PRISM: Multi-Agent Reinforcement Learning for Automated Service Assurance in B5G Networks

Mukesh Kumar Angrish

A Thesis
in
Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master of Computer Science at
Concordia University
Montréal, Québec, Canada

October 2025

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Mukesh Kumar Angrish**

Entitled: **PRISM: Multi-Agent Reinforcement Learning for Automated Service Assurance in B5G Networks**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Sandra Cespedes

_____ Examiner
Dr. Sandra Cespedes

_____ Examiner
Dr. Lata Narayanan

_____ Thesis Supervisor
Dr. Brigitte Jaumard

_____ Thesis Supervisor

Approved by _____
Dr. Denis Pankratov, Graduate Program Director

October 23, 2025 _____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

PRISM: Multi-Agent Reinforcement Learning for Automated Service Assurance in B5G Networks

Mukesh Kumar Angrish

The exponential growth in mobile data traffic and connected devices necessitates intelligent management solutions for 5G and B5G networks. Modern telecommunications networks must simultaneously support diverse services ranging from video streaming to autonomous vehicles, each requiring different performance guarantees. This thesis presents a comprehensive framework that uses artificial intelligence to automatically manage network resources and maintain service quality. We developed a closed-loop control system where reinforcement learning agents continuously monitor network performance, make resource allocation decisions, and learn from the consequences of those decisions as they ripple through the network. Each action taken by the system influences future network conditions, creating a complex feedback loop that traditional management approaches struggle to handle. Our framework learns to dynamically adjust computational resources (CPU, memory, storage) and network bandwidth in response to changing traffic patterns, preventing service degradation before it occurs. The system eliminates the need for manual tuning by automatically learning the delicate balance between maintaining service quality and minimizing resource consumption. We implement both single-agent and multi-agent architectures, where multiple independent agents can manage different network segments simultaneously. Validation using packet-level network simulator with traffic patterns derived

from urban mobility data demonstrates that our approach successfully maintains performance metrics such as delay, packet-loss, and jitter within acceptable bounds while adapting to dynamic conditions. The multi-agent configuration achieves efficient resource management with reduced computational overhead. This work establishes foundational principles for autonomous network management, providing practical frameworks for deploying intelligent, adaptive, and energy-efficient control mechanisms in next-generation telecommunications infrastructure.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Brigitte Jaumard, for her invaluable guidance and support throughout this research. Her insightful feedback and encouragement to articulate complex technical concepts in clear and accessible language have been instrumental in shaping this thesis. I am deeply grateful to Dr. Oscar Delgado for his continuous guidance and technical expertise; his willingness to engage in detailed discussions was crucial in bringing this project to fruition. This research was supported by a MITACS Internship (Grant Number IT 16296) in partnership with Ciena and Concordia University, and I acknowledge their financial support with appreciation. I extend my heartfelt thanks to my parents, whose teachings and unwavering support have enabled me to reach this milestone in my academic journey. Finally, I am grateful to my friends who stood by me during the most challenging moments of this endeavor, providing encouragement and perspective when it was needed most.

Contents

List of Figures

List of Tables

List of Abbreviations

5G Fifth Generation.

6G Sixth Generation.

AI Artificial Intelligence.

B5G Beyond 5G.

C-RAN Cloud Radio Access Network.

CLA closed-loop assurance.

CPU Central Processing Unit.

DDPG Deep Deterministic Policy Gradient.

DQN Deep Q-Network.

DRL Deep Reinforcement Learning.

eMBB enhanced Mobile Broadband.

ICT Information and Communication Technology.

IoT Internet of Things.

KPI Key Performance Indicator.

MARL Multi-Agent Reinforcement Learning.

MEC Multi-Access Edge Computing.

ML Machine Learning.

NFV Network Function Virtualization.

NS Network Slicing.

NWDAF Network Data Analytics Function.

O-RAN Open Radio Access Network.

OpEx operating expenses.

POMDP Partially Observable Markov Decision Process.

PPO Proximal Policy Optimization.

PPO proximal policy optimization.

QoS Quality of Service.

RAM Memory.

RAN Radio Access Network.

RL Reinforcement Learning.

SA Service Assurance.

SA service assurance.

SAC Soft Actor-Critic.

SDN Software-Defined Networking.

SLA Service Level Agreement.

SLA service level agreement.

URLLC Ultra-Reliable Low-Latency Communications.

VNF Virtual Network Function.

