilience. This thesis investigates SAC in a resource-constrained, inter-domain 5G network, with a focus on improving the utility, fairness, scheduling, and network resilience of infrastructure providers (InPs).

Firstly, this thesis develops an inter-domain resource allocation framework that elucidates the inherent non-linearity of its formulation. The resulting formulation is presented as a mixed-integer non-linear programming (MINLP) problem and is proven to be NP-hard. In this context, the study conducts a comprehensive analytical comparison of various feasible solution approaches to address this problem, including branch and bound (BnB), successive convex approximation (SCA), the alternating direction method of multipliers (ADMM), heuristic methods, genetic algorithms (GA), and machine learning (ML) techniques.

Secondly, this thesis introduces a multi-server, multi-queue resource scheduling framework aimed at accurately predicting costs and resource availability over a 24-hour period. By leveraging virtualized inter-domain resource blocks, the transient probabilities of queues are

derived. The investigation finds that the predictions of the deep Q-learning (DQL) agent generally fall within an acceptable average variation of 6%.

Thirdly, this study investigates the impact of integrating slice admission control (SAC) with an auction-based game, which is double-ended bidding mechanism that allow both network resource buyers and sellers to place preference so that resources can only be allocated to the most deserving bidder, this aim to maximize overall utility and enhance fairness. The thesis introduces inter-domain resource models for n-class zoned 5G slices. By increasing the probability of admission during periods of resource availability, the findings demonstrate that the reinforcement learning agent improves long-term utility and fairness by 68.2%.

Finally, this thesis proposes a novel sequential twin-actor critic (STAC) method that optimizes a two-stage action process—namely, slice admission control (SAC) and resilience maintenance—by adjusting network resource blocks (RBs) to maximize overall utility. Additionally, the probability of slice acceptance is evaluated and compared with that of similar schemes. Given the anticipated density of up to one million devices per square kilometer, this study provides supplementary insights into SAC in an anomalous multi-node environment by analyzing data from admitted slice requests to detect any irregular patterns. The results demonstrate that the adopted reinforcement learning (RL) scheme outperforms the compared approaches.

List of Abbreviations

3GPP	Third Generation Partnership Project
4G	Fourth Generation
5G	Fifth Generation
6G	Sixth Generation
ADDD	Alternating Direction Dual Decomposing
ADMM	Alternating Direction Method of Multipliers
BBU	Baseband Unit
BnB	Branch and Bound
BS	Base Station
CA	Carrier Aggregation
CAC	Conventional Actor Critic
CAPEX	Capital Expenditure
CN	Cloud Network
CNN	Convolution Neural Network
C-Plane	Control Plane
CPU	Central Processing Unit
C-RAN	Cloud RAN
D2D	Device-to-Device
DDPG	Deep Deterministic Policy Gradient
DL	Downlink
DQL	Deep Q-Learning
DS-DDQL	Dual-Stage Double Deep Q-Learning
DU	Distribution Unit
DVP	Delay Violation Probability
E2E	End-to-End
EMB	Enhanced Mobile Broadband
eNB	evolve Node-B
EPC	Evolved Packet Core

FCE	Function Chain Embedding
FDD	Frequency Division Duplexing
GA	Genetic Algorithm
GA-AMD	Genetic Algorithm Based Adaptive Mating Distance
Gbps	Giga bits per second
GDA	Greedy Based Admission
GHz	Giga Hertz
gNB	gNodeB
IoT	Internet of Things
InP	Infrastructure network Provider
ITU	International Telecommunication Union
J	Joule
JRRA	Joint Radio Resource Allocation
KPCA	Kernel Principal Component Analysis
KPI	Key Performance Indicator
LBA	Learning Base Admission
LP	Linear Programming
LTE	Long Term Evolution
M2M	Machine-to-Machine
MAC	Medium Access Control
Max	Maximum
MBS	Macro Base Station
MDP	Markov Decision Process
MHz	Mega Hertz
Min	Minimum
MINLP	Mixed Integer Non-Linear Programming
MILP	Mixed Integer Linear Programming
MME	Mobility Management Entity
MSE	Mean Square Error
MNO	Mobile Network Operator
NE	Nash Equilibrium
NFV	Network Function Virtualization
NP	Non-deterministic Polynomial time
NR	New Radio

ONU	Optical Network Unit
OPEX	Operating Expenditure
ORANS	Online Resource Allocation for Network Slicing
PCA	Principal Component Analysis
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RB	Resource Blocks
ReLU	Rectified Linear Unit
RBA	Random Base Auctioning
RRB	Radio Resource Block
RRH	Remote Radio Head
RL	Reinforcement Learning
RRA	Radio Resource Allocation
RRM	Radio Resource Management
SAC	Slice Admission Control
SCA	Successive Convex Approximation
SDE	Stochastic Differential Equation
SDN	Software Defined Networking
SE	Spectrum Efficiency
SINR	Signal-to-Interference-plus-Noise Ratio
SLA	Service Level Agreement
SMDP	Semi-Markov Decision Process
SNR	Signal-to-Noise Ratio
ST	Slice Tenant
STAC	Sequential Twin Actor-Critic
SU	Secondary User
TDD	Time Division Duplexing
TN	Transport Network
UE	User Equipment
UL	Uplink
mMTC	Massive Machine Type Communication
UAV	Unmanned Aerial Vehicles
U-Plane	User Plane
URLLC	Ultra-Reliable Low Latency
VCG	Vickrey-Clarke-Groves
VNF	Virtual Network Functions
VNO	Virtual Network Operator
WEN	Wireless Edge Networks

List of Tables

List of Figures

Chapter 1

Introduction

1.1 General Introduction

Wireless communication has taken a new paradigm shift in the way network users are assigned network resources, deviating from the monolithic nature of the fourth generation (4G) wireless system where "one fits all" resource assignment was employed. The new mode of communication which in-cooperating the 5th generation (5G), and the 6th generation (6G) is characterized by a customized resource allocation strategy. This strategy aims to match unique network requirement of each user with tailored network resource assignment. The key enabler to non-monolithic network resource allocation is known as network slicing[1]. Network slicing allows the creation of logical network instances known slices that can be assigned to specific user requirement and be deployed on a common physical infrastructure to meet the slice instance quality of service(QoS) [2, 3].

Compared to its predecessor, 5G incorporates key network capabilities, such as Network Function Virtualization (NFV) [4] and its corresponding virtual network functions (VNFs), which are implemented using Software Defined Networking (SDN) technology. This allows for the orchestration of VNFs on commodity servers, leading to increased resource flexibility and isolation, and enabling highly customizable network slices. The integration of NFV and SDN in 5G represents a significant advancement in network architecture and functionality.

Network users can request network resources based on specific demands. However, due to resource scarcity, slice providers have the ability to scrutinize these requests and only admit slices that align with specific provider objectives.

Slice admission control algorithms can provide effective resource allocation and reliable function chaining by setting specific objectives based on analyzing user requests and network status. These algorithms are designed to efficiently manage network resources while ensuring that the required QoS is maintained. By employing slice admission control algorithms, network operators can optimize their resource utilization and provide differentiated services to meet the diverse needs of their customers.

Network slices can be constructed partially or end to end and then launched as a service function. This process is known as service orchestration. The orchetrator role is also to supervise and manage such network slices [5]. Further, the success of 5G communication has leveraged the ability to decouple the user data plane (U-plane) and the control plane (C-plane) which was first formulated in Phantom Cell Concept (PCC) [6] to improve scalability, security, flexibility and reliability. Indeed, the ability to instantiate VNFs on commodity servers have enabled improved control allowing the underlying physical network to perform fast data forwarding.

In the context of 5G, slice admission control (SAC) plays a crucial role in efficiently managing network resources. SAC involves several key players, including the slice tenant (ST) or virtual network operator (VNO), who is a third-party provider that obtains a bundle of virtual network functions (VNFs) from the infrastructure provider (InP). The InP, on the other hand, owns the physical substrate that is abstracted from the VNO. This abstraction

allows the VNO to perform slice scheduling and assign cost unit to resource usage per user while an InP can assign cost per VNO. The interaction between a VNO and a user, dictates that a user can be admitted to a slice in a process known as intra-slice admission. Meanwhile, the InP can admit a VNO into a resource pool, which is referred to as inter-slice admission. These interactions and admission processes are essential for enabling network virtualization and ensuring efficient utilization of resources in the 5G ecosystem[7].

The unclustered data from network users represents tailored user resource demands [8]. To standardize the provisioning of network slices, the International Telecommunication Union (ITU) has defined categories for slices. These classifications include massive machine type communication (mMTC), which is designed to support bursty and lightweight payloads, making it suitable for applications such as the Internet of Things (IoT) that require massive connectivity, low latency, and low power consumption.

Another category defined by the ITU is enhanced mobile broadband (eMBB), which is geared towards data-intensive applications such as high-density (HD) video streaming and virtual reality (VR). Lastly, the ITU has also defined an ultra-reliable low latency (URLLC) slice category, which is ideal for delay-sensitive services such as autonomous vehicles, remote healthcare, and unmanned aerial vehicles (UAVs). These standard slice categories provide a framework for designing and provisioning network slices that serve specific application requirements in the 5G environment[9].

As part of the slice admission control process, the InP is responsible for defining admission control objectives. Efficient optimization of these objectives can lead to improved network fairness, resilience, and profitability. Additionally, it enables the investigation of various

relevant questions related to network reach and performance. For instance, an InP may define a slice with the objective of optimizing revenue by selectively admitting only requests that have short-term network occupancy and stringent latency constraints. This strategic admission control objective allows the InP to make informed decisions about which slice requests to admit, based on their potential revenue generation and the network's capacity and performance. By carefully setting admission control objectives, InPs can align slice provisioning with business goals and optimize resource allocation, leading to more effective and profitable network operations [10, 11]. Other slice admission control objectives include QoS maintenance, admission control fairness, inter and intra-slice admission congestion control [12–14].

The flexibility of network slicing and the evaluation of admission control objectives are still evolving, and there is room for improvement. This is primarily due to the diverse nature of user requests and behavior, as well as the dynamic nature of 5G networks. Therefore, generally speaking, the objective of this research is to investigate and understand the ever-changing user demands and their impact on network usage in the context of slice admission control. By studying and analyzing user behavior and demands, this research aims to identify ways to improve the efficiency and specific effectiveness of slice admission control in 5G networks, ultimately enhancing the overall performance and operability of the network.

1.2 Research Motivation

Slice admission control (SAC) methodology in 5G communication networks interrogates user and network parameters in order to uncover abstracted patterns that can be utilized to

optimize network performance and maximize market returns for operators[15]. By analyzing user requests and network status in real-time, SAC can provide a comprehensive view of network conditions. These conditions can be employed to dynamically adapt the network resources, improve network fairness, resilience, and profitability, and respond proactively to changing user demands[1].

The inherent dynamics and unpredictability of networks and user behavior present significant challenges in adopting a standardized approach for optimizing network resource allocation and slice admission control. Traditional methods relying solely on historical data may not suffice in making informed decisions about the current network status, particularly in the context of rapidly evolving 5G and 6G networks. Therefore, it becomes crucial to implement an intelligent approach that leverages temporal and non-static data to optimize admission control.

To address this issue, a more generalized approximation is required, enabling efficient adaptation to rapid changes in user demands and network status. This approach should allow real-time adjustments and informed decision-making regarding resource allocation and slice admission control. By incorporating temporal and dynamic data into the optimization process, a more intelligent and adaptable framework can be developed to optimize network resource allocation and effectively handle evolving user demands and network conditions.

The practical application and adaptability of reinforcement learning (RL) techniques offer a powerful tool for effectively interacting with temporal network environments and making decisions that closely mimic human responses more rapidly [16]. RL can be leveraged to optimize admission control in the context of network slicing, addressing the challenges posed

by granular and unpredictable user and network data. Additionally, the exploration of multidimensional data for admission control optimization is an under-explored area. By utilizing RL, it becomes possible to streamline the process by reducing dimensionality and optimizing efforts towards specific admission control objectives, thereby providing a more efficient and compact approach. The rate of admission, which represents the number of admitted slices divided by the total number of slice requests, serves as a crucial performance indicator in slice admission control. However, this rate alone does not provide significant value unless it is utilized to optimize both user and network objectives. Therefore, various challenges persist in the field, including admission fairness, efficient scheduling, network resilience maintenance, and profit optimization, which remain open problems requiring further exploration and solution. These challenges necessitate the development of novel approaches to effectively address them and enhance the overall performance and effectiveness of slice admission control.

1.3 Problem Statement

Investigations into slice admission control optimization with RL have predominantly focused on static network operations, often targeting single QoS metrics and leveraging single-dimensional data to enhance profitability. However, there remains a gap in the literature regarding a comprehensive study on slice admission control that incorporates multidimensional data and accounts for hidden user and network patterns. This study aims to address these limitations by examining slice patterns to enhance considerations of fairness, efficient resource scheduling, resilience, and slice profitability.

1.4 Research Aims and Objectives

This study aims to tackle the problem of slice admission control with a focus on revenue optimization, taking into account both user and network objectives. Specifically, we address the challenges of maintaining slice admission fairness, efficient slice scheduling, network resilience maintenance, and profitability within an end-to-end 5G ecosystem.

In contrast to existing literature, which often restricts investigation to slice admission using single-dimensional data solely for QoS maintenance, inter-slice and intra-slice congestion control, this thesis takes a broader perspective and considers multiple objectives to optimize revenue generation in a comprehensive manner. By exploring these additional dimensions, we aim to provide a more holistic and effective approach to slice admission control in 5G networks.

The approach chosen in this investigation relies on the formulation of non-standard network slicing but still aligning to the known eMBB, uRLLC, and mMTC slices while considering spatio-temporal user and network parameters to achieve the following objectives.

- To perform comprehensive complexity analysis for slice admission control optimization algorithms as a motivation for machine learning approach to slice admission control.
- To propose and formulate analytical expression for slice scheduling and analyze the performance considering resource and cost prediction over the selected period with Deep Q-Learning scheduler for multi-queue multi-server.

- To formulate the expression for slice admission control to implement fairness through slice auctioning and evaluate the performance through Q-learning.

- To develop and analyze slice admission control for resilient 5G network which improves network utility using a novel approach known as sequential twin-actor critic (STAC) in a multidimensional state space while efficiently adjusting throughput, computation and memory resources.

1.5 Research Questions

In this section we outline the corresponding research questions targeted to be answered through out this Thesis.

- What are the optimal methodologies to address the challenges posed by the intricate dynamics of network resource allocation and admission control, particularly in environments characterized by highly dynamic and nonlinear fluctuations in network resources?

- What factors contribute to the suitability of RL as an effective approach for addressing slice admission control alongside broader objectives such as resource scheduling, equitable admission control, and the preservation of network resilience?

- What is the significance of resource scheduling within the context of slice admission control, and how can this process be executed with precision while simultaneously accurately estimating the anticipated slice cost?

- How can slice admission control be performed fairly while improving an InP's revenue?
- How can network resilience be maintained in a highly dynamic 5G network while leveraging slice admission control for revenue optimization?
- How can slice admission control enhance anomaly detection where there are massive connectivity of 5G network nodes?

1.6 Research Contributions

This study addresses slice admission and associated objectives in 5G wireless network. By extensive study of existing literature, it is determined that, slice admission control yield immense data that can be employed to improve network performance. This study stands on this foundation to employ slice admission control as a method of improving slice fairness enhancement, enabling efficient scheduling, resilience improvement and improved anomaly detection. The contributions are detailed as follows.

- The study outlines a comprehensive complexity analysis into classical resource allocation scheme and employs the analysis as a starting point into the motivation to employing machine learning in solving slice admission control problems. Successive Convex Approximation (SCA), Alternating Direction Method of Multipliers (ADMM), Branch and Bound(BB) and Heuristic approaches are studied and analyzed.
- This thesis also presents efficient slice scheduling as scheme of improving revenue by adopting stochastic differential equations (SDE) to formulate a solution for multi-queue multi-server approach in resource allocation

- The study further presents enhanced fairness in resource allocation. The problem is addressed by adopting an auction game and proving incentive compatibility through Nash Equilibrium. This approach is then solved using reinforcement learning.

- Further contributions of this study involve the formulation of resilience-aware strategy within a multi-dimensional state space leveraging slice admission control. A novel Sequential Twin Actor-Critic (STAC) model is developed to tackle this problem.

1.7 Thesis Outline

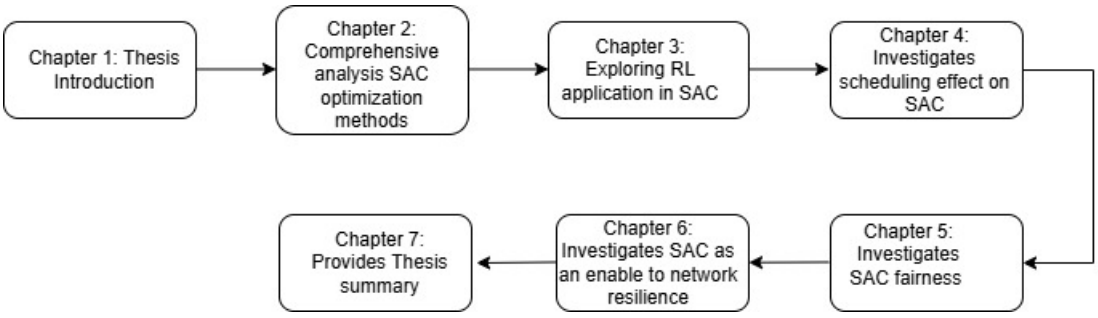

Figure 1.1: Summary diagram for Thesis outline

This thesis has seven chapters summarized in Figure 1.1, the rest of the chapters are organized as follows.

Chapter 2 provides a comprehensive complexity analysis for slice admission control optimization schemes for problems formulated as mixed-integer non-linear programming (MINLP) as a motivation for machine learning approach to slice admission control. This analysis have been published in journal paper [J-1]

Chapter 3 explores the reinforcement learning (RL) approach to slice admission control, examining the suitability of each approach in achieving the intended goal of market optimization. This chapter constitutes a segment of the research published in Journal [J-2].

Chapter 4 investigates the effect of slice scheduling to market optimization, aligning to deep Q-learning ability to deal with continuous data presenting evaluation of server and queue transients and resource predictions. The work in this chapter is published in conference paper [C-1]

Chapter 5 delves into the impact of SAC to fair resource allocation through the utilization of an auctioning game. It outlines the anticipated enhancements in market performance compared to random and greedy approaches. The results in this chapter are partly published in conference [C-2]

Chapter 6 investigates slice admission control as an enabler to network resilience by leveraging a novel approach known as sequential twin actor-critic (STAC) a subset of deep deterministic policy gradient method. The results in this chapter are published partly in [C-3] and [J-2]

Chapter 7 concludes the investigation and provides a summary of the thesis with highlights in results and contribution. Suggested future work are also highlighted in this chapter.

The appendices provides extra investigations not included in the main chapters alongside extended results puplished in [C-4].

1.8 Publications

[J-1] Ojijo, Mourice Otieno and Ramotsoela, Daniel and Oginga, Ruth Anyango, Slice Admission Control in 5g Wireless Communication with Multi-Dimensional State Space and Distributed Action Space: A Sequential Twin Actor-Critic Approach. Available at SSRN: https://ssrn.com/abstract=4696985 or http://dx.doi.org/10.2139/ssrn.4696985

[J-2] M. O. Ojijo and O. E. Falowo, "A Survey on Slice Admission Control Strategies and Optimization Schemes in 5G Network" IEEE Access, vol. 8, pp. 14 977˘14 990, 2020

[C-1] Mourice Otieno Ojijo , Neco Ventura; "Queuing based multi -InP resource scheduling for slice admission control with Reinforcement Learning"Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2022
https://www.satnac.org.za/proceedings ISBN : 978-0-6397-2773-8:8–13, 2022

[C-2] Mourice Ojijo and Dismas Ombuya. "Slice admission control by resource auction game in 5g network with reinforcement learning". Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2021 -https://www.satnac.org.za/proceedings, 2021ISBN : 978-0-6397-2773-8:8-13, 2021

[C-3] M. O. Ojijo, N. Ventura and D. N. Ondwari, "Deep Reinforcement Learning Approach to Slice Admission Control for Resilient 5G Wireless Network with Multidimensional

State Space," 2023 IEEE AFRICON, Nairobi, Kenya, 2023, pp. 01-06, doi: 10.1109/AFRICON55910.2023.10293471.

[C-4] Mourice Otieno Ojijo , Neco Ventura; "Reinforcement Learning Approach to Slice Admission Control for Anomaly Detection in Massive Machine Type Communication" Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2023

https://www.satnac.org.za/proceedings ISBN : 978-0-6397-2773-8:8–13, 2023

Chapter 2

Background

This chapter focuses on the background study of slice admission control optimization techniques, their limitation and complexity analysis. Brief introduction to reinforcement learning is provide in section 2.1.2.11. Finally , Section 2.1.3 concludes the chapter.

2.1 5G Network Slicing and Admission Control

The field of information and communication technology is currently undergoing a significant technological revolution, driven by the growing demands for broadband Internet, massive connectivity, and the Internet of Things (IoT). As these demands continue to increase at a rapid pace, there is need to evolve towards the next level of mature technology.

This evolution involves the development of intelligent networks that can deliver tailored services to meet specific user demands in a faster and more efficient manner[17]. The adoption of network slicing enables the fulfillment of diverse use and application requirements by facilitating the delivery of tailored network slices. In the context of 5G networks, network slicing allows for the partitioning of key subsystems, namely the Radio Access Network (RAN), the Transport Network (TN), and the Cloud Network (CN). This capability is referred to as end-to-end (E2E) slicing.

E2E slicing provides the ability to create virtualized and independent network instances that are customized to meet the specific needs of different use cases and applications. Each network slice is designed to provide dedicated resources and functionalities, including RAN,

TN, and CN components. With E2E slicing, the RAN slice focuses on managing and optimizing radio resources to meet the specific requirements of different applications and services. The TN slice ensures efficient and reliable transport of data between network elements, while the CN slice provides cloud-based resources and services for processing and storage. Figure 2.1 illustrates an E2E network slicing scenario.

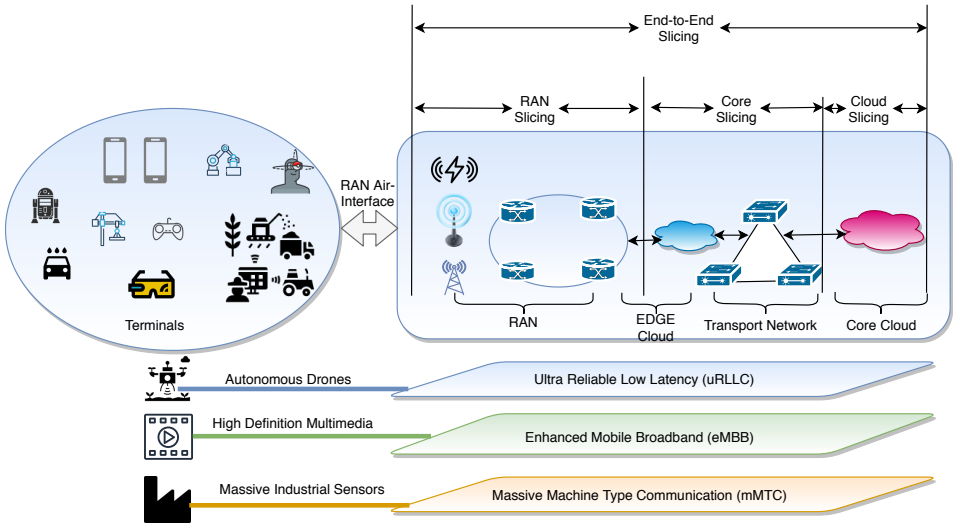

Figure 2.1: E2E 5G network slicing

In practice, the physical substrate of the 5G ecosystem consists of three main components: the access network, the transport network, and the device network.

The access network serves as the entry point to the 5G ecosystem, where radio resources are allocated to establish connectivity between a User Equipment (UE) and the network. This is achieved through the air interface, utilizing a radio resource management system. Within the access network, Radio Resource Blocks (RRBs) play a crucial role by providing defined radio bands that can be measured and allocated for orchestrating network slices. It is essential to optimize the allocation of these RRBs to ensure efficient communication within the network.

The transport network acts as the interconnection between data centers, facilitating the most optimal route selection for carrying massive data streams. This network is responsible for ensuring reliable and efficient data transfer while considering the constraints of resource availability. Optimizing the transport network enables the seamless flow of data between different network elements, improving overall system performance.

The device network encompasses User Equipment (UE) devices that are 5G-enabled. These devices can establish connections with each other using a Device-to-Device (D2D) criterion. This capability allows for direct communication between devices, bypassing the need for routing through the network infrastructure. D2D communication can enhance the efficiency and speed of data exchange between devices in proximity, enabling various applications and services.

The deployment of E2E resources require optimum resource selection to achieve maximization of revenue[11]. The resource units in each domain can be defined as an integer or non-integer quantity thereby presenting a mixed integer optimization problem. The challenge here involves finding the optimal resource allocation in all the domains considering specific requirements and constraints of the network slice. If the optimization problem can be defined as a mixed integer nonlinear problem, it is only feasible to identify an appropriate mathematical solution to maximize the objective.

2.1.1 General slicing model as a mixed-integer non-linear programming problem (MINLP).

Firstly, let us consider the following optimization problem.

$$\max f(x_i, x_j.x_k) \tag{2.1}$$

for

$$x_i, x_k, x_j \in \mathbf{x} \text{ for } i \neq j \neq k \tag{2.2}$$

Where $\mathbf{x}$ is the set of all possible decision variables required to optimize $f(x_i, x_j, x_k)$, i, j, k are unique subset of variables. The function $f(x_i, x_j, x_k)$, has coefficients $\beta_i, \beta_j, \beta_k$ summing to 1 representing the trade off among resource allocation balancing and objective maximization. The variables $x_i \subseteq \mathbb{W}_1^{n_1}, x_j \subseteq \mathbb{W}_2^{n_2}$ and $x_k \subseteq \mathbb{W}_m^{n_3}$ are distinct elements which represent scalable quantities of network resources applicable in a slice construction and are algebraic composing sums and products of multivariate functions[18]. The entire space encompases the union of all variables given by $\mathbb{W}_1^{n_1} \cup \mathbb{W}_2^{n_2} \cup \mathbb{W}_m^{n_3}$. The space containing the decision variables is defined as x_i as the subset of continuous decision variables, x_j contains the hard constrained integer variables while x_k is the subset comprising the binary variables. The optimality of the objective function in 2.1 is dependent on a combination of constrained

linear and non-linear functions, most of which are given by:

$$\begin{aligned} f(x_1, x_2, ..x_i) &\leq b_1 \\ f(x_1, x_2, ..x_j) &\leq b_2 \\ &. \\ &. \\ &. \\ f(x_1, x_2, ..x_k) &\leq b_m \\ \mathbf{x} &\in \mathbb{W}_1^{n_1} \times \mathbb{W}_2^{n_2} \times \mathbb{W}_m^{n_3} \end{aligned} \tag{2.3}$$

Notably the variable x is said to belong to a multidimensional state space given by $\mathbb{W}_1^{n_1} \times \mathbb{W}_2^{n_2} \times \mathbb{W}_3^{n_3}$. The function $f(x_i, x_j, x_k)$ can be rewritten in a more restricted formed as[19].

$$\max f(x_i, x_j, x_k) \tag{2.4}$$

s.t

$$\mathbf{Ax} \leq \mathbf{b} \tag{2.5}$$

where $\mathbf{A}$ and $\mathbf{b}$ form a rational matrix of $m \times n$ dimension and $\mathbf{x}$ is a n dimension rational vector respectively.

The constraint $\mathbf{Ax} \leq \mathbf{b}$ is fundamental because it defines the feasible region for the integer programming problem. It restricts the candidate solutions to those integer vectors that

satisfy the linear equations, thereby ensuring that the set of feasible solutions have well-defined structure. This structure allows us to take the convex hull, $x \in \mathbb{W}^n : \mathbf{Ax} \leq \mathbf{b}$, which is crucial for developing and solving the optimization algorithms. The polyhedral nature of this set enables the use of edge-directions and other combinatorial properties that underpin the strongly polynomial oracle-time.

Moreover, enforcing $\mathbf{Ax} \leq \mathbf{b}$ ensures that the algorithm respects the problem's inherent linear constraints, which often represent balance or conservation laws in practical applications like resource allocation or network flow.

The function can still be considered as NP-hard, this makes it much harder and complex to solve. Moreover, unless the complexity of the problem of class P equals to NP which is still unlikely, we can not estimate an approximate polynomial time to solve this problem. This implies that finding an efficient computational algorithm to solve this problem is highly challenging. Notably, if the dimension of the continuous polynomial is fixed to just n_1 then it can be solved in polynomial time, however, the complexity of solving this problem increases to hard immediately another integer is added[20].

The complexity of solving $f(x_i, x_j, x_k)$ requires the maximizing of polynomial of degree $n_1 + n_2 + n_3$ covering a lattice points of a concave polygon[21][22]. There cannot exist a general algorithm that can solve this problem even when the number of integer variables are reduced to a small number as the problem become worse than NP-hardness[23][24].

More precisely, a slice objective is dependent on actual network constraints in the RAN enabled by cloud RAN (C-RAN), while TN, and CN complete the E2E architecture. Typically to slice the C-RAN, the main slicable resources such as remote radio heads

(RRH), fronthaul links and base band units (BBUs) are considered. On other hand, the TN optimal path selection is a more complex procedure and virtualizing it means traversing VNFs through service function chain (SFC) in a netwrok function virtualization environment in order to achieve for optimal function placement[25]. The CN slicing is more elastic and involves resources that can be modeled by relaxable constraints in a mixed integer problem. Data centers in the CN can provide cache, central processing units (CPU) and network resources to maintain a slice .

The elasticity or inelasticity of a network resource matches the set of hard integer and soft integer variables in a MINLP.

2.1.2 Specific slicing model as a mixed-integer non-linear programming and complexity analysis of optimization approaches

The slicing model for E2E network resource allocation is based on a non liner estimation given by;

$$\mathcal{P}(x) = f_1(x_1) + f_2(x_2) + f_3(x_3) \tag{2.6}$$

where $\mathcal{P}$ is the aggregated revenue estimation, $f_1(x_1)$ and $f_2(x_2)$ represent non-linear functions of the objective function and $f_3(x_3)$ denotes the linear function. Each component of the objective function represents resource revenue model in each domain, in this case there are three domains namely C-RAN, TN and CN. These functions represent non-linearity in

resource allocation optimization but do not translate into proportional changes in revenue. Further, they represents the system variables x_1 and x_2 represent continuous resources in C-RAN and TN which are generally hard to slice while x_3 is a set of discrete cloud resources represent the units allocated to complete a slice setup. The aim is to maximize $\mathcal{P}(x)$ under constrained environment as:

$$\max[c.\mathcal{P}(x)] \tag{2.7}$$

s.t

$$\begin{aligned} f_1(x_1) &\leq \eta_1, \\ f_2(x_2) &\leq \eta_2, \\ f_3(x_3) &\leq \eta_3, \end{aligned} \tag{2.8}$$

The variable $c \in [0, 1]$ represents a binary decision indicating slice admission status where a 0 implies no slice admission and 1 otherwise. It is expected that the objective $\mathcal{P}(x)$ is non-convex if any of the functions $f_1(x_1), f_2(x_2), f_3(x_3)$ are also non-convex. The constraints function upper bounds are denoted by η_1, η_2 and η_3. Each of the function defines resource limitation in each domain. The functions are non-linear because revenue obtained when network resources are allocated or sliced are not linearly proportional to the revenue obtain. For instance, when resources are inelastic and on high demand revenue acquired is logarithmic[1]. This revenue model is typical with uRLLC and mMTC slice types.

Generally speaking in single resource domain, resources quantities consisting of continuous and discrete elements differentiable to the second order and Riemann summable exist in hull $\mathbb{W}$ providing the condition for convexity defined as[26]

$$conv(\mathbb{X}) := \{x : x\beta x^k + (1-\beta)x^{k-1}, \forall 0 \leq \beta \leq 1, \forall x^{k-1}, x^k \in \mathbb{W}\} \tag{2.9}$$

the state variable x belongs to the set $\mathbb{W}^{n_1}$ where x is bounded as $x_{min} \leq x \leq x_{max}$ and $x_{min} \in \mathbb{W}^{n_1}$ and $x_{max} \in \mathbb{W}^{n_1}$. The convexity of $\mathbb{X}$ within $[x_{min}, x_{max}]$ is hypercube. However this does not make it easy to solve $conv(X)$. Although the convexity of $\mathbb{X}$ can help in solving mixed integer linear program (MILP) by obtaining the convexity of the linear program (LP). This is still not feasible in solving MINLP. Therefore a more concrete solution to the a non-convex MINLP has to be found. Stating the slicing objective once again as

$$\max \mathcal{P}(x) = f_1(x_1) + f_2(x_2) + f_3(x_3) \tag{2.10}$$

The functions $f_1(x_1)$ and $f_2(x_3)$ are also non-convex indicating that there is no global maximum or minimum while $f_3(x_2)$ is not guaranteed to be summable over the minimum and maximum resource limits. This implies that the optimization problem 2.10 is still NP-hard under the conditions already provided.

To obtain an optimal solution for $\mathcal{P}$, a stricter and more viable approach is required. Many authors have attempted to solve an MINLP problem employing several techniques. One of the most popular technique is the branch and bound method developed by Land et al [27], successive convex approximation, heuristic approach and machine learning models. We

proceed to discuss these approaches and presenting limitation and issues concerning these schemes which later motivate our interest in machine learning approach for solving such a problem.

2.1.2.1 Brach and Bound (BB) approach to solving MINLP

The BB method of solving MINLP relies on two main conditions namely: 1) If the continuous variables are convex then the global solution can be found without relaxing the discrete variables. 2) If the continuous variables are non-convex then the discrete variables must be relaxed in order to find all the global solutions.

Assuming the continuous variables are non-convex then the first step results from relaxing the discrete variables . This becomes the optimal solution if the relaxed variables yields an integral solution and the operation is stopped. On the other hand, the relaxed variables will represent the lower bound and iteration continues for each tree node which becomes a sub-problem of the master problem. The upper bound is the set if any integer solution is obtained. Any node that exist beyond the upper bound is pruned while the search is conducted until entire tree is known[28]. Limitations with BB method is that the tree can grow extremely large further the sub-problems created can also be non-linear which are hard to solve. Consider a convex MINLP problem with variable set upper bounded at $\mathbb{W}^{upper} = \infty$

containing and a heap-wise empty set $G = \emptyset$ given by :

$$\begin{aligned}
&\max \mathcal{P} = [D^T x_1 + f(x_2) + f(x_3)] \\
&\text{s.t} \\
&Ax_1 + Bx_2 + Cx_3 \geq 0 \\
&x_2 \in X = \{x_2 | x_2 \in \mathbb{R}^{n_2}, x_2^{lower} \leq x_2 \leq x_2^{upper}, \} \text{ (MINLP)} \\
&x_3 \in X = \{x_3 | x_3 \in \mathbb{R}^{n_3}, x_3^{lower} \leq x_3 \leq x_3^{upper}, \} \text{ (MINLP)} \\
&x_1 \in X = x_1 | x_1 \in \{0, 1\}
\end{aligned} \tag{2.11}$$

The set containing x_2 and x_3 can be continually relaxed within $\{x_2^{lower}, x_2^{upper}\}$ and $\{x_3^{lower}, x_3^{upper}\}$. The solution to the above problem proceeds as follows. Considering a tolerance $\epsilon > 0$. We add the extreme bounds $MINLP\{-\infty, \infty\}$ of the MINLP to the heap G such that $G \cup MINLP\{-\infty, \infty\}$. The next step is to remove the bounded function $MINLP(lower, upper)$ from the heap by $\{G = G - MINLP(lower, upper)\}$, this means that the heap G is nolonger an open problem. The $MINLP(lower, upper)$ is then solved and the solution becomes $x_2^{lower,upper}$ or $x_3^{lower,upper}$ assuming the double MINLPs problems are solved simultaneously. If the solution does not satisfy all the constraints then that node is pruned. Otherwise if the functions $f(x_2^{lower,upper})$ and $f(x_3^{lower,upper})$ are outside $\mathbb{W}^u$ then that node consists of mostly upper bounds and is not feasible therefore is pruned. If the solutions $x_k^{lower,upper}$ is an integer then the current best solution in $\mathbb{W}$ is $f(x^{lower,upper})$ and $x^{'} = x^{lower,upper}$. Further, a non integral solution requires branching by introducing two new MINLP bounded child nodes one to the left and the other to the right. The two problems are then stored in the heap G.

The complexity of solving the slicing problem in Eq. 2.10 is much higher than the modified problem in Eq. 2.11. With a convex problem having integral solutions it is possible to have a fairly less mathematical complexity. However, in slicing environment where many of the optimal resource allocation models in a domain are non-convex and non-integral, the complexity of solving such problems using BB is in the order $\mathcal{O}(e^{\iota})$ where ι can be very large, denoting the number of sub-problems which could still be complex to solve.

2.1.2.2 Implementation Outline for 5G Slice Admission Control using Branch-and-Bound

In the SAC strategy, a BB approach is implemented as follows: First, the number of applicable slice requests within a slicing domain is defined, along with the target objective for each slice. Each slice request is then mapped to a corresponding set of bandwidth, compute, and latency requirements.

The general outcome of slice admission control is to either admit or reject a slice request, which is represented as the decision variable. The objective function is then defined as the reward obtained from admitting a slice, multiplied by the decision variable. Additionally, resource constraints are established to ensure that the total admitted slice blocks do not exceed the available inelastic resources.

The BB method begins by initializing a priority queue and setting the initial lower bound using a greedy heuristic, which is executed in two steps. First, slices are sorted based on their reward-to-resource ratio. Second, slices are selected sequentially until the available resources are exhausted. Finally, the obtained solution is stored as the lower bound. The upper

bound is determined by solving a relaxed version of the problem, which involves a linear programming (LP) relaxation where the decision variable can take fractional values. A slice branch is then selected, prioritizing the one with the highest reward-to-revenue ratio. The two SAC decisions—slice admission and slice rejection—are then implemented as branches, with the constraints updated accordingly. The LP is solved to compute the upper bound for the new node. If the upper bound is worse than the feasible solution then the branch is pruned and if the decisions are integer, the feasible solution is updated.

Constraint-based pruning is applied to discard branches that violate resource constraints. Finally, the branch implementation strategy, bound computation, and pruning process are repeated until no branches remain in the queue. The feasible SAC solution obtained at the end is considered the optimal solution.[29][9],[30][31]

2.1.2.3 Successive Convex Approximation (SCA) and MINLP slicing problem

The N-P hardness of an MINLP problem requires a feasible solution to obtain an optimal resource allocation model. SCA approach can assist in solving such problems. This scheme requires an iterative adjustment of objective function $P(x)$ 2.12 with x as the resource allocation vectors under strict convex constraint [10][32]

$$\begin{aligned} &\max \mathcal{P}(x) \triangleq f(x_1) + f(x_2) + ... + f(x_k) \\ &s.t \\ &P\,(x+1\,) \triangleq f\,(x_1+1\,) + f\,(x_2+1\,) + ... + f(x_{k+1}) \geq 0 \quad \forall \quad x \end{aligned} \tag{2.12}$$

The function set containing $f(x_1), f(x_2), ...f(x_k)$ consists of both linear and non-linear functions. They represent resource allocation models that are both differentiable and integrable. The aim is to find an optimal solution that maximizes profit P. This is obtained by iteratively reducing the loss function $x_k \leftarrow x_k - \Delta x_k$. Initially the feasible point is set at $x_k = 0$ and step size $\Delta \in (0, 1)$ and a loop counter $\mathbf{m} = 0$. While the optimal solution of $P(x)$ is not reached, set $x_k^{\mathbf{m}}$ as the solution. Find $\{\max f\,(x_k, x_k^{\mathbf{m}})\}$ then check if

$$P\,(x_{k+1}\,) \triangleq \sum_{k \in K} f(x_{k+1}\,) \geq 0 \quad \forall \quad x_k \tag{2.13}$$

then the iteration index $\mathbf{m} \leftarrow \mathbf{m} + 1$ is adjusted and then the loss function is updated as

$$x_k^{\mathbf{m}+1} \leftarrow x_k^{\mathbf{m}} - \Delta x_k \tag{2.14}$$

the iteration is repeatedly performed until an optimal solution is reached. SCA can be improved when semidefinite relaxation[10] is performed, this is useful in SAC where the is need to solve a multi-stage problem. SCA requires convexing a non-convex problem by choosing the most suitable step size Δ. Generally speaking, slice models can be highly non-convex meaning applying SCA would require very many small steps during linearization of the non-convex function. The complexity of solving MINLP using SCA is controlled by the number or iterations performed before global convergence is reached, this is inversely proportional to the step size Δ. The rate of convergence is in the order $\mathcal{O}(\frac{1}{\mathbf{m}})$. However despite promising performance of MINLP, any model modification would required the entire

process being repeated for another optimal solution. This inhibits the efficiency of obtaining a more optimal revenue model in the expected dynamic 5G resource allocation.