Chapter 1

Introduction

The telecommunications industry is experiencing a profound shift as connected devices and data traffic surge with the advent of Fifth Generation (5G) and the emergence of Sixth Generation (6G) on the horizon, expanding far beyond human-centric communication to sensors, industrial systems, autonomous vehicles, and drones [20]. Mobile data traffic is projected to triple between 2024 and 2030 to over 473 exabytes per month, while Internet of Things (IoT) connections may exceed 38 billion by 2030 [22]. This expanding ecosystem underpins services such as cloud gaming, augmented/virtual reality, high-definition streaming, and autonomous control, which require ultra-reliable, high-throughput, low-latency connectivity, intensifying computational and energy pressures on both radio access and core networks [27].

These gains arrive with sustainability risks. The Information and Communication Technology (ICT) sector already contributes an estimated 2-4% of global greenhouse gas emissions, with communication networks responsible for roughly 20–30% of that share [26]. Without targeted efficiency measures, densified 5G deployments and expanded edge computing could drive a 160% rise in energy consumption by 2030 relative to 2020. Integrating Artificial Intelligence (AI) and Machine Learning (ML) workloads at the edge may further aggravate demand if not paired with efficient

orchestration and adaptive resource management [25]. To meet these challenges, network architectures are being reworked around Software-Defined Networking (SDN), Network Function Virtualization (NFV), Network Slicing (NS), Multi-Access Edge Computing (MEC), and advanced radio access technologies, introducing programmability, automation and agility. In parallel, AI and ML are emerging as pivotal enablers of self-optimizing and self-organizing networks [19]. Looking ahead, forecasts indicate 4.4 billion 5G subscriptions by 2027 (48% of mobile), while early 6G visions target up to 100x higher data rates and sub-millisecond latencies by 2030[20, 32]. In this context, intelligent, adaptive, and energy efficient AI-driven solutions are essential to manage heterogeneity, dynamism, and scale, ensuring seamless interoperability among service assurance processes while optimizing resource utilization in next-generation networks.

1.1 Motivation

The transition to intelligent, energy-efficient, and adaptive networks rests on foundational technologies that form the backbone of 5G and Beyond 5G: SDN, NFV, MEC, and End-to-End Network Slicing, each detailed in Section 2.1. Together, they introduce programmability, elasticity, and locality. Building on these enablers, 5G supports heterogeneous use cases from high-throughput broadband access to ultra reliable low-latency control systems and large-scale IoT deployments [2]. Departing from the traditional model of "one-size-fits-all", 5G enables multiple logical networks to coexist and operate over shared physical infrastructure.

The 3GPP and ITU-T standards define three primary service categories: enhanced Mobile Broadband (eMBB) for applications such as 4K/8K streaming and virtual reality, Ultra-Reliable Low-Latency Communications (URLLC) for autonomous vehicles and industrial automation, and massive Machine-Type Communications (mMTC) for

pervasive, low-power IoT deployments [1]. The unprecedented scale and complexity of 5G and Beyond 5G (B5G) networks, with their diverse service requirements and massive data volumes, makes the integration of AI and ML essential for network operations. In essence, AI/ML provides the capability to optimize network and service performance with this increased operational complexity while reducing costs through enabling automation [37]. This autonomous operation is enabled by learning from real-time and historical data to predict traffic demand, detecting anomalies, optimizing resource allocation, and adapting policies dynamically without any manual intervention.

Reflecting this shift, 3GPP introduced the Network Data Analytics Function (NWDAF) as a standard Control Plane function within the core network system [4]. NWDAF collects diverse data from distributed network functions and employs AI algorithms for analytics tasks supporting load prediction, quality-of-service estimation, user mobility analytics, and slice performance forecasting.

1.2 Problem Statement

The convergence of diverse service requirements, dynamic traffic patterns, and shared infrastructure in 5G networks creates unprecedented operational challenges that traditional management approaches cannot address. Network operators must simultaneously orchestrate resources across multiple network slices, each serving applications with fundamentally different performance requirements. An enhanced Mobile Broadband (eMBB) slice supporting video streaming may tolerate moderate latency but requires sustained high throughput, while a Ultra-Reliable Low-Latency Communications (URLLC) slice controlling autonomous vehicles demands microsecond precision with near perfect reliability. These slices compete for finite computational and network resources that must be allocated dynamically as traffic patterns shift throughout

the day and across geographic regions.