2.1.2.4 Implementation Outline for 5G Slice Admission Control Using Successive Convex Approximation (SCA)

This section outlines the implementation of SCA in solving the SAC problem. The initial variables required for SAC implementation with SCA include decision variables—typically representing slice admission or rejection—along with allocated spectral resources, compute resources, and the latency budget. The objective function is formulated as a non-convex function to maximize utility based on the decision variable, spectral resources, compute resources, and latency budget.

Resource constraints are formulated to prevent the over-provisioning of network resources. Since the decision variable is binary, the problem becomes a MINLP problem. To simplify this, the binary variable is relaxed into a continuous variable between zero and one. A logarithmic approximation is used to convexify the non-convex functions, while latency and bandwidth constraints are approximated using a modified version of the Taylor series expansion. Finally, the resource constraints are transformed into convex quadratic forms.

The iteration process for solving the SCA begins by initializing the iteration index, all variables, and the stopping criteria. The solution process then take the following steps: First, the non-convex constraints are linearized, and the original constraints are replaced with tighter convex approximations. A convex solver is then applied to solve the resulting convex

optimization problem. The variables are updated in a stepwise format, and convergence is checked before the process is repeated.

2.1.2.5 Alternating Direction Method of Multipliers(ADMM) for solving an MINLP slicing problem

The complexity of resource allocation in multi-domain environment 5G communication system requires that the resource allocation optimization problem be decomposed and distributed, ADMM is a known solution suited to tackle such problems [33]. Further, ADMM is also highly adaptive especially when dealing with large scale problems. A variant of ADMM known as alternating direction dual decomposing (ADDD) improves stability and convergence, this is particularly vital in when performing VNF admission and function chain embedding (FCE) [34]. Recalling the slice revenue optimization MINLP problem and adopting a simplified version of it we have

$$\begin{aligned} &\max P(x) = f(x_1) + f(x_2) + f(x_3) \\ &s.t \\ &Ax_1 + Bx_2 + Cx_3 = z \end{aligned} \tag{2.15}$$

In this case x_1, x_2 and x_3 still represent the resource types while A, B and C denote coefficients that determine how these resources are used to maximize the objective, z is the maximum limit for x_1, x_2 and x_3. Solving this problem requires a loss function formulated

as Lagrangian approximation given by

$$L\left(x_1, x_2, x_3\right) = f\left(x_1\right) + f\left(x_2\right) + x_3^T\left(Ax_1 + Bx_2 + Cx_3 - z\right) \tag{2.16}$$

The next step is to iterative dual update for x_1, x_2 over unit time $\mathbf{m}$ such that;

$$\begin{aligned} x_1\left(\mathbf{m}+1\right) &\leftarrow \arg\max L\left(x_1, x_2, x_3\right) \\ x_2\left(\mathbf{m}+1\right) &\leftarrow \arg\max L\left(x_1, x_2, x_3\right) \end{aligned} \tag{2.17}$$

then update

$$x_3\left(\mathbf{m}+1\right) \leftarrow x_3 + \Delta^{\mathbf{m}}\left(\sum_{m\in\mathcal{M}} Ax_2^{\mathbf{m}+1} + Bx_3^{\mathbf{m}+1} - z\right) \tag{2.18}$$

The complexity of ADMM lies on the dual update process, hence for the gradient accent in Eq. 2.18 converges at the order $\mathcal{O}\frac{1}{(\mathbf{m}^2)}$ see e.g[35]. The stepwise adjustment of the gradient ascent optimization can achieve optimal solution when the objective function is assumed convex, otherwise the complexity rises with non-convex problems associated with inter-domain slice admission control. Recent development in dynamic RL techniques have proven critical in solving problems with no guaranteed optimality and where conventional techniques rely on convexity and system parameter relaxation to achieve near optimality.

2.1.2.6 Implementation Outline for 5G Slice Admission Control Using Alternating Direction Method of Multipliers (ADMM)

The ADMM method closely aligns with the SCA approach in defining network slicing decision variables, the objective function, and constraints. The ADMM reformulation follows a three-stage process: first, the slice admission control decision variable is relaxed; second, the problem is decomposed into sub-problems; and finally, Lagrangian multipliers are introduced to enforce the constraints. The remaining process involves an iterative procedure where variables are first initialized. Then, the decision variable is updated locally for each slice, followed by the update of dual variables using Lagrangian multipliers. Finally, the problem is checked for convergence, and the process is repeated if necessary[2][36][37][33].

2.1.2.7 Heuristic Approach in Network Slicing

Slice admission control for more flexible slices such as eMBB may not require strict system constraint and therefore is more elastic. In such conditions heuristic approach is considered as a technique for resource allocation. The technique strives to achieve a best effort policy for allocating resources without considering absolute accuracy. It is computationally faster but less intensive. Many researchers have applied this method in slice admission control see e.g [38] and [39] even when the problem has characteristics of NP-hardness. A priority based heuristic resource allocation can align each slice to its quality of experience (QoE) index. Instead of solving Eq. 2.7 each resource value is allocated a Boltzmann distribution normalized priority index p for each slice category $s \in S$. In a total slice pool denoted by s_k for $k = [1, 2..K]$, a probability distribution $p_k^s = \frac{e^{s_k}}{\sum_{k \in K} e^{s_k}}$ is assigned to a resource

pool. Compare the priority to a threshold γ, such that any value $p_k^s \geq \gamma$ is admitted or else rejected. It is clear that the resource allocation will not be optimal, however the solution will be obtained much faster with less computational requirement. The complexity of a heuristic approach depends on the search steps. For any search requiring J steps in a single loop in n loops, the complexity of a such algorithm will be in the order $\mathcal{O}(nJ)$ and a time duration of $(t = \tau J + \sigma)$ where, τ and σ are the unit time required to execute a code-line, and time variance respectively.

2.1.2.8 Implementation Outline for 5G Slice Admission Control Using Heuristic Approach

The greedy heuristic approach presented in this thesis aims to maximize utility after each slice admission while considering a constrained resource pool that includes spectral, compute, and latency requirements. A score is calculated as the ratio of the utility obtained to the total resource consumption. Slices are then sorted in descending order based on this score and admitted greedily until the resource pool is exhausted[40][41][42].

2.1.2.9 Genetic Algorithm (GA) for network slicing

GA is an optimization scheme which relies on natural selection of evolution which can be used to obtain a solution which optimally selects a set of resources for network slice orchestration[9]. The random sample selection is assigned to the initial population which is eventually evaluated for its fitness. The optimization process in GA generally involves three main steps.

1. Reproduction stage: This is where policies are copied in a new set. The best performing policy is allocated higher priority and arranged to the top while the worst performing policy is arranged to the bottom. In network slicing revenue optimization, resource selection strategy that generates more revenue is considered as the fittest policy and is allocated a higher priority[1].

2. Crossover stage: In this stage, the reproduced policies are paired in a random manner, new policy is the "child" which is considered to be more superior than the parents by virtue of inheriting "genes" from both the parents[1].

3. Mutation stage: The process involves multiple rounds of bit inversions with the purpose of generating genetic variation from the original policy. It is important to keep the mutation probability low to avoid a purely random search. In the context of slice admission control, the mutation process aims to create an improved policy for selecting the best requests[1].

In this context, consider a random population given by $x_{(n)} = \{x_{(1)}, x_{(2)}, ..x_{(N)}\}$, a candidate in the population can be selected by a probability p. The candidate is considered to have a binary string b of length l. A solution of the objective can fall among any binary set $b_{i \in I}$ [43]. Based on a Boltzmann distribution, each candidate probability of selection falls within $p_{(n)} = \frac{e^{x_{(n)}}}{\sum_{n=1}^{N} e^{x_{(n)}}}$. The crossover state follows a binary process with probability $p_{co} = [0, 1]$. In the mutation stage any n^{th} candidate can mutate with a probability p_q. For instance if a candidate has a binary string $b = 001000$ and another $\bar{b} = 110111$ of length j then by mutation

$$p_q \cdot p_q \cdot (1 - p_q) \cdot p_q \cdot p_q \cdot p_q = p^j (1 - p_q)^{l-j} \tag{2.19}$$

.

For a non superior "child", the parents must be have had similar genes. Then the probability that the parents with b and $\bar{b}$ are the same is given by

$$P(b \to \bar{b}) = p_q^{H(b,\bar{b})} (1 - p_q)^{l - H(b,\bar{b})} \tag{2.20}$$

Where $H(b, \bar{b})$ is the Hamming distance between b and $\bar{b}$[43]. The adoption of GA in network slicing may suffer the following limitations i) evaluating a policy repeatedly ii) lack of scalability, ii) inefficient decision making for binary options and iv) sub-optimal settling. In the context of complexity, if a slice s has N resource features to complete for full orchestration, also assuming each individual feature can be mapped into a binary control condition indication feature allocation or not, then the entire service construction is in the order $\mathcal{O}(2^{sN})$. The complexity to convergence can further worsen if there are M reproduction stages and K iterations in the crossover stage. Hence the complexity to convergence becomes $\mathcal{O}(2^{s \times N} \times M \times K)$[7] [43][44].

2.1.2.10 Implementation Outline for 5G Slice Admission Control Using Genetic Algorithm

The methodology for slice resource initialization remains consistent across all the discussed approaches. The following section outlines the steps for using GA to solve the SAC problem.

Initially, a finite binary vector is created to represent slice admission (1) or rejection (0). The values in this vector can be randomly assigned but must remain within the limits of available resources and network constraints. A fitness function is then generated. The process begins with a selection procedure to identify the fittest slices for admission, followed by a crossover operation to combine solutions. Next, a mutation process is applied to explore new slice configurations. Constraint handling is achieved through penalty methods, which reduce fitness for constraint violations, and a repair method that adjusts values to ensure constraints are met. Finally, a stopping criterion is applied, terminating the process when the maximum number of generations is reached or when no further improvement in fitness is observed[45][29].

2.1.2.11 Reinforcement Learning Approach

The adoption of machine learning approaches to resource management in 5G communication system is currently on the upward trend and many literature have been published on this area see e.g [46],[47],[48],[49],[11],[50],[12],[51],[52],[53]. Machine learning, particularly RL which is the focus of this research provides a robust methodology for solving complex problems in a manner that mimics human behavior. RL provides superior performance in control and has been applicable in robotics and gaming[54].

The benefit of using RL methods lies in their ability to eliminate the necessity of constructing a system model. A general presentation of RL method is provide in Fig. 2.2

RL works by interacting with an environment. All the essential information required for an agent to facilitate learning is encompassed within the environment. This information

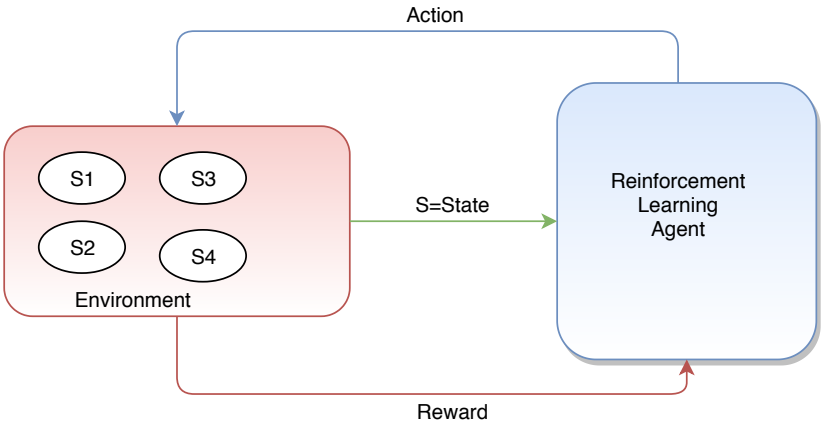

Figure 2.2: An Illustration of Reinforcement Learning

predominantly comprises system states, rewards, and available actions. A learning agent visits the environment and based on the system state, an action is taken. This action is valued by a reward. A good action attracts more reward while a bad is less rewarded. Precisely the main elements of the environment are defined as follows.

- State: A state is a mathematical scalar, vector or a closed function that represent a unique condition of a system. An agent improves its behavior by making sure similar states attracts same or improved responses.

- Action: An action is the agent response to a particular state. In a physical environment, action can be actual movement on an object. However, in an emulator an action, is represented by an index. With experience, an agent learns to take the best action in a particular state.

- Reward: A reward is a scalar quantity chosen or computed based on the action taken. Similar actions must attract similar rewards otherwise, the reward function or selection policy is deemed to be poorly constructed. A mathematical formulation is usually

constructed to map a reward to its temporal characteristic. Meaning that the current reward may not be equal to a future reward even when the same action is taken.

The goal in RL is to maximum the long term reward. The set of probabilities that map states action pairs leading to reward maximization is known as a policy. In a model based learning, these probabilities are well known in advance usually constructed using Markov Decision Process (MDP), conversely a model free environment does not require any complete knowledge of the mapping between state-action pairs. In this case, the policy is typically learned dynamically as the agent interacts with the environment, adapting and improving over time.

The complexity of employing RL is solving revenue optimization in a constrained network slicing ecosystem depends on the size and complexity of the environment. In this regard an environment will constitute the training data-set which can be historical or dynamically generated, model constraints, the set of actions and the reward function. One of the simpler form of RL solutions is the Q-learning approach. Consider a state matrix $\mathbb{S}^{|\mathbf{U}\times\mathbf{V}|}$ the constructed experience table known as the Q-table will be in the order $\mathcal{O}(\mathbf{U}\times\mathbf{V}\times\mathbf{Z})$ where $\mathbf{Z}$ is the set actions. The size of the Q-table can grow extremely as the environment becomes more complex usually known as the curse of dimensionality.

2.1.2.12 Reinforcement Learning Justification

The adoption of RL in this study has been well interrogated and justified in our publication in [1]. Further, we can draw a summary of justifications as follows. First, in terms of adaptability, RL learns optimal policies by interacting with an environment enabling it to

learn the dynamic conditions and gain experience [16]. In contrast, the heuristic approach requires manual tuning, while the GA, which employs an evolutionary strategy may also require manual tuning to adapt to new states. Further, both ADMM and SCA heavily rely on the convexity of the problem which limits their adaptability[10].

Second, to achieve non-biased target selection, a SAC solution must generalize learned policies across the dynamic and non-similar environments enabling it to accurately respond to never-seen-before states. Both ADMM and SCA are designed for specific problem structures, a property that hinders their ability to generalize responses. The heuristic approach does not have any ability to generalize solutions. The GA may often demand more computational capability and tailored design to work well in heterogeneous environments[36][55][38][56][7].

Third, RL naturally balances the exploration of new actions with the exploitation of known strategies to maximize long-term rewards on the path to optimal solutions. In contrast, heuristics, GA, ADMM, and SCA primarily emphasize exploitation. They face challenges such as inefficient utilization of high-quality solutions, reliance on a fixed update scheme, and the absence of an explicit exploration mechanism[55][57][57][58].

Fourth, although RL techniques can be computationally expensive, they scale effectively with deep learning and can efficiently handle non-convex and high-dimensional problems. Additionally, RL approaches are easily automated, with convergence largely dependent on the specific algorithm and hyper-parameters. Finally, RL is highly suitable for real-time and online learning[12, 49, 50, 59–65].

Clearly, RL provides the agility and flexibly required to solve a dynamic system state in heterogeneous network is discussed in this chapter. Section 1.5 has provided the research questions of which bullet two is answered this Chapter.

2.1.3 Chapter Summary

In this chapter, slice admission control has been presented as an optimization problem. A comprehensive analysis has been conducted to highlight the feasibility of each optimization scheme in solving slice admission control. The motivation for employing machine learning lies in its ability to provide generalization during the search and selection of state entities, enhancing decision-making and adaptability in dynamic network environments.

Chapter 3

Reinforcement Learning for Slice Admission Control

Reinforcement learning is a machine learning technique designed to mimic human learning, an agent which interacts with an environment (*states*) makes decisions (*actions*) with main objective of maximizing returns (*rewards*)[12]. RL is different from other forms of learning in that, the is no need to have a pre-existing data-set for training. This improves automation in a highly dynamic environment[56]. Further, Keonig et al in [66] have categorically inserted that, RL are highly tractable without any need for augmentation confirming the assertion in Chapter 2 about the run-time being small polynomial in state space. This of course is highly suitable in SAC online learning. Despite numerous research efforts employing other forms of machine learning, system performance has not been entirely decoupled from dependence on specific datasets. In contrast, RL's minimal reliance on specific datasets makes it well-suited for SAC environments. [67]. In 5G resource management, on-demand adjustments are crucial. For example, radio resource slicing requires that, despite its inelastic nature, slice requests be evaluated based on long-term benefits. Consequently, decisions must be made instantly based on the observed state, without relying on complex mathematical extrapolation.

3.0.1 State Space

In many slice admission control problems, a state space S comprises all the information required to make a decision [68]. Specifically, consider slice type $k = [1, 2, 3, \cdots K]$ with r_k, ω_k and δ_k being radio,computing, and storage resource blocks required per slice[69]. The state space $\mathbf{S}$ is therefore defined by a $K \times N$ matrix given by Eq 3.1.

$$\mathbf{S} \triangleq \begin{Bmatrix} r_{1,1} & \omega_{1,2} & \delta_{1,3} \\ r_{2,1} & \omega_{2,2} & \delta_2, 3 \\ . & . & . \\ r_{k,1} & \omega_k, 2 & \delta_k, n \\ . & . & . \\ . & . & . \\ r_{K,1} & \omega_{K,2} & \delta_{K,N} \end{Bmatrix} \tag{3.1}$$

where n is the number of resource blocks required per slice. The learning agent iterates through the entire state space and during each iteration a decision is made at the decision epoch. A decision epoch is the duration between when a slice request is encountered and the time a new request is read.

3.0.2 Action Space

Typically in slice admission control there are mainly two actions in the action space namely: slice request accept or reject denoted by $\mathbf{A} = [0, 1]$ where a 1 implies slice request accept

and 0 otherwise. Other actions may include slice upgrade or downgrade and slice release. In an RL based slice admission control, for any given state an action associated to a state $s \in \mathbf{S}$ is reinforced by a positive reward value r. An optimal policy π^* will always return the highest reward for every action taken.

3.0.3 State Transition

State transition is a method by which an agent determines the next state. In slice admission control, considering the metrics that influence the decision of admitting or rejecting a slice request. Such metrics may include the spectral resource, optical resources , computing resources number of served users, the coverage area and slice duration [15]. To derive an optimal path, an RL agent choice actions are dependent on the state transition which can be classified as either Markovian or Semi-Markovian[70]

3.0.3.1 Markov Decision Process (MDP)

If a slice admission control problem can be formulated as a Markov decision process (MDP) then it can be solved using RL. An MDP follows a tuple given by $M = (\mathbf{S,A,P,R})$ where **S** denotes the state space, **A** denotes the action space **P** is the finite transition probability and **R** is a reward function which is maximized during learning[62]. The RL agent observes these unique metrics as states (**S**), takes an action (**A**), moves to the next state (**S'**) with a probability of transition (**P**) and then observes reward (**R**)[56]. A policy π is evaluated as a mapping between states and a probability distributions over actions such that:$\pi \rightarrow \mathbf{P}(\mathbf{A} = a|s)$. This policy influences both state-action selection and reward retrieval which leads to

long-term return given by $\mathbf{R} = \sum_{k=0}^{K} \gamma^k r_{k+1}$, moreover, the aim is to determine an optimal policy π^* that always leads to maximum return every time an action is taken[54]. $\gamma \in \{0, 1\}$ is a discount factor which reduces the significance of far off returns of each discrete step k . Considering a finite state space where at any time instant the agent must be in one of the states. The agent must decide the next state **S'**. The transition to the next state **S'** $= S_{k+1}$ is by a probability $\mathbf{P} = Pr(\mathbf{S'} = S_{k+1}|S_k, A_k)$, the set of all probabilities at a time instant that influences the choice of the next state is given by $\sum_{S' \in S} \mathbf{P}(\mathbf{S'}|S, A) = 1, \mathbf{P}(\mathbf{S'}|\mathbf{S}, \mathbf{A}) \geq 0$ noting that the transition probability function depends only on the current state and action. Let $\{y_0, y_1, y_2 \dots y_k \dots y_K\}$ be states in a Markov process of size K in a state space S. The transition probability from state k to state $k + 1$ is given by Eq 3.2

$$(P)_{s_k, s_{k+1}} = p_{s_k, s_{k+1}} = \mathbb{P}(y_1 = s_{k+1}|y_0 = s_k) \tag{3.2}$$

Proof: For the two-state transition provided in 3.2 let us state the following:.

$$\begin{aligned} \mathbb{P}(y_0 \to y_0) &= \varepsilon \\ \mathbb{P}(y_0 \to y_1) &= 1 - \varepsilon \\ \mathbb{P}(y_1 \to y_0) &= \overline{\varepsilon} \\ \mathbb{P}(y_1 \to y_1) &= 1 - \overline{\varepsilon} \end{aligned} \tag{3.3}$$

Since it holds, the sum of all transition probabilities must be equal to 1. We have;

$$\begin{aligned} \mathbb{P}(y_0 \to y_0) + \mathbb{P}(y_0 \to y_1) = \varepsilon + (1-\varepsilon) = 1 \\ \text{and} \\ \mathbb{P}(y_1 \to y_0) + \mathbb{P}(y_1 \to y_1 = (1-\overline{\varepsilon}) + \overline{\varepsilon} = 1 \end{aligned} \tag{3.4}$$

end of proof. For a k step transition, the probability distribution is given by;

$$P(y_k|y_0) = P^k \tag{3.5}$$

Proof:

Let us consider the distribution probability of a single state y_0. The transition from y_0 to y_1 has K probability option given by an $K \times 1$ matrix denoted by;

$$\beta_0 = \left\{ \begin{array}{c} \beta_1 \\ \beta_2 \\ . \\ . \\ \beta_k \\ . \\ . \\ \beta_K \end{array} \right\} = \left\{ \begin{array}{c} \mathbb{P}(y_0 = 1) \\ \mathbb{P}(y_0 = 2) \\ . \\ . \\ \mathbb{P}(y_0 = k) \\ . \\ . \\ \mathbb{P}(y_0 = K) \end{array} \right\} \tag{3.6}$$

It follows that;

$$\begin{aligned}\mathbb{P}(y_1 = s_k) &= \sum_{k=1}^{K} \mathbb{P}(y_1 = s_{k+1}|y_0 = s_k)\mathbb{P}(y_0 = s_k) \\ &= \sum_{k=1}^{K} p_{s_k,s_{k+1}}\beta_k \\ &= \sum_{k=1}^{K} \beta_k p_{s_k,s_{k+1}} \\ &= (\beta^T P)\end{aligned} \tag{3.7}$$

for any k step transition starting from the initial state y_0 we have;

$$\mathbb{P}(y_k) = [\beta]^T P^k \tag{3.8}$$

where β^T is a row matrix obtained from β_0.

The optimality equations incorporating transition probabilities are the key merits of Markov decision problems. They are often referred to as *functional equations* or *Bellman equations*[56][71]. They are given by

$$V^*(s) = \max_a \sum_{S' \in S} P(S'|S, A)[R + \gamma V^*(S')] \tag{3.9}$$

or

$$Q^*(S, A) = \max_a \sum_{S' \in S} P(S'|S, A)[R + \gamma Q^*(S', A')] \tag{3.10}$$

where 3.9 is known as the state value equation and 3.10 is known as the action value equation. By adopting MDP in optimal policy evaluation, a priori knowledge of the environment

is necessary. Conversely, this is not always the case in slice admission control, where states are dynamic and non-deterministic and, in numerous cases, may be partially or fully unknown[72]. One way of solving this problem is by adopting a semi-Markov decision process. We proceed by discussing this concept in the next section.

3.0.3.2 Semi-Markov Decision Process (SMDP)

Unlike model-based algorithms, where an agent tries to learn how the environment works and provide a complete trajectory for optimal solution, in model-free algorithms, the agents do not try to learn the environment but instead try to estimate the goodness of any state-action pair in order to derive an optimal policy[46].

Many model-free algorithms often adopt a semi-Markov decision process to deal with the uncertainty of the environment. Unlike MDP, an SMDP is represented by a five-tuple given by $M = (S, A, P, R, D)$ [70]. The new parameter D is a random value that represents how long an agent remains in a state or how long an agent waits for a state to arrive. Intuitively, both P, R, D are Markovian. Therefore, in order to obtain an optimal policy, the three parameters (P, R and D) must be considered i.e P provides the trajectory towards an optimal policy, R value that ensures the agent is always rightly reinforced for an action taken after visiting a state, and D is the mean transition period from one state to another. In slice admission control, slice requests are often discrete network parameters [69] whose arrival rates are stochastic. The sojourn time of a slice in the network can be modeled as a continuous distribution. From this concept it is therefore important to derive the

semi-Markov process that will lead to optimal slice admission control. Recall that P is the transition probability that must be derived.

Assuming that a state ($\mathbf{s}$) arrival has a of mean μ_s, the probability distribution can be modeled by the Poisson distribution given by

$$P(\mathbf{s}; \mu_s) = \frac{e^{-\mu_s}(\mu_s)^{\mathbf{s}}}{\mathbf{s}!} \tag{3.11}$$

the probability that a particular desired state will arrive in the next time interval Δt is given by

$$P\{s(t + \Delta t)\} = v \cdot \Delta t \times P(\mathbf{s}) + o(\Delta t) \tag{3.12}$$

the probability more than one state will arrive in the same time interval is given by $P\{s(t + \Delta t)\} = o(\Delta t)$ and finally the probability that no states will arrive in the same time interval is given by

$$P\{s(t + \Delta t)\} = 1 - v \cdot \Delta t \times P(\mathbf{s}) - o(\Delta t) \tag{3.13}$$

where $\mu_s = vt$, for $1 > v > 0$ and t is the time interval under consideration v is the occurrence rate.

When a state arrives, it has a sojourn time in the queue. This duration is continuous and can be modeled by exponential distribution in the format.

$$P(\mathbf{s}) = \frac{1}{\lambda} e^{-s/\lambda}, \quad s > 0 \tag{3.14}$$

where λ is the mean rate. The probability that state will last additional time t in the queue is independent of the fact that it had lasted time t previously. Hence each state's sojourn time does not affect how long it lasts in the queue. Since the probability of staying in the queue for exactly $t = t_0$ is equal to zero, we assume a time interval Δt as duration within which a state can stay in the queue. Therefore the probability that a state will stay in the queue with the time duration Δt is given by

$$P\{s(\Delta t)\} = \frac{1}{\lambda t} e^{-s/\lambda t} \tag{3.15}$$

3.0.4 Policy Evaluation

Policy evaluation involves determining the set of probabilities that map states to actions in a trajectory which leads to maximizing the long term reward function, this can also be called prediction problem[56]. There are two techniques of policy evaluation: *on-policy* and *off-policy*.

3.0.4.1 On-policy

In the on-policy mode, the general assumptions is that the agent has a complete knowledge of the environment. More explicitly, the on-policy method endeavors to improve a policy π based on the mean returns from every value function evaluation in an episodic iteration[73]. An example of on-policy evaluation is the Monte-Carlo methods for solving RL.

In 5G network slice admission control, resources can be scheduled then orchestrated based on instantaneous end to end resource blocks and the QoS requirements. Therefore, having

a prior knowledge of the complete system is highly infeasible. In this case, an algorithm employing in on-policy method will have to perform endless number of episodes to reach near optimal solution. This is computationally expensive and not attainable.

An on-policy method relies on initial arbitrary policy π_{init} which is constructed on a soft $\epsilon - greedy$ and the consequent set $\pi = \{\}$ is evaluated in an episode such that $\pi(a \mid s) = 1 - \epsilon + \frac{\epsilon}{A(s)}$ for greedy actions and $\pi(a \mid s) = \frac{\epsilon}{A(s)}$ for non greedy actions. Once $\pi(a \mid s)$ is completely evaluated, the $\epsilon - greedy$ shifts closer to non-stochastic optimal policy. Further, inefficiency caused by sample collection during each iteration increases computation cost for practical cases, however there have been attempts to improve stability in some on-policy approaches such actor-critic , trust region policy optimization (TRPO) and proximal policy optimization (PPO) by taking broadest variance and by replacing the hard constraints in TRPO with penalties respectively [74].

3.0.4.2 Off-policy

Due to the massive temporal dynamics of 5G networks, resource management and slice admission control require a general approach to solving the admission control problem. An approach that is not deterministic but can still converge to an optimal or near-optimal solution. Off-policy techniques can fit this requirement because they have additional concepts that make them more powerful, albeit slow to convergence.

Off-policy algorithms do not require complete knowledge of the environment but only an estimate of it. They work by employing what is known as *importance sampling*. Importance sampling allows both the learned and target policies to be compared by estimating the

probability distribution of the expected target policy from the behavioral policy and reducing the variance between them so that, in the end, the importance sampling ratio becomes almost equal to one. Assuming that the behavioral policy π_{be} have a distribution $\mathbb{P}_{\pi_{be}}$ and the target policy π_{ta} has probability distribution $\mathbb{P}_{\pi_{ta}}$ for their trajectories, thus for an episodic event starting from time t ending at T, has an importance sampling ratio given by [75]

$$\mu : T = \prod_{t}^{T} \frac{\pi_{be}\mathbb{P}_{\pi_{be}}(a_t \mid s_t)}{\pi_{ta}\mathbb{P}_{\pi_{ta}}(a_t \mid s_t)} \tag{3.16}$$

All these trajectory probabilities are highly correlated and are obtained based on anonymous MDP. For an optimal slice admission control the distributions cancels each other ending up with the two policies which are independent of the MDP[56].

In general, off-policy techniques rely on experiences stored as data set in replay memory to update a target policy from the behavioral policy and are known to have high sampling efficiency and can tackle continuous state-action space and can also be made to perform more exploration by adding entropy[54][76]. In our slice admission control setup, network parameters to be considered are non deterministic and time constrained, thereby employing off-policy RL optimization is the most appealing way of achieving near optimal or optimal admission control. Some off-policy techniques include Q-learning and deep Q-learning (DQL), deep deterministic policy gradient (DDPG) and off-policy actor-critic (Off-PAC)[74].

3.0.5 Model Based verses Model Free Reinforcement Learning

In model-based learning, an agent first determines the environment model and then utilizes this knowledge to address subsequent encounters, devising an appropriate policy. A classic instance of model-based learning is the multi-armed bandit game. Conversely, in model-free learning, the agent assesses the state and its corresponding optimal actions to formulate a policy [73]. For instance, Q-learning exemplifies model-free learning, wherein the evaluation process involves constructing an action value function termed the Q-value. Initially, the agent explores the environment with a certain random probability but gradually transitions to selecting actions with high average returns, adopting a more greedy approach. The agent lacks the ability to predict the subsequent state except through the returns garnered from actions.

3.1 From Q-learning to Function Approximation Models

3.1.1 Q-learning

Q-learning approach implies an RL technique where the learning process is model free and off-policy. The learning process is based on value function update. A finite size two dimensional table known as a Q-table representing all possible states and actions is employed to store gradually updated action values[77]. An action **a** is initially chosen with an exponentially decaying probability $1 - \epsilon$, a reward **r** is observed which is used to update a value function

known as a Q-value. To gain experience, an agent chooses other action with probability ϵ and observes the corresponding reward. A Q-value in the Q-table is again updated in a manner as.

$$Q(\mathbf{s,a}) \leftarrow Q(\mathbf{s,a}) + \alpha(R(\mathbf{s,a}) + \gamma \max_{a'} Q(\mathbf{s'},\mathbf{a'})) \tag{3.17}$$

where α and γ are the learning rate and discount factor respectively.

As the network complexity grows, so does the Q-table, leading to memory limitations and slow convergence. This "curse of dimensionality" restricts Q-learning to considering only a subset of potential states, compromising its effectiveness in a highly dynamic scenario.

3.1.1.1 Complexity analysis of Q-learning

Slice admission control inherently involves evaluating a slice request alongside its value function and assessing the potential impact on network constraints upon admission. Employing Q-learning to determine the suitable set of slice requests within non-deterministic network constraints necessitates a balance between exploration and exploitation within a finite-size state space. A worst case scenario arises when, in each learning episode, the agent encounters a novel state with tabula rasa. Additionally, owing to the rapid temporal shifts in network dynamics, an online-based learning model seems most plausible. In this approach, each state reflects the current network condition, and a slice request typically leads to encountering a new state every time the state space is observed.

Considering a state space comprising N states, each state being unique and encountered only once by the agent, implies that the state space could be considered infinite or, at the very least, a subset of a Euclidean space, even if an episode concludes upon reaching the

N^{th} state. The complexity of reaching an optimal solution has an upper bound of $\mathcal{O}(N^{\infty})$. In this regard the agent will never gain any experience and the probability of exploiting any event is zero. On the other hand, if a Q-learning is zero initialized and one-step invertible then the complexity of reaching optimality is $\mathcal{O}(N^2)$ with a worst case being $\mathcal{O}(N^3)$[78].

3.1.2 Function approximation

As previously highlighted, model-free algorithms, including techniques like Q-learning, operate without prior knowledge of the environment. Nevertheless, Q-learning encounters the curse of dimensionality when confronted with a larger state space, especially in attempts to emulate a practical environment for network slicing.

Effectively assessing a realistic network environment characterized by a continuous state space often necessitates the use of SMDP. Function approximation methods, leveraging deep learning through artificial neural networks, offer a solution for addressing SMDP by estimating either the value function of an event or its policy function. We illuminate this process by conducting a complexity analysis of three function approximation methods, ultimately determining the most suitable approach for resolving the suggested slice admission control challenge.

3.1.3 Deep deterministic policy gradient

Deep deterministic policy gradient(DDPG) are methods of actor-critic deep reinforcement learning where the policy update is deterministic[16] and are mainly used for solving problems with continuous action spaces. Since DDPG evaluates a deterministic policy, initial

exploration of a known environment may not yield a wide enough experience, in this regard DDPG adds what is known as a mean-zero Gaussian noise to improve exploration. The concept relies a finite size memory buffer for storing experiences. A mini-batch sample of the experience is employed to train a policy network by gradient ascent while the Q-function is updated by gradient descend.

Slice admission control at its simplest form is a binary decision problem, where a decision has to be made by admitting or rejecting a slice request considering real-time network constraints. However some literature have implied multiple decisions including resource upward or downward resource adjustment, nonetheless the decisions are still discrete and finite[2][17]. Employing DDPG to solve such non-differentiable problems adds unnecessary complexity.

3.1.3.1 Complexity analysis of DDPG

Notably, policy gradient methods are designed to improve the policy by generating causal vectors along the trajectory of the objective function. By evaluating learning samples over a certain number of episodes, the aim is to reach an $\epsilon - optimal$ policy where ϵ is sufficiently small. For some parameter vectors $[\Gamma]^{|K\times T|}$ of the target policy approximator in DDPG, the actor critic converges to the global optimum $\delta_k = r_k + \gamma J^{\pi^*} - J^{\pi} \leq \epsilon$ having a sample complexity of $\mathcal{O}(\epsilon^{-3})$ without considering a bias[79]. Where r_k is the sampled reward value at state $k = [1, 2..K]$ for a total for T time steps, J^{π^*} is the function value of the actor under optimal policy π^* and J^{π} is the value function of the critic under the natural policy π.

3.1.4 Deep Q-learning

Deep Q-learning (DQL) was first proposed by Google's DeepMind to play Atari games. DQL and has been improved to play other games such as Connect 4 from Kaggle, and Tetris. While there have been modified approach to DQL such as deep duelling Q-learning. A critical survey by [16] argued that DQL rendered improved stability when used with many inputs compared to other policy gradient methods.

DQL adopts both exploration and exploitation to improve a behavioral policy. The policy network stores its experience in a finite buffer called replay memory, where a target network estimates the next state by picking a random sample from the replay memory. The outputs of both networks provide value functions that are non-correlated. By this, a loss function $\mathcal{L}(\theta)$ estimated through mean square error (MSE) is minimized. Generally, actor critic have promising improvement is the field of deep reinforcement learning but they do not always have improved performance compared to DQL[16, 62]

A DQL can be combined with convolution neural network (CNN) to improve learning where highly dimensional data is adopted. An efficient slice admission control scheme must map multidimensional data input to a unique objective defined by an InP. By considering DQL with CNN, a slice request from multiple mobile network operators (MNOs) can be value-approximated to reveal its uniqueness. A technique that is more intelligent and generic. Whereas, classical DQL utilizes a single neural network requiring a second pass through the network to calculate the Q-values, a double DQL is a more stable version which employs two networks for both policy and target evaluation[69].

3.1.4.1 DQL complexity analysis

On a common setup where a state space is considered to be a 2 dimensional tensor, we extend the complexity analysis to prove that double DQL yields a reduced complexity performance when compared with both Q-learning and DDPG methods. Further, as mentioned in[16] there is no proof that any DDPG method offers better stability compared to double DQL unless the action space is continuous.

Considering a square frame of width w and a kernel of size z employed in a convolution neural network of $\mathbb{L}$ layers, c nodes and $\mathbb{L}_f$ fully connected layers then this stage complexity has the order $\mathcal{O}(w^2 z^2 \mathbb{L}_f)$. Further, the fully connected layers of DQL will then have complexity of $\mathcal{O}(2c^2 \mathbb{L}_f)$. Consequently, the overall complexity of a double DQL is given by $\mathcal{O}(w^2 z^2 \mathbb{L}_c + 2c^2 \mathbb{L}_f)$

3.1.5 Chapter Summary

In this chapter, the study has outlined reinforcement learning as a unique approach to solving SAC. The formulation of state space, action space, state transition, reward functions, and policy evaluations have been comprehensively discussed. Further, the chapter has presented the significance of each RL variant beginning with Q-learning and gradually progressing to function approximation techniques. The complexity analysis of RL has been discussed

Chapter 4

Queuing based Multi-InP Resource Scheduling for Slice Admission Control with Deep Reinforcement Learning

Slice admission control key objectives have mainly encompassed profit maximization [10][11], QoS control[80][38], slice fairness [14] and inter/intra slice admission control[13]. Singular objective approach is less conclusive in terms of determining the true revenue implication of a slice resource allocation. In this chapter a queuing based multi-InP resource scheduling for slice admission control is introduce. This scheme leverages server scheduling to improve allocation efficiency and reduce cost associated with in-server waiting time. This approach promised reduced SLA penalties thereby improving revenue obtained by network slice operators. We begin this chapter by motivating the need to perform slice scheduling and then derive the analytical presentation of queuing model. Later we present the problem formulation and solving it using reinforcement learning. In the end the simulation results are discussed.

4.1 Overview of Slice Scheduling in a Multi-Tenant Multi-InP Ecosystem

The network slicing architecture must support a multi-user multi-service environment in which each use case can be associated with a specific service. Many of these services are requested by a user, and the MNO subsequently forwards them as a slice bundle to an InP[13]. Because the time of a slice request is typically random, the provider cannot manually foresee a service and prepare resources in advance. However, with huge cloud distribution, a slice can still be orchestrated close to a user, raising the question of whether it is possible to accurately perform slice scheduling and instantiate it only when and where it is required. On that note, an InP can decrease OPEX by limiting costly delays and get more revenue by performing efficient scheduling.

A network operator can create more income by simply admitting as many users and as many slices as possible; however, the trade-off between reducing OPEX and producing more revenue is a complicated balancing act, as the latter may lead to network overloading. Using resource scheduling, an MNO can learn network dynamics and identify slice cost changes[15]. We offer a typical scheduling model in which slices are cheaper when the network is less crowded and delays are projected to be minimal. These network dynamics are thought to be advantageous when creating consistent decision variables for machine learning systems.

Different slice classes eg eMBB, uRLLC and mMTC have specific requirements which can not be met under a common network status[39]. It is therefore key to separate temporal instances

under which these slices are instantiated, this can only be done through efficient scheduling. If an MNO can determine when and where it is convenient to deploy a slice, a revenue model can be placed on such slices and scheduling effect considered. With consistently varying network conditions, relying on historical data for proper scheduling is not efficient, as such data may not be available or simply takes too long to collect [81]. Hence, the choice of efficient machine learning algorithm for resource scheduling should be flexible enough to adopt both real-time and historical data. The reward based approach in RL allows for real-time learning and experience used in improving scheduling.

In recent times, researchers have attempted to perform resource scheduling majorly as a means of reducing slice orchestration delays but not as away of minimizing OPEX. The study presented in [82] introduced an approach for resource scheduling in mobile edge computing within ultra-dense networks. The primary objective was to enhance efficiency by implementing offloading strategies. The problem was tackled using the Newton-IPM (Interior Point Method) based computing resource allocation algorithm. However, this technique lacked the capability to forecast future resource fluctuations.

Further, a task scheduling approach formulated as an NP-hard problem presented in [83] aimed at optimizing network utility. The researchers employed a GA based adaptive mating distance (GA-AMD). By dynamically adjusting the mating distance between fitter parents a broader range of exploration can be achieved however, more complexity and computational time is increased, further, GA-AMD generally is a problem specific solution and can not perform generalization as expected when network conditions changes rapidly. In [41], the authors performed delay violation probability (DVP) minimization by scheduling the wireless

edge network (WEN). The authors objective was to evaluate the effect of semi-static and dynamic scheduling policies.

When faced with multi-service scheduling on the cloud a technique called Fregata which is a partial machine learning based has been applied[84]. This technique allows a scheduling agent to perform dependency evaluation. Jobs with high dependencies are prioritized and provided with low latency, this approach has been performed on Amazon EC2 cloud services. The work in [85] introduced a resource scheduling for long term evolution (LTE) where the objective was to optimize reuse distance in vehicle to everything (V2X) so as to improve QoS in mode 3 of V2X. Other researchers have investigated adaptive scheduling mainly to improve response and processing time for tasks that are loaded onto servers[86]. Although response and processing time are key performance indices that are of interest in this thesis, to infer efficiency in an algorithm, deploying an efficient queuing can massively reduce processing time.

In [78], Li et al. proposed a cooperative approach for E2E scheduling. The focus of this approach was to address scheduling challenges in a coexistence environment. By applying cooperative strategies, the authors aimed to enhance the efficiency and performance of the scheduling process, ultimately improving the overall communication quality for users.

Similarly, in [87], a cooperative approach investigating low complexity in radio resource allocation was presented. This investigation was focused on uRRLC and eMBB slices. By employing cooperative methods, the authors intended to efficiently resolve conflicts that may arise as a result of the coexistence of these slices, thereby ensuring enhanced resource allocation and performance for both uRRLC and eMBB services.

In slice admission control revenue optimization and OPEX reduction through efficient queuing is a complex trade-off between rapidly processing slice requests for admission control and limiting over-subscription. In this chapter, the effect of efficient queuing and scheduling on OPEX reduction and revenue optimization is investigated. Deep RL is implemented at the scheduling stage while a multi-server queuing precedes the scheduling processing.

The rest of this chapter is organized as follows. The system model is presented in Section 4.1.1, while in Section 4.1.2, we present the solution using Deep RL.

4.1.1 System Description for Slice Queuing and Scheduling

4.1.1.1 Queuing Model

Consider $m \in M$ InPs with multiple resource pools and $n \in N$ tenants/MNOs receiving multiple request from multiple users for slice onboarding. In this scheme both InPs and MNOs must collaborate since it is assumed that at no point the resource aggregation pool can be depleted. Figure 4.1 represents a multi-InP, multi-tenant and multi-server scheduling and queuing process.

Each InP periodically provides a network update and resource status; this information must be stored on a central server known as the update server. Each tenant is assumed to have access to the update server and can see resource availability. Each tenant request is queued, and the queue manager strives to manage the process as efficiently as possible by reducing queue delays. Nonetheless, collaboration with the scheduling algorithm remains necessary. It is anticipated that the cost of network resources will continue to fluctuate due to rapidly

changing network conditions. This study adopts an M/M/C queuing system which is a multi-server multi-queue approach. The justification for adopting this scheme is that: The M/M/C queuing model is the most suitable for this scenario due to its ability to efficiently handle multiple concurrent tenant requests while ensuring optimal resource allocation and minimal delays. Given that multiple infrastructure providers (InPs) maintain separate resource pools and serve numerous tenants with varying demands, a multi-server queuing system is essential to balance load distribution and maintain system efficiency[88].

First, the Markovian (M/M) assumptions of Poisson arrival and exponentially distributed service times align well with the dynamic nature of network slicing, where tenant requests arrive unpredictably, and service times vary based on resource availability and network conditions. These assumptions allow for a realistic modeling of stochastic traffic flows in a 5G environment[89].

Second, the multi-server configuration ensures that multiple service resources are available simultaneously, preventing bottlenecks and improving system responsiveness. Since network resource aggregation ensures that no pool is completely depleted, an M/M/C system effectively distributes requests across available servers, reducing congestion and preventing service disruptions[90].

Additionally, real-time updates from the InPs are crucial for tenants to make informed decisions regarding resource availability. The central update server acts as a shared information hub, and the queue manager utilizes this data to dynamically manage slice onboarding. An M/M/C system accommodates this by adjusting service rates based on real-time demand and resource status, ensuring a seamless and efficient queuing process[91].

Lastly, fluctuating network costs and varying service demands necessitate a flexible queuing approach. The M/M/C model supports adaptive resource allocation, allowing for efficient tenant request scheduling while minimizing queue delays. Collaboration with a scheduling algorithm further enhances its performance by optimizing server utilization and prioritizing critical requests[92].

Overall, the M/M/C queuing model provides a robust and scalable framework for managing slice admission in a dynamic network environment. It ensures efficient resource utilization, minimizes delays, and adapts to fluctuating network conditions, making it the most suitable choice for this study[88].

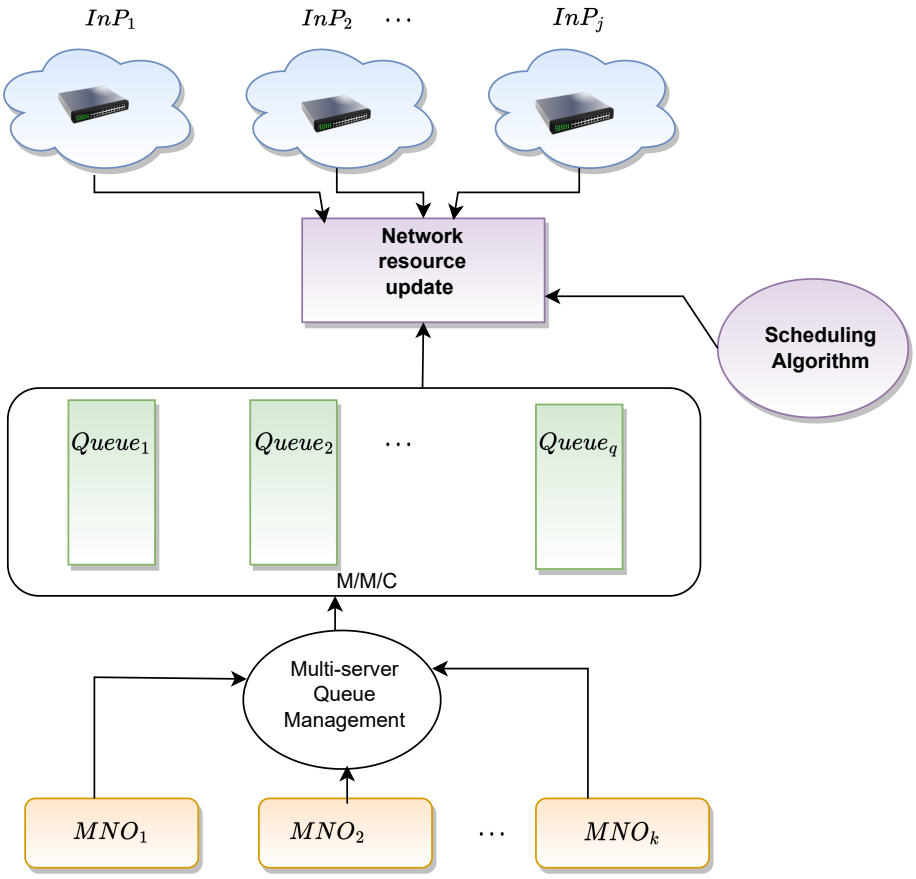

Figure 4.1: Multi-InP, multi-tenant and multi-server queuing and scheduling

The model depicts a multi-server queuing management system with non-homogeneous server-level functionality. The mean task arrival rate on server $i = [1, 2, ..I]$ from queue $j = [1, 2...J]$ is given by a Poisson process $\lambda_{i,j}$, this is because user requests for network slices are computational tasks that typically arrive randomly and independently. This is particularly relevant in network slicing environments, where multiple MNOs submit slice requests to InPs at unpredictable intervals. The randomness of user demand, coupled with the assumption that each task arrival is an independent event, makes the Poisson distribution a natural and mathematically sound choice for modeling task arrivals. Each server can be served from one queue at a time, however a task can transient from one server to another, this is due to heterogeneity in task execution. Hence at full capacity there are q servers. Each server has an exponential service rate given by μ. Let us denote a slice request from an MNO n to an InP m as $x_{m,n}$. Slice queue occupancy can be estimated by an exponential expression given by

$$\rho = \frac{\lambda_{i,j}}{\mu_{m,n}} \tag{4.1}$$

Assume there are $X(t) > 0$ representing slice requests at time t, the probability that there are $l = [0, 1, 2...I]$ requests being processed is $P_l\{X(t) = l\}$ can be modeled from the following transient probability [93].

$$\frac{dP_l(t)}{dt} = P_0(t) - \left(\lambda_{i,j} + \rho + \sum_{n=1}^{N} \mu_{m,n}\right) P_l(t) + \lambda_{i,j} P_{l-1}(t) + \sum_{n=1}^{N} \mu_{m,n} P_{l+1}(t) \tag{4.2}$$

Equation 4.2 represent transient probability of l slice requests at time t. It represents time-dependent probabilities of the system size of a Markovian queuing model with heterogeneous

servers which are governed by the differential-difference equations[93]. It is clear that these equations may not have closed-form solutions but these stochastic differential equations (SDE) may be solved to the closest estimate. These equations represent the probability that a slice may transit from one server to another, considering waiting time and queue size. They are important in this section since they highlight the complexity of managing a multi-queue, multi-server environment that anticipates stochastically arriving slice requests. Therefore, an alternate solution for slice scheduling is required, where an RL method is applied in this chapter.

The linear SDE in 4.2 can be expressed as

$$dP_l(t) = P_0(t) - \left(\lambda_{i,j} + \rho + \sum_{n=1}^{N} \mu_{m,n}\right) P_l(t)dt + \lambda_{i,j} P_{l-1}(t)dt + \sum_{n=1}^{N} \mu_{m,n} P_{l+1}(t)dt + \sigma dW \quad (4.3)$$

Where σ is the variance and dW represents the Brownian motion. Moving forward the probability generating function can be derived from 4.3 as:

$$P_l(t) = A(t) \int_0^t \left[P_0(t) - A(t)^{-1} \left(\lambda_{i,j} + \rho + \sum_{n=1}^{N} \mu_{m,n}\right) P_l(t) + A(t)^{-1} \lambda_{i,j} P_{l-1}(t) + A(t)^{-1} \sum_{n=1}^{N} \mu_{m,n} P_{l+1}(t) \right] d$$
$$+ \sum_{j=0}^{l} \int_0^t A(t)^{-1} dW(t) \quad (4.4)$$

The function $A(t) \in \mathbb{R}^{l\times l}$ represents the fundamental matrix that denotes the solution to the SDE. The solution can now be expressed as:

$$dA(t) = A(t)dt + \sum_{j=1}^{l}(t)A(t)dW_j, \tag{4.5}$$

if the initial condition for the function $A(0)$ is an identity matrix $\mathbf{t} \ \in \mathbb{R}^{l\times l}$ then 4.4 and 4.5 are the solution to 4.2

Proof: By rearranging 4.4 we have:

$$P_l(t) = A(t)\int_0^t \left[P_0(t) - A^{-1}dX(t)\right] \tag{4.6}$$

where

$$dX(t) = \left[\left(\lambda_{i,j} + \rho + \sum_{n=1}^{N}\mu_{m,n}\right)P_l(t) + \lambda_{i,j}P_{l-1}(t) + \sum_{n=1}^{N}\mu_{m,n}P_{l+1}(t)\right] + \sum_{j=0}^{l}\sigma dW_j(t) \tag{4.7}$$

hence

$$P_l(t) = A(t)Z(t),\ Z(t) = \left(P_0(t) - \int_{j=0}^{l} A^{-1}dX(t)\right) \tag{4.8}$$

$$dZ(t) = A^{-1}dX(t) \tag{4.9}$$

Using Itô lemma we obtain

$$\begin{aligned} dP_l(t) &= A(t)dZ(t) + dA(t)Z(t) + \sum_{j=1}^{l} C_j(t)A(t)A^{-1}(t)dt \\ &= dX(t) + A(t)Z(t)dt + \sum_{j=1}^{l} A(t)Z(t)\sigma dW_j(t) \end{aligned} \tag{4.10}$$

therefore

$$dP_l(t) = [P_0(t) + C(t)P_l(t)]dt + \sum_{j=1}^{l} P_l(t) + \sigma dW_j(t) \tag{4.11}$$

where $C(t) \in \mathbb{R}^l$. This ends the proof. It is important to acknowledge that a significant number of stochastic differential equations (SDEs) lack analytical solutions. Consequently, numerical methods are employed to approximate the sample trajectories of the desired solution. The initial assumption at time $t = 0$ is that there are no slice requests in the queue, and there is no server currently fulfilling any request. The probability $P_0(t)$ denotes the likelihood of having no requests in the queue system. Typically, the derivation of the transient probability will contribute to the following KPIs.

1. The probability that there are l requests in a queue.

2. The probability that there are j servers out of l being utilized and $l - j$ servers are not utilized.

3. The chance of having l slices serviced at given time t which encompasses both slices in the wait and those presently being served.

The closed form expression estimation of Equation 4.4 is given by[94]

$$P_l = \rho^l P_0 \tag{4.12}$$

where P_0 estimation takes the form

$$P_0 = \left(\frac{1}{\sum_{j=0}^{l-1} \frac{\rho}{j!} + \frac{\rho}{l!} \cdot \frac{l\mu}{l\mu - \lambda_{i,j}}} \right) \tag{4.13}$$

Equation 4.13 can be modified to fit the KPIs mentioned earlier.

The cost of each slice is originally determined by the duration of time spent in the queue. Longer delays in the queue result in higher penalties, leading to an increase in OPEX and subsequently raising the cost of orchestrating a slice. The primary objective of an InP is to decrease the duration of queue occupancy. In order to emphasize this point, we use the assumption that the cost function C_F exhibits a linear relationship with a slice request $x_{m,n}$ for all admitted slices from all MNOs as:

$$C_F = \beta \sum_{l=0}^{C-1} \frac{\Delta t}{P_l(t)} \forall t = [0, 1, 2...\mathcal{T}] \tag{4.14}$$

The system occupancy duration is given by $\Delta t = \mathcal{T}_{x_{k,n}} - t_{x_{k,n}}$ indicate the difference between the arrival time and exit time. The first objective is therefore to minimize the cost C_F as:

$$\min_{\Delta t} C_F \tag{4.15}$$

subject to

$$\Delta t \leq \omega \tag{4.16}$$

If a slice request's queue time ω exceeds the maximum permitted by the constraint in 4.16, the request is removed from the queue.

In Figure 4.2, the queuing model derived is presented. This M/M/C queuing model is a plausible approach due to its ability to enable resource alignment to specific slice. The multi-queue multi-server system observes slice request arrival at time t and then sent to the queue

with the lowest waiting time Δt. The number of requests in the queue is also expected to determine the choice of queue. It is therefore important to define an optimization procedure for Figure 4.2. From Equation 4.13 the expected waiting time in the queue q is given by

$$W_q = \frac{P_0}{l\mu - \lambda_{i,j}} \tag{4.17}$$

Each queue operates under Poisson arrivals with rate λ and exponential service times with rate μ per server. The expected number of customers in queue is governed by $L_q = \lambda W_q$ (Little's Law) [95]. Henceforth, the choice of queue by a slice is an optimization function given by,

$$\min_q = \alpha W_q + \beta Q_q \tag{4.18}$$

where Q_q is the number of slices in queue W_q represents the expected waiting time in slices. The weight coefficients α and β adjust the priority between queue length and latency-sensitive slices respectively. Each slice request must chose a queue that has the available capacity in that $Q_q \leq l$ and the service and arrival rate must ensure system stability such that $\lambda \leq l\mu$ In a 5G network slicing environment, optimizing queue selection ensures efficient resource allocation across different slices, each catering to unique service requirements such as eMBB, URLLC, and mMTC. A multi-server M/M/C queuing model represents scheduling decisions in network slicing, where multiple slices share a set of servers (computing, network and storage resources). This is highly adaptable network slicing model presented in this study. The arrival of data packets in each slice follows a Poisson process with rate and service times are exponentially distributed with rate per server. The probability that all servers are

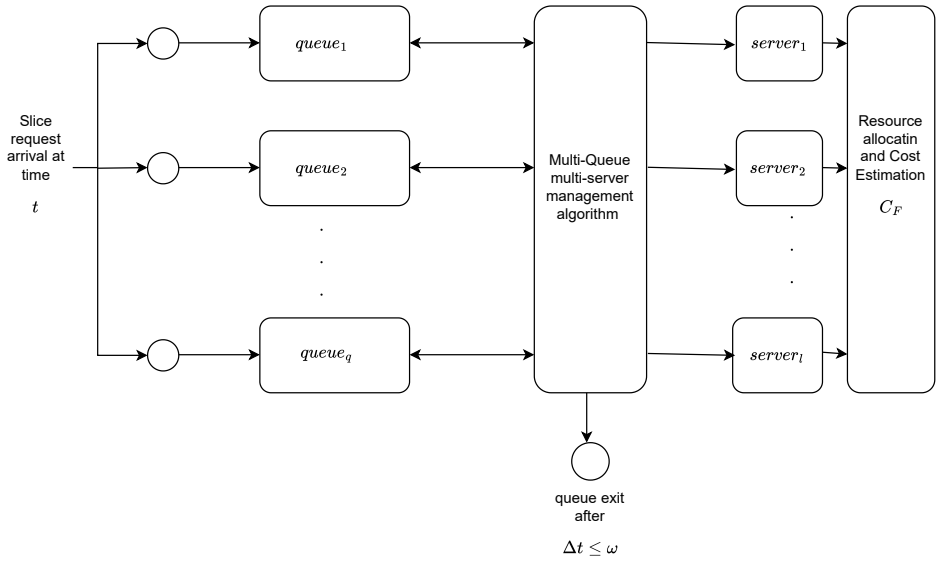

Figure 4.2: Resource queuing and system occupancy in an M/M/C model.

occupied is determined by the Erlang-C formula, influencing whether an incoming packet experiences a delay. The expected queue waiting time W_q depends on system utilization $\rho = \lambda/(C\mu)$ and the probability P_w that an incoming packet must wait for service. Given the dynamic nature of 5G traffic, a queue selection mechanism must minimize both the latency experienced by packets and the congestion level in each slice, ensuring that critical services such as URLLC maintain ultra-low latency while balancing network resources across all slices see Equation 4.18.

Comparatively, a greedy scheduling approach prioritizes the slice with the lowest function value, ensuring immediate resource allocation for the most time-sensitive packets[40]. A DP method considers future traffic conditions to optimize long-term load balancing among slices[16]. RL-based scheduling, continuously adapt to real-time traffic variations, learning optimal queue selection policies based on past network states.

In the next section, we present a model for the resource scheduling scenario. It is important to note that throughout this thesis, the resource categories may be altered, but the core concept of network slice admission control and embedded objective remains consistent.

4.1.1.2 Resource scheduling

The combined prediction (*deterministic and stochastic nature*) of slice servicing by multi-server queuing system in Section 4.1.1 provides a platform for slice scheduling and an enhancement for determining how resources levels will fluctuate temporally per time unit.

When considering the complex issue of scheduling, we propose the use of a well specified set of parameters. In this case, we focus on a group of users, represented as $u = [1, 2...U]$, who are effectively served by a selected slice of class c. Each slice request is placed by a tenant $n \in \mathcal{N}$ to an InP $i \in \mathcal{I}$. In order to determine the amount of spectral resources, which is a crucial measure in the RAN domain, we carefully calculate the required baseband units (BBUs) for each slice, represented by the variable $b \in \mathcal{B}$. It is important to highlight that our study carefully separates the functions associated with the centralised unit (CU) from the operational domain being examined. As a result, our assessment solely concentrates on the distributed unit (DU) due to the real-time events considered.

The slicing of the optical carrier in the core presents a greater challenge, therefore necessitating the inclusion of an Optical Network Unit (ONU) in the access network, which can be sliced with less complexity [96]. The collection of ONUs assigned to a slice of type c, is denoted as $O_c \in \mathcal{O}$, represented by the cost $\mathcal{C}_c^o = \phi^o O_c$. The provision for delay-sensitive applications is addressed by the implementation of a dynamically deployed edge cloud, which

utilises commodity servers equipped with virtual central processing units (vCPUs). Each vCPU denoted by g belongs to a set $\mathcal{G}$ having a cost $\mathcal{C}_c^g = \sum \phi^g g_c$, and to a slice service node (SSN). An SSN servicing a slice type c has a measurable distance in terms of hops denoted as h_c. An InP strives to orchestrate a slice as close as possible to a user.

The cost linked to the quantity of hops may be represented as $\mathcal{C}_c^h = \sum \phi_h h_c \forall h \in H$. It is assumed that slices encounter an equal processing delay on every node along the way. The values ϕ_o, ϕ_g, and ϕ_h represent the mean cost per hour per ONU, vCPU, and hop respectively. Each slice service possesses distinct QoS requirement meaning that a conventional antenna design is not optimally effective in terms of spectrum utilization. To address this, an InP has the flexibility to deploy either a single antenna or multiple antenna setup, depending on the specific QoS of a given slice. This is denoted by the parameter a, which can take on values from the set $a = [1, 2...A]$. It is crucial to note that the value of $\mathcal{U} << A$ in a multiple antenna configuration. The power budget of each antenna arrangement is denoted as $\mathbf{P}_a$, which must satisfy the condition $\mathbf{P}_a < \mathbf{P}_A$, where$\mathbf{P}_A$ represents the power budget that takes into account all antennas within the setup.

The power budget calculation is based on the antenna and BBU power consumption during a slice service. Following [97] we can denote the BBU power as:

$$\mathbf{P}_{bbu} = \sum_{a \in \mathcal{A}} \mathbf{P}_a^{std} + \Delta p \star \mathbf{P}_a^{load} r_b \tag{4.19}$$

where $\mathbf{P}_a^{std}$ determines the standard power per antenna a when the BBU is on no load while $\mathbf{P}_a^{load}$ is the power under load, Δp is the power model rate and r_b denotes the traffic load per BBU $b \in \mathcal{B}$. The cost associated with the BBU power consumption is estimated as

$C_c^{bbu} = \phi_b \mathbf{P}_{bbu}$ where ϕ_b represents the cost per unit time charged per BBU. For simplicity, the dynamic power due to beamforming vector is disregarded. The RB aggregation per slice is given by $\mathcal{O} \cup \mathcal{B} \cup \mathcal{G}$.

The pricing structure employed by the tenant is determined by two factors: the duration of the wait time in the queue and the level of resource required for deploying a slice in each individual slice request. The anticipated fluctuation in the cost of the system is attributed to the dynamic nature of the network within a 24-hour timeframe. This suggests that the price for each individual slice is variable. Tenants possess the capacity to decrease their request for a slice in instances when the cost of the system is increases, hence leading to a diminished payment. From an economic perspective, it can be observed that there exists a positive connection between the cost of a system and the price per slice. This implies that as the system cost rises, there is a corresponding increase in the price per slice, and conversely, as the system cost decreases, the price per slice also decreases. To accomplish this objective, each resource pool is allocated a metric specified by

$$\mathcal{V} = \frac{1}{\mathcal{O}log(\mathcal{O} - \sum_{k=1}^{K} O_k)} + \frac{1}{\mathcal{B}log(\mathcal{B} - \sum_{k=1}^{K} b_k)} + \frac{1}{\mathcal{G}log(\mathcal{G} - \sum_{k=1}^{K} g_k)}. \tag{4.20}$$

This metric represents the normalized value of the resource pool. A comparison is made between this metric and resource fluctuation later in the simulations. The cost of the remaining resource block is directly proportional to this metric whenever a resource is allocated to service a slice of any type c. Based on the aforementioned framework, we can

now proceed to establish the second objective where the slice subscription cost is minimized as follows:

$$C_R = \min_{C^{bbu}_{i,n,c}, C^{o}_{i,n,c}, C^{g}_{i,n,c} C^{h}_{i,n,c}} \sum_{i \in \mathcal{I}} \sum_{n \in \mathcal{N}} \sum_{c \in C} \Gamma_c (C^{bbu}_{i,n,c} + C^{o}_{i,n,c} + C^{g}_{i,n,c} + C^{h}_{i,n,c}) \tag{4.21}$$

subject to:

$$\Gamma_c = [0, 1] \tag{4.22}$$

$$\sum_{n \in \mathcal{N}} \sum_{c \in \mathcal{C}} \Gamma_c x_{c,n} b_{c,n} \leq \mathcal{B}_i \quad \forall i \in \mathcal{I} \tag{4.23}$$

$$\sum_{n \in \mathcal{N}} \sum_{k \in \mathcal{K}} \Gamma_c x_{c,n} O_{c,n} \leq \mathcal{O}_i \quad \forall i \in \mathcal{I} \tag{4.24}$$

$$\sum_{n \in \mathcal{N}} \sum_{c \in \mathcal{C}} \Gamma_c x_{c,n} g_{c,n} \leq \mathcal{G}_i \quad \forall i \in \mathcal{I} \tag{4.25}$$

$$\sum_{n \in \mathcal{N}} \sum_{c \in \mathcal{C}} \Gamma_c x_{c,n} a_{c,n} \leq \mathcal{A}_i \quad \forall i \in \mathcal{I} \tag{4.26}$$

$$\sum_{u \in \mathcal{U}} a_u \leq \mathcal{A} \tag{4.27}$$

Constraint 4.22 is the admission control variable, which can be either one (1) for admitted or zero (0) for rejected slice requests. Constraints 4.23 to 4.26 ensure that available resources, such as BBUs, ONUs, CPUs, and antennas, do not exceed the maximum limit. Constraint 4.27 ensures that the total number of antennas provided to all users in a multi-antenna setup does not exceed the maximum available. The joint objective function may now be written as:

$$C = \xi C_F + (1 - \xi) C_R \tag{4.28}$$

The parameter ξ which is constrained to the interval $(0, 1)$, is utilized to achieve an optimal equilibrium between the trade-off concerning latency and the costs related to resources. This effect arises from the observation that when demand for resources are kept lower, the likelihood of processing slice requests rapidly increases, leading to increased costs associated with queue processing but reduced resource expenses. The lack of convexity in Equation 4.28 requires the implementation of a robust approach to address the problem. In general, it is important to note that there is no global optimum for this problem. Therefore, a feasible approach is to approximate the output, which can be effectively estimated using machine learning techniques. The utilization of deep reinforcement learning has been employed in the form of Deep Q-Network (DQL) to address the problem in question.

4.1.2 Deep Reinforcement Learning

The methodology employed for making judgments entails the development of a systematic technique to effectively manage the allocation of slices. This approach entails prioritizing non-deterministic slice requests as the initial step, followed by evaluating the consequences of queue delay, and subsequently analyzing variations in the resource pool. The objective is to reduce the expenses incurred by the system within a designated time frame Δt. The approach employed in this study is Markovian, and it may be effectively tackled through the utilization of DQL algorithm. DQL is a reinforcement learning approach that does not require prior knowledge of the environment and operates in an off-policy manner. Through the examination of action values, it is possible to formulate a policy that assigns substantial rewards to good actions and lesser rewards or penalties to bad actions. In the DQL framework

Figure 4.3[40], an initial finite buffer called replay memory is established. This buffer is responsible for storing the experiences that have been accumulated by the learning agent. Subsequently, the acquired experiences are employed to update the neural networks. General, DQL adopts twin neural networks namely, the policy and target networks. During the learning process, the policy network estimates the current action values given by $Q(\mathbf{s}, \mathbf{a}, \theta)$, the target is then updated periodically with the weights of the policy network picked randomly from the replay memory, this approach improves stability during learning and prevents what is known as "*dog follows its tail effect*". A soft update employing $\tau \in (0, 1)$ is employed and is chosen as a hyper parameter.

The optimization process of DQL is performed by computing the loss function between the action values of the policy and the target network given by $Q(\mathbf{s}, \mathbf{a}, \theta')$.

The DQL environment holds all the information required for training the agent. The environment holds the state space, the action space, the reward, and a terminal state, since we have opted to employ an episodic state space. We proceed to define the state space, the action space, and the reward functions.

State Space: The set of states required by the training agent are represented as a state space. In this study the state-space contain the transient probabilities of the servers, the resource requirement of a slice and the resource pool status. Each transient probability represents the following states;

- $P_l^{req}(t)$:The probability that there are l slice requests at time t
- $P_l^{que}(t)$:The probability that there are l requests in a queue at time t.

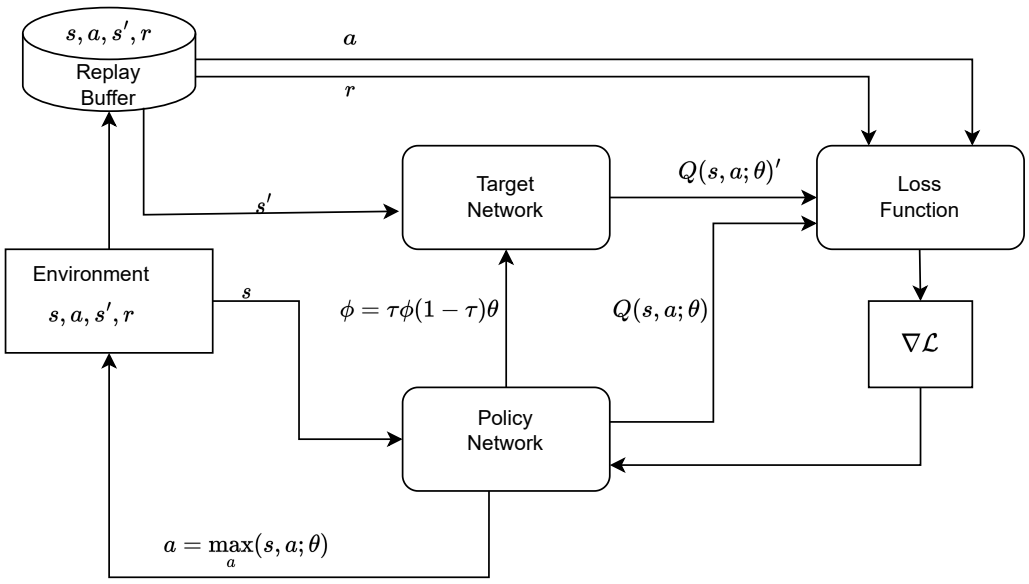

Figure 4.3: DQL architecture depicting the replay buffer, the environment and the twin neural networks

- $P_j^{serv}(t)$: The probability that there are j out of l servers being utilized and $l-j$ servers are not utilized at time t.

- $P_l^{slice}(t)$:The chance of having l slices serviced at given time t.

Additionally, the state space also encompasses the slice request $x_{k,n} = 1$, the real-time network status and queue and server occupancy time δt. We represent these as follows.

$$\mathcal{S}(t) = \{P_l^{que}(t), P_l^{serv}(t), P_l^{slice}(t), O_c(t), b_c(t), g_c(t), \mathcal{O}_c(t), \mathcal{B}_c(t), \mathcal{G}_c(t), \mathbf{P}_{bbu}(t), \Delta t, x_{c,n}(t)\} \tag{4.29}$$

Each of the state value is fed into the training agent through the inputs of fully connected deep neural networks.

Actions and Action Space: The set of actions necessary is intended to depict the agent's response to its interaction with the environment. In this scheduling scheme, the agent undergoes training to execute three actions. 1) The agent predicts when to transfer a slice request to another server considering the current queue and server status. 2) Performs admission control 3) Performs prediction of resource fluctuation. The set of actions are represented as follows.

$$\mathbf{A}(t) = \{\mathbf{a}_1(t), \mathbf{a}_2(t), \mathbf{a}_3(t)\} \tag{4.30}$$

Specifically, the action values represents quantity of resource requirement which must be the server adjustment, number of vCPU required by each server and the predicted resource allocation per user cluster.

Reward function The performance of the agent is heavily influenced by the quality of the reward function. Our objective is to develop a reward function that effectively reinforces actions of the agent, ultimately improving performance. To this end, the cost pertaining to each resource value has been defined, also a metric for network status. We incorporate this parameters into the reward function as follows

$$\mathbf{R}(t) = \left[\frac{\mathbf{A}(t)x_{n,c}(t)}{\mathcal{V}}\right] \frac{P_l^{que}(t)P_l^{serv}(t)P_l^{slice}(t)}{\xi(\beta \sum_{l=0}^{C-1} \frac{\Delta t}{P_l(t)}) + (1-\xi)(C_{i,n,c}^{bbu} + C_{i,n,c}^{o} + C_{i,n,k}^{g} + C_{i,n,c}^{h})} \tag{4.31}$$

The algorithm for implementing admission control based on DQL for resource scheduling and cost prediction is implemented in **Algorithm** 1. The design of DQL requires a finite-size memory buffer known as the replay memory. In this work, we set the buffer size to 1200. Initially, there is no resource allocation; hence, it is assumed that the resource pool is

full. Slice requests have been modeled with a non-deterministic Poissonian arrival rate λ. Each of the neural networks has 12 inputs, with 2 hidden layers corresponding to each set of states. The outputs represent 3 defined action values already mentioned. The training of the agent is episodic and ends after $M = 2000$ episodes where each episode is divided into time steps given by $t = [0, 1, 2...T]$.

Algorithm 1 Deep Q learning for slice scheduling

Initialize the replay buffer size $\mathcal{D}$
Initialize the policy network with random weights θ
Create a copy of the policy network (Target) with weights θ'
for Episode = 1 to M **do**
  Initial starting state from 4.29
  for $t = 0$ to T **do**
    Select action via ϵ or $1 - \epsilon$
    Execute an action according to 4.30
    Observe reward 4.31
    Store experience (s, a, r, s')
    Move to $next\_state$
    if $batch\_size \leq M$ **then**
      Sample batch of size B
      Policy network output = $Q(s, a : \theta)$
      Target network output = $Q(s', a' : \phi)$
      $\mathcal{L}(\theta) = \mathbb{E}[\{Q(s, a : \theta) - (r + \gamma \max_a Q(s'a' : \phi))\}^2]$
      $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$
    else
      Continue
    end if
    $\phi' \leftarrow \tau\phi + (1 - \tau)\theta$
  end for
  $\epsilon \leftarrow \min(\epsilon) + (\{\max(\epsilon) - \min(\epsilon)\} \star \exp(-\zeta \star M))$
end for

Prior to the commencement of the training process, the policy network is duplicated and thereafter utilized as the target network, possessing similar weights θ. A copy of θ is created as ϕ for the target network. During the training phase, the selection of actions is initially exploratory with a probability of $1-\epsilon$. As the agent acquires more experience, this probability

decreases and transitions to what is often referred to as exploitative action selection. After the selection of an action, an emulation of the action is executed using a separate function call that is not integrated inside the algorithm. Each batch consists of 64 elements given by 8 features per row, this corresponds to each state set. The current Q-values of both the policy network $Q(s, a : \theta)$ and target network $Q(s', a' : \phi)$ are utilized to compute the loss Equation 4.32 which is the mean square error temporal difference between the outputs of the two networks.

$$\mathcal{L}(\theta) = \mathbb{E}[\{Q(s', a' : \phi) - (r + \gamma \max_{a} Q(s, a : \theta))\}^2] \tag{4.32}$$

To obtain this difference, the agent require a second pass through the target network to obtain $Q(s', a' : \theta)$. The objective is to minimize the error then optimize the policy network by stochastic gradient descent (SGD) by updating the network weights in the policy networks as

$$\theta' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta) \tag{4.33}$$

The target network weights are updated after a finite duration through a soft updated as;

$$\phi' \leftarrow \tau\phi + (1 - \tau)\phi' \tag{4.34}$$

where τ is a hyperparemeter chosen as $\tau = (0, 1)$. At the end of an episode the exploration rate ϵ is updated.

4.1.2.1 Space and Time Complexity Analysis of Algorithm 1

The algorithm's space complexity is largely determined by the replay buffer, which is initialized to hold up to D experiences. Each experience (consisting of a state, action, reward, and next state) typically occupies constant space, leading to an overall storage requirement of $\mathcal{O}(D)$. In addition to the replay buffer, the algorithm maintains two neural networks (the policy network and its target copy) whose memory footprint depends on the number of parameters P in the network. Since P is fixed and generally much smaller than D in practical settings, the dominant space usage is from the replay buffer, resulting in a total space complexity of $\mathcal{O}(D + P)$, which can be approximated as $\mathcal{O}(D)$ when D is large. The algorithm operates over M episodes, with each episode running for T time steps, leading to a total of $\mathcal{O}(M \times T)$ iterations. At each time step, the actions are chosen and executed in constant time, $\mathcal{O}(1)$, and the environment is updated similarly in $\mathcal{O}(1)$. Once the replay buffer contains enough experiences, a mini-batch of size B is sampled, which is typically an $\mathcal{O}(B)$ operation. The core computation during each update is the forward and backward pass through the neural network, with a time cost proportional to the number of parameters P given by $\mathcal{O}(P)$. Thus, each gradient descent update requires $\mathcal{O}(B \times P)$ time. In the worst case, assuming an update occurs at every time step, the overall time complexity becomes $\mathcal{O}(M \times T \times B \times P)$, capturing the cumulative cost of sampling mini-batches and performing network updates throughout the training process.

4.1.3 Simulation and Results

In the simulation process, the parameters in Table 4.1 were employed, the results are then discussed. The simulations were performed in Python with Scipy.optimize employed to optimize the queuing process. The resource scheduling framework was then carried out as a DRL process with DQL to performed resource prediction. The following hardware was used in the simulation: $Intel(R)\ Core(TM)\ i7-8665U\ CPU\ 1.90GHz,\ 2112\ Mhz, 4Core(s)\ , 8\ Logical\ Processors$. The simulation results were recorded in a Python lists stored in .csv files then later plotted.

The discount rate (0.09) is set low to prioritize short-term rewards over distant future rewards, which is suitable for queuing systems where immediate resource allocation decisions significantly impact performance. The learning rate (0.001) is relatively small to ensure stable updates to the value function, preventing drastic changes that could lead to suboptimal policies. The maximum exploration rate (1) encourages full exploration at the beginning, while the minimum exploration rate (0.001) ensures that the agent gradually shifts toward exploitation as learning progresses. The exploration decay rate (0.01) controls the transition between exploration and exploitation, ensuring that learning stabilizes over time. Finally, 2000 episodes provide sufficient training iterations for the RL model to converge to an optimal policy without excessive computational overhead.

For the system parameters, the time step (1 sec) aligns with real-time queuing dynamics, ensuring that network changes are captured at an appropriate resolution[15]. The resource limits—maximum BBUs (100) [40], CPUs (1000)[98], ONUs (20) [96], and hops (100)—reflect

realistic infrastructure constraints in network slicing, ensuring that the simulation adheres to practical deployment scenarios. The antenna configurations (4–360 MIMO with Massive MIMO support) enable an accurate representation of wireless network performance, influencing how resources are allocated across different slices. These parameters ensure that the RL model effectively learns to manage queuing delays, and optimize resource allocation. The number of hops is a metric employed in this study to estimated the effect of distance to the user and is given a maximum value of 100.