While initial AI/ML deployments in 5G networks have shown promise for specific tasks like traffic prediction and anomaly detection, existing solutions typically address isolated aspects of network management rather than providing comprehensive end-to-end service assurance. Many implementations rely on offline training with simplified models that fail to capture the full complexity of production networks, or they focus on single-slice optimization without considering the intricate resource sharing dynamics across multiple concurrent slices. The complexity stems from multiple interacting factors: network state information arrives with inherent delays and measurement noise, resource allocation decisions at one layer impact performance at others, and the time available for decision making is constrained by strict control loop requirements. Furthermore, the relationship between resource allocation and resulting Key Performance Indicators (KPIs) exhibits nonlinear behavior that varies with network load, making static provisioning strategies either wasteful through overprovisioning or inadequate during peak demand.

The fundamental challenge we address is developing an intelligent control mechanism capable of proactive resource management in this complex environment. This mechanism must continuously monitor network conditions across all active slices, predict potential service degradations before they impact users, and autonomously adjust both computational resources (Central Processing Unit (CPU), Memory (RAM), storage allocations for Virtual Network Functions (VNFs)) and network resources (per-slice bandwidth allocations) to maintain Quality of Service (QoS) within specified bounds. The solution must operate under partial observability, where complete network state is never fully known, process noisy telemetry data that may be delayed or incomplete, and make decisions within millisecond timescales to prevent service disruption.

Beyond maintaining service quality, the control system must minimize resource consumption to reduce operational costs and environmental impact. This creates competing objectives: ensuring robust service assurance while avoiding wasteful over-provisioning. The algorithm must learn to balance these goals without manual tuning of trade-off parameters, adapting its strategy as network conditions evolve. Additionally, the solution must scale from managing individual slices to orchestrating multiple concurrent slices without requiring architectural changes or extensive reconfiguration, ensuring practical deployability in production networks where new slices are frequently instantiated and existing ones modified. The system must maintain Service Level Agreement (SLA) compliance across heterogeneous service types while operating within the computational constraints of real-time network control, addressing the gap between theoretical Reinforcement Learning (RL) approaches and the practical requirements of production 5G network operations.

1.3 Contributions

Current reinforcement learning approaches for network management typically address isolated aspects of the problem such as bandwidth allocation or scheduling, without considering the holistic view of service assurance that encompasses both network and compute resources. Many solutions are evaluated in simplified simulators with synthetic traffic patterns rather than realistic network conditions. Additionally, most existing multi-agent implementations do not leverage truly parallel agent architectures that align with the distributed nature of network slice management.

This work proposes a reinforcement learning-based closed-loop algorithm for 5G service assurance that:

- Formulates elastic per-slice resource control under shared constraints as a partially

observable control problem with KPI-aware reward engineering to balance QoS adherence against resource and thus energy savings.

- Employs stable policy-optimization updates to adapt compute (CPU, RAM, Storage) and network (link bandwidth) capacities online, guided by short-horizon telemetry and prediction.

- Supports both single-agent and multi-agent control aligned with slice modularity, and validates the approach in packet-level OMNeT++/Simu5G simulation environment with realistic urban mobility traffic from the city of Montreal.

- Demonstrates high KPI adherence with reduced computational time for multi-agent control, and accurate per-slice resource predictions under dynamic loads, without requiring parameter returning when extending from single-slice to multi-slice operation.

1.4 Plan of Thesis

The remainder of this thesis is organized as follows. Chapter 2 provides a comprehensive background on the foundational technologies enabling intelligent 5G networks as well as an overview of Reinforcement Learning (a subset of Machine Learning that automates control tasks). It also covers the 5G network architecture including the roles of Software-Defined Networking and Network Function Virtualization, Multi-Access Edge Computing, End-to-End Network Slicing concepts, Service Assurance mechanisms, and AI/ML-driven orchestration approaches. The chapter concludes with a detailed examination of reinforcement learning approaches in 5G networking contexts.

Chapter 3 presents a detailed analysis of three key research papers that provide important insights related to our approach. These papers are selected based on their

relevance to understanding critical aspects of reinforcement learning in network management, multi-agent systems, and service assurance mechanisms.

Chapter 4 presents our main contribution, a journal paper titled "". This paper develops a reinforcement learning framework for automated service assurance in 5G and B5G networks, building upon key papers analyzed in section 2.

Chapter 5 summarizes the contributions presented in this thesis and outlines potential directions for future work, including extensions of the proposed reinforcement learning framework to emerging 6G network paradigms and integration with other AI/ML techniques for enhanced network intelligence.