Table 4.1: Simulation parameters

Hyper parameters	
Parameter	Value
Discount rate	0.09
Learning rate	0.001
Maximum exploration rate	1
Minimum exploration rate	0.001
Exploration decay rate	0.01
Number of episodes	2000
Other parameters	
Parameter	Value
Time step Δt	1 sec
Maximum No. BBUs	100
Maximum No. CPUs	1000
Maximum No. ONUs	20
Maximum No. Hops	100
Antennas per Massive MIMO	4-360

In Figure 4.4, the optimization process is presented. The plot presents loss function minimization verses epoch. The agent action starts by exploring the environment which reduces exponentially to the minimum exploration rate ϵ_{min}. The agent is seen to converge when the number of epochs approaches 2500. At this point the agent exploits its experience

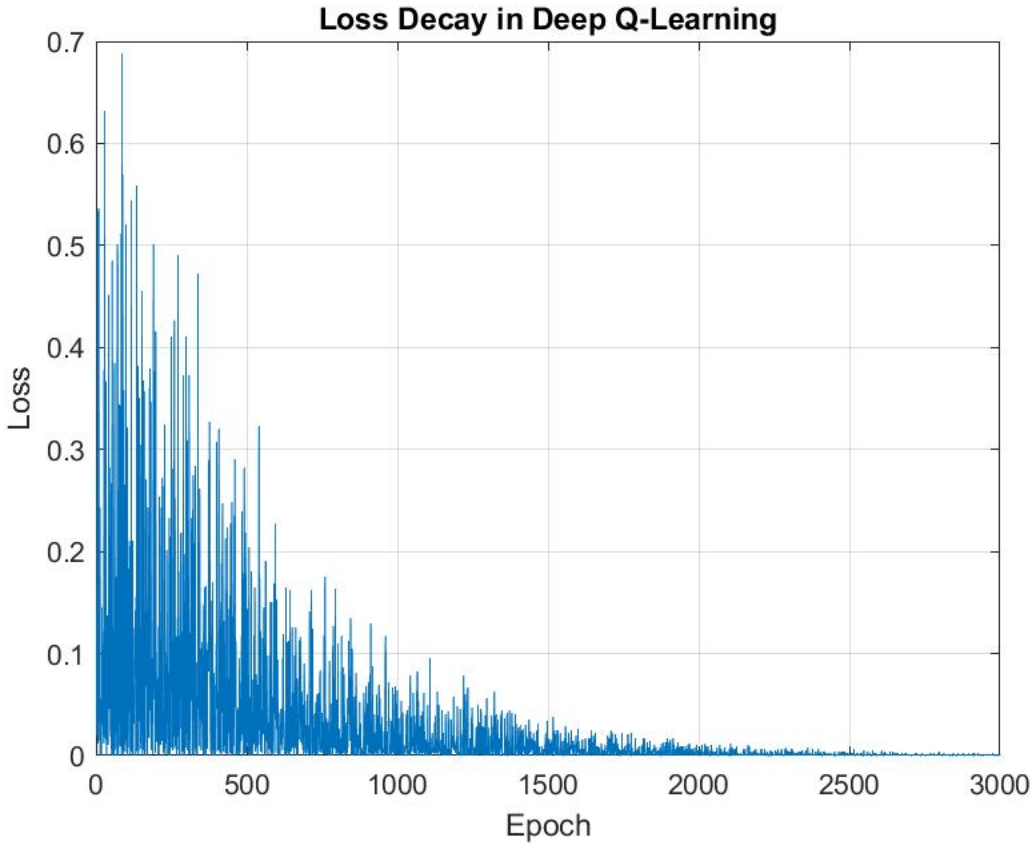

Figure 4.4: DQL optimization through loss minimization over 3000 epochs for resource scheduling

and is said to have converged. Epochs above 3000 showed no further improvement. The loss calculation is obtained through a temporal difference in Equation 4.32.

The efficiency of the queuing process employed in slice admission control depended on the performance of the multi-server multi-queue process.

In Figure 4.5, we illustrate the impact of multi-queue multi-server slice processing on transitioning from a queue or a server. With fewer than 15 slice requests, the relative transition probability of a slice request moving to another server is approximately 0.02. This is highly expected as an acceptable confidence interval of 95% where the mean probability is 0.018 giving a mean error of 0.002. However, as the number of requests increases, the

service volatility also increases. Starting from request 25, the trend shows that a server with a relatively lower service rate μ becomes more volatile, allowing services to be transferred to a server with a higher service rate. For instance, with 30 slice requests in the system, servers 1-4 with service rates of 0.4, 0.6, 0.8, and 1.2 had transition probabilities of 0.5, 0.42, 0.28, and 0.16 respectively.

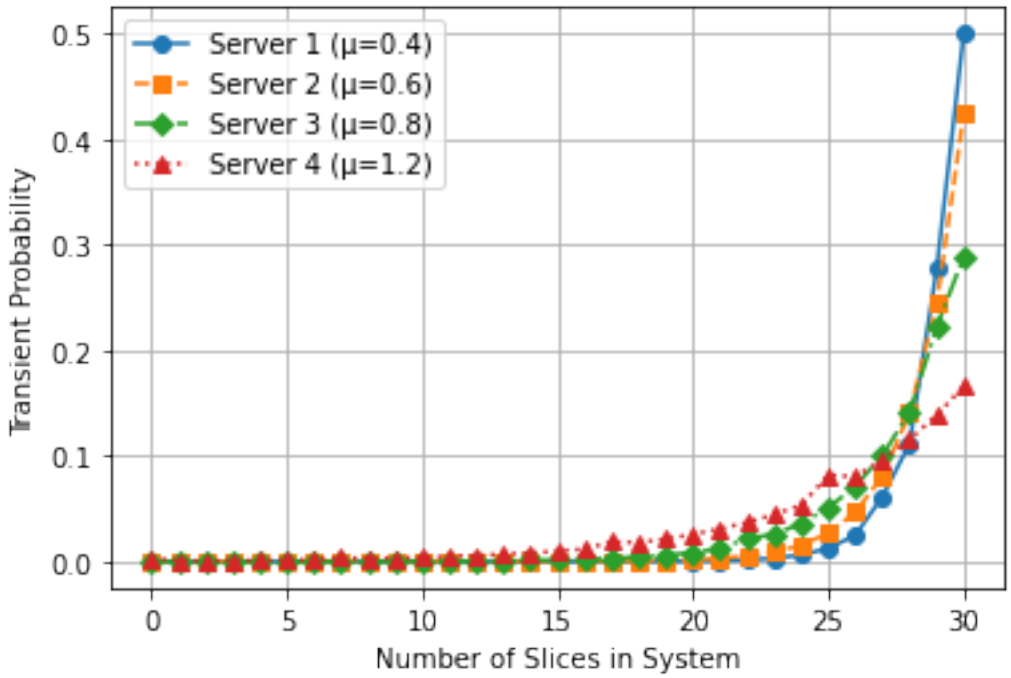

Figure 4.5: Transient Probabilities vs. Number of Slices (Multiple Servers)

Figure 4.6 to 4.10 depict resource prediction over a 24-hour. Initially, Figure 4.6 illustrates radio resource availability and prediction. The compared agents, QL and DQL, were trained on dynamic resource orchestration, where resources were allocated such that services requesting resources they no longer needed relinquished them back to the system.

The actual resource balance depicted by "Available DQL" and "Available QL" followed a similar pattern over the 24-hour period, reflecting a pre-tuned allocation plan. This time

duration is sufficient considering the granularity of slice request arrival of 1 sec. With the goal of determining how the agents would predict the fluctuations, the plots "Forecast DQL" and "Forecast QL" present these predictions. The prediction by DQL had an average error of approximately 20% over a period of 24 hours. However, the prediction improved to less than 15% over the remaining time. Meanwhile, QL remained at an overall accuracy average of over 20% with no further improvement.

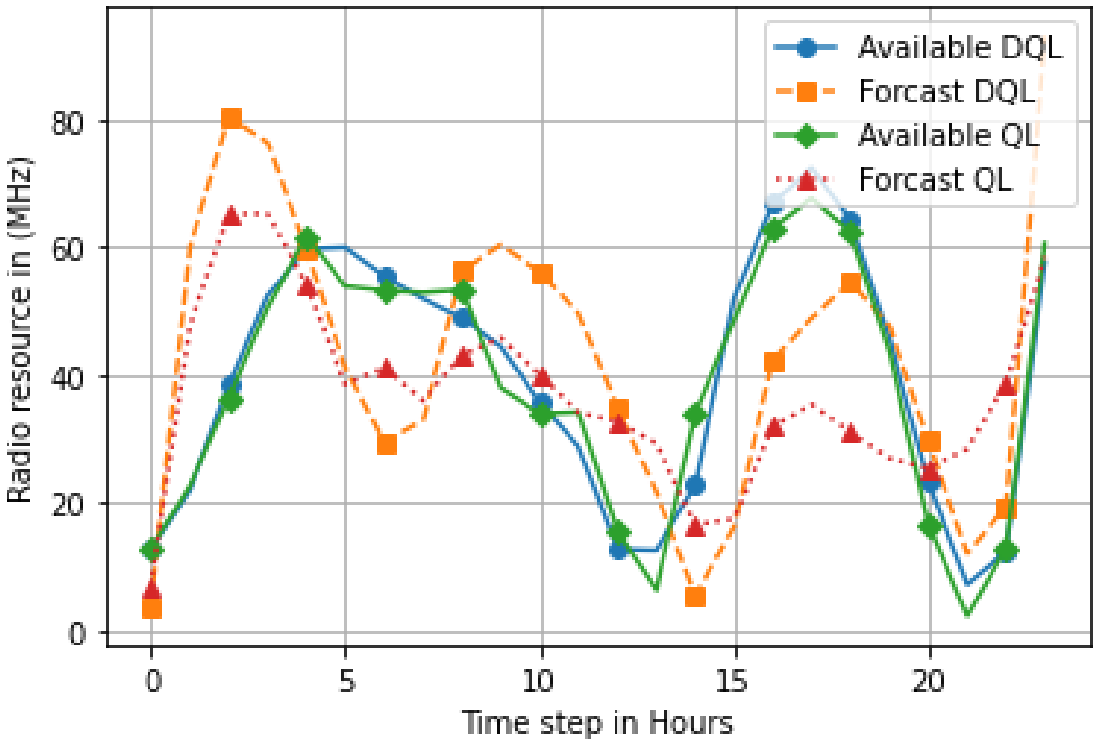

Figure 4.6: Comparison of both actual and predicted radio resource

In Figure 4.7, we present radio resources in RB units. Here, each RB in the BBU considered is a 5G NR FR1 15KHz unit. The fluctuations predicted over 24 hours are plotted against the actual available RBs. Additionally, the agent estimated the cost variation, converted into USD. The cost estimation (USD) is employed in this study as a general indication of cost variation during the set period. The predicted and actual resource values closely matched

each other over the 24-hour period, indicating accurate predictions by the agent. The overall average error between the predicted and actual resources remained at less than 2%. Indeed, the agent's prediction of reduced RB cost for the first 10 hours aligns with typical market behavior. However, considering that users tend to lose interest in network resources when QoS drops, the agent shifted its strategy by increasing the cost when more resources were available and reducing the cost when it anticipated limited resources.

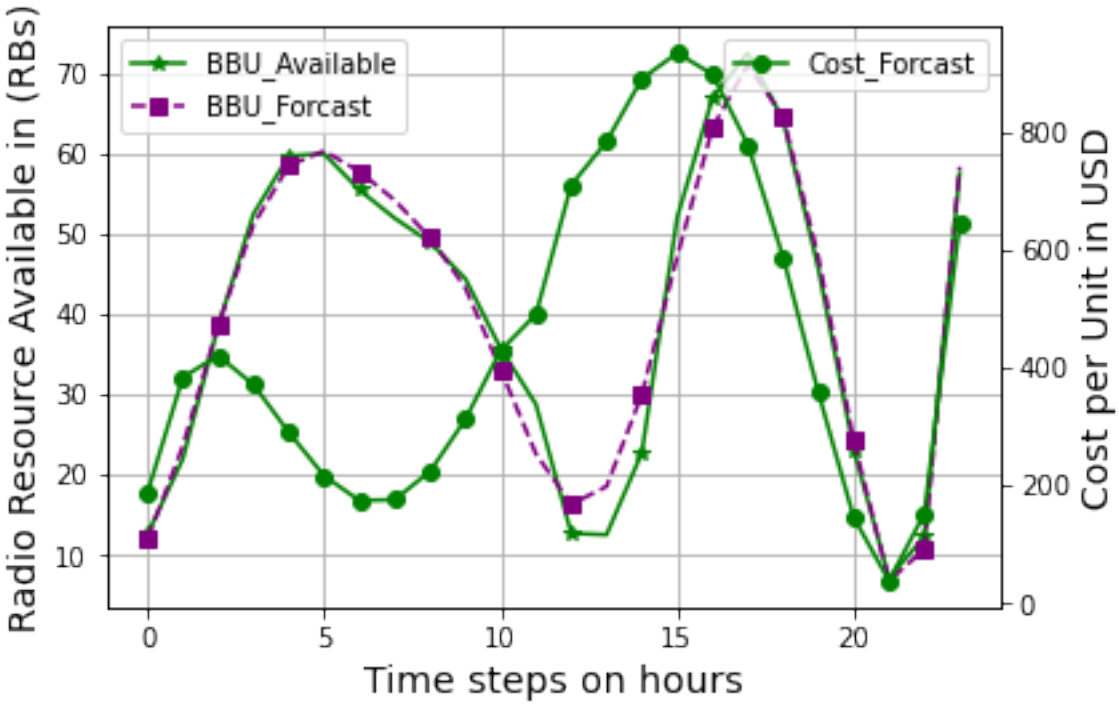

Figure 4.7: Comparison of actual and memory resource against the predicted cost

The memory resource slicing was performed per gigabyte demonstrated in Figure 4.8. The agent's prediction of available resources closely matched the actual values. Furthermore, the cost prediction leveraged the typical network behavior of employing a "high cost for improved QoS" strategy. This prediction criteria is visible throughout the entire time range with minimal deviation between the 5th and 10th hour.

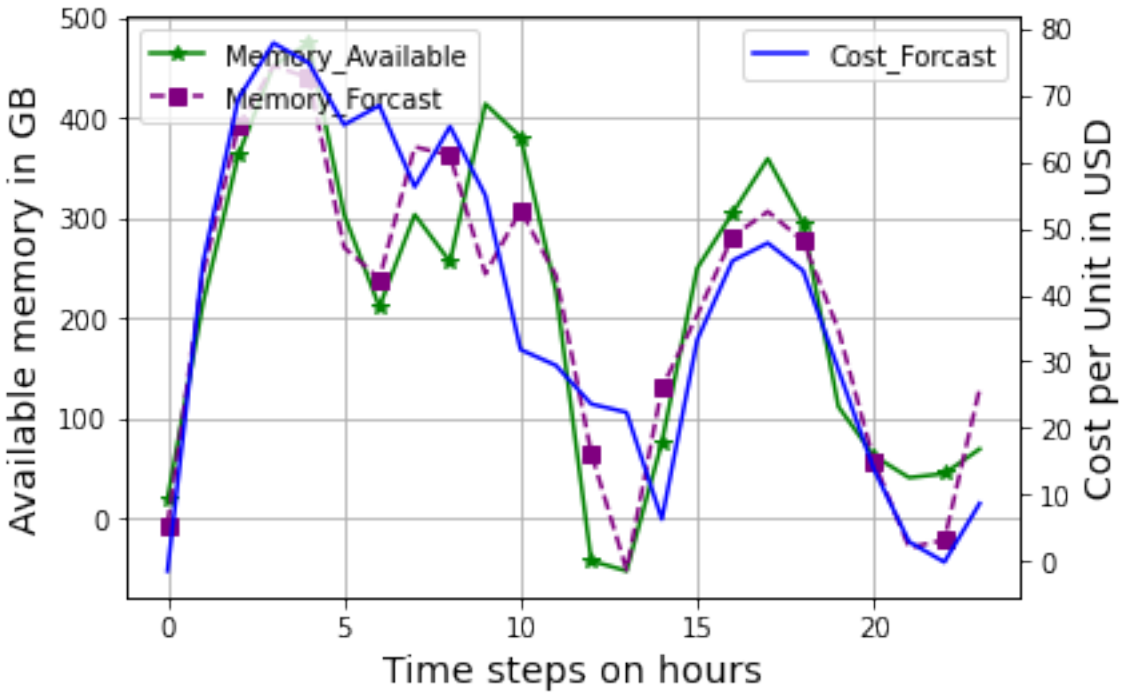

Figure 4.8: Comparison of actual and predicted memory against the predicted cost

Slice task processing requires the availability of logical processors. If the InP can predict the availability pattern, slices can be scheduled accurately. Figure 4.9 presents the vCPU availability pattern over a 24-hour period. The predicted quantity is seen to match the actual values with an average minimal error of less than 7%.. The predicted vCPU cost shows initial increase due to demand at low availability and the agent's lack of experience. The agent improves its estimation and provides a varying cost estimate over 15 hours from the 5th hour. This prediction, similar to previously discussed strategies indicates that improved QoS attracts improved revenue.

The plot in Figure 4.10 illustrates the fluctuation and prediction of ONU resources by the trained agent over a 24-hour period. The predicted and actual resource variations closely match each other, with an approximate temporal error of $\pm 5\%$, consistent with expectations.

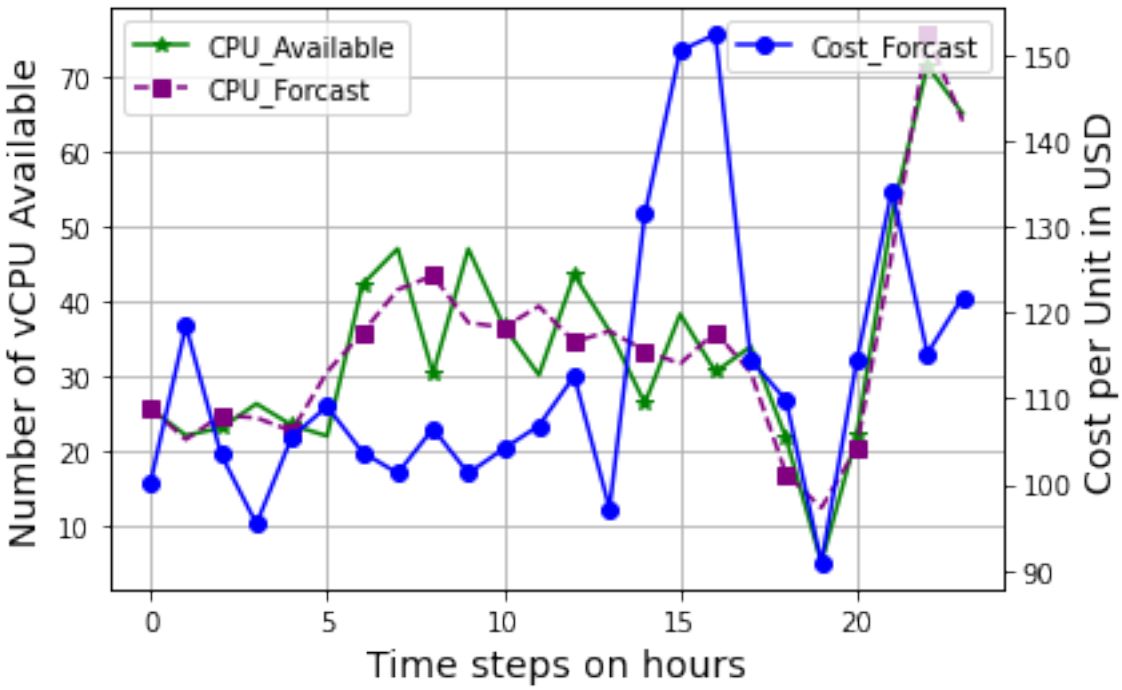

Figure 4.9: Comparison of actual number of vCPU against the predicted cost

Additionally, the plot depicts the expected cost variations for the same period as predicted by the agent.

The plot also illustrates the agent's strategy to balance the trade-off between adopting a market strategy by reducing costs for abundant resources until the 8th hour and after the 15th hour, while increasing resource costs to improve QoS between the 8th and 13th hour. In general, the agent behavior completely complies with the study's intention of evaluating an optimum policy for improving the long term revenue for the InP. The simulation results majorly fall within the confidence interval 95% for the duration of 0 hours to 24 hours in terms of resource forecast and the available.

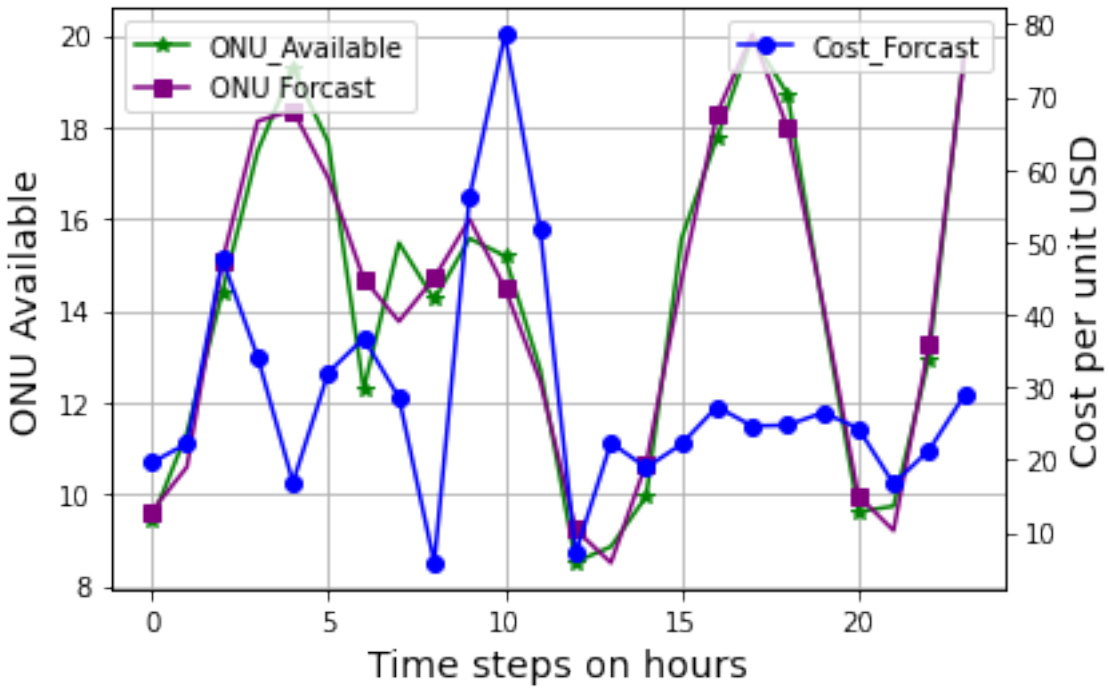

Figure 4.10: Comparison of actual number of ONU against the predicted cost

4.1.4 Chapter Summary

This chapter focuses on resource scheduling for 24 hours. The fundamentals of multi-tenant multi-InP are discussed. We also presented a comprehensive overview of literature describing the state of the art in slice scheduling leveraging machine learning, specifically RL. The queuing model for multi-tenant scenario are presented and discussed.

The transient probabilities are modeled by SDE in Section 4.1.1, and proof provided. The cost model is also provided in this section which aligned to the reward function. In resource scheduling, power allocation for BBU and the overall resource estimation is presented. The objective function for minimizing the overall cost is provided in 4.21

The DQL model for slice scheduling is discussed in Section 4.1.2 provisioning the environment, the replay buffer and both the target and policy networks. In Section 4.1.3 the

simulation parameters are provided and the comparison between actual resource quantity and the predicted values are presented. Similarly, the expected cost estimation for resources are also presented.

Chapter 5

Slice Admission Control Incorporating Resource Auction Game with Reinforcement Learning for Social Welfare Maximization

As presented in the preceding chapter, the establishment of a predictive resource allocation strategy hinges upon the effective implementation of resource scheduling. In order to ascertain the financial impact of resource allocation, it is imperative to develop a model that aligns with the fundamental economic framework of network resource trading. 5G network resources can be traded for revenue by the InP, however this is subjected to aggregated resource constraint.

While an InP has the option to accept or decline slice requests made by a Virtual Network Operator (VNO), they can choose for an economic model that offers the potential for enhanced revenue. One emerging paradigm shift is resource auctioning. This chapter employs Q-learning (QL) specifically designed to address slice admission control using resource auctioning.

5.1 Introduction

The challenge of ensuring equitable resource allocation becomes increasingly complex with the rise of data-centric applications on 5G wireless networks. Furthermore, it is worth noting that the utilization of normal resource costing may not be the most effective approach for maximizing income generation. In this context, it is necessary for an InP to manually examine and assess each resource request, afterwards determining an appropriate price to be paid. However, over time, a slice may unfairly escalate its requirement for data. In order to address this issue, we suggest the implementation of an online auction game that assesses bidding data as a series of requests and subsequently applies reinforcement learning technique to perform admission control.

The exponential increase in the quantity of interconnected devices necessitates autonomous analysis of massive data, specifically bid information, to facilitate intelligent decision-making[99]. Machine learning approach is a tractable technique highly suited to deal with massive and granular data expected during auctioning

When resource auctioning is implemented, it addresses two primary purposes. There are two primary concerns in the context of auctions: the first pertains to the broader economic objective of optimizing the overall welfare of auctioneers and bidders, while the second involves enhancing technical efficiency by means of resource allocation and ensuring fairness[100]. By leveraging QL we prove that a machine learning approach to slice admission control through auction improves social welfare for the seller and fairness in network slice management. In auctioning, a bidding information vector is mapped to RL state attributes.

The learning agent in RL resolves the best bid based on these attributes during the learning process. Various bidding methods, such as the Vickrey-Clarke-Groves (VCG) approach, exhibit bidders who possess dominating strategies that are rooted in incentive compatibility. Our approach involves aligning our model with universal models, while simultaneously accommodating bidders' individual preferences. We propose a non-greedy policy that aims to maximize utility, taking into account resource constraints and fairness. In a multi-tenant environment, the InP is faced with the challenge of managing various requests for shared resources, resulting in resource competition. The practice of the InP opting to consistently sell to a tenant/VNO based on their commitment to generate substantial positive revenue can be characterized as a greedy approach, which is deemed unfair.

Intelligent resource auctioning allows InPs to evaluate tenants and sell resource to the most deserving bidders. This is why governments around the world use auctioning to allocate spectrum licenses to the most deserving telecommunication companies[101][102][103]. The scarcity of network resources implicates that, the slice provider can only admit a qualified bidder and allocate resources from the pool. Moreover, a tenant may define a slice requiring multiple subsets from various classifications. An InP must allocate and maintain such slices for the duration of the request, this implies that a continuous auctioning process must be maintained.

Significantly, the implementation of 5G network slicing allows for the allocation of a slice tenant into distinct categories. However, it is worth noting that at present, the majority of slice requests are generic in nature and lack specific classification. As a result, the feasibility of optimization and admission methods that rely on slice categorization is diminished. Hence,

in order to facilitate the implementation of a dynamic tenant admission and slice allocation system that is based on slice auctioning, it is imperative to develop a fast and intelligent method for resource auction[36]. Further, the data presented by a bidder conveys to the seller the bidder's valuation and preference. These preferences are always changing due diverse use cases. Moreover, the complexity of the auctioning process is exacerbated by the exponential growth in the number of interconnected devices. Consequently, there is a need for autonomous analysis of a significant volume of data that encompasses bidding information. This analysis is crucial for enabling intelligent decision-making[99] and eventually the winning bid is resolved by its ability to meet the slice admission objective of optimizing economic utility.

5.1.1 Overview of Network Resource Auctioning

In recent times, scholars have put forth theoretical frameworks pertaining to the management of resources and auctioning models in the context of 5G technology. The study presented in [104] introduces an online auction-based approach called online auction-based resource allocation for Network Slicing (ORANS). The concept offers a framework for the allocation of online resources, leveraging identification of the winning bid and the subsequent payment process. The algorithm suggested operated on the assumption of a bidding mechanism in which each offer is designated for a particular slice. However, it should be noted that this assumption may not lead to the most effective discovery of revenue, as slices are now not limited to the conventional categories of uRLLC, eMBB and mMTC.Moreover, a thorough assessment of the impact of auctioning on revenue optimization might be more definitive

when supported by a significant amount of bidding data, a methodology that has been less extensively investigated by the authors cited..

In their study, the authors [100] introduced a two-tier cloud radio access network (C-RAN) resource auction approach. This technique consisted of two auctions: one conducted between the end user and the virtual network operator, and another between the virtual network operator and the C-RAN operator. This approach aimed to optimize resource allocation in C-RAN. Although the inclusion of a vertical auction in this method is an interesting methodology, it presented challenges in terms of coordination and ensuring its effectiveness. Further, the authors have only focused on two specific features of the auction, namely the resource block and the amount paid by the winning bidder. The consideration of critical features, such as the length of resource allocation, is vital and should not be overlooked.

In their respective studies, Liu et al. [105] and Zhu et al. [106]] examined a resource allocation and pricing mechanism for 5G slices in an online combinatorial auction setting. They employed an exclusive OR (XOR) mechanism to optimize social welfare and assumed equal value for each resource, this assumption does not hold in practical scenarios where 5G slicing is implemented. A closely related work is presented in [107]. The utilization of a RL algorithm enables the independent assessment and valuation of resources. This enables virtual operators with the highest merit to secure bids. Furthermore it is essential, for providers of slices to maintain fairness when selecting the winning bidder. To achieve this goal it is crucial for the RL based algorithm to adhere to a strategy, which can be achieved through a learning process.

The concept of a 5G network slice refers to a collection of diverse virtual resources. However, it is inadequate to solely focus on the allocation of spectral resources, as demonstrated by the studies conducted by Tadayon et al. in [108] and Zhao et al. in [109].

Moreover the VCG[110] [111] mechanism, extensively used for addressing resource auction problems relies on utilitarian welfare functions. Each bidding participant provides their value function. The optimal choice is determined based on that. The payment made by each agent is determined by the collective valuation of the remaining agents. Although this methodology retains a certain level of truthfulness, it may fall short in identifying the bidder with the highest merit. For instance, participants in an auction may engage in deceptive practices with the intention of altering the final result, which may negatively impact other participants. Tied to auctioning 5G network resources, the authors in [112] and [113] presented auctioning mechanisms but never attempted to deal with the intricate problem of fair resource allocation.

Although auctioning mechanisms are commonly used for fair network resource allocation a due and to model interactions between InPs and VNOs, as well as optimization, analysis, and revenue maximization, the potential of machine learning and intelligent techniques in this context has not been thoroughly investigated. We strive to leverage resource auctioning to enable slice admission by employing state-of-the-art reinforcement learning.

The complexity of achieving fairness in resource management for 5G is further compounded by the presence of multi-tenant multi-resource based slice admission control. To address this challenge, we propose a solution that involves constructing a state-space comprising of

resource demand and auction constraints. Additionally, we introduce a rewarding objective that is specifically designed to optimize revenue maximization.

This chapter aims to achieve the following:

1. Model an RL environment containing bidding information considering tenant QoS demands.

2. Employ Q-learning to perform slice admission control

3. Presents a positive economic utility maximization and an admission control mechanism based on RL outcome.

4. Presents an analysis from simulations to prove that RL based technique can improve positive utility while maintaining fairness in slice admission control.

5.2 System Model

As part of the work in [77], this chapter considers a set of users who are associated to a VNO. In a multi-slice scenario a VNO may subscribe to one or more slices of similar or different classes. A VNO is considered to place a slice request as a bid information through a bidding controller owned by an InP. Users with similar latency requirements are virtually zoned together analogous to [104], this allows the VNO to uniquely define a slice request and the bid data as shown in Figure 5.1.

Each bidding information describes a VNO's preference. These preferences are unique to a slice request and a VNO. Network functions virtualization (NFV) is utilized to orchestrate a

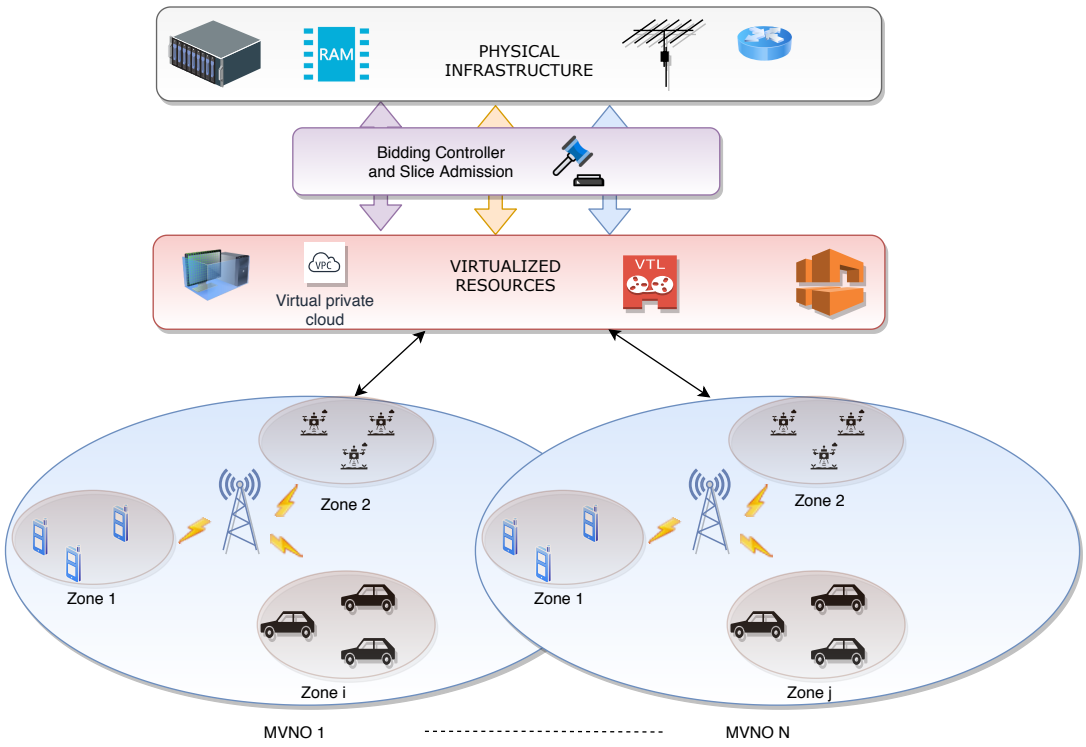

Figure 5.1: Network Model with an Auction Controller

slice, which is a collection of virtual network functions (VNFs). The resource pool consists of the non-shareable radio RBs in terms of BBU which is measurable in Hz, computing resources in CPU cycles and storage resources in GB. A state of the art scheme is adopt to enable resource virtualization which are orchestrated as a chain of VNFs. A multi-server scheduling scheme implemented in Chapter 4 is adopted to manage slice arrival prior to slice auctioning.

5.2.1 Network Model and Architecture

The system model for slice scheduling was introduced in Chapter 4. It is important to acknowledge that the thesis consistently maintains the shared target of resource management and admission control. However, several scenarios may be delineated to illustrate the

specific goals of individual chapters and the overarching objectives of the thesis as a whole. Furthermore, we shall uphold the establishment of slice models that encompass radio, computer resources, and any closely associated variations.

In this consideration, a set of users denoted by $U = \{1, 2, 3..., |U|\}$ are served by a base station $v \in V$ is given by $u_v \in U$. A VNO provides a zoning $i = [1, 2, 3..I]$ criterion for users with similar latency requirement ω. Therefore the QoS requirement for a user u belonging to zone i under the service of a base station v is give by $\omega^i_{u_v}$. The power allocation to a user in zone i is given $p^i_{u_v}$, similarly each user is allocated a portion of a radio resource $c_u \subset \mathcal{C}$, where $\mathcal{C}$ is the total available bandwidth assigned to a slice class . Each zone is subsequently allocation $\mathcal{B}_i \in \mathcal{C}$. We donate a slice class as $n = (1, 2, 3..., N)$ specific to a set of users within a zone known as n-Class (nC) slice , hence we can have nCmMTC, nCeMBB and nCuRLLC as $nCm, nCe,$ and nCu slice categories respectively such that $nCm \cap nCe \cap nCu$. It is assumed that the channel state in between users served by a base station is known hence the channel coefficient is given by h_v[106],[114], hence the received signal is given by;

$$x^i_{u_v} = \sqrt{p^i_{u_v}} h_v q_v + \sum_{j=1, j\neq v}^{J} \sqrt{p^i_{u_j}} h_j, q_j + n \tag{5.1}$$

where q_v is the transmitted signal from a base station and n is the additive white Gaussian noise(AWGN). The expected signal strength to maintain a slice for the users in being served by a slice of type n is given by

$$X_n = \mathbb{E}[\sum_{u}^{U} x^u_{v,i}] \forall v \in V. \tag{5.2}$$

Realistically, the signal from another base station would cause interference to a user on another base station. Hence the signal to interference-plus noise ratio is given by

$$SINR_u = \frac{p^i_{u_v} h_v}{\sum_{j=1, j\neq v}^{J} p^i_{u_j} |h_j|^2 + \sigma^2} \tag{5.3}$$

where σ^2 is the noise power. We now define the total ergodic downlink throughput per slice as

$$r_n = \sum_{u \in \mathcal{U}} \mathcal{B}_i \log_2(1 + SINR_u) \tag{5.4}$$

During the bidding process, the bid information must be populated with the slice resource requirements, namely; data throughput, latency requirement, slice lease duration, and the computing requirement, we continue the model description by illustrating the computing resource model.

5.2.1.1 Computing Resource Model

The computing resource required per slice class incorporates the virtualized CPU cores and random access memory (RAM). In the zoning criterion employed, users with similar computing requirement are zoned together. Each user in a zone is associated to a portion $z^u \in Z$ of a finite size memory and RAM $g^u \in G$. The memory requirement can only be orchestrated if there is a hit. We denote the probability that a memory and RAM of size z and g respectively are placed as part of slice request as;

$$\mathcal{P}_{z,g} = \frac{1/(z \cdot g)^{\varsigma}}{\sum_u^U u^{\varsigma}} \tag{5.5}$$

similarly the probability that the requested memory and RAM will be matched is denoted by $h_{z,g}$ hence from 5.5 the hit ratio is given by

$$\mathcal{H} = \mathcal{P}_{z,g} h_{z,g} \tag{5.6}$$

where $\varsigma \in (0, 1)$ is the coefficient that follows Zipf distribution [115]. Obtaining the exact value of $h_{z,g}$ is a complicated task, hence we obtain is estimate by adopting the admission control policy. To improve efficiency, an InP must refresh the memory unit or dynamically reassign RAM once a slice exits.

The computing resource hit indicator $d = 1$ for resource match while $d = 0$ indicates no computing resource is available or not enough; hence $d \in \{0, 1\}$ is a two state Markov model [116] given by.

$$d_t = \begin{bmatrix} d_{t=0} & d_{t=1} & d_{t=2} & ... & d_{T-1} \end{bmatrix} \tag{5.7}$$

here t denotes the instantaneous time for which the computing resource is required. The total memory and RAM requirement per slice is therefore denoted by

$$\rho_{mem,i,n} = \sum_{u=1}^{U} \sum_{t=0}^{T-1} d_t z_t^u \tag{5.8}$$

$$\rho_{ram,i,n} = \mathcal{H} \sum_{u=0}^{U} \sum_{t=0}^{T-1} d_t g_t^u \tag{5.9}$$

The computational model comprises of the size of task and the number of CPU core required per slice . We denote a CPU task as $\Omega\{o_t, q_t\}$, where o_t is the instantaneous size of the task and q_t is the task size per CPU. The computational capacity given by CPU cycles per second

assigned to a slice n is therefore given by β_n. By modifying model in [116] the numbers of CPUs per slice is given by

$$f_n = U \sum_{t=0}^{T-1} \frac{\Omega(o_t)}{\beta_n \Omega(q_t)} \times d_t \tag{5.10}$$

In the next section we outline the auctioning process employed for multi-tenant leveraging granular bidding information while aiming to improve fairness.

5.2.2 Online Slicing Auction Problem

In the conventional auctioning framework, bidders cannot express their preferences, resulting in a significant degree of inefficiency. In the proposed model, a VNO is granted the explicit permission to possess a bidding preference, as the underlying resource demand is determined by use cases of different users. However, in order to acquire a network slice, the tenant's slice request must go through a competitive bidding procedure. Every VNO is associated with a set of users linked to a particular slice. The VNO resolves the issue of slice demand by purchasing a resource block from the InP specifically for user centric services launched at the access end. In order to acquire a slice, the VNO engages in a resource auctioning game that is overseen by an agent known as an auctioneer. To efficiently carryout the auctioning process, a new double sided auctioning game is employed. Conveniently, both the VNO and InP send there valuation of a slice to a central agent as $\mathfrak{V}_{bid}$ and $\mathfrak{V}_{ask}$ respectively. We denote $\mathfrak{V}^+$ as the positive utility achievable when $\mathfrak{V}_{bid} > \mathfrak{V}_{ask}$ hence:

$$\mathfrak{U}^+ = \mathfrak{V}_{bid} - \mathfrak{V}_{ask} \tag{5.11}$$

During the bidding process, a record containing both bid information and a slice valuation is sent by a VNO. This is defined as a tuple given by:

$$\phi_{bid,n} = (r_n, \rho_{mem,i,n}, \rho_{ram,i,n}, f_n, T_n, \mathfrak{V}_{bid,n}, l_n) \tag{5.12}$$

where T_n is the slice duration and $l_n = \frac{1}{\mathcal{U}} \sum_{u \in \mathcal{U}} \omega_{u_v}^i$ is the average latency requirement per slice. The profitability of the auctioning game is achieved by maximizing the positive utility given by equation (5.13)

$$\max_{n \in \mathcal{N}} \mathfrak{U}_n^+ = \sum_{i=1}^{I} \sum_{n=1}^{\mathcal{N}} (\mathfrak{V}_{bid,i,n} - \mathfrak{V}_{ask,i,n}) \tag{5.13}$$

and the selling price given by::

$$\eta_n = \sum_{i=1}^{I} (\sum_{n=1}^{N} \eta_{opt,i,n} - \sum_{\jmath \neq n}^{\mathbb{J}} \eta_\jmath) \tag{5.14}$$

Where $\eta_\jmath$ represents other valuations as if the winning bidder was absent and

$$\eta_{opt} = \sum_{n \in N} \frac{\mathfrak{U}_n^+ T_n}{l_i (r_{tot,i,n} \delta_{data} + \rho_{mem,i,n} \delta_{mem} + \rho_{ram,i,n} \delta_{ram} + f_{i,n} \delta_{cpu})} \forall i \in I \tag{5.15}$$

the variables δ_{data}, δ_{mem},δ_{ram},δ_{cpu} represents the unit prices per data rate, memory, RAM and CPU respectively.

Problems 5.12, 5.13 and 5.15 are subject to:

$$C1 : \omega_{u_v}^i < \omega_{th} \tag{5.16}$$

$$C2 : d_t \in \{0, 1\} \tag{5.17}$$

$$C3 : \mathfrak{U}^{+} > 0 \tag{5.18}$$

$$C4 : \sum_{n \in N} [\rho_{mem,i,n} + \rho_{ram,i,n}] < \rho_{tot} : \quad \forall i \in \mathcal{I} \tag{5.19}$$

Equation 5.14 conforms to the VCG auctioning model in[117]. Constraint $C1$ represent the QoS threshold required by each slice. The control variable for slice admission in the cloud network for computing resources is $C2$. The constraint $C3$ indicates that the social welfare must remain positive for the maximization of profit. The overall computing memory resources allocation per slice is denote by constraint $C4$.

The auctioning process outcome combined with resource constraints are limiting parameters that periodically control slice admission process leading to a dominant strategy. In this work, the dominant strategy is achieved when an action with a positive utility coupled with a resource request that incur improved revenue for short term contracts are admitted. In this regard we define a parameter known as the *value indicator* V_I given by $\mathcal{V}_{\mathcal{I}} = \frac{Total\ slice\ cost}{Total\ slice\ duration}$ which is applied so that if a bidder pays more for a short term slice then VI is improved. When a resource block or slice is sold at selling price $\{\eta_{i,n} = \bar{\eta}\}$ [117], under dominant strategy then the agent is on track to reach optimality. where $(\bar{\eta}_n \geq \mathfrak{V}_{ask})$ is known as the hammer price.

5.2.2.1 Auction controller and Bid Price Update under Resource Constraint and Peak Load Time

Algorithm 2 below illustrates the auctioning processes by the auction controller. A bid can be accepted only if it generates a positive utility $\mathfrak{U}^{+}$ or when it is withing the NE criterion.

Algorithm 2 Auctioning Process by the Auction Controller

1: **Result**: Successful bid
2: **Input**:Observe bids($\mathfrak{V}_{bid}$ and $\mathfrak{V}_{ask}$)
3: **if** $\mathfrak{V}_{bid} > \mathfrak{V}_{ask}$ **then**
4: $\mathfrak{U}^{+} = \mathfrak{V}_{bid} - \mathfrak{V}_{ask}$
5: **else**
6: Calculate ($\mu = \log\left(\frac{\mathfrak{V}_{bid}}{l_n}\right)$)5.20
7: **end if**
8: **if** $\mu < Threshold$ **then**
9: Flag bid as unfair
10: **else**
11: Obtain the next slice request and move to step 2
12: **end if**

The resource bid update procedure by the InP depends on resource availability and unsuccessful bid due to high prices and bid requests during resource scarcity due to peak work load time. To tackle this problem, the following assumptions are initially made.

- An auction is truthful only when a bidder reveals true valuation.
- The InP adjusts prices dynamically based on resource availability and demand parameters.

Algorithm 3 describes the bid price update procedure. The InP monitors available resources periodically and then aligns each bid to current bid price. Resource constraints described in section 5.2.2 can not be violated in this bid update procedure. The InP sets a competitive

current bid price for auctioning. This bid price cannot be revealed to the a VNO. A minimum step to increase bid is also set by the InP as well as the maximum bid limit. If a bid is successful then the InP has to maintain the bid price or slightly decrease it to remain competitive. Conversely, if there is an unsuccessful bid and resources are available then the current bid has to be updated. Further, if available resources are less than the threshold and demand level becomes high then the current be is updated according to line 17 in the algorithm. If peak load persists the the updated bid remains the maximum bid limit.

Algorithm 3 Bid Price Update under resource constraint and peack overload time

1: **Result**:New_Bid_Price
2: **Input**:Available_Resource
3: **Input**:Number_of_Incoming_Request
4: **Input**:Current_Bid_Price
5: **Input**:Minimum_Acceptable_Resource_Level
6: **Input**:Flag_Indicating_Workload_Periods
7: **Initialize**:Initial_Resource_Value
8: **Initialize**:Minimum_Step_To_Increase_Bid
9: **Initialize**:Max_Bid_Limit
10: **if** Bid_Succesful **then**
11: Maintain bid or slightly decrease bid to remain competitive.
12: **else**
13: Bid Unsuccessful and Resource Available > Threshold_Resource
14: Update_bid= Current_Bid + Min_Bid_Increment
15: **end if**
16: **if** Resource_Available $\leq$ Threshold_Resource and Demand_Level is High **then**
17: Update_Bid = Current_Bid + (2*Min_Bid_Increment)
18: **else**
19: Peak_Load Active
20: Updated_Bid = Max_Bid_Limit
21: **end if**
22: Repeat

5.2.3 Incentive Compatibility and The Nash Equilibrium(NE)

The competitive approach of network slice acquisition employed by an InP is likely to necessitate untrue preference declaration by a VNO in order to win a bid, this is an unfair strategy. As a result, a valuable bidder is likely be bypassed. If a resource buyer changes its valuation $\mathfrak{V}_{bid}$ with the intention of winning, our approach enforces fairness by providing the Nash equilibrium (NE) which goes as follows. The QoS indicator l_n is used as a weight balancing controller between the latency requirement of a slice and the resource valuation. For instance, the higher the valuation the smaller the latency requirement.

We illustrate this in equation (5.20)

$$\mu = \log\left(\frac{\mathfrak{V}_{bid}}{l_n}\right) \tag{5.20}$$

where l_n is the average slice latency requirement. It should be noted that, any bidder cannot alter the bidding information without affecting the QoS index requirement and the InP's response. To explain this concept further, consider a bidder who wishes to maintain the same QoS index but unfairly increases the valuation in order to win a bid. Equation 5.20 will not be satisfied without reducing the latency requirement, furthermore, such a move is likely to violet the SLA between the VNO and the InP. Finally the fairness index [117] [118] is defined by equation (5.21) as

$$\mathbb{D}(\mathfrak{U}) = \frac{\left(\sum_{n=1}^{N} \mathfrak{U}_n\right)^2}{N(\sum_{n=1}^{N} \mathfrak{U}_n^2)} \tag{5.21}$$

5.2.4 Reinforcement Learning: State, Action and Reward Function

As already postulated, RL relies on a Markov decision process (MDP) formed as tuple $\{\boldsymbol{s},\ \boldsymbol{a},\ \boldsymbol{r}\ ,\ \boldsymbol{s'}\}$ where $\boldsymbol{s}$ is the observed state, $\boldsymbol{a}$ is the action taken, $\boldsymbol{r}$ is the reward obtained and $\boldsymbol{s'}$ is the next state under state transition model [46]. We revise the definition of the state,action and reward values to fit the context in this section. The environment comprises states, possible actions and the reward value. Since Q-leaning is employed, the algorithm is an off-policy and state transition is dependent on state arrival and not transition probability.

5.2.4.1 System State

The system state is modeled from the bidding information provided by each tenant n at discrete time $t = \{0, 1, 2, ...T-1\}$ which is given by a tuple in equation (6.26)

$$\boldsymbol{s}(t) = \{\phi_{bid,n}(t), \mathfrak{V}_{ask,n}(t), \Gamma_n\} \tag{5.22}$$

where $\phi_{bid,i,n}(t), \mathfrak{V}_{ask,i,n}(t), \Gamma_n$ are the bidding information, the InP's asking price and the total remaining resource units at time t respectively. Between time $t = 0$ to $T-1$ the InP reads $\mathcal{M}$ requests from a holding queue. We assume that each bidding information forms a part of a finite state space $\boldsymbol{s} \in \boldsymbol{S}$. The dynamic state space $\boldsymbol{S}$ changes during each iteration as it must be updated to reflect the total available resources. Similarly, the ability to meet the QoS requirement after each admission must also be updated. The InP cannot admit a slice when resources are depleted hence the agent must learn this strategy.

5.2.4.2 Action Space

The action space is modeled as a binary variable $\mathbf{a}(t) = \{\mathbf{a}(0), \mathbf{a}(1) \ldots \mathbf{a}(T-1)\}$ such that $\boldsymbol{a}(t) \in \{0, 1\}$ represents an action at time t where 1 indicates a successful resource auction and an eventual slice admission and 0 otherwise. After every action $\mathbf{a}(t)$ the agent obtains the corresponding reward $\mathbf{r}(t)$ which is used to update the Q-values in the Q-table [46].

5.2.4.3 Reward

Intuitively, RL learns by maximize the reward through the evaluation of each action. The reward function is modeled to optimize action selection (slice admission control) while considering all constraints. A reward $\mathbf{r}(t) = \chi$ is maximum when right action is taken and minimum otherwise.

In order to assure complete slice isolation, the InP must guarantee all resources required to instantiate a slice. Let $\lambda = 1$ denote full resource availability and successful auction, 0 otherwise. The reward function employed can therefore given by equation (5.23 and 5.24)

$$\mathbf{r} = \mathbf{a}(t) log(\eta_{opt,i}) \mu V_I : \quad \text{if } \lambda = 1 \tag{5.23}$$

else

$$\mathbf{r} = (1 - \mathbf{a}(t)) log(\eta_{opt,i}) \mu V_I : \quad \text{if } \lambda = 0 \tag{5.24}$$

5.2.5 Q-Learning Model

The objective of Q-learning in **Algorithm 3** is to build a map of optimal actions and their corresponding states $\boldsymbol{s} \in S$. This however, is not complete until this map is retrieved. The retrieval of this mapping is known as policy retrieval. Once the policy retrieval is complete the system is now aware of all possible bids under resource constraint for slice admission control. The policy retrieval algorithm is given in **Algorithm 4**. The Q-learning is built based in the classical Bellman's equation given in 5.25.

$$Q(\boldsymbol{s,a}) = (1-\alpha)q(\boldsymbol{s,a}) + \alpha\{\mathbf{r}_t + \gamma \max_{a'} q(\boldsymbol{a',s'})\} \tag{5.25}$$

Where α is the learning rate, γ is the discount factor and R long-term reward. $Q(\boldsymbol{s,a})$ is the Q-value at action $\boldsymbol{a}$ after visiting state $\boldsymbol{s}$

5.2.5.1 Space and Time Complexity of Q-learning in Algorithm 3

The dominant storage requirement of the algorithm is the Q-table, which holds a value for each state-action pair. Thus, if there are $|S|$ states and $|A|$ actions, the Q-table requires $O(|S| \times |A|)$ space. In addition to the Q-table, the algorithm may also maintain a copy of a policy network (the target network) and several environment parameters. However, these additional components typically require much less memory compared to the potentially large Q-table, so overall the space complexity is primarily $O(|S| \times |A|)$.

The algorithm features nested loops over episodes and within each episode, over iterations defined by indices m and t. If we denote the number of episodes by E, the number of

Algorithm 4 Q-Learning for Auction based slice admission control

Result: Q-Table, Reward Per Episode
Input:Initialize learning parameters$(\gamma, \alpha, \epsilon_{max})$
Input: Initialize environment parameters $(\phi_{bid,n}(t), \mathfrak{V}_{ask,n}(t), \Gamma_n, \mathbf{r}(t), \mathbf{a}(t))$
Input: Initial Q-table ($[Q] = 0$)
Create a copy of the policy network (Target) with weights θ'
while inEpisode **do**
  GetInitialState$(\phi_{bid,n}(t=0), \mathfrak{V}_{ask,n}(t=0), \Gamma_n(t=0))$
  for m=0 to M-1 **do**
    for $t = 0$ to T **do**
      if exploration rate $> \epsilon$ **then**
        Take greed action
      else
        Choose random action ($\mathbf{a} \in [0, 1]$)
      end if
      Update Γ_n
      Find next-state($\mathbf{s} = \mathbf{s}_{t+1}$)
      Obtain reward by Eq 5.23 or 5.24
      Update Q-table by Eq.5.25
      m=m+1
    end for
    Update Exploration rate using $(\epsilon = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min})e^{-\delta \times \text{episode}})$
  end for
end while
Return Q-table

Algorithm 5 Q-Learning Policy Retrieval for Auction based slice admission control

Result: Table of actions and states
Input:Initialize States$(\gamma, \alpha, \epsilon_{max})$
Input: Initialize environment parameters $(\phi_{bid,n}(t), \mathfrak{V}_{ask,n}(t), \Gamma_n, \mathbf{r}(t), \mathbf{a}(t))$
Input: Initial Q-table ($[Q] = 0$)
Create a copy of the policy network (Target) with weights θ'
while inState and Action Space **do**
  GetInitialState
  Obtain Optimal Action and Corresponding State
  Obtain next state m=m+1
end while
Return optimal actions and corresponding states

Table 5.1: Simulation parameters

Hyper parameters	
Parameter	Value
Discount rateγ	0.09
Learning rate α	0.001
Maximum exploration rate ϵ_{max}	1
Minimum exploration rate ϵ_{min}	0.001
Exploration decay rate δ	0.01
Number of episodes	1000
Other parameters	
Parameter	Value
Latency range	1ms-100ms
Carrier Frequency	400MHz
Subcarrier bandwidth	60KHz,120KHz
Thermal noise	-174dBm
Task per cpu	10GB
Path loss	72+30log (d)[119]

iterations (or sub-episodes) by M, and the number of time steps per sub-episode by T, the total number of iterations is $O(E \times M \times T)$. Within each iteration, actions are chosen (either greedily or randomly), rewards are computed, and the Q-table is updated—each of which is typically an $O(1)$ operation. However, if the greedy action selection requires scanning through all actions, that could introduce an $O(|A|)$ factor per iteration. Therefore, the overall time complexity becomes $O(E \times M \times T \times |A|)$. In practice, if $|A|$ is small or optimized, the per-iteration work remains effectively constant, yielding a time complexity linear in the number of iterations.

5.2.6 Simulation and Results

The simulation process and results are discussed in this section. Table 5.1 refers to the simulation parameters. The hyper-parameters are carefully chosen and tuned to fit the simulation environment. The other parameters are derived from existing standards applicable to 5G networks and similar criterion[4][120][121]. We leveraged 5G new radio (NR) FR2 parameters considering the time division duplex (TDD) major bands on a single tier macro base station. In this case we consider a maximum achievable bandwidth of 400MHz at a maximum sub-carrier bandwidth of 120kHz with 264 channels for the radio RBs. The cloud based computing requirement is divided into the number of CPU cores and RAM requirement per slice per unit time normalized into unit RBs.

To validate this study we compared the work with well researched admission criteria, notably, random based admission (RBA) and greedy based admission (GBA) and published part of the study in [77]. Perhaps, we need to cautiously state that, from literature no work was found to have implemented auction based slice admission control with RL criterion at the time of publication.

In the simulation, we considered a single task of 10Gigabytes requiring 10^6 CPU cycles per second. Initially, we simulated the probability of accepting a slice request to justify the quest for this approach. The findings depicted in Figure 5.2

Following every admission control instance, the agent ability to admit more slice request is evaluated. The revelation from Figure 5.2 implies that the three techniques tested had similar behavior for the first 1500 slice requests depicted an average probability of accepting

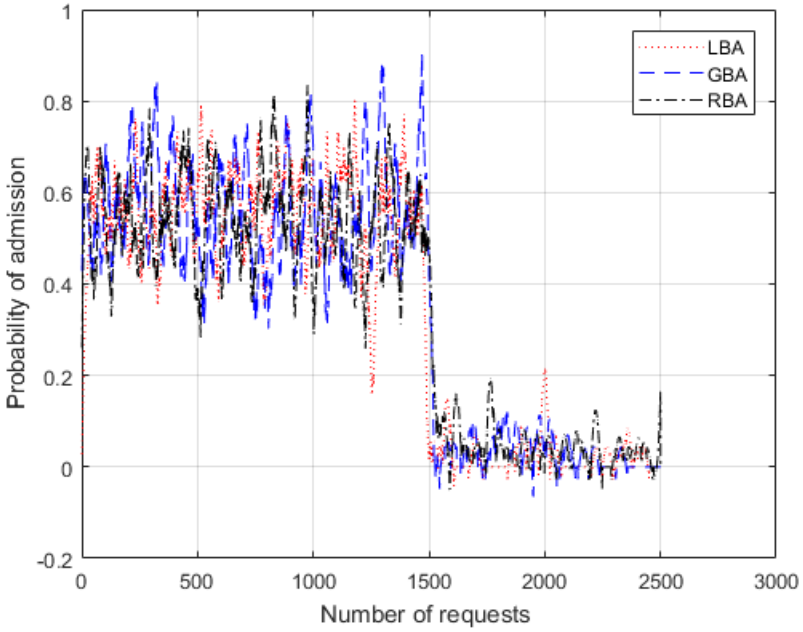

Figure 5.2: Admission probability of each scheme

a request at 0.6 before resource depletion where the probability average dropped to 0.2. This probability is calculated as a ratio of the overall available network resources to the required portion.

The Jain's fairness indicator in Eq. 5.21 provides a comparative measure of how fair each scheme performed. With 68.2% level, the LBA presents an improved fairness in distributing admission consent among nCm, nCe, and nCu slice categories in comparison to 27.8% and 4.0% of the RBA and GBA respectively. Concretely, as the agent policy evaluation improves, the algorithm learns to evaluate between short-term contracted slices with strict QoS demands and long-term applications with relaxed elastic QoS. Further, the agent learns to balance the trade-off between generating more long-term revenue by allowing resources to be held by slices for extended period or by admitting short-term slices with increased slice request turn-over but moderate revenue.

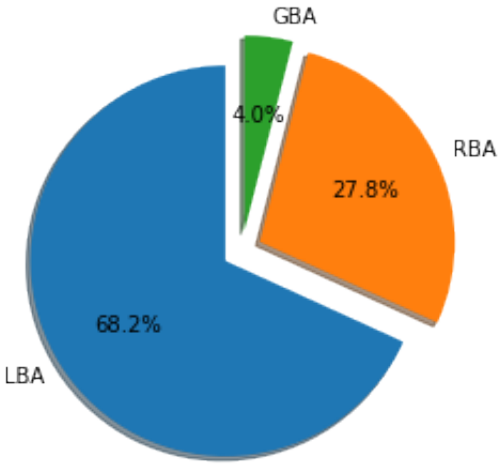

Figure 5.3: Fairness percentage for each slice admission scheme

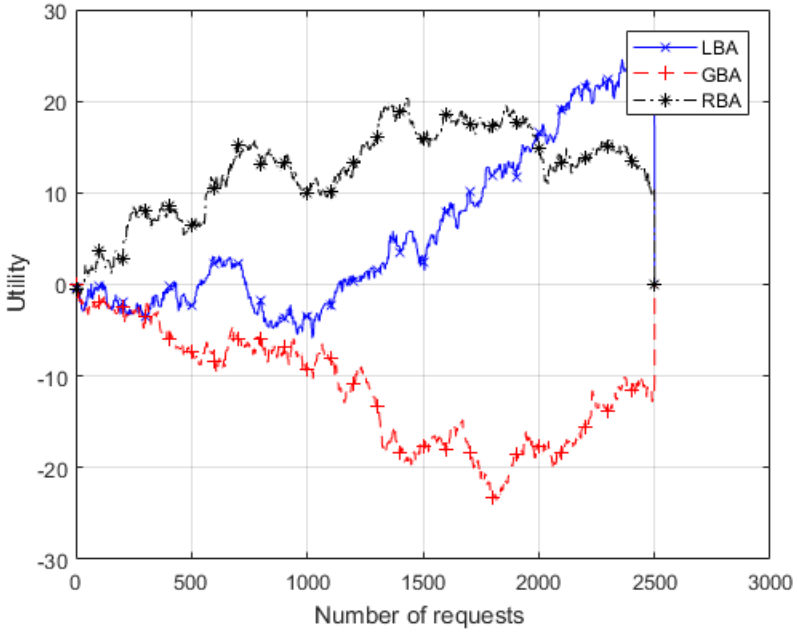

Figure 5.4: Projected social welfare

In Figure. 5.4, we present the effect of auctioning by determining the social welfare which is the cumulative utility over all episodes. The RBA scheme progressed with improved social welfare prediction initially due to numerous successful random auctions and eventual slice admissions. This randomness however, remained inconsistent and failed to produce positive utilities. This indicates that, the social welfare is dependent on accurate slice admission by reducing the gap between the social welfare obtained through learning and slice with global optimal revenue. The LBA through initial exploration and eventual exploitation gradually learned when to auction resource and admit slice requests and presented a progressive improvement in long-term social welfare for the tested 2500 slice requests. The GDA performed poorly due to consistent exploitation of non optimal states leading to largely negative utilities.

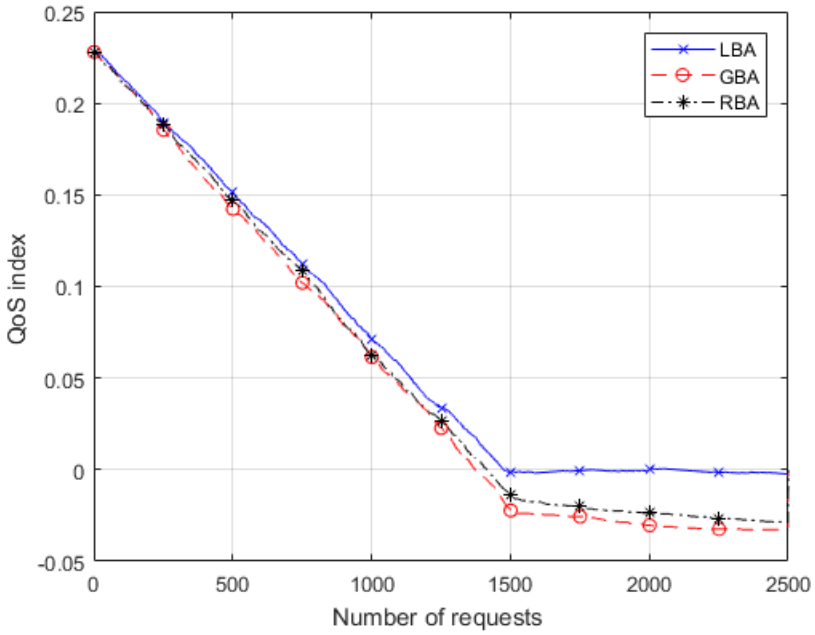

Figure 5.5: QoS Index vs slice admission

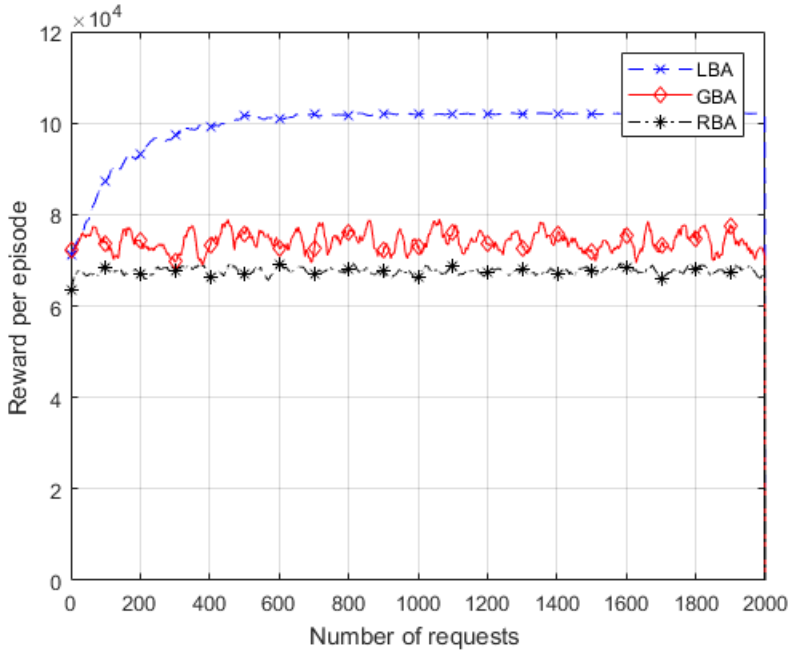

Figure 5.6: Illustration of average reward build up during learning

After every slice admission the InP must check its ability to meet the QoS for remaining slice requests. This ability is calculated as a function of the a quantity which represents the remaining resources known as the QoS index in Equation 5.20 shown in Fig. 5.5. The QoS index deterioration is considerably similar in all the schemes, however LBA still performs better than the other two schemes. This prediction shows that, the QoS cannot remain constant in an active episode as such implication is a false representation of the network status. The turning point at 1500 slice requests represents the resource depletion region. At this stage network resource are considered to be depleted and any subsequent slice admission only attracts penalties due negative QoS indicator or index.

In Fig 5.6 we finally demonstrate that LBA reaches near optimal at approximately episode 600. This contrast both GBA and RBA which remains non optimal through all iterations. It

is clear that LBA performs better for the tested events meeting the goals set at the beginning of this paper.

5.3 Chapter Summary

To address the challenge of slice admission control and efficient resource allocation, this chapter presents an auction based slice admission control to provide a QoS aware admission control while maximizing revenue. We have shown that, compared to RBA or GBA, the LBA scheme has greater control on admission selection enabling valuable resource tenants to be admitted. We have shown that the proposed algorithm allows the InP to limit losses by restricting slice admission to those that can achieve the auction goal and have the highest value.

Chapter 6

Enhancing 5G Network Resilience with Machine Learning based on Network Slice Admission Control

6.1 Introduction

Network slicing is still the main strategy in the allocation of resources for various 5G network functions via network function virtualization. This advancement is aimed at facilitating the logical allocation of resources to accommodate the expected increase in network resource requirements. Several benefits can by achieved through automated scheduling, auctioning, and orchestration for effective management. This chapter introduces network resilience maintenance leveraging slice admission control. While 5G network resources can be allocated to sustain a slice, ongoing examination and adjustment of logical allocation and real-time network assessment are necessary to maintain network resilience. The intricate task of utilizing slice admission control to uphold 5G network resilience is investigated henceforth. To tackle this issue, this chapter presents a machine learning-based approach for slice admission control and resource allocation optimization to ensure network resilience.

Machine learning algorithms provide a potent means of making robust and autonomous decisions, essential for effective slice admission control. By intelligently allocating resources in response to real-time demand and network conditions, these algorithms can contribute to

long-term network resilience and the achievement of key objectives. While various machine learning algorithms show promise for 5G resource management and admission control, reinforcement learning (RL) has emerged as an especially promising solution. Its capacity to mimic human learning processes renders it a versatile solution well-suited for addressing the persistence challenges of network control. To address this need, we propose a new technique called sequential twin actor-critic (STAC). Simulations demonstrate that STAC enhances network resilience through improved admission probability and overall utility.

The consensus among major stakeholders and industry players in fifth-generation (5G) communication technology is that network slicing will serve as the primary facilitator of dynamic resource allocation.[15].Network slicing enables users to establish precise quality-of-service (QoS) criteria for resources, allowing them to efficiently accommodate virtual functions on the shared physical infrastructure. Despite offering improved flexibility, scalability, and adaptability, the success of this technology relies on the implementation of agile optimization methods to meet predefined slicing objectives.

The application of these optimization methods equips InPs with effective means to efficiently achieve their slice admission control objectives. Some of the widely recognized slice admission control objectives encompass user maximization, revenue optimization, QoS control, fairness, and slice isolation [1].

To enhance resilient outcomes in slice admission control, an InP must choose an optimization approach that offers both flexibility and intelligent decision-making capabilities. This decision should be based on interaction with either available data or the contextual information surrounding it. While historical data have been used to forecast crucial network

behavior, aiding informed decision-making [122], they might be costly and scarce [15]. Additionally, given the anticipated rapid evolution of network services and functions [64], depending solely on historical data may not yield accurate prediction into network behavior. Slice admission control can initiate network status updates by enabling a trained machine learning agent to respond based on an aggregated dataset comprising both slice requests and real-time network status information. Furthermore, RBs can be allocated and adjusted to a network slice for service chain orchestration in both RAN and CNs after evaluating the network condition, thus completing the slice admission control process.

The use of a non-optimized slice admission control process can be misleading, as each admission control is closely tied to an objective specifically designed to fulfill a user-defined QoS, which must be maintained throughout the slice orchestration process. In this context, any QoS deterioration will always attract SLA penalties. On this note, an InP is motivated to deploy autonomous self healing resilient system.

Moreover, optimal slice admission control must account for numerous network constraints, some of which are inelastic and cannot be relaxed, in conjunction with the non-linearity of network entities. As highlighted in [1], the formulation of the objective function tends to become mixed-integer and non-linear. Many of the approaches employed to solve mixed-integer problems introduce a certain level of computational complexity. This challenge can be addressed by leveraging the generalization ability of machine learning, thereby obviating the need to relax many constraints.

Among various machine learning techniques, reinforcement learning (RL) has been a prominent method advocated by numerous researchers to improve network control and

optimization [64]. Slice admission control objectives have been addressed using RL, taking into account individual user and network scenarios. For instance, an RL-based slice admission policy can be employed to maximize long-term time-varying profit over an episode duration, denoted as t, as illustrated in Equation 6.1 [123].

$$P = \int p_i(t)dt \tag{6.1}$$

where p_i is an instantaneous profit acquired when a slice i is admitted. The learning agent controls the admission process by evaluating resources at the central office, optical back-haul and remote data centers.

6.2 Overview of Machine Learning based Slice Admission Control for Network Resilience.

Precise predictive modeling is crucial for effective decision-making in network management, presenting a multifaceted challenge. Furthermore, as the granularity of the evaluated data expands, so does the complexity of this task. In the realm of slice admission control and network resilience maintenance for 5G networks, there is a notable lack of comprehensive research exploring the utilization of multidimensional data in multi-stage reinforcement learning (RL). This gap underscores the necessity for investigating efficient slice admission control and resource adjustment strategies to enhance the resilience of these networks. In this section, we delve into related works in this field, highlighting their constraints. These limitations have acted as the driving force behind the inquiry in this chapter.

The study presented in [124] introduced a joint resource allocation and slice admission control mechanism for fog-RAN. By incorporating the advantage actor-critic (A2C) technique, notable enhancements were achieved in managing continuous action spaces, leading to improved efficiency and stability. However, the research overlooked the complex challenge arising from multi-dimensional state spaces in scenarios involving multiple users and slices, indicating the need for additional exploration and consideration coupling the need to maintain network resilience.

Bakri et al. in [125] made a significant contribution to this field by exploring the feasibility of utilizing the stability of deep Q-learning and regret matching. They comparatively evaluated the effectiveness of these techniques in maximizing slice admission while penalizing violations of service level agreements (SLAs). However, the study's simplistic focus on maximizing slice admission without considering the adjustment of network resources to ensure resilience highlights a limitation in assessing the broader network requirements and is essentially greedy in nature.

The continuous and enduring pursuit of minimizing time complexity in training reinforcement learning (RL) models is a broad challenge. In the work presented in [126], the authors proposed RL-based ensemble learning specifically for slice admission control, aiming to reduce learning time as a primary objective. The state-space elements in their approach were limited to throughput rate, error rate, and available resource blocks. However, a more flexible and scalable state space, as demonstrated in our study, would represent an enhancement in the effectiveness of a model for reducing time complexity during the learning process in the pursuit resilient performance. Continuing the exploration of slice admission control,

the authors in [127] introduced dynamic slice allocation and admission control to optimize the number of users, while evaluating network parameters such as QoS, network load, and predictive load.

6.3 System model

The continuing quest to improve slice admission control and its application to achieve a set network objective has been established in Chapter 4 and 5. We continue on the same note by revisiting some key entities in network slicing referenced in Figure 6.1

In our scenario, we have three key participants. Firstly, there is the user who requires a portion of slice service to run an application and may not be aware of service limitations. Secondly, we have the tenant, who initiates a slice request to an Infrastructure Provider (InP). The tenant faces the challenge of crafting an appropriate slice request that closely aligns with the demands of a group of users. Lastly, there is the InP, who owns the physical infrastructure and has the ability to create virtual resource pools to accommodate a specific slice. The InP generates revenue by admitting the requested slice and assigning a price to it. Additionally, the InP has the capability to adjust a slice to prevent compatibility issues and duplication.

An InP plays a pivotal role in overseeing a slice package by ensuring several key parameters: maintaining the user count within specified limits, preserving spectral resource availability, managing cloud resources, preventing the orchestration of duplicate or incompatible slices, allocating optimal power per user, and constraining interference ratios. However, these

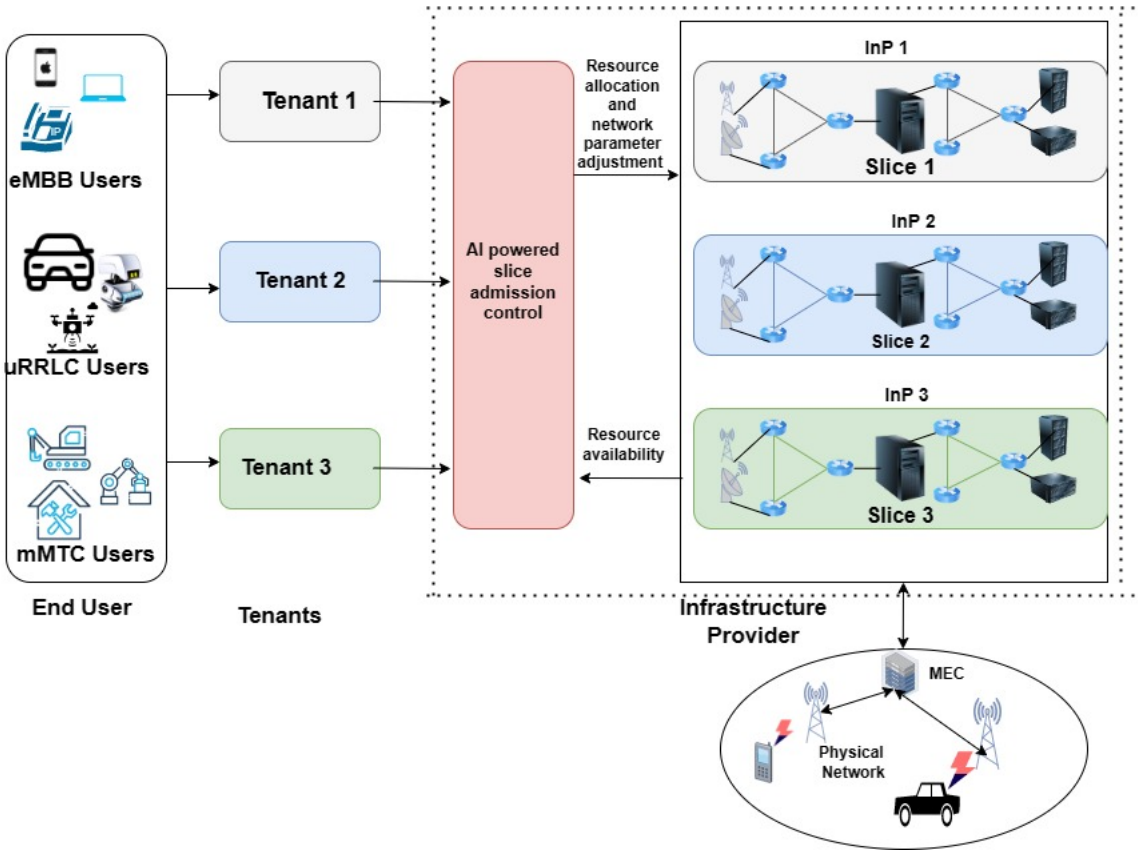

Figure 6.1: Network model

constraints collectively give rise to a complex, non-linear, and non-convex problem, presenting significant challenges even in relatively straightforward configurations.

To address the slicing problem, this section introduces a model for the individual components needed to formulate a multidimensional slice request. Each slice is supported by an asymmetric physical downlink shared channel (APDCH), whereas the uplink connection is assumed to have minimal data rate requirements with acceptable latency.

We denote the set of slice classes for each user application as $c \in \mathcal{C}$ with a cardinality of $C = |\mathcal{C}|$. A slice request of class c is characterized by a Poissonian arrival rate λ_c (requests/sec) and a processing delay of t_c. Each user $u \in \mathcal{U}$ is associated with a gNodeB (gNB) $b \in \mathcal{B}$ and receives fixed-sized packets according to the slice type. Each user can be allocated a portion

of power $p_{u,b}$ assuming each user associates with one gNB at a time. The expected power required to service a slice of type c serving u is given by:

$$\mathcal{P}_c = \sum_{u \in \mathcal{U}} p_{u,b} \tag{6.2}$$

However, optimal power allocation can be achieved by employing the water filling algorithm. We proceed by deriving the power allocation per user leveraging a modified water-filling algorithm in [128]. If the are $n \in \mathbb{N}$ downlink channels serving all the users such that $\mathcal{U} = \mathbb{N}$, it is then convenient to mention that, the total allocated power $\mathbb{P}_\mathbb{N}$ serves all users. Let $\mathbb{G}_n$ denote sub-channel coefficient assuming the channel information is known then from [129] we derive the optimum allocated power to each channel. For a convex objective function representing a maximum channel rate $\mathbb{C}$ with $\mathbb{G}_n$ as the sub-channel gain and $\mathbb{P}_n$ we have:

$$\mathbb{C} = \max_{\mathbb{P}_1, \mathbb{P}_2 \ldots \mathbb{P}_\mathbb{N}} \sum_{n \in \mathbb{N}} \log_2(1 + \mathbb{G}_n \mathbb{P}_n) \tag{6.3}$$

s.t $\sum_{n \in \mathbb{N}} \mathbb{P}_n \leq \mathbb{P}_\mathbb{N}$

To solve optimization problem in 6.3 we introduce Lagrangian function of the form;

$$\mathbb{L}(\mathbb{P}_1, \mathbb{P}_2 .. \mathbb{P}_\mathbb{N}, ß) = \sum_n \int_0^\mathbb{N} h_c \log_2 \left(1 + \frac{\mathbb{P}_n \mathbb{G}_n}{\mathbb{P}_\mathbb{N}}\right) \wp(\sigma) d\mathbb{P}_n - ß \sum_{n \in \mathbb{N}} \int_0^\mathbb{N} (\mathbb{P}_n - \mathbb{P}_\mathbb{N}) \wp(\sigma) d\mathbb{P}_n \tag{6.4}$$

The Lagrangian multiplier ß accounts for the accumulated power constraint, $\wp(\sigma)$ is the probability distribution of the received SNR and h_c is the total channel allocated bandwidth. By differentiating 6.4 and equating to zero we have

$$\frac{\partial \mathbb{L}(\mathbb{P}_n)}{\partial(\mathbb{P}_n)} = \left[\left(\frac{h_c/ln(2)}{1+\sigma\mathbb{P}_n/\mathbb{P}_\mathbb{N}}\right)\frac{\sigma}{\mathbb{P}_\mathbb{N}} - ß\right]\wp(\sigma) = 0 \tag{6.5}$$

By solving 6.5, the required water-filling power allocation is obtained as:

$$\frac{\mathbb{P}_n}{\mathbb{P}_\mathbb{N}} = \begin{cases} \frac{1}{\sigma_0} - \frac{1}{\sigma} & \text{for} \quad \sigma \geq \sigma_0 \\ 0 & \text{for} \quad \sigma < \sigma_0 \end{cases} \tag{6.6}$$

The expected maximum down-link data rate achievable with the allocated bandwidth h_c is $\mathcal{W}_b$ given by;

$$\mathcal{W}_b = \mathbb{E}\left[\sum_{u\in\mathcal{U}} \frac{h_c \log_2(1+\Delta_{u,b})}{\lambda_c t_c}\right] \tag{6.7}$$

The signal-to-interference ratio $\Delta_{u,b}$ is estimated as follows:

$$\Delta_{u,b} = \frac{p_{u,b}}{\sigma^2 + P_r p_{int}} \tag{6.8}$$

Where $p_{int} = 128.1 + 3737.6log(d)$ [9] is the interfering power from a co-slice adjacent user and σ^2 is the noise power. At this point we can equate $p_{u,b}$ to $\mathbb{P}_n$ and adopt the optimum power allocation model in 6.6. P_r and p_{int} are the probability of inter-slice inference and the interference from other gNBs respectively [130].

The allocation of computing resources involves determining the CPU cycles per second per slice task and the required memory. To handle the anticipated non-homogeneous tasks, the CPU performance is graded. The system calculates a unique cost for each task. While some computing resource requirements can be managed by the Baseband Unit (BBU) function, this operation operates on the physical layer, and for efficient provisioning, the resources must be deployed as a logical process. We will proceed by elaborating on the computing model at the edge cloud, beginning with the modeling of the CPU core requirement based on the chosen task.

Given a slice type $c \in \mathcal{C}$ associated with a tenant $e \in \mathcal{E}$, the slice requires a processing rate of ζ_c bits/sec and has an average duration of l_c secs in the server. Each slice task can be represented as a tuple $Q_c = \{\zeta_c, l_c\}$. The CPU grade assigned to a slice of type c is denoted by $\eta_c^m = \{\eta_c^0, \eta_c^1, \eta_c^2, \ldots, \eta_c^{M-1}\}$, consisting of incremental values divided into M discrete grades. The realization of each grade depends on the CPU task, and transitions between grades follow a Markovian process that can be modeled as follows. The transition matrix containing the probability of moving from one grade to another is given by.

$$\mathcal{Y}_c(t) = [\varkappa(t)]^{M \times \mathcal{C}} \tag{6.9}$$

with $\varkappa(t) = \mathbb{P}r(\mathcal{Y}_c(t_c + 1) = \overline{a} | \mathbb{P}r(\mathcal{Y}_c(t_c) = \overline{b})$ such that $\overline{a}, \overline{b} \in \eta$. The computation cost estimation is provided as follows:

$$\omega_c^m = \frac{\zeta_c l_c}{f_c} \eta_c^m a_c^m(t) \tag{6.10}$$

where f_c is the number of CPU cycles and $a_c^m(t) \in \{0,1\}$ represents whether a CPU grade m has been realized (1) for slice type c or not (0). Ultimately, the total cost incurred for computing resource subscription is given by $\Omega_c^{cpu} = \omega_c^m \mathcal{T}_c$ where $\mathcal{T}_c$ is the slice duration.

The processing of tasks also relies on the cache memory requirement per slice, which is determined by a hit ratio. This ratio indicates the probability of establishing an exact cache match. However, the InP must periodically refresh cache resource blocks to enhance the efficiency of the hit ratio[131]. If we consider an i^{th} slice request arriving with a Poisson process of rate λ_i, then similarly, if v_i is a finite cache size then its realization probability is given by [116].

$$Pr(v_i) = \frac{1}{i^{\psi} \sum_{i=1}^{\mathcal{I}} \frac{1}{i^{\psi}}} \tag{6.11}$$

where $0 < \psi \leq 1$ follows a Zipf slope. Hence the cache memory cost per slice at time t is given by;

$$\Omega_c^{v_i} = Pr(v_i)\lambda_i(t) \tag{6.12}$$

A successful hit leads to cache RB orchestration otherwise the slice request cannot be completed due RB deficiency.

6.4 Problem formulation

This section addresses the problem formulation with the goal of achieving optimal revenue while also addressing the ongoing pursuit of maintaining network resilience.

6.4.1 Optimizing Data Rate

The admission criteria for network slices specify required data rates for both the transport network and the RAN. Achieving these rates, particularly for the downlink, relies on the processing capacity of the BBU and the mitigation of air interface interference. Fortunately, the BBU has the capability to dynamically adjust its performance to mitigate interference, given accurate control information about downlink channel conditions. Intelligent negotiation plays a crucial role in this process. Through intelligent negotiation, critical system adjustments are made to meet the required data rates. Specifically, after admitting a slice,RL is employed to take additional actions. These actions involve adjusting system parameters with the objective of minimizing interference and maximizing the downlink data rate.

Increasing data rates in transport networks relies on two primary strategies: minimizing processing time and selecting optimal routes for data packets. While control plane controllers such as OpenFlow and iMaster NCE in 5G core are primarily recognized for their proficiency in route selection (typically via segment routing in the next-generation core), reducing processing time necessitates an additional layer of optimization. This optimization is achieved through the implementation of multi-processing mechanisms and multi-queue scheduling at each network node.

When considering a fixed-size packet $\rho \in \mathcal{P}$ of length ι_ρ arriving at a BBU node g with ϱ_g processors of equal performance (measured in bits processed per second), and given that the node performs a multi-queue scheduling of q_g queues, where each queue holds an equal portion of the arrived packet, we can derive the processing time per slice as follows:

$$t_{pro} = \begin{cases} \frac{\iota_\rho}{\varrho_g} & \text{for} \quad q_g = 1 \\ \sum_{\rho \in \mathcal{P}} \frac{\iota_\rho}{\varrho_g \times q_g} & \text{for} \quad q_g \geq 1 \end{cases} \tag{6.13}$$

We therefore formulate the data rate maximization problem which indeed is the minimization of the mean squared bit processing time error given by;

$$\boldsymbol{P1} : \min \varphi_1 = \frac{1}{|\mathcal{E}|} \left(\sum_{g \in \mathcal{G}} t_{pro,e,c,g} - \left\{ \frac{1}{\mathcal{W}_{b,e,c}} + \frac{1}{\mathcal{F}_b} \right\} \right)^2 \tag{6.14}$$

subject to:

$$C1 : \sum_{g \in \mathcal{G}} t_{pro,e,c,g} \leq \mathcal{T}^{max} \quad \forall e \in \mathcal{E} \quad \forall c \in \mathcal{C} \tag{6.15}$$

$$C2 : \left\{ \frac{1}{\mathcal{W}_{b,e,c}} + \frac{1}{\mathcal{F}_b} \right\} \leq \mathcal{T}^{pro} \quad \forall e \in \mathcal{E} \quad \forall c \in \mathcal{C} \tag{6.16}$$

where $\frac{1}{\mathcal{W}_{b,e,c}}$ is the downlink bit duration, $t_{pro,e,c}$ is the bit processing time in the BBU pool and $\frac{1}{\mathcal{F}_b}$ is the BBU bit processing time adjustment in seconds which is set by the InP in anticipation of and downlink air interface interference An InP can increase $\mathcal{F}_b$ cycles per second to reduce the bit processing time.

Constraint C1:represents the overall bit processing duration limit set by a VNO which contributes to the overall latency between the BBU and an end node where the maximum limit is set at $\mathcal{T}^{max}$.

Constraint C2: denotes the down-link bit duration plus a positive adjustment to cater for variation in network load the maximum limit is set at $\mathcal{T}^{pro}$.

6.4.2 Optimizing computing resource allocation

Allocating slice RB presents a challenging trade-off between minimizing slice resource requirements and maximizing revenue. Additionally, tenants subscribed to a slice aim to minimize the cost incurred per slice without compromising the requested QoS. Achieving this balance is complex for the InP, as resource over-subscription tends to degrade the network performance and result in penalties.In computing resource optimization, we take into account both the CPU and memory requirements. The objective is to maximize each resource allocation while ensuring that it precisely matches the requirements of each slice, without oversubscribing or duplicating slices.

Considering the computing resource requirement, a slice RB is instantiated if there exists an initial requirement RB^{inti} at a cost Ω^{init}. Furthermore, an adjustment RB^{adj} at Ω^{adj} is added only if it is necessary. We adapt the static resource allocation method described in [132] to facilitate controlled resource matching. This modification ensures that a computing RB is instantiated only if an exact match can be allocated. This is give by;

$$\Upsilon_c = \begin{cases} 1 & \text{for} \quad [\Omega^{init} \geq \Omega \leq \Omega^{init} + \Omega^{adj}] \\ 0 & \text{Otherwise} \end{cases} \tag{6.17}$$

Each initial RB cost is defined by:

$$\Omega^{init} = \sum_{c \in \mathcal{C}} \Omega_c^{cpu} + \sum_{c \in \mathcal{C}} \Omega_c^{v_i} \tag{6.18}$$

To improve RB assignment efficiency, the RB adjustment cannot be more than the initial requirement, this is also fair as a tenant has to place an estimated, accurate demand that is capped at a cost Ω^{ref}. The controlling constraint is given by.

$$\Omega^{adj} = \Omega^{init}.\min(1, \frac{\Omega^{init}}{\Omega^{ref}}) \tag{6.19}$$

Finally to prevent resource overlap and duplication,we have

$$\begin{aligned} &[RB_c^{init}, RB_c^{init} + RB_c^{adj}] \bigcap [RB_{c'}^{init}, RB_{c'}^{init} + RB_{c'}^{adj}] = \emptyset \\ &\forall c \neq c', \forall c, c' \in \mathcal{C} \end{aligned} \tag{6.20}$$

The ultimate goal of the InP is to maximize revenue obtained from slice resource allocation. The RB orchestration from the computing block accounts for a portion of expected revenue, this is given by ***P2***

$$\boldsymbol{P2} : \max \varphi_2 = \left[\frac{1}{\Omega_{ref}} \left\{ (1-\Gamma)\Omega_c^{init} + \Gamma \left(\Omega_c^{init} + \Omega_c^{adj} \right) \right\} \Upsilon_c \right] \tag{6.21}$$

subject to:

$$C2 : \Omega_{c,e}^{init} \leq \Omega^{adj} \leq \frac{\Omega^{init}}{\Omega^{ref}} \tag{6.22}$$

$$C3 : \theta_c = [0, 1] \tag{6.23}$$

The parameter $\Gamma \in (0, 1)$ ensures that revenue obtained per slice is not due to overpricing despite adjusting resource allocation to match the QoS demand. Constraint $C2$ describes the cost adjustment limits for RBs allocated to balance resource fluctuations. Constraint $C3$

represents the admission control variable which implies admit if 1 or reject otherwise. The control parameter is the cost due to resource adjustment level Ω^{adj}

The joint optimization problems is now given as follows;

$$\varphi_{tot} = \mathcal{T} \sum_{n=1}^{\mathcal{N}} \sum_{c \in \mathcal{C}} \left[\frac{\vartheta_c}{\mathcal{V}} \log(\varphi_1^n) / \log(\varphi_2^n t_c) \right] \theta_c \tag{6.24}$$

where $\theta_c = [0, 1]$ is an admission control variable which implies admit if 1 or reject otherwise, ϑ_c is the radio RB unit price and $\mathcal{V}$ is the maximum chargeable price per slice of type c.

6.5 Problem solution

In this section we introduce a new actor-critic framework with a multi-stage action capability. By leveraging continuous action space in real time, the system can realistically adjust system parameters

Considering the temporal nature of the state space, we utilize a stacked three-dimensional matrix as input to the Convolutional Neural Network (3D-CNN). Each matrix represents a two-dimensional state space at a specific time. While a Virtual Network Operator (VNO) typically sends a single slice request to the Infrastructure Provider (InP), our multidimensional state allows for multiple slice requests to be placed simultaneously.

A slice s is defined as comprising data representing a slice request from a VNO, along with the measured system conditions. A trap signal delivers network condition parameters, enabling an Infrastructure Provider (InP) to assess the admissibility of a slice request. Typically, a state $s(\kappa)$ is given by equation 6.26, where $\kappa = [1, 2, 3, ..\mathbb{K}]$ denotes the κ^{th} request from a

set of VNO $\mathcal{E}$. The parameter τ_c represents the slice latency requirement, which is a key QoS metric.

From equation 6.26, it is evident that a single state has a size of $10 \times \Theta$, while the entire state space has a size of $10 \times \Theta \times \mathbb{K}$. Navigating through the entire state space has a complexity of $\mathcal{O}(\mathbb{K})$, which is advantageous as an entire state matrix can be fed into a convolutional neural network input embedded in the Deep Reinforcement Learning (DRL) algorithm. The parameters $\mathcal{H}(\kappa), \mathcal{V}(\kappa), D(\kappa)$ in the state space represent the cumulative remaining system resource levels for bandwidth RB, cache, and CPU grades, respectively.

$$\begin{aligned} \mathcal{H}(\kappa) &= \mathcal{H}(\kappa - 1) - h_c(\kappa) \\ \mathcal{V}(\kappa) &= \mathcal{V}(\kappa - 1) - v_{i,c}(\kappa) \\ D(\kappa) &= D(\kappa - 1) - q_c(\kappa) \end{aligned} \tag{6.25}$$

It has to be noted that each RB can be allocated independently. If a tenant requests for a single domain resource it implies that only the requested resource is allocated and charged with minimal allowable network support.

$$s(\kappa) = \begin{bmatrix} \mathcal{W}^1_{b,c}(\kappa) & \mathcal{F}^1_{b,c}(\kappa) & \tau^1_c(\kappa) & h^1_c(\kappa) & q^1_c(\kappa) & \mathcal{T}^1_c(\kappa) & v^1_{i,c}(\kappa) & \mathcal{H}^1(\kappa) & \mathcal{V}^1(\kappa) & D^1(\kappa) \\ \mathcal{W}^2_{b,c}(\kappa) & \mathcal{F}^2_{b,c}(\kappa) & \tau^2_{c,}(\kappa) & h^2_{c,}(\kappa) & q^2_{c,}(\kappa) & \mathcal{T}^2_c(\kappa) & v^2_{i,c}(\kappa) & \mathcal{H}^2(\kappa) & \mathcal{V}^2(\kappa) & D^2(\kappa) \\ \mathcal{W}^3_{b,c}(\kappa) & \mathcal{F}^3_{b,c}(\kappa) & \tau^3_c(\kappa) & h^3_{c,}(\kappa) & q^3_c(\kappa) & \mathcal{T}^3_c(\kappa) & v^3_{i,c,}(\kappa) & \mathcal{H}^3(\kappa) & \mathcal{V}^3(\kappa) & D^3(\kappa) \\ . & & & & & & & & & \\ . & & & & & & & & & \\ . & & & & & & & & & \\ \mathcal{W}^\Theta_{b,c}(\kappa) & \mathcal{F}^\Theta_{b,c}(\kappa) & \tau^\Theta_c(\kappa) & h^\Theta_c(\kappa) & q^\Theta_c(\kappa) & \mathcal{T}^\Theta_c(\kappa) & v^\Theta_{i,c}(\kappa) & \mathcal{H}^\Theta(\kappa) & \mathcal{V}^\Theta(\kappa) & D^\Theta(\kappa) \end{bmatrix} \tag{6.26}$$

6.5.1 Distributed Action Space

An action in slice admission control is a command initiated by the learning agents after visiting a particular state. The primary objective is to select a slice request, evaluate its advantage, and decide whether to admit the request or not. This decision-making process typically relies on a predefined objective, such as achieving the expected Quality of Service (QoS). However, the commonly used binary decision is insufficient for adjusting system parameters. In traditional approaches, a separate external process is initiated to perform system adjustments, which is disadvantageous as it is not integrated into the learning process. In this work, we employ an action space that reflects practical network observations. A slice request is admitted with the aim of matching requirements through optimal resource allocation and improving expected revenue. If a slice cannot be admitted, the system status remains unchanged. To implement this solution, a multilevel action space is necessary. The first step involves deciding whether to admit a slice or not, followed by adjusting system parameters based on the initial decision.

There are two sets of action spaces. The initial space is designed to perform slice admission control A^{cont} while the latter space is employed for system parameter adjustment or compensation A^{adj}. We define the action spaces at time t as:

$$\mathcal{A}(t) = \{A^{cont}, A^{adj}\} \tag{6.27}$$

where $A^{cont} = [a^{admit}, a^{rej}]$ are slice level actions meaning admit or reject while $A^{adj} = (a^{dat}, a^{cpu}, a^{mem}, a^{int})$ are data rate, cpu cycles, memory adjustments and interference compensation.

6.5.2 Reward Function

The reward function is designed to evaluate the performance the agent. Each action attracts a unique reward. We define them as follows.

$$r(t) = \begin{cases} (r^{dat} + r^{cpu} + r^{mem} + r^{int} + r^{rej})\varphi_{tot} & \text{for} \quad A^{cont} = a^{admit} \\ \frac{\mathcal{W}_{b,c}(\kappa)h_c(\kappa)\mathcal{T}_c(\kappa)}{\mathcal{H}(\kappa)\times\mathcal{W}_{max}(\kappa)\times\tau_c(\kappa)} & \text{for} \quad A^{adj} = a^{dat} \\ \frac{q_c(\kappa)\times\mathcal{T}_c}{D(\kappa)} & \text{for} \quad A^{adj} = a^{cpu} \\ \frac{v_{i,c}(\kappa)\times\mathcal{T}_c(\kappa)}{\mathcal{V}(\kappa)} & \text{for} \quad A^{adj} = a^{mem} \\ \log(\mathcal{F}_{b,c}(\kappa)) & \text{for} \quad A^{adj} = a^{int} \\ \frac{1}{r^{dat}+r^{cpu}+r^{mem}+r^{int}} & \text{for} \quad A^{cont} = a^{rej} \end{cases} \tag{6.28}$$

Each reward value is formulated to reinforce each action the agent takes. This decomposition is advantageous since it eliminates the "one fit all" reward function which indeed can be very complex.

6.5.3 State Space Dimensionality Reduction

The proposed reward functions are not directly applicable to the expected multidimensional state space. Additionally, to ensure scalability, the state space is allowed to grow either linearly or non-linearly, a decision left to the Infrastructure Provider (InP). To address the

challenge of the curse of dimensionality, a tractable procedure is to perform dimensionality reduction. Kernel Principal Component Analysis (KPCA) is adopted in this study for dimensionality reduction. KPCA enables extract of important principal components of a dataset.

If we consider Θ training samples $s(\kappa) \in \mathbb{R}^{\Theta}$, the goal is to map the samples into a high-dimensional space $\mathcal{H}$ under a non-linear transformation. This transformation allows the samples to become linearly separable for PCA. By zero-centering $s(\kappa)$, the covariance matrix can be obtained by:

$$\mathcal{S} = \frac{1}{\Theta} \sum_{1}^{\Theta} s(\kappa) s^{T}(\kappa) \tag{6.29}$$

further, the eigenvectors φ and eigenvalues ψ of $\mathcal{S}$ are determined as follows

$$\mathcal{S}\varphi = \psi\varphi \tag{6.30}$$

Generally, mapping the high-dimensional feature map $\mathcal{H}$ to the state matrix $s(\kappa)$ is not computationally efficient as any increase in dimension would further increase the dimensions in $\mathcal{H}$. A suitable kernel function is highly appropriate. In this work a kernel matrix $\mathcal{K}$ is derived from the state matrix as

$$\mathcal{K} = s(\kappa)^{T} s(\kappa) \tag{6.31}$$

The eigenvectors and eigenvalues (φ,ψ) of $\mathcal{K}$ can be presented as

$$\psi_{\Theta}\varphi_{\Theta} = s(\kappa)^{T} s(\kappa)\varphi_{\Theta} \tag{6.32}$$

Following equations 6.29,6.30,6.31,6.32 the principal component Υ projection and evaluating according to [84] is derived as:

$$\Upsilon_{\Theta} = \frac{1}{\sqrt{\psi_{\Theta}}}(\varphi_{\Theta})^{T}\mathcal{K} \tag{6.33}$$

6.6 Algorithms

The action determination for a policy based evaluation method, stochastic policy approach with parameter $\theta : \pi(s, a, \theta)$ can adopt stochastic policy gradient method where actions are sampled. However for a multidimensional action space, this can be computationally intensive. An appropriate approach is to adopt a deterministic policy gradient (DPG) [115]. DPG does not perform action sampling instead provides a single value $a = \Pi_{\theta}(s)$ which is the probability of taking action a after a state s is visited. As already mentioned, an actor critic approach provides a mechanism for off-policy evaluation in deterministic policy while maintaining the trade-off between exploration and exploitation. We proceed by briefly describing the actor-critic method then present the proposed sequential twin-actor critic (STAC) method.

6.6.1 The Actor Critic Algorithm

The actor-critic framework integrates both value function evaluation and policy determination. The actor component outputs an action based on a deterministic policy, specifically the probability of taking an action under the policy parameter θ_a. In the learning process, the deterministic policy gradient method is employed for policy evaluation. On the other

hand, the critic component, whose parameters are updated via gradient descent, evaluates the action-value using the loss function, which samples parameters from the critic network and the replay buffer. The deterministic policy of the actor is updated by a parameter update given by

$$\theta'_a = \theta_a + \alpha \bigtriangledown_{\theta_a} \mathcal{J}(\pi_{\theta_a}) \tag{6.34}$$

Where α is the learning rate $\bigtriangledown_{\theta_a}$ is the update gradient under parameter θ_a, $\mathcal{J}(\pi_{\theta_a})$ is the objective function of the critic under policy π_{θ_a} of the actor. The actor gradient can be estimated by the following expression.

$$\bigtriangledown_{\theta_a} \mathcal{J}(\pi_{\theta_a}) \approx \sum_{s \in S} \bigtriangledown_{\theta_a} \log \pi_{\theta_a}(s) A_{\pi_{\theta_a}}(s, a) \tag{6.35}$$

where $\log \pi_{\theta_a}(s)$ is the state visitation probability distribution under the policy π. The value function of the critic is denoted by $Q_{\theta_c}(s, a)$ where θ_c is the critic parameter. Further, the critic is updated by TD as:

$$\delta_t = r\prime + \gamma Q_{\theta_c}(s\prime, a\prime) - Q(s, a) \tag{6.36}$$

and the critic parameters are updated as follows

$$\theta'_c = \theta_c + \alpha \delta_t \bigtriangledown_{\theta_a} Q_{\theta_c}(s, a) \tag{6.37}$$

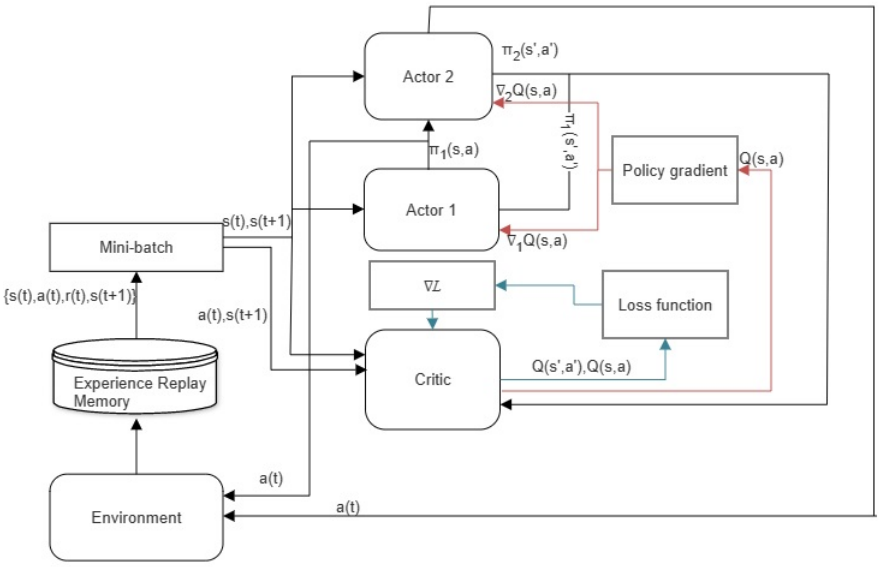

Figure 6.2: Sequential twin-actor critic algorithm for multi-stage slice admission control

6.6.2 Proposed Sequential Twin-Actor Critic Algorithm

As previously discussed in Chapter 1, slice admission control is a binary decision process: either admitting or rejecting a slice request. However, this method is incomplete as any subsequent tasks dependent on the admission decision must be initiated by an external process. This issue is addressed through the implementation of STAC, which utilizes a multistage procedure for both slice admission control and system parameter adjustment.

The STAC algorithm operates in two stages for slice admission control, each mapped to distinct action spaces. In the first stage, the action space comprises a decision denoted by $A^{cont} = \{a^{admit}, a^{rej}\}$. The second stage involves a resource adjustment phase represented by $A^{adj} = \{a^{dat}, a^{cpu}, a^{mem}, a^{int}\}$.

The STAC algorithm, illustrated in Fig. 6.2, employs a multi-stage action process for slice admission control utilizing a deep neural network (DNN), actor1 approximates a policy

function, denoted as $\pi_1(s, A^{cont})$, to predict whether a slice should be admitted or rejected. Here, the input to actor1 is the state at time slot t, denoted as $s(t)$, and the output is the probability of taking an action in the action space A^{cont} after observing the state $s(t)$.

Actor2, forming the second stage of the twin-actor network, consists of a Deep Neural Network (DNN) tasked with evaluating a policy based on the decision made by actor1, denoted as $\pi_1(s, A^{cont})$, and the current state $s(t)$. The prediction of actor1 serves as a bias for actor2, strongly influencing its decision-making process. The output of actor2 is the policy function $\pi_2(A^{cont}, (s, A^{adj}))$, representing the probabilities of taking an action in the second stage given the current state $s(t)$ and the previous action probability.

The critic component of the STAC algorithm evaluates the performance of the twin-actor section. It takes as input the current state, action, and the response from the twin-actor. The output of the critic represents the value function $Q(s', a'), Q(s, a)$, assessing the performance of the system.

This algorithm summarizes the STAC approach, where the agent observes the environment at each time-step during training. It performs admission control based on the current network status and slice request demand. If a slice is admitted, the agent performs resource allocation; otherwise, network adjustment is executed. The policy parameters are updated using the actor-critic method at the end of each time step.

6.6.2.1 Training STAC

The proposed STAC is a variant of DRL which employs two sets of actor DNN and one critic DNN. Once the replay buffer is full, the training of the twin-actor is initiated. The DPG

Algorithm 6 The STAC algorithm for slice admission control and resource allocation algorithm

Input actor1 and actor2 networks $\pi(s,a)$ and $\pi_2(s,a)$, critic network $Q(s, A^{cont}, a_2)$, the discount rate γ, the learning rates α_1 and α_2, the soft update ς
Output: Update network weights

STAC tranining
Initialisation : network parameters θ_1 and θ_2, critic parameter Ψ, target network parameters
for *episode* $= 1, 2...\mathbb{K}$ **do**
  Initialize state $s(k, \psi)$
  for each time step $t = 1, 2...\mathcal{T}$ **do**
    Choose random action in A^{cont}, choose action $A^{adj} = \pi_2(s, A^{cont})$
    Execute A^{adj}, move to state $s(t+1)$ and obtain reward $r(t)$ according to 6.28
    Store $s(t), a(t), r(t), s(t+1)$ in the replay buffer
    Retrieve $\mathcal{I}$ samples from replay buffer
    Update the loss function $\mathcal{L} = r(t) + \gamma$(critic policy output) $-$ (critic target output)
    Update critic parameters
    Calculate gradients: $\bigtriangledown_{\theta_1} \mathcal{J}_{\theta_1}$ and $\bigtriangledown_{\theta_2} \mathcal{J}_{\theta_2}$
    update actor1 and actor2 policy network parameters
    Update actor and critic target parameters
  end for
end for

method is employed to obtain the policy parameters θ_1 and θ_2 independently as

$$\bigtriangledown_{\theta_1} \mathcal{J}(\pi_{\theta_1}) \approx \sum_{s \in S} \bigtriangledown_{\theta_1} \log \pi_{\theta_1}(s) A_{\pi_{\theta_1}}(s,a) \tag{6.38}$$

and

$$\bigtriangledown_{\theta_2} \mathcal{J}(\pi_{\theta_2}) \approx \sum_{s \in S} \bigtriangledown_{\theta_2} \log \pi_{\theta_2}(s) A_{\pi_{\theta_2}}(s,a) \tag{6.39}$$

The subsequent step is to update the actor parameters as follows:

$$\theta_1(t+1) = \theta_1(t) + \bigtriangledown_{\theta_1} \mathcal{J}(\pi_{\theta_1}) \tag{6.40}$$

and

$$\theta_2(t+1) = \theta_2(t) + \bigtriangledown_{\theta_2} \mathcal{J}(\pi_{\theta_2}) \tag{6.41}$$

Once the parameters are updated, the aggregated action and the current state becomes the input to the critic. The critic parameter is denoted by ψ The target network of the critic obtains value approximation from the two target networks of the actors as inputs. The temporal difference error of an $n-th$ sample minibatch data is given by the current $n-th$ reward plus the weighted value approximation difference of the critic. The critic target parameter is then updated. The actor target parameters are finally updated using the DPG criterion described earlier. The training procedure is summarized in **Algorithm 6**.

6.6.2.2 Space and Time Complexity Analysis of STAC in Algorithm 6

The space complexity also determined by the memory required to store the network parameters and the replay buffer. The algorithm uses two actor networks and one critic network, along with their respective target copies. Since these networks are of fixed size (with a total of P parameters), their storage requirement is $O(P)$. Further, the replay buffer, which holds past experiences (each comprising a state, action, reward, and next state), requires $O(D)$ space, where D is the size of the buffer. Thus, the overall space complexity is $O(P+D)$, with the replay buffer often dominating in large-scale implementations.

The algorithm iterates over K episodes with T time steps per episode, leading to a total of $O(K \times T)$ iterations. At each time step, operations include selecting actions (which are typically $O(1)$), storing experiences, and sampling a mini-batch of I experiences from the replay buffer. The core computational cost comes from the neural network updates—each

requiring forward and backward passes through the networks. If each update takes $O(P)$ time (with P being the number of network parameters), then the mini-batch update incurs $O(I \times P)$ time per time step. Consequently, the overall time complexity for the training process is $O(K \times T \times I \times P)$, which simplifies to a linear relationship with respect to the total number of time steps when I and P are treated as constants.

6.7 Results and Performance Evaluation

In this section we provide a discussion for the simulation results for the study in this Chapter. We utilize Python 3 along with the corresponding TensorFlow libraries to build deep neural networks for the STAC algorithm. The algorithm is executed for 10,000 episodes, with each episode containing a variable number of time steps. Table 6.1 outlines the simulation parameters for slice admission and resource allocation using STAC. The simulation parameters table contain the hyper-parameters which are carefully tuned to fit the simulation environment.

The rest of the simulation parameters are chosen to reflect realistic network dynamics and resource constraints in a 5G slicing environment while enabling effective learning by the RL agent. For instance, a slice arrival rate (λ_c) of 15 and a processing rate (μ) of 20 indicate a moderately loaded system where demand is slightly below capacity, allowing the RL algorithm to manage queues efficiently. The configuration of 3 queues with up to 10 slices each introduces priority or service class differentiation, mirroring practical multi-service scenarios. Moreover, provisioning 960 RBs along with a downlink data rate range from 10MB/s (minimum) to 200 M/Bs (maximum) ensures that the simulation captures a broad

spectrum of radio resource allocations and quality-of-service levels, from baseline connectivity to high-demand applications.

On the resource management and economic side, the parameters further incorporate cost and capability factors that are critical for the RL-based decision-making process. The CPU grade ranging from 1 to 10 models heterogeneous computing capabilities, forcing the RL framework to optimally match processing power with slice demands. Pricing for communication, computing, and caching resources—set at 10 units/kHz, 10 units per CPU grade, and 20 units/MB respectively—introduces realistic economic incentives and trade-offs. These cost settings drive dynamics within the simulation, ensuring that the RL agent not only meets technical performance targets but also considers economic efficiency when allocating scarce network resources.

A comparative analysis between the proposed model and existing algorithms including conventional actor-critic(CAC), DQL, and Greedy based learning is performed. To validate our results in this study, the Twin-Actor Deep Deterministic Policy Gradient (DDPG)[115] and the conventional ADMM approaches are compared

Given the anticipated network fluctuations, our model's response is to adjust various network parameters, including the downlink data rate ($\mathcal{W}_b$), the computing grade (η_c^m), and the cache size (v_i). We regulated the downlink rate within the range of 20-100 MB/s. As previously mentioned, the computing grade ranges from 0 to 10, while the cache hit ratio is defined by the probability of hit $Pr(v_i) \in (0, 1)$ as in Equation 6.11. The InP aims for seamless system performance by automatically adjusting these parameters based on data obtained from temporal network states, as described in Equation 6.26.

To simplify the deep neural network (DNN) architecture, we performed dimensionality reduction using KPCA. The number of principal components Υ_Θ was limited to match the DNN inputs, set at 56. The DNN architecture consists of two actor networks and their respective target networks. Additionally, a single critic network and its corresponding target network are included. The simplified architecture is illustrated in Figure 6.2. Each network is fully connected, comprising only two layers of 50 neurons each.The activation function adopted for the first actor is binary step while for the second actor we have adopted rectified linear unit (ReLU). Table 6.1 contains the simulation parameters.

Table 6.1: Simulation parameters

Parameter	**Value**
Discount rate γ	0.09
Learning rate α	0.01
Soft update ξ	0.01
Zipf slope Ψ	0.44
Slice arrival rate λ_c	15
Slice processing rate μ	20
The number of queues q	3
The number of slices per queue	10
The number of resource blocks (RBs)	960
Minimum downlink data rate $\mathcal{W}_b min$	10MB/s
Maximum downlink data rate $\mathcal{W}_b max$	200MB/s
CPU grade η_c^m	1-10
Price for communication resource	10units/kHz
Price for computing resource ζ_c	10 units/grade
Price for caching resource	20 units/MB
Replay Buffer Size	1000
Mini-batch	56

In Figure 6.3, the overall utility is depicted, which is derived from the cost of resources considering 60 admitted slices. This utility reflects the optimization process outlined in

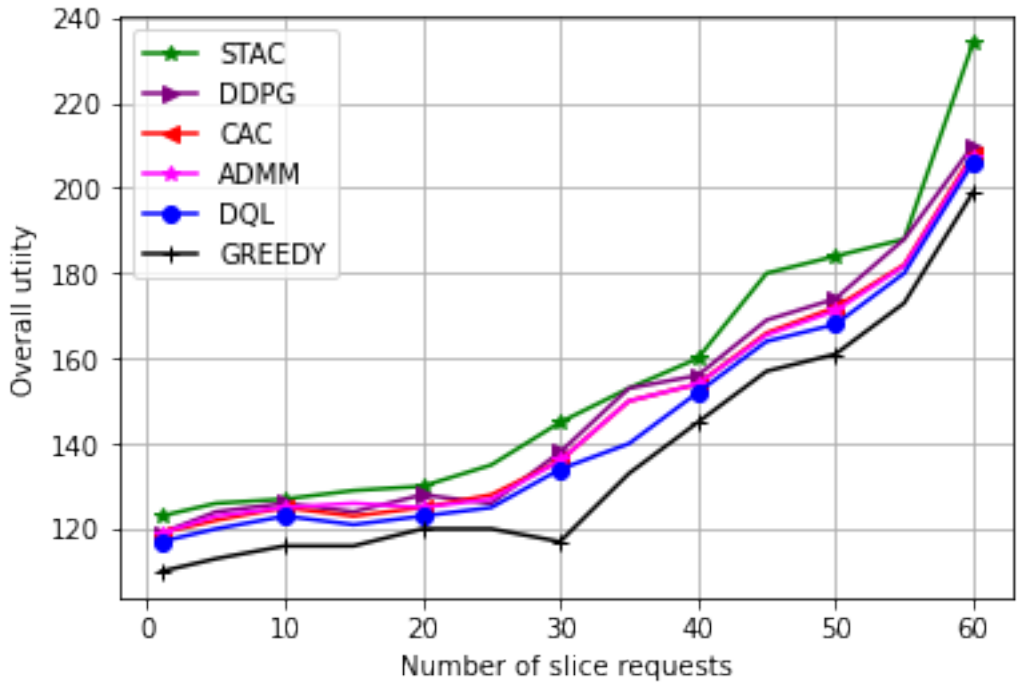

Figure 6.3: Overal utility vs slice request for complete allocation

Equation 6.24. The graph illustrates a steady increase in utility across the tested algorithms. Notably, the proposed STAC algorithm demonstrates an average utility improvement of approximately 4% for the admitted slices compared to the baseline. This improvement is attributed to the enhanced accuracy of admission control in the initial network stage and the autonomous resource adjustment facilitated by the latter stage of the STAC algorithm. Moreover, this automatic adjustment feature enables the algorithm to effectively respond to non-deterministic network fluctuations, enhancing its overall resilience. Both the DDPG, CAC, ADMM and DQL have an average difference of 2% between them for the considered admitted slices. The Greedy approach has the worst performance with a fall-off of 6% due to sub-optimal convergence. We further examine the system performance by evaluating the communication resource blocks (RBs). Figure 6.4 illustrates the utility plotted against the

allocated communication RBs. In this analysis, we allocated 1100 RBs across 60 admitted slices. Specifically, we considered the 5G FR1 New Radio (NR) configuration for this investigation. We selected a cell bandwidth of 100MHz sliced at a single carrier RB of 30KHz.

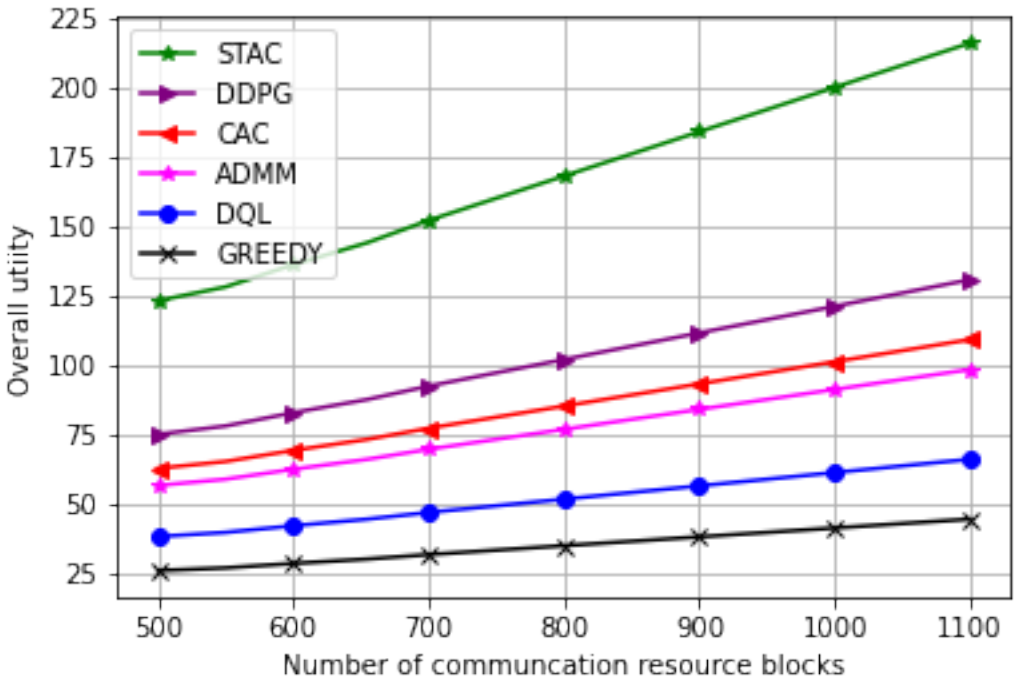

Figure 6.4: Overal utility vs slice request for communcation RB

This configuration does not support an RB of 15KHz. From this plot, it is observed that the proposed algorithm outperforms the Baseline with an average improvement of over 50% across 1100 slice RBs. The linear increase in communication RBs depicts strict adherence to the standard NR FR1 slicing criterion in 5G wireless networks. The allocation of sliced computing resources involved determining the number of CPU cycles required by the cache for each task. Additionally, it's important to note that the grading system associated with the vCPUs is computed based on the number of cycles required per slice task. Figure 6.5 represents the utility plot versus the computing resources allocated for 60 admitted slices.

The STAC algorithm demonstrated an overall improved utility, as VNOs tends to benefit from the balanced resource allocation procedure outlined in 6.19.

In Figure 6.6, the slice request probability is plotted against the cache size. Generally, with minimal cache availability, the hit ratio is subdued. However, as the cache size increases, there is a noticeable rise in the hit ratio, leading to a higher probability of accepting a slice. With more than 15 gigabytes of cache available, the STAC algorithm consistently maintains a probability of slice acceptance of over 90%, in contrast to the inconsistencies observed in the compared strategies. The InP allocates memory resources based on the hit criterion. The STAC algorithm's agent is optimized to provide a match for each memory resource request with minimal adjustment, as depicted in 6.17 and 6.18.

Finally, Figure 6.7 illustrates the effect of data dimensionality reduction on CPU training time. In this regard, we employed KPCA for non-linear dimensionality reduction. The simulation results clearly demonstrate that by reducing the data dimension, the training time is significantly decreased. When testing with a maximum data dimension of 1000, it took approximately 25 minutes to reach an acceptable near optimality without dimensionality reduction (No Dim_reduction), compared to only 10 minutes with KPCA.

As outlined in 6.33, we matched the principal components Υ_{Θ} to 56 data sets, which was the minibatch size employed during the training process. It's important to note that the initial calculation of both the eigenvectors and eigenvalues is computationally intensive. However, as the data dimensionality increases, the benefits of reduced dimensions outweigh these calculations, leading to reduced CPU time for training larger data sets.

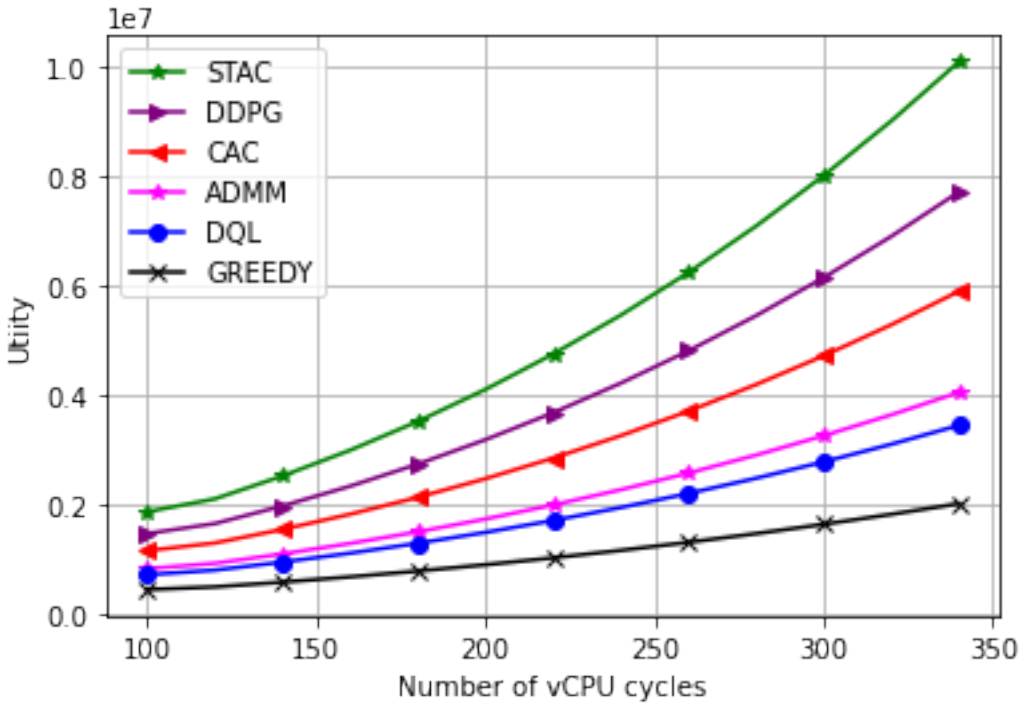

Figure 6.5: Overal utility vs slice request for computing resources

Considering the computational complexity of PCA and KPCA is crucial in determining the training efficiency of the agent. Both dimensionality reduction techniques are known to maintain statistical features despite reducing the principal components. In general, when a dataset with size m and n features is used, PCA has a computational complexity of $\mathcal{O}(m^2n + m^3)$, while KPCA has a computational complexity of $\mathcal{O}(m^2)$ [133].

To estimate the computational complexity of STAC, we need to consider the rate of convergence of a parameterized dual-stage actor-critic. According to [134], the rate of convergence of a critic can be characterized for a k-step policy evaluation as:

$$\mathbb{E}||\Xi_k - \Xi_*||^2 \leq \mathcal{O}(k^{-2/3}) \tag{6.42}$$

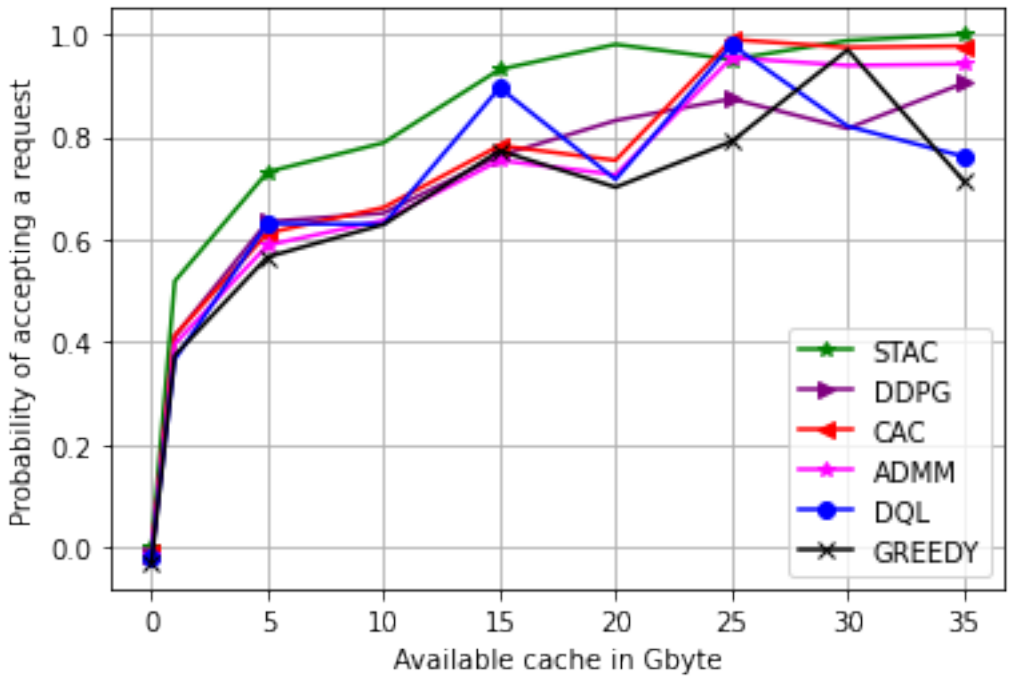

Figure 6.6: Probability of accepting a slice considering the available cache size

where Ξ_k and Ξ_* are the actual and estimated TD parameters for estimating the value functions of the critic. The overall convergence rate for both the critic and actor stages of STAC is therefore estimated by

$$\mathbb{E}||\Xi_k - \Xi_*||^2 \leq \mathcal{O}(k^{-2/3} + 2k^{-5/2}) \tag{6.43}$$

In conclusion, the comparison presented in Figure 6.7 highlights the computational complexity difference between dimensionality reduction with KPCA and no dimensionality reduction. The experimental results demonstrate an exponential trajectory for both strategies. It's evident that dimensionality reduction with KPCA leads to reduced CPU time when analyzing a complex environment, as observed from the experimental attributes tested.

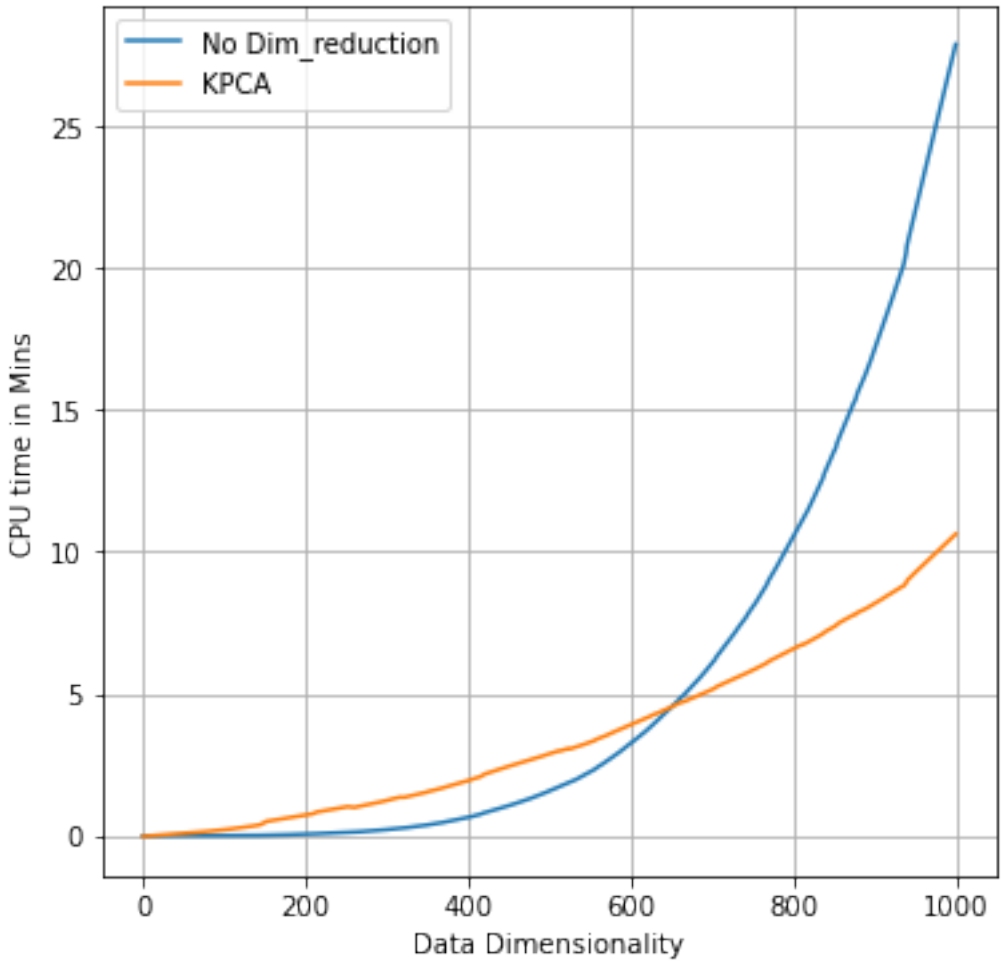

Figure 6.7: Time Complexity Analysis between PCA and KPCA

6.8 Chapter Summary

This chapter presented a slice admission control optimization method called STAC, based on a multi-stage actor-critic approach. Through an evaluation of 5G resource allocation for cloud-based slicing, we have demonstrated that this method enhances network resilience by dynamically adjusting communication, computing, and cache resources which eventually improves overall utility. Performance evaluation involved comparing the expected utility of our algorithm against DDPG,ADMM and other variant schemes. Across all resource

domains, our algorithm consistently outperforms the alternatives, showcasing its effectiveness in optimizing slice admission control.

Chapter 7

Conclusion and Future Work

This chapter present highlights of each chapter and a summary of main contributions, results are summarized inline with set objectives.

7.1 Summary and Main Contribution of the Study

This thesis has presented a study on SAC for 5G wireless network. In this study, cross domain resource allocation has been considered, specifically focusing on RAN/CRAN, TN and CN. Resource formulation have been developed for each domain. Complexity analysis have been attempted, optimization of slicing objective and introduction of a novel multistage RL for solving multi-dimensional state spaces. The contribution of this thesis are covered in Chapters 3-6.

This thesis has been organized in 7 chapters with Chapter 1 introducing the thesis by presenting research motivation, problems state, research objectives, research questions, research contributions and publications. Chapter 2 presents the research background by formulating complexity analysis for classical network resource slicing. Specially, comprehensive analysis is performed on SCA, ADMM, BB,heuristic approaches and briefly introducing RL.

Chapter 3 investigates RL techniques for network SAC. RL modules namely state space, action space, reward formulation and state transition models namely (MDP,SMDP) are discussion focusing mainly of network resource formulation and slicing. Admission control

policy evaluation schemes are discussed presenting complexity levels from Q-learning to function approximation.

Chapter 4 investigates the effect of multi-queue multi-server scheduling on slice admission control. The scenario presents a multi-InP where each InP's slice request is served by one of the multiple servers. The study in this Chapter adopts an M/M/C server system for efficient queue management. The Chapter contributions are stated as follows.

- The transient probabilities are derived and proved. By adopting SDE, the probability of slices transiting from one queue is presented. It is established that, no closed form solution to the SDEs exist but a closest estimate may obtain by solving the SDE representing the transient probabilities.
- The cost function for slice queue occupancy is derived as the main objective in optimizing slice scheduling. Efficient slice scheduling is expected to minimize the cost function C_F
- The overall scheduling cost is derived a function of both latency and resource related costs.

This chapter has the following results

- The proposed slice queuing model improves overall efficiency on slice processing as transition from a server with low processing capability to a server with high processing capability.

- The proposed DQL provides improved resource prediction over a 24 hour period compared to similar schemes. This is particular to the radio resource slicing which comprises the RB logically sliced by the BBUs.

- The resource cost prediction is shown to align with user behavior considering when resource availability improves. This is a result of improved accuracy in prediction over the set period.

Chapter 5 investigated resource allocation fairness and how to optimized overall utility during SAC. The investigation proceeds to derive a model for improving incentive compatibility through NASH equilibrium. To tackled the problem of fairness, resource auctioning game has been employed in this Chapter. Resource modeling for each domain has been presented and applied as elements of SAC. Policy evaluation with Q-Learning has been employed and later retrieved. The following are the Chapter contribution.

- Proposed a double side online auction model for a multi-stage SAC where bidders and auctioneers are allowed to express interests which enhances competition.

- Modeled a double sided bidding information as a learning state-space for the proposed auction game.

- Proved that, by implement Nash equilibrium, untrue preferences are eliminated thereby improving fairness in auction based SAC.

The insights gained from the Chapter are listed as follows.

- The probability of slice admission is accurately mapped to resource availability by the training agent in all the three compared instances namely: LBA,GBA and RBA.

- The fairness comparison is presented in this Chapter. By adopting the Jain's fairness index measurement, each strategy is evaluated based on how fair it performed SAC. The LBA is established to have an improved fairness of 62.8% compared to 27.8% and 4.0% of RBA and GBA respectively. The fairness evaluation is a degree of measures where each strategy is interrogated on its ability to achieve a trade-off between greedy and exploration based admissions.

- The long term utility predicted by LBA, GBA and RBA are presented showing consistency with the stipulated objective. The LBA algorithm improves long term utility as opposed to GBA and RBA which shows that the projected utility dips in the long run.

- The QoS determination is also presented for the three schemes. With increasing slice admission enabling network resource consumption, the QoS which is a measure of the ability by the orchestrator to meet a slice requirement is calculated. LBA is shown to prevent negative QoS

Chapter 6 provides an RL based SAC scheme that enables network resilience in 5G. The scheme employs a dual RL known as STAC to evaluate slice admissibility and then employed to adjust network parameters to prevent complete QoS deterioration. In this scenario, An InP plays a pivotal role in overseeing a slice package by ensuring several key parameters: maintaining the user count within specified limits, preserving spectral resource availability, managing cloud resources, preventing the orchestration of duplicate or incompatible slices. In this Chapter, the following contributions are noted.

- Derived the formal resource model containing, the spectral, computing and storage resources.
- Constructed the optimization problems for optimizing data rate and computing resources.
- The complexity of the learning state-space is tackled by adopting KPCA to identify principal components.
- Constructed a reward function suitable for optimizing stipulated objectives for enhancing resilience and the solved by a multi-stage RL known as STAC.

The results from Chapter 6 are stipulated below.

- The overall utility is plotted against complete resource allocation. This is determined by obtaining a normalized resource allocation per slice and calculating the running sum over 60 slices. The results show, that STAC has 6% improved utility over the total number of request evaluated.
- Spectral RB sliced on 5G FR1 numerology is also presented. The overal RB covered 100MHz with each RB slice at 30KHz. By calculating the overall utility obtained when resource adjustment due to resilience maintenance is plotted, it is observed the STAC still maintains a 50% overall improvement above Baseline.
- Computing resources in terms of vCPU cycles are normalized and allocated per slice task. The plot shows STAC maintaining a superior performance over the Baseline.

- The determination of network resilience is calculated based on the probability of a slice being accepted considering the available cache. This is one of the results presented. As the available cache improves so does the probability. The proposed STAC still out performs the compared schemes in terms of hit ratio.

The Appendices provide further research contribution and results by determining the application of SAC in improving QoS for 5G enabled mMTC setup. The study investigates effect of SAC in anomaly aware clustered multi-node mMTC network. The research employs an RL scheme known as DS-DQL for admission control and anomaly detection. The contributions from the appendices include a complete model for anomaly-aware multi-node mMTC network.

For the RL adoption, multiple action-space is required and constructed, this is implemented on DS-DQL scheme. Results show that DS-DQL improves F1 score compared to the Q-learning and DQL for all the sets of anomalies.

7.2 Suggestion for Future Work

Although 5G network slicing is being gradually implemented, there still open challenges which still require further investigation. Some of these open research areas are given below:

- In this thesis, RAN,TN and CN resource domains and slicing have been considered in slice admission control, it would be highly interesting to investigate the effect of non-terrestrial domain on the overall slicing objective.

- Application of double ended resource auctioning has been implemented and results showed that when combine with Q-learning, a higher degree of fairness is achieved. This scenario can be extended to the employment of function approximation in DRL and a comparison made on the results.

- Effect of $M/M/C$ queuing in resource scheduling was investigated, however a comparison with similar queuing schemes such as $M/M/C/K/K$, $D/M/1$ and $\mathrm{M}/E_k/1$ was not investigated. This is an interesting gap that can still be tackled by researchers.

- While employing multi-stage RL to solve slice admission control problem and improve network resilience a dual stage RL scheme known as STAC was introduced. An interesting endeavor would be to investigate if it is possible to construct a triple stage RL and apply it in resource slice in 5G and beyond

Appendices

A Appendix I

A.1 Extending Slice Admission Control to Anomalous Massive Machine Type Communication with RL

mMTC networks are designed to facilitate the Internet of Things (IoT), where millions of low-powered nodes transmit stochastic telemetric payloads to centralized or decentralized controllers [9]. These IoT systems serve diverse domains, including smart agriculture, factories, and time-critical services, enabling efficient and reliable production [135]. However, faults and anomalies within these systems can lead to significant repercussions, causing disruptions in production and resulting in financial losses. As the expectation of accommodating more than one million devices per square kilometer in 5G networks becomes a reality, the influx of massive traffic flow directed towards a centralized controller is anticipated [136]. However, this surge can lead to faults and anomalies becoming increasingly granular and complex to detect. Traditional methods that rely on deterministic and regular data may not be suitable for detecting faults in these scenarios[137]. Several schemes have been employed to perform anomaly detection with limitations, see[138] [135][139][139][140][141][142], Appendix A-F provides a new approach leveraging slice admission control.

B Appendix II

B.1 Anomaly Aware Clustered Multi-node mMTC Network

The radio network comprising nodes adopts the Narrowband IoT (NB-IoT) access scheme as per 3GPP Release 13 standards. Typically, this scheme utilizes sub-200kHz bands in standalone operation, independent of LTE and GSM carriers.

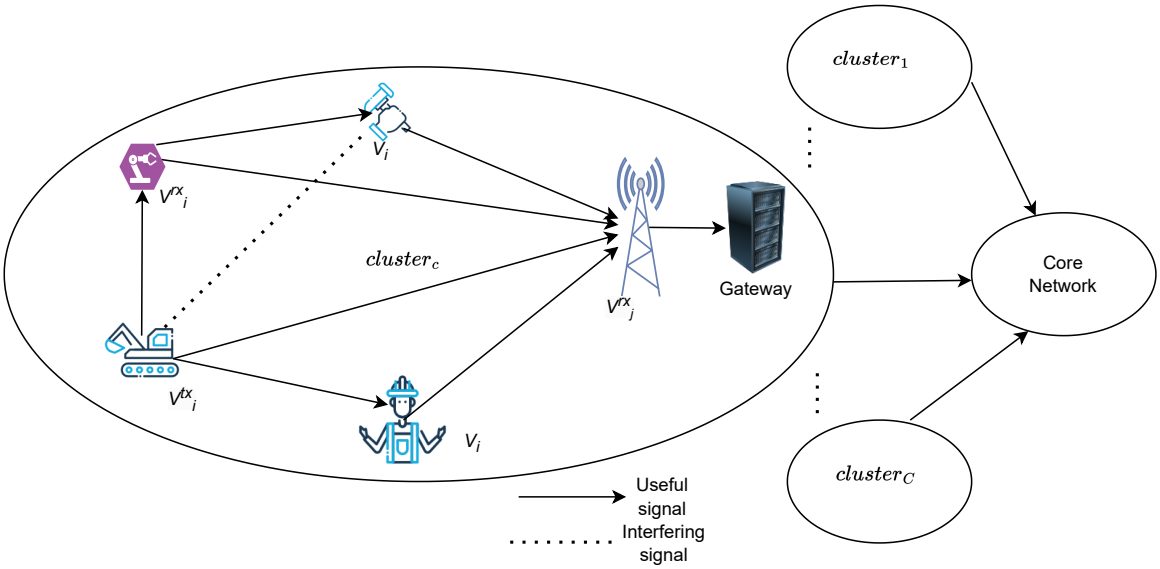

Figure A-1: System model with communication and interference links between devices

The system also adheres to a standard architecture for massive Machine Type Communication (mMTC) based on OpenFog [120][121]. In this setup, a cluster of nodes running IoT services sends data to a shared nano data center, also known as an edge node, located within the base station [138]. Each nano data center is connected to a gateway, which manages the data flow from the local edge fog. It is assumed that the local edge fog has a direct link to the core network, although other fog nodes may exist between the edge fog and the

core. Each cluster is considered to exhibit similarities with any existing co-cluster, thereby providing homogeneous data.

At the core network, an orchestrator with the capability of performing slice admission control and anomaly detection is deployed. However, there is also the possibility of conducting federated anomaly detection at the edge fog. The set of events that can trigger anomalies are as follows:

1. Congestion: This anomaly occurs when many devices attempt to communicate through a highly constrained uplink in the bottom-up setup of mMTC architecture, leading to inconsistent jitter.

2. Energy Usage: mMTC devices rely on battery power. Excessive power transmission implies inefficient energy usage, meaning that devices are consuming more energy than necessary.

3. Radio Interference: mMTC devices utilize open radio access to communicate with edge nodes. Due to the homogeneity in the allocated communication bands, severe interference may occur. While interference cannot be completely eliminated, extreme cases must be flagged as anomalies.

Appendix III provides the interference modeling considering the consumed power from each base station and adjacent clusters.

C Appendix III

C.1 Interference Modeling for Multi-node Clustered mMTC Network

Considering $n = \{1, 2, \ldots |\mathcal{N}|\}$ as the number of devices per cluster. Each cluster c also belongs to the set of clusters served by a single core processor, this is given by $c \subseteq \mathcal{C}$. The core network aggregates data from all clusters and perform SAC.

To identify jitter anomalies, the orchestrator maintains the average packet delay from each end node $i \in I$ in a cluster c as a reference. Any fluctuations beyond the predefined threshold jitter are flagged as anomalies. This is given by:

$$t_{i,c}^{delay} = t_{i,c}^{up} \pm \delta t \tag{1}$$

where $t_{i,c}^{up}$ is the average uplink packet duration from node i in cluster c and δt is the time delay variance.

The power consumption of each node can be approximated by evaluating the energy consumption of its transmitter components, including signal processing, transceiver, and power amplifier units. This estimation assumes that each end node is equipped with a micro antenna for upward communication. The dynamic power model is derived from the power consumption under no load and the power consumption under network load conditions. as[143]:

$$P_{i,c}^{load} = m_{sec}(m_{ant}(P_{amp} + P_{_{trans}}) + P_{proc}) \tag{2}$$

where m_{sec} and m_{ant} are the number of antennas and sectors respectively, P_{amp}, P_{trans} and P_{proc} are the internal amplifier, transmitter, and processing power of the end node respectively. The total consumed power per cluster can be estimated from 2 as

$$P_{cluster} = \sum_{i \in \mathcal{N}} P_{i,c}^{load} \tag{3}$$

Due to the limited spectral resources, devices within mMTC networks typically share common frequency bands [144]. Communication can occur either between mMTC devices or between an mMTC device and a central node, leading to potential interference scenarios. It is assumed that the channel gain remains constant throughout a transmission period, whether between devices or between a device and the central node, although it may vary independently between symbols, a condition known as quasi-static. Let the communication between a device-to-device (D2D) pair be represented by $\mathcal{V}_i$, and the communication between an mMTC node and the central node/edge node be denoted by $\mathcal{V}_j$.

The network model adopted assumes an Okumura-Hata model as described in [145], where the signal-to-interference ratio between a D2D pair can be estimated as outlined in [146]:

$$\gamma_{\mathcal{V}_i} = \frac{P_{i,c}^{\mathcal{V}_i} \cdot (g^{\mathcal{V}_i})^2}{\sigma^2 + \sum_{i' \neq i \in \mathcal{N}} P_{i' \in \mathcal{N},c}^{\mathcal{V}_{i'}} \cdot (g^{\mathcal{V}_{i'}})^2} \tag{4}$$

Where $P_{i,c}^{\mathcal{V}_i}$ is the transmitter power from node i in cluster c on link $\mathcal{V}_i$, $g^{\mathcal{V}_i}$ is the channel gain in link $\mathcal{V}_i$. σ is the average noise power. $P_{i,c}^{\mathcal{V}_{i'}}, g^{\mathcal{V}_{i'}}$ is the interfering power and channel gain from node i' on link $\mathcal{V}_{i'}$ respectively. For the link between a node and the edge node,

the signal to interference can be estimated as.

$$\gamma_{\mathcal{V}_j} = \frac{P_{i,c}^{\mathcal{V}_j}(g^{\mathcal{V}_j})^2}{\sigma^2 + \sum_{j' \neq j \in \mathcal{N}} P_{i' \in \mathcal{N}, c}^{\mathcal{V}_{j'}}(g^{\mathcal{V}_{j'}})^2} \tag{5}$$

Where $P_{n,C}^{\mathcal{V}_j}$ is the transmitter power from node n in cluster C on link $\mathcal{V}_j$, $g^{\mathcal{V}_j}$ is the channel gain in link $\mathcal{V}_j$. σ is the average noise power. $P_{n,C}^{\mathcal{V}_{j'}}, g^{\mathcal{V}_{j'}}$ is the interfering power and channel gain from node n' on link $\mathcal{V}_{j'}$. The overall signal to noise interference ratio estimation can be represented as.

$$\gamma = \sum_{i \in \mathcal{N}} \gamma_{\mathcal{V}_i} + \sum_{j \in \mathcal{N}} \gamma_{\mathcal{V}_j} \tag{6}$$

D Appendix IV

D.1 RL Solution to Anomaly Detection Leveraging SAC

SAC and anomaly detection represent a combinatorial problem addressed through a novel dual-stage double deep Q-learning (DS-DDQL) approach. The DS-DDQL proposed here consists of two stages of DDQL, where the initial stage handles SAC and the subsequent stage deals with anomaly detection. When a slice request is made by a virtual network operator (VNO), a decision must be made regarding its admission. This decision hinges on the impact of the slice on the quality of service (QoS) level of the network. In the following section, we define the environment attributes in DS-DDQL.

D.2 State Space

The state space comprises the network status and the slice request. The training agent observes the environment at time slot t and collects a state s, defined by:

- $s_{req}^{pow}(t)$: The average power level per node as requested by the VNO
- $s_{req}^{del}(t)$: This is the acceptable average slice packet delay variation requirement sent by the VNO
- $s_{req}^{int}(t)$: The maximum link interference stated by the VNO
- $s_{act}^{pow}(t)$: The actual average power transmitted per node.
- $s_{act}^{del}(t)$: This the actual average packet delay
- $s_{act}^{int}(t)$: This is the actual link interference as seen by the agent.

The state space can be seen as a dual vector parameter given by $s = \{s_{req}(t) = [s_{req}^{del}(t), s_{req}^{del}(t), s_{req}^{pow}(t)], s_{act}(t) = [s_{act}^{del}(t), s_{act}^{pow}(t), s_{act}^{jit}(t)]\}$. Where $s_{req}(t)$ is the input to the initial stage of DS-DDQL agent while $s_{act}(t)$ influences the anomaly detection. The decision to admit a slice and to classify a status as an anomaly are based on the states, with each state being mapped to the predefined models in Equations 1, 2, and 4. A slice is admitted if the following constraints are achievable, considering state $s_{req}(t)$:

$$\frac{1}{\mathcal{N}} \sum_{i \in \mathcal{N}} P_{i,c}^{req} \geq P_c^{min} \tag{7}$$

where $P_{i,c}^{req}$ is the requested average power per end node in a cluster and P_{c}^{min} is achievable power to sustain a slice. The requested time delay constraint is denoted by

$$t_{up}^{req} \pm \delta t \leq t_{up}^{max} \tag{8}$$

where t_{up}^{max} is maximum time delay acceptable for the slice. On a similar note, the noise interference level must be constrained in order to maintain a slice. We have

$$\sigma^2 + \sum_{j' \neq j \in \mathcal{N}} P_{n' \in \mathcal{N}, C}^{\mathcal{V}_{j'}} g^{\mathcal{V}_{j'}} \leq p_{int}^{req} \tag{9}$$

where p_{int}^{req} is the acceptable interference power level per cluster. To identify a status as an anomaly, the agent examines the second set of states denoted by $s_{act}(t)$. To avoid considering the average cluster power level as anomalous, we define an upper bound constraining the average transmitted power per cluster as:

$$P_{av} = \frac{1}{N} \sum_{i \in \mathcal{N}} P_{i,c} + p \leq p_{max} \tag{10}$$

where p_{max} represent the maximum power level in watts allowable in a cluster. The expected jitter on the uplink is only upper bounded because a reduced average delay is advantageous and cannot be considered an anomaly. We modify Eq. 8 as $t_{up}^{req} \pm \delta t^{act} + \tau \leq t_{up}^{max}$ where δt^{act} is the average actual jitter measured on the uplink. For the interference anomaly, the edge node V_i^{tx}, which has the capability of estimating the average interference level, relays information on spectrum sharing and average interference on a selected cluster. The actual

signal-to-noise-plus-interference ratio is also constrained. The agent can label this parameter as an anomaly if:

$$\gamma_{av} = \frac{\gamma}{N} + \Gamma \geq \gamma_{\mathcal{V}_i}^{th} \forall i \in N \tag{11}$$

where $\gamma_{\mathcal{V}_i}^{th}$ is the threshold level of SINR that can be sustained in the duration a slice is deployed. The parameters p, τ, and Γ are carefully chosen signal adjustments to create a gap between the normal value and an anomaly.

D.3 Action Space

The proposed algorithm is designed as a multistage action process where the initial action space is to perform SAC, while in the next stage, a state is labeled as either belonging to the anomalies set or not. We present these action spaces as $\mathbf{a} = \{a^{ad}(t), a^{an}(t)\}$, denoting the actions for admission control and anomaly detection, respectively. Specifically, $a^{ad}(t) = \{1, 0\}$ indicates slice admission or rejection. On the other hand, $a^{an}(t)$ has eight possible actions, defined as follows:

- $a_0^{an}(t)$: The agent labels the network status as having no anomaly
- $a_1^{an}(t)$: There is anomalous level of interference in the cluster.
- $a_2^{an}(t)$: There is an anomalous power transmission in the cluster.
- $a_3^{an}(t)$: There is both anomalous power and interference levels in the cluster.
- $a_4^{an}(t)$: There is anomalous jitter in the cluster.
- $a_5^{an}(t)$: There is anomalous jitter and signal interference

- $a_6^{an}(t)$: There is anomalous jitter and power level in the cluster
- $a_7^{an}(t)$: There is anomalous jitter, power and interference in the cluster

D.4 The reward functions

The agent strives to maximize the reward considering each action. An action $a^{an}_{k=[0,1,2..K=7]}$ attracts a unique reward to update the long term action value. The reward function is carefully formulated to accommodate each action taken by the agent this is given by;

$$r(t) = \begin{cases} \beta + a^{ad}(t)\left(\frac{\gamma_{av}}{P_{av}-\log t_{up}}\right)^2 & \text{if } a^{an}(t)=0 \\ \beta + a^{ad}(t)\left(\frac{1}{\gamma_{av}+P_{av}-\log t_{up}}\right)^2 & \text{if } a^{an}(t)=1 \\ \beta + a^{ad}(t)\left(\frac{\gamma+P}{-\log t_{up}}\right)^2 & \text{if } a^{an}(t)=2 \\ \beta + a^{ad}(t)\left(\frac{P_{av}}{\gamma_{av}-\log t_{up}}\right)^2 & \text{if } a^{an}(t)=3 \\ \beta + a^{ad}(t)\left(\frac{\gamma_{av}-\log t_{up}}{P_{av}}\right)^2 & \text{if } a^{an}(t)=4 \\ \beta + a^{ad}(t)\left(\frac{-\log t_{up}}{P_{av}+\gamma_{av}}\right)^2 & \text{if } a^{an}(t)=5 \\ \beta + a^{ad}(t)\left(\gamma_{av}+P_{av}-\log t_{up}\right)^2 & \text{if } a^{an}(t)=6 \\ \beta + a^{ad}(t)\left(\frac{P_{av}-\log t_{up}}{\gamma_{av}}\right)^2 & \text{if } a^{an}(t)=7 \end{cases}$$

The constant β is a hyper-parameter chosen to make the reward function more positive.

E Appendix IV

E.1 Dual Stage Double Deep Q-learning Algorithm

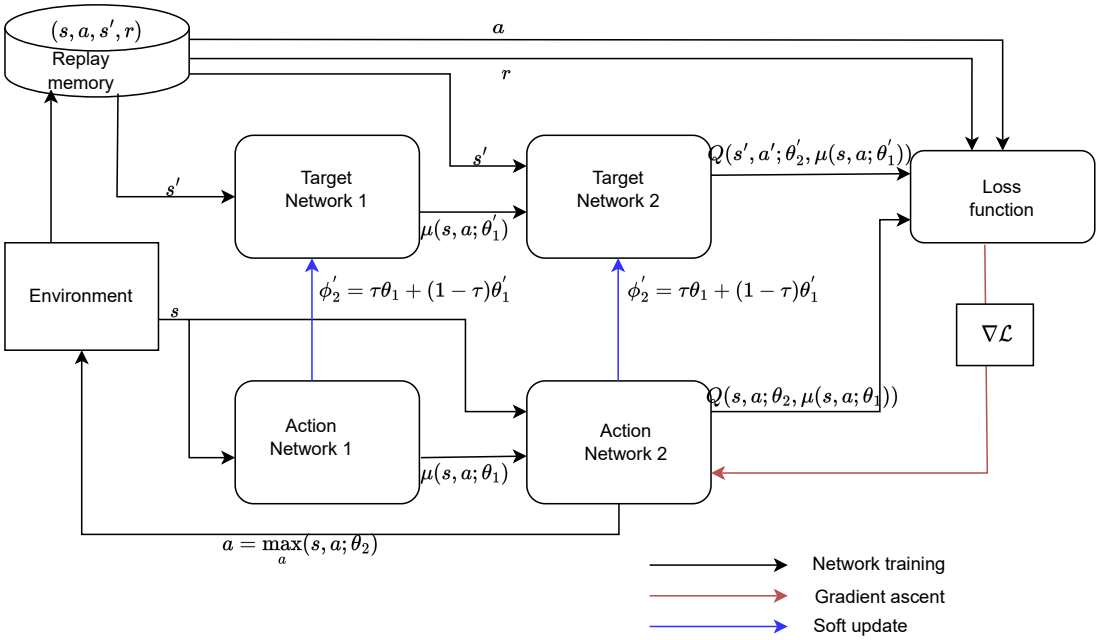

Figure A-2: Dual stage double deep Q learning architecture

The proposed algorithm presented in **Algorithm 1** relies on deep neural network to estimate both the admission policy in the initial stage and the value function for anomaly detection in the latter stage. Fig A-2 represents the DS-DQL architecture. To optimize the algorithm we set up an objective function given by the temporal difference expressed as

$$\max_{s,a} \mathcal{R} = \sum_{k=1}^{K} \sum_{t=0}^{T-1} (r(t) + \zeta \max_{a} Q(s_k', a'; \theta_2', \mu(s_k, a; \theta_1')) - Q(s, a; \theta_1, \mu(s_k, a; \theta_2))^2). \tag{12}$$

The gradient of the objective function parameterized by θ_1 and θ_2 is obtained by

$$\nabla\mathcal{L}(\theta_1) = \sum_k \nabla Q(s, a; \theta_1)|_{(a(t)=(\mu(s_k, a^{ad}))} \tag{13}$$

$$\nabla\mathcal{L}(\theta_2) = \sum_k \nabla Q(s_k, a; \theta_2,)|_{a(t)=(Q(s_k, a^{ad}))} \tag{14}$$

Where $\mathcal{L} = Q(s'_k, a'; \theta'_2, \mu(s_k, a; \theta'_1)) - Q(s, a; \theta_1, \mu(s_k, a; \theta_2))^2)$. Each of the action network parameters are updated based on the gradient accent as

$$\theta_1(t+1) = \theta_1(t) + \alpha\nabla\mathcal{L}(\theta_1) \tag{15}$$

and

$$\theta_2(t+1) = \theta_2(t) + \alpha\nabla\mathcal{L}(\theta_2) \tag{16}$$

The target networks are updated through a soft update as $\phi'_1 = \tau\theta_1 + (1-\tau)\theta'_1$ and $\phi'_1 = \tau\theta_1 + (1-\tau)\theta'_1$

The two stages of deep neural networks are utilized to estimate both the action policy and the action value respectively. The step by step procedure in DS-DDQLThe proposed algorithm presented in **Algorithm 1** relies on deep neural network to estimate both the admission policy in the initial stage and the value function for anomaly detection in the latter stage. Fig A-2 represents the DS-DQL architecture. To optimize the algorithm we set

up an objective function given by the temporal difference expressed as

$$\max_{s,a} \mathcal{R} = \sum_{k=1}^{K} \sum_{t=0}^{T-1} (r(t) + \zeta \max_{a} Q(s_k', a'; \theta_2^{'}, \mu(s_k, a; \theta_1^{'})) \\ -Q(s, a; \theta_1, \mu(s_k, a; \theta_2))^2). \tag{17}$$

The gradient of the objective function parameterized by θ_1 and θ_2 is obtained by

$$\nabla \mathcal{L}(\theta_1) = \sum_{k} \nabla Q(s, a; \theta_1)|_{(a(t)=(\mu(s_k, a^{ad}))} \tag{18}$$

$$\nabla \mathcal{L}(\theta_2) = \sum_{k} \nabla Q(s_k, a; \theta_2,)|_{a(t)=(Q(s_k, a^{ad}))} \tag{19}$$

Where $\mathcal{L} = Q(s_k', a'; \theta_2^{'}, \mu(s_k, a; \theta_1^{'})) - Q(s, a; \theta_1, \mu(s_k, a; \theta_2))^2)$. Each of the action network parameters are updated based on the gradient accent as

$$\theta_1(t+1) = \theta_1(t) + \alpha \nabla \mathcal{L}(\theta_1) \tag{20}$$

and

$$\theta_2(t+1) = \theta_2(t) + \alpha \nabla \mathcal{L}(\theta_2) \tag{21}$$

The target networks are updated through a soft update as $\phi_1^{'} = \tau\theta_1 + (1-\tau)\theta_1^{'}$ and $\phi_1^{'} = \tau\theta_1 + (1-\tau)\theta_1^{'}$

The two stages of deep neural networks are utilized to estimate both the action policy and the action value respectively.

F Appendix IV

F.1 Simulation Outcome for the implemented DS-DDQL

The simulation parameters are provide in Table F.1. After training the model in 2000 episodes with 1000 data-sets, we sample 100 with 5% anomaly per classification. We present the model performance against conventional DQL and Q-learning.

Table F.1

Hyper parameters	
Parameter	Value
Discount rate	0.09
Learning rate	0.001
Maximum exploration rate	1
Minimum exploration rate	0.001
Exploration decay rate	0.01
Number of episodes	2000
Time step Δt	1 sec
Other parameters	
Parameter	Value
Number of data-sets	1000
Normal transmitted power range	10dBm - 20dBm
Normal packet delay range	10ms - 20ms
Normal SINR range	40dB+

The simulation results provide visualized comparison between our proposed model, DQL and Q-learning. In Fig 3 the reward distribution is presented, the proposed algorithm provide a tightly packed cumulative reward distribution compared to the two baseline.

Q-learning depicts worst performance with a larger third quartile representing $\approx 60\%$ of the reward values above the median accompanied with a wider whisker and numerous outliers. Fig 4 to Fig 7 denotes the F1-score [147] which is a metric deployed to evaluate the performance of the model as:

$$F1\_score = 2 \times \left(\frac{precision \times recall}{precision + recall} \right)^2 \tag{22}$$

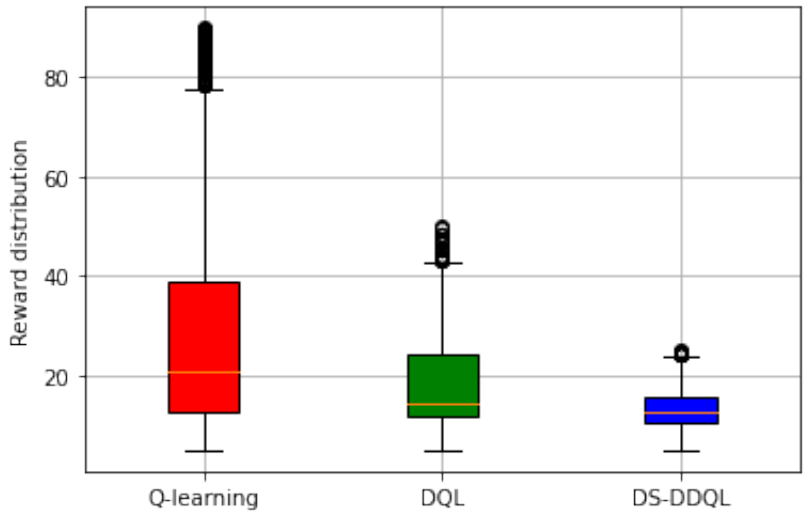

Figure A-3: Performance evaluation of Q-learning, DQL and DSDDQL during learning

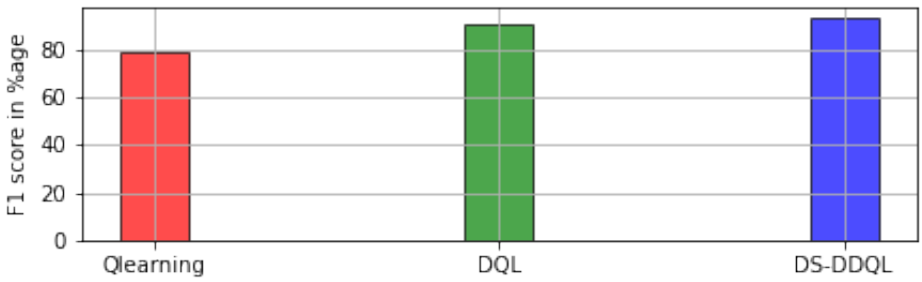

Figure A-4: F1 score with no anomaly

From the figures our model maintained anomaly detection accuracy of $\approx 90 - 95\%$ while DQL and Q-learning presented anomaly detection of $\approx 77 - 89\%$ and $\approx 63 - 78\%$

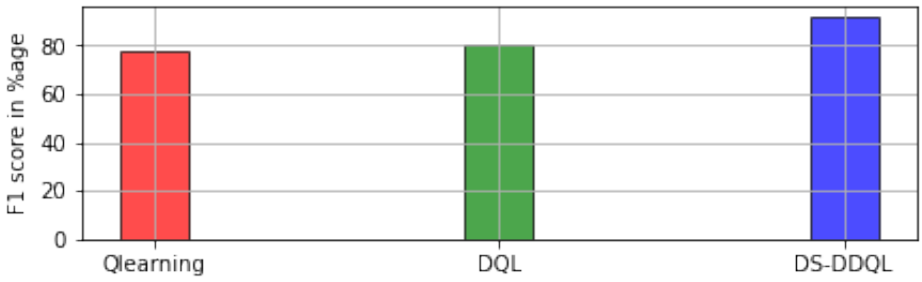

Figure A-5: F1 score with one set of anomaly

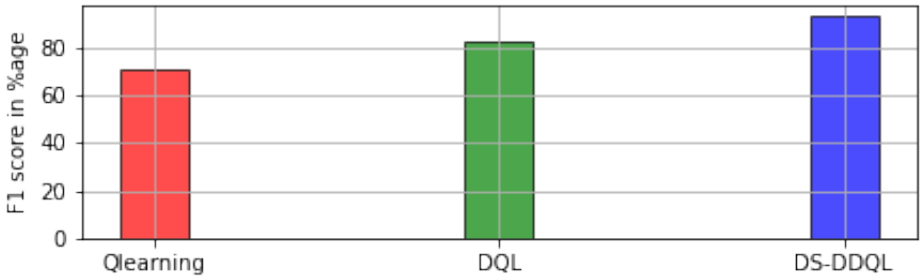

Figure A-6: F1 score with two sets of anomalies

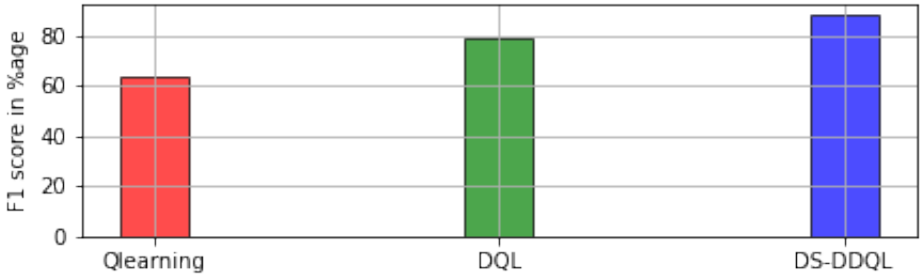

Figure A-7: F1 score with three sets of anomalies

respectively, when there was no anomaly, one set of anomaly, two sets of anomalies and three sets of anomalies.

In general, the DS-DDQL approach provides an improved control in determining a policy for anomaly detection thereby presented an improved rate of anomaly detection in a continuous state space with a multistage action space.

References

[1] M. O. Ojijo and O. E. Falowo, "A Survey on Slice Admission Control Strategies and Optimization Schemes in 5G Network," *IEEE Access*, vol. 8, pp. 14 977–14 990, 2020. 1, 5, 21, 33, 37, 121, 122

[2] G. Wang, G. Feng, W. Tan, S. Qin, R. Wen, and S. Sun, "Optimizing Network Slice Dimensioning via Resource Pricing," *IEEE Access*, vol. XX, no. XX, pp. 1–13, 2019. 1, 31, 54

[3] X. Li, J. Rao, H. Zhang, and A. Callard, "Network slicing with elastic SFC," *IEEE Veh. Technol. Conf.*, vol. 2017-Septe, pp. 1–5, 2018. 1

[4] European Telecommunications Standards Institute (ETSI), "Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action," ETSI, Sophia Antipolis, France, Technical Report 002, Oct. 2012. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf 1, 114

[5] T. Taleb, I. Afolabi, and M. Bagaa, "Orchestrating 5G Network Slices to Support Industrial Internet and to Shape Next-Generation Smart Factories," *IEEE Netw.*, vol. PP, pp. 1–9, 2019. 2

[6] H. Ishii, Y. Kishiyama, and H. Takahashi, "A novel architecture for lte-b: Cplane/u-plane split and phantom cell concept," in *IEEE Globecom Workshops*, Anaheim, 2012, pp. 624–630. 2

[7] B. Han, D. Feng, and H. D. Schotten, "A Markov Model of Slice Admission Control," *IEEE Netw. Lett.*, vol. 1, no. 1, pp. 2–5, 2018. [Online]. Available: http://arxiv.org/abs/1804.01861 3, 34, 38

[8] M. G. Kibria, K. Nguyen, G. P. Villardi, O. Zhao, K. Ishizu, and F. Kojima, "Big Data Analytics, Machine Learning, and Artificial Intelligence in Next-Generation Wireless Networks," *IEEE Access*, vol. 6, pp. 32 328–32 338, 2018. 3

[9] S. O. Oladejo and O. E. Falowo, "Latency-aware dynamic resource allocation scheme for multi-tier 5g network: A network slicing-multitenancy scenario," *IEEE Access*, vol. 8, pp. 74 834–74 852, 2020. 3, 26, 32, 128, 164

[10] M. A. T. Nejad, S. Parsaeefard, M. A. Maddah-Ali, T. Mahmoodi, and B. H. Khalaj, "VSPACE: VNF Simultaneous Placement, Admission Control and Embedding," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 542–557, 2018. 4, 26, 27, 38, 57

[11] D. Bega, X. Costa-Perez, M. Gramaglia, V. Sciancalepore, and A. Banchs, "A Machine Learning approach to 5G Infrastructure Market optimization," *IEEE Trans. Mob. Comput.*, vol. XX, no. X, pp. 1–1, 2019. 4, 16, 35, 57

[12] 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*, vol. 7, pp. 45 758–45 772, 2019. 4, 35, 38, 40

[13] B. Han, A. Dedomenico, G. Dandachi, A. Drosou, D. Tzovaras, R. Querio, F. Moggio, O. Bulakci, and H. D. Schotten, "Admission and Congestion Control for 5G Network Slicing," *2018 IEEE Conf. Stand. Commun. Networking, CSCN 2018*, 2018. 57, 58

[14] B. Han, V. Sciancalepore, D. Feng, X. Costa-Perez, and H. D. Schotten, "A Utility-Driven Multi-Queue Admission Control Solution for Network Slicing," no. May, 2019. [Online]. Available: http://arxiv.org/abs/1901.06399 4, 57

[15] D. Bega, M. Gramaglia, A. Garcia-Saavedra, M. Fiore, A. Banchs, and X. Costa-Perez, "Network Slicing Meets Artificial Intelligence: An AI-Based Framework for Slice Management," *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 32–38, 2020. 5, 42, 58, 82, 121, 122

[16] E. O. Arwa and K. A. Folly, "Reinforcement learning techniques for optimal power control in grid-connected microgrids: A comprehensive review," *IEEE Access*, vol. 8, pp. 208 992–209 007, 2020. 5, 38, 53, 55, 56, 70

[17] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, and X. Costa-Pérez, "A machine learning approach to 5g infrastructure market optimization," *IEEE Transactions on Mobile Computing*, vol. 19, no. 3, pp. 498–512, 2020. 14, 54

[18] N. V. Sahinidis, "Mixed-integer nonlinear programming," *Optimization and Engineering*, vol. 20, no. 2, pp. 301–306, 2019. [Online]. Available: https://doi.org/10.1007/s11081-019-09438-1 17

[19] R. Hemmecke, M. Köppe, J. Lee, and R. Weismantel, *50 Years of Integer Programming 1958-2008*. Springer, 2010. 18

[20] R. Hemmecke, M. Köppe, J. Lee, and R. Weismantel, *Nonlinear integer programming*, 2010. 19

[21] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. New York, NY: W. H. Freeman and Company, 1979. 19

[22] J. A. De Loera, R. Hemmecke, M. König, and R. Weismantel, "Integer polynomial optimization in fixed dimension," *Mathematics of Operations Research*, vol. 31, no. 1, pp. 147–153, 2006. 19

[23] Y. V. Matiyasevich, "Enumerable sets are diophantine," *Soviet Mathematics Doklady*, vol. 11, no. 2, pp. 354–357, 1970. 19

[24] ——, *Hilbert's Tenth Problem*. Cambridge, MA, USA: The MIT Press, 1993. 19

[25] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2179–2192, 2019. 20

[26] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, "Mixed-integer nonlinear optimization," *Acta Numerica*, vol. 22, p. 1–131, 2013. 22

[27] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," *Econometrica*, vol. 28, no. 1, pp. 49–54, 1960. 22

[28] I. Quesada and I. E. Grossmann, "An lp/nlp based branch and bound algorithm for convex minlp optimization problems," *Computers & Chemical Engineering*, vol. 16, no. 10–11, pp. 937–947, 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0098135492800288 23

[29] S. O. Oladejo and O. E. Falowo, "Latency-Aware Dynamic Resource Allocation Scheme for 5G Heterogeneous Network: A Network Slicing-Multitenancy Scenario," *Int. Conf. Wirel. Mob. Comput. Netw. Commun.*, vol. 2019-October, pp. 1–7, 2019. 26, 35

[30] J.-J. Kuo, S.-H. Shen, H.-Y. Kang, D.-N. Yang, M.-J. Tsai, and W.-T. Chen, "Service chain embedding with maximum flow in software defined network and application to the next-generation cellular network architecture," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9. 26

[31] X. Li, J. Rao, H. Zhang, and A. Callard, "Network slicing with elastic sfc," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, 2017, pp. 1–5. 26

[32] A. Dissertation, "Successive Convex Approximation : Analysis and SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL. pp8-48," 2014. 26

[33] Z. Allybokus, K. Avrachenkov, J. Leguay, and L. Maggi, "Lower Bounds for the Fair Resource Allocation Problem," no. February, 2018. [Online]. Available: http://arxiv.org/abs/1802.02932 29, 31

[34] P. T. A. Quang, A. Bradai, K. D. Singh, G. Picard, and R. Riggio, "Single and Multi-Domain Adaptive Allocation Algorithms for VNF Forwarding Graph Embedding," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 1, pp. 98–112, 2019. 29

[35] W. Tian and X. Yuan, "An alternating direction method of multipliers with a worst-case convergence rate," vol. 88, no. 318, pp. 1685–1713, 2019. 30

[36] J. Tang, B. Shim, and T. Q. S. Quek, "Service Multiplexing and Revenue Maximization in Sliced C-RAN Incorporated with URLLC and Multicast eMBB," *IEEE J. Sel. Areas Commun.*, vol. 8716, no. 2016, pp. 1–1, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8638932/ 31, 38, 95

[37] C. Song, M. Zhang, X. Huang, Y. Zhan, D. Wang, M. Liu, and Y. Rong, "Machine learning enabling traffic-aware dynamic slicing for 5g optical transport networks," in *2018 Conference on Lasers and Electro-Optics (CLEO)*, 2018, pp. 1–2. 31

[38] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing management & prioritization in 5G mobile systems," *Eur. Wirel. 2016; 22th Eur. Wirel. Conf.*, pp. 197–202, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7499297/ 31, 38, 57

[39] T. Guo and A. Suarez, "Enabling 5G RAN Slicing with EDF Slice Scheduling," *IEEE Trans. Veh. Technol.*, vol. PP, no. c, pp. 1–1, 2019. 31, 58

[40] D. J. Birabwa, D. Ramotsoela, and N. Ventura, "Multi-agent deep reinforcement learning for user association and resource allocation in integrated terrestrial and

non-terrestrial networks," *Computer Networks*, vol. 231, p. 109827, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128623002724 32, 70, 76, 82

[41] S. Zoppi, J. P. Champatiy, J. Grossz, and W. Kellerer, "Scheduling of wireless edge networks for feedback-based interactive applications," *IEEE Transactions on Communications*, pp. 1–1, 2022. 32, 59

[42] Y. Li, E. Pateromichelakis, N. Vucic, J. Luo, W. Xu, and G. Caire, "Radio resource management considerations for 5g millimeter wave backhaul and access networks," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 86–92, 2017. 32

[43] G. Rudolph, "Convergence Analysis of Canonical Genetic Algorithms," *IEEE Trans. Neural Networks*, vol. 5, no. 1, pp. 96–101, 1994. 33, 34

[44] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. 34

[45] B. Han, J. Lianghai, and H. D. Schotten, "Slice as an evolutionary service: Genetic optimization for inter-slice resource management in 5g networks," *IEEE Access*, vol. 6, pp. 33 137–33 147, 2018. 35

[46] Q. Sun, C.-L. I, Z. Zhao, H. Zhang, X. Chen, C. Yang, R. Li, and M. Zhao, "Deep Reinforcement Learning for Resource Management in Network Slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018. 35, 46, 109, 110

[47] P. Monti, C. Natalino, M. R. Raza, P. Ohlen, and L. Wosinska, "A Slice Admission Policy Based on Reinforcement Learning for a 5G Flexible RAN," no. 1, pp. 1–3, 2018. 35

[48] S. E. Elayoubi, S. B. Jemaa, Z. Altman, and A. Galindo-Serrano, "5G RAN Slicing for verticals: Enablers and challenges," *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 28–34, 2019. 35

[49] H. Tong and T. X. Brown, "Adaptive Call Admission Control Under Quality of Service Constraints : A Reinforcement Learning Solution," vol. 18, no. 2, pp. 209–221, 2000. 35, 38

[50] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource Management with Deep Reinforcement Learning," 2016, pp. 50–56. 35, 38

[51] T. Maksymyuk, J. Gazda, O. Yaremko, and D. Nevinskiy, "Deep learning based massive MIMO beamforming for 5G mobile network," *Proc. 2018 IEEE 4th Int. Symp. Wirel. Syst. within Int. Conf. Intell. Data Acquis. Adv. Comput. Syst. IDAACS-SWS 2018*, pp. 241–244, 2018. 35

[52] L. V. Le, B. S. P. Lin, L. P. Tung, and D. Sinh, "SDN/NFV, Machine Learning, and Big Data Driven Network Slicing for 5G," *IEEE 5G World Forum, 5GWF 2018 - Conf. Proc.*, pp. 20–25, 2018. 35

[53] "Neural network-based autonomous allocation of resources in virtual networks," *EuCNC 2014 - Eur. Conf. Networks Commun.*, no. August, 2014. 35

[54] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, 2017. 35, 43, 50

[55] A. Yadav, O. A. Dobre, and N. Ansari, "Distributed energy and resource management for full-duplex dense small cells for 5G," *2017 13th Int. Wirel. Commun. Mob. Comput. Conf. IWCMC 2017*, pp. 133–139, 2017. 38

[56] R. S. Sutton and A. G. Barto, "Reinforcement learning Complete Draft," 2017. 38, 40, 42, 45, 48, 50

[57] R. Sawant and S. Nema, "Outage probability analysis in hybrid cooperative cognitive radio networks," *2017 Int. Conf. Big Data, IoT Data Sci. BID 2017*, vol. 2018-Janua, pp. 77–81, 2018. 38

[58] V. S. Reddy, A. Baumgartner, and T. Bauschert, "Robust embedding of VNF/service chains with delay bounds," in *2016 IEEE Conf. Netw. Funct. Virtualization Softw. Defin. Networks, NFV-SDN 2016*. IEEE, 2017, pp. 93–99. 38

[59] "Machine Learning Aided Orchestration in Multi-Tenant Networks," *IEEE Photonics Soc. Summer Top. Meet. Ser. SUM 2018*, pp. 125–126, 2018. 38

[60] T. Maksymyuk, J. Gazda, O. Yaremko, and D. Nevinskiy, "Deep Learning Based Massive MIMO Beamforming for 5G Mobile Network," in *2018 IEEE 4th Int. Symp. Wirel. Syst. within Int. Conf. Intell. Data Acquis. Adv. Comput. Syst.* IEEE, sep 2018, pp. 241–244. [Online]. Available: https://ieeexplore.ieee.org/document/8525802/

[61] L. V. Le and B. S. Lin, "Machine Learning Enabling Traffic-Aware Dynamic Slicing for 5G Optical Transport Networks," *IEEE Access*, vol. 6, no. c, p. JTu2A.44, 2018.

[62] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuška, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 6, pp. 1291–1307, 2012. 42, 55

[63] Y. Li, K. Li, S. Ci, R. Chen, Y. Yang, S. Zhao, and J. Wang, "DECCO: Deep-Learning Enabled Coverage and Capacity Optimization for Massive MIMO Systems," *IEEE Access*, vol. 6, pp. 23 361–23 371, 2018.

[64] C. Zhang, P. Patras, and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," pp. 1–67, 2018. [Online]. Available: http://arxiv.org/abs/1803.04311 122, 123

[65] L. L. Vy, L. P. Tung, and B. S. P. Lin, "Big data and machine learning driven handover management and forecasting," *2017 IEEE Conf. Stand. Commun. Networking, CSCN 2017*, pp. 214–219, 2017. 38

[66] S. Koenig and R. G. Simmons, "Complexity analysis of real-time reinforcement learning," in *Proceedings of the Eleventh National Conference on Artificial Intelligence*, ser. AAAI'93. AAAI Press, 1993, p. 99–105. 40

[67] M. Al-Naday, M. Reed, V. Dobre, S. Toor, B. Volckaert, and F. De Turck, "Service-based federated deep reinforcement learning for anomaly detection in fog ecosystems,"

in *2023 26th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2023, pp. 121–128. 40

[68] M. M. Kokar and S. A. Reveliotis, "Reinforcement learning: Architectures and algorithms," *International Journal of Intelligent Systems*, vol. 8, no. 8, pp. 875–894, 1993. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/int.4550080805 41

[69] N. Van Huynh, D. Thai Hoang, D. N. Nguyen, and E. Dutkiewicz, "Optimal and Fast Real-Time Resource Slicing with Deep Dueling Neural Networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1455–1470, 2019. 41, 46, 55

[70] A. Khodadadi, P. Fakhari, and J. R. Busemeyer, "Learning to maximize reward rate: a model based on semi-markov decision processes," *Frontiers in Neuroscience*, vol. 8, p. 101, 2014. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2014.00101 42, 46

[71] M. L. Puterman, "Chapter 8 markov decision processes," in *Stochastic Models*, ser. Handbooks in Operations Research and Management Science. Elsevier, 1990, vol. 2, pp. 331–434. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927050705801720 45

[72] R. Srinivasan and A. K. Parlikad, "Semi-Markov Decision Process With Partial Information for Maintenance Decisions," vol. 63, no. 4, pp. 891–898, 2014. 46

[73] Y. Liu, J. Ding, and X. Liu, "Resource allocation method for network slicing using constrained reinforcement learning," in *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1–3. 48, 51

[74] C. Banerjee, Z. Chen, N. Noman, and M. Zamani, "Optimal actor-critic policy with optimized training datasets," *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–11, 2022. 49, 50

[75] D. Urgun and C. Singh, "Composite power system reliability evaluation using importance sampling and convolutional neural networks," in *2019 20th International Conference on Intelligent System Application to Power Systems (ISAP)*, 2019, pp. 1–6. 50

[76] X. Tang, B. Huang, T. Liu, and X. Lin, "Highway decision-making and motion planning for autonomous driving via soft actor-critic," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4706–4717, 2022. 50

[77] M. Ojijo and D. Ombuya, "Slice admission control by resource auction game in 5g network with reinforcement learning," *Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2021 -https://www.satnac.org.za/proceedings*, vol. 2021, no. ISBN: 978-0-6397-2773-8, pp. 8–13, 2021. 51, 98, 114

[78] G. Li, Y. Wei, Y. Chi, Y. Gu, and Y. Chen, "Sample Complexity of Asynchronous Q-Learning: Sharper Analysis and Variance Reduction," *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 448–473, 2022. 53, 60

[79] S. Khodadadian, Z. Chen, and S. T. Maguluri, "Finite-sample analysis of off-policy natural actor-critic algorithm," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 5420–5431. [Online]. Available: https://proceedings.mlr.press/v139/khodadadian21a.html 54

[80] G. Sun, K. Xiong, G. O. Boateng, D. Ayepah-Mensah, G. Liu, and W. Jiang, "Autonomous resource provisioning and resource customization for mixed traffics in virtualized radio access network," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2454–2465, Sep. 2019. 57

[81] M. R. Raza, A. Rostami, L. Wosinska, and P. Monti, "A Slice Admission Policy Based on Big Data Analytics for Multi-Tenant 5G Networks," *J. Light. Technol.*, vol. PP, no. c, pp. 1–1, 2019. 59

[82] Y. Lu, X. Chen, Y. Zhang, and Y. Chen, "Cost-efficient resources scheduling for mobile edge computing in ultra-dense networks," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022. 59

[83] R. Chen and X. Wang, "Situation-aware orchestration of resource allocation and task scheduling for collaborative rendering in iot visualization," *IEEE Transactions on Sustainable Computing*, pp. 1–1, 2022. 59

[84] X. Liu and X. Guo, "Ship image compression method based on hsv color space and kernel principal component analysis," in *2022 International Symposium on Sensing and Instrumentation in 5G and IoT Era (ISSI)*, 2022, pp. 175–180. 60, 140

[85] K. Kim, D. Kim, S. Choi, D. Kwon, and J.-W. Choi, "Location-based maximum reuse distance resource scheduling for lte cellular-v2x sidelink mode 3," in *2022 International Conference on Electronics, Information, and Communication (ICEIC)*, 2022, pp. 1–4. 60

[86] D. Mukherjee, S. Ghosh, S. Pal, A. A. Aly, and D.-N. Le, "Adaptive scheduling algorithm based task loading in cloud data centers," *IEEE Access*, pp. 1–1, 2022. 60

[87] S. Skaperas, N. Ferdosian, A. Chorti, and L. Mamatas, "Scheduling of heterogeneous services by resolving conflicts," *IEEE Access*, vol. 10, pp. 36 576–36 591, 2022. 60

[88] S. A. AlQahtani and W. A. Alhomiqani, "A multi-stage analysis of network slicing architecture for 5g mobile networks," *Telecommunication Systems*, vol. 73, pp. 205–221, 2020. 62, 63

[89] 5G Americas, "5g americas urllc white paper," Tech. Rep., 2019, accessed: 2025-02-06. [Online]. Available: https://www.5gamericas.org/wp-content/uploads/2019/07/5G_Americas_URLLLC_White_Paper_Final__updateJW.pdf 62

[90] L. Chinchilla-Romero, J. Prados-Garzon, P. Ameigeiras, P. Muñoz, and J. M. Lopez-Soler, "5g infrastructure network slicing: E2e mean delay model and effectiveness assessment to reduce downtimes in industry 4.0," *Sensors (Basel, Switzerland)*, vol. 22, no. 1, p. 229, 2021. 62

[91] B. Han, V. Sciancalepore, D. Feng, X. Costa-Perez, and H. D. Schotten, "A utility-driven multi-queue admission control solution for network slicing," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 55–63. 62

[92] J.-Y. Lin, P.-H. Chou, and R.-H. Hwang, "Dynamic resource allocation for network slicing with multi-tenants in 5g two-tier networks," *Sensors*, vol. 23, no. 10, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/10/4698 63

[93] "Transient solution of a Markovian queuing model with heterogeneous servers and catastrophes," *Opsearch*, vol. 52, no. 4, pp. 810–826, 2015. 64, 65

[94] N. Sandhiya and R. Varadharajan, "A study on single and multi server queuing models using interval number," *Journal of Physics: Conference Series*, vol. 1000, no. 1, p. 012133, apr 2018. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1000/1/012133 67

[95] J. D. Little and S. C. Graves, "Little's law," in *Building Intuition*, D. Chhajed and T. J. Lowe, Eds. Boston, MA: Springer, 2008, pp. 81–100. 69

[96] S. Li, Q. Tian, F. Wan, X. Xin, Q. Zhang, Y. Wang, F. Tian, and L. Yang, "A dynamic optical network units slicing algorithm for centralized flexible time- and wavelength-division multiplexing passive optical network," in *2021 19th International Conference on Optical Communications and Networks (ICOCN)*, 2021, pp. 1–3. 71, 82

[97] B. Tian, Q. Zhang, X. Xin, Q. Tian, X. Wu, Y. Tao, Y. Shen, G. Cao, and N. Liu, "Recursive neural network based rrh to bbu resource allocation in 5g fronthaul

network," in *2018 Asia Communications and Photonics Conference (ACP)*, 2018, pp. 1–3. 72

[98] G. Sun, K. Xiong, G. O. Boateng, D. Ayepah-Mensah, G. Liu, and W. Jiang, "Autonomous resource provisioning and resource customization for mixed traffics in virtualized radio access network," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2454–2465, 2019. 82

[99] V. P. Kafle, Y. Fukushima, P. Martinez-Julia, and T. Miyazawa, "Consideration On Automation of 5G Network Slicing with Machine Learning," *2018 ITU Kaleidosc. Mach. Learn. a 5G Futur. (ITU K)*, pp. 1–8, 2019. 93, 95

[100] M. Morcos, T. Chahed, L. Chen, J. Elias, and F. Martignon, "A Two-level Auction for C-RAN Resource Allocation," *2017 IEEE Int. Conf. Commun. Work. (ICC Work.*, pp. 516–521, 2017. 93, 96

[101] H. Wang, E. Dutkiewicz, G. Fang, and M. D. Mueck, "An Auction Framework Based on Flexible Transmit Powers in the Licensed Shared Access Systems," *2015 15th Int. Symp. Commun. Inf. Technol.*, pp. 269–272, 2015. 94

[102] B. Kasberger, "Can Auctions Maximize Welfare in Markets After the Auction?" *Mimeo*, no. NOeG, pp. 1–44, 2017. 94

[103] N. H. Almofari, S. Kishk, and F. W. Zaki, "Auction Based Algorithm for Distributed Resource Allocation in Multitier-Heterogeneous Cellular Networks," *2016 11th Int. Conf. Comput. Eng. Syst.*, pp. 426–433, 2016. 94

[104] L. Liang, Y. Wu, G. Feng, S. Member, X. Jian, and Y. Jia, "Online Auction-Based Resource Allocation for Service-Oriented Network Slicing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8063–8074, 2019. 95, 98

[105] X. Liu, Q. Qiu, and L. Linyun, "An Online Combinatorial Auction Based Resource Allocation and Pricing Mechanism for Network Slicing in 5G," *2019 IEEE 19th Int. Conf. Commun. Technol.*, pp. 908–913, 2019. 96

[106] K. Zhu and E. Hossain, "Virtualization of 5G Cellular Networks as a Hierarchical Combinatorial Auction," *IEEE Trans. Mob. Comput.*, vol. 15, no. 10, pp. 2640–2654, 2016. 96, 100

[107] X. Zhang, H. Qian, K. Zhu, and R. Wang, "Virtualization of 5G Cellular Networks : A Combinatorial Double Auction Approach," 2017. 96

[108] N. Tadayon and A. Sonia, "Radio Resource Allocation and Pricing : Auction-Based Design and Applications," *IEEE Trans. Signal Process.*, vol. 66, no. 20, pp. 5240–5254, 2018. 97

[109] F. Zhao, Y. Zhang, and Q. Wang, "Multi-Slot Spectrum Auction in Heterogeneous Networks Based on Deep Feedforward Network," *IEEE Access*, vol. 6, pp. 45 113–45 119, 2018. 97

[110] M. Harishankar, S. Pilaka, P. Sharma, N. Srinivasan, C. Joe-wong, and P. Tague, "Procuring Spontaneous Session-Level Resource Guarantees for Real-Time

Applications : An Auction Approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 7, pp. 1534–1548, 2019. 97

[111] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing in 5G : an Auction-Based Model," *2017 IEEE Int. Conf. Commun.*, pp. 1–6, 2017. 97

[112] U. Habiba and E. Hossain, "Auction Mechanisms for Virtualization in 5G Cellular Networks : Basics , Trends , and Open Challenges," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 3, pp. 2264–2293, 2018. 97

[113] S. Swain, N. Mishra, S. Rath, and B. P. S. Sahoo, "Spectrum Sharing for D2D Communication in 5G Cellular Networks : An Auction-Based Model." 97

[114] F. Zhao and Q. Tang, "A KNN Learning Algorithm for Collusion- Resistant Spectrum Auction in Small Cell Networks," *IEEE Access*, vol. PP, no. c, p. 1, 2018. 100

[115] Z. Wang, Y. Wei, F. R. Yu, and Z. Han, "Utility optimization for resource allocation in multi-access edge network slicing: A twin-actor deep deterministic policy gradient approach," *IEEE Transactions on Wireless Communications*, vol. 21, no. 8, pp. 5842–5856, 2022. 102, 140, 147

[116] F. R. Yu and Y. He, *Deep Reinforcement Learning for Interference Alignment Wireless Networks*, 2019. 102, 103, 130

[117] Y. Zhang, C. Lee, D. Niyato, and P. Wang, "Auction approaches for resource allocation in wireless systems: A survey," *IEEE Commun. Surv. Tutorials*, vol. 15, no. 3, pp. 1020–1041, 2013. 105, 108

[118] M. A. Rostom, A. H. A. El-malek, M. Elsabrouty, and M. Abo-zahhad, "Cognitive Radio Users Admission and Channels Allocation in 5G HetNets : A College-based Matching and Auction Game Approach," *2019 Int. Conf. Wirel. Mob. Comput. Netw. Commun.*, pp. 1–6, 2019. 108

[119] J. Deng, O. Tirkkonen, R. Freij-Hollanti, T. Chen, and N. Nikaein, "Resource Allocation and Interference Management for Opportunistic Relaying in Integrated mmWave/sub-6 GHz 5G Networks," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 94–101, 2017. 113

[120] 3rd Generation Partnership Project, "Massive machine type communication (mmtc) reference architecture," https://www.3gpp.org/ftp/Specs/archive/23_series/23.501/23501-e40.zip, 2022, accessed: April 19, 2023. 114, 165

[121] O. Consortium, "Openfog reference architecture for fog computing," https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf, 2017, accessed: April 19, 2023. 114, 165

[122] P. Ge and T. Lv, "Data enabled Self-Organizing Network with Adaptive Antennas based on Proactive Prediction for Enabling 5G," *2018 IEEE/CIC Int. Conf. Commun. China*, no. Iccc, pp. 52–56, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8641135/ 122

[123] M. R. Raza, C. Natalino, P. Öhlen, L. Wosinska, and P. Monti, "A slice admission policy based on reinforcement learning for a 5g flexible ran," in *2018 European Conference on Optical Communication (ECOC)*, 2018, pp. 1–3. 123

[124] Y. Ai, G. Qiu, C. Liu, and Y. Sun, "Joint resource allocation and admission control in sliced fog radio access networks," *China Communications*, vol. 17, no. 8, pp. 14–30, 2020. 124

[125] S. Bakri, B. Brik, and A. Ksentini, "On using reinforcement learning for network slice admission control in 5G: Offline vs. online," *International Journal of Communication Systems*, vol. 34, no. 7, pp. 1–12, 2021. 124

[126] S. II Moon, H. Hirayama, Y. Tsukamoto, S. Nanba, and H. Shinbo, "Ensemble learning method-based slice admission control for adaptive ran," in *2020 IEEE Globecom Workshops (GC Wkshps*, 2020, pp. 1–6. 124

[127] A. Perveen, M. Patwary, and A. Aneiba, "Dynamically reconfigurable slice allocation and admission control within 5g wireless networks," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019, pp. 1–7. 125

[128] G. Tseliou, F. Adelantado, and C. Verikoukis, "Netslic: Base station agnostic framework for network slicing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3820–3832, 2019. 127

[129] Q. Qi, A. Minturn, and Y. Yang, "An efficient water-filling algorithm for power allocation in ofdm-based cognitive radio systems," in *2012 International Conference on Systems and Informatics (ICSAI2012)*, 2012, pp. 2069–2073. 127

[130] M. O. Ojijo, D. Ramotsoela, and R. A. Oginga, "Slice admission control in 5g wireless communication with multi-dimensional state space and distributed action space: A

sequential twin actor-critic approach," 2024, available at SSRN: https://ssrn.com/abstract=4696985 or http://dx.doi.org/10.2139/ssrn.4696985. 128

[131] H. Gomaa, G. G. Messier, C. Williamson, and R. Davies, "Estimating instantaneous cache hit ratio using markov chain analysis," *IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1472–1483, 2013. 130

[132] R. Schmidt, C.-Y. Chang, and N. Nikaein, "Slice scheduling with qos-guarantee towards 5g," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–7. 133

[133] K. Simonov, F. Fomin, P. Golovach, and F. Panolan, "Refined complexity of PCA with outliers," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 5818–5826. [Online]. Available: https://proceedings.mlr.press/v97/simonov19a.html 152

[134] H. Kumar, A. Koppel, and A. Ribeiro, "On the sample complexity of actor-critic method for reinforcement learning with function approximation," 2023. 152

[135] H. Wu, J. He, M. Tömösközi, Z. Xiang, and F. H. Fitzek, "In-network processing for low-latency industrial anomaly detection in softwarized networks," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 01–07. 164

[136] A. Perveen, M. Patwary, and A. Aneiba, "Dynamically Reconfigurable Slice Allocation and Admission Control within 5G Wireless Networks," *2019 IEEE 89th Veh. Technol. Conf.*, pp. 1–7, 2019. 164

[137] M. Mohammadkarimi and M. Ardakani, "Efficient massive machine type communication (mmtc) via amp," *IEEE Wireless Communications Letters*, vol. 12, no. 6, pp. 1002–1006, 2023. 164

[138] M. Al-Naday, M. Reed, V. Dobre, S. Toor, B. Volckaert, and F. De Turck, "Service-based federated deep reinforcement learning for anomaly detection in fog ecosystems," in *2023 26th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2023, pp. 121–128. 164, 165

[139] G. Raja, M. Begum, S. Gurumoorthy, D. S. Rajendran, P. Srividya, K. Dev, and N. M. F. Qureshi, "Ai-empowered trajectory anomaly detection and classification in 6g-v2x," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 4599–4607, 2023. 164

[140] S. B. Prathiba, G. Raja, S. Anbalagan, A. K. S, S. Gurumoorthy, and K. Dev, "A hybrid deep sensor anomaly detection for autonomous vehicles in 6g-v2x environment," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 3, pp. 1246–1255, 2023. 164

[141] A. Yahja, S. Kaviani, B. Ryu, J. Kim, and K. Larson, "Deepadmr: A deep learning based anomaly detection for manet routing," in *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*, 2022, pp. 412–417. 164

[142] H. Szostak and K. Cohen, "Decentralized anomaly detection via deep multi-agent reinforcement learning," in *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2022, pp. 1–4. 164

[143] L. M. Margot Deruyck, Wout Joseph, "Power consumption model for macrocell and microcell base stations," *Transactions on Emerging Telecommunications Technologies*, 2011. [Online]. Available: https://doi.org/10.1002/ett 167

[144] W. Hou, S. Li, Y. Sun, J. Zhou, X. Yun, and N. Lu, "Interference-Aware Subcarrier Allocation for Massive Machine-Type Communication in 5G-Enabled Internet of Things," 2019. 168

[145] M. I. Nashiruddin and A. A. F. Purnama, "Nb-iot network planning for advanced metering infrastructure in surabaya, sidoarjo, and gresik," in *2020 8th International Conference on Information and Communication Technology (ICoICT)*, 2020, pp. 1–6. 168

[146] S. Zhang, L. Liu, Y. Cheng, X. Cao, and L. Cai, "Energy-efficient beamforming for massive mimo with inter-cell interference and inaccurate csi," in *2018 International Conference on Computing, Networking and Communications (ICNC)*, 2018, pp. 518–523. 168

[147] J. Gorodkin, "Comparing two k-classifiers: The wilcoxon-mann-whitney approach," *Journal of Machine Learning Research*, vol. 5, pp. 203–228, 2004. 178