Chapter 2

Background

2.1 5G Network Architecture

The 5G system architecture is designed to accommodate data connection and services, leveraging techniques such as software-defined networking (SDN) and network function virtualization (NFV) [7]. SDN promotes network adaptability by separating the network's control and data planes, allowing network managers to be directly programmable while abstracting the underlying infrastructure for applications and network services. NFV enables the replacement of network services on specialized appliances with virtualized instances operating as software on commercially available hardware. The cellular network offers wireless access to mobile devices (User Equipment, or UE) which may include smart devices, automobiles, drones, industrial and medical equipment [6]. 5G provides wireless cell networks with high speed, enhanced reliability, and low latency via a unified, more capable air interface developed to enable next-generation customer experiences, support new deployment patterns, and offer new services. A common 5G cellular network consists of three main components: Radio Access Network (RAN), Transport Network (TN), and Core Network (CN) [24]. The RAN maintains the radio spectrum, ensuring effective utilization

while satisfying quality-of-service needs for every user. The transport network handles connection services required by the 5G network, including access, aggregation, and core layers. The core network serves as the central component that enables access to services and delivers multiple services to interconnected customers.

2.1.1 Software-Defined Networking (SDN) and Network Function Virtualization (NFV)

Software-Defined Networking (SDN) represents a fundamental paradigm shift in network architecture design, enabling programmable and centralized control of network behavior through the separation of control and data planes [14]. SDN allows network operators to apply high-level policies, dynamically reconfigure routes, and optimize performance in real-time based on network conditions and objectives such as latency, bandwidth, or energy efficiency. In the context of 5G networks, SDN provides the foundation for implementing dynamic service provisioning, traffic engineering, and quality of service management across heterogeneous network infrastructures.

Network Function Virtualization (NFV) complements SDN by enabling the deployment of network functions as software applications running on general-purpose computing hardware, rather than dedicated proprietary appliances [42]. NFV enables the decoupling of network functions from proprietary hardware appliances, allowing them to run as Virtual Network Functions (VNFs) on general-purpose computing infrastructure, reducing both CapEx (Capital Expenditures) and OpEx (Operating Expenses) while increasing scalability and resource utilization. The combination of SDN and NFV creates a highly flexible and programmable network environment that is essential for implementing intelligent service assurance mechanisms.

In the 5G RAN architecture, functional disaggregation has been implemented to enable independent scaling and deployment of different network functions. The

traditional monolithic base station (eNodeB) has been decomposed into three logical units that can be deployed independently:

- The **Radio Unit (RU)** is responsible for analog and digital signal processing at the radio interface, including digital beamforming, analog-to-digital conversion, and low-level physical layer functions. The RU is typically deployed close to or integrated with antenna systems and connects to the Distributed Unit through fronthaul interfaces. The strategic placement of RUs enables efficient radio resource utilization and supports advanced antenna technologies such as massive MIMO and beamforming.

- The **Distributed Unit (DU)** executes real-time functions including Medium Access Control (MAC), Radio Link Control (RLC), and portions of the physical layer processing. Due to strict timing requirements, DUs are typically located near RUs to ensure low-latency communication. The virtualization of DU functions enables flexible deployment on edge computing platforms, supporting localized processing and rapid response to changing radio conditions.

- The **Centralized Unit (CU)** handles non-real-time functions including Radio Resource Control (RRC) and Packet Data Convergence Protocol (PDCP). The less latency-critical nature of CU functions allows for centralized deployment in data centers, enabling resource pooling and coordination across multiple cells. The CU connects to DUs through midhaul interfaces and to the core network through backhaul connections.

The **User Plane Function (UPF)** serves as a critical component in the 5G core network architecture, responsible for packet forwarding and traffic handling between access and data networks. The UPF operates in the user data plane and is often distributed closer to the network edge to reduce latency and offload traffic from

centralized cores through Multi-Access Edge Computing technology. The strategic placement and dynamic scaling of UPF instances enable efficient traffic routing and support for low-latency applications while facilitating load distribution across network infrastructure.

2.1.2 Multi-Access Edge Computing

Multi-Access Edge Computing (MEC) emerges as a critical enabler for 5G networks, bringing computational capabilities, data storage, and application services closer to end users to minimize latency and improve quality of experience [10]. Application of intelligent data analytics using machine learning in management of 5G networks can enable autonomous networking capabilities, with management systems supporting detection of anomalous network behavior and resource optimization. MEC platforms are strategically deployed at network edges, including base station sites, aggregation points, and local data centers, creating a distributed computing infrastructure that complements centralized cloud resources.

The integration of MEC with 5G network slicing enables the deployment of slice-specific applications and services at edge locations, reducing end-to-end latency and improving service responsiveness. MEC platforms can host virtual network functions, application servers, and AI/ML processing engines that support real-time analytics and decision-making for service assurance applications. This distributed deployment model is particularly important for supporting URLLC applications that require guaranteed low-latency responses and for implementing intelligent network management functions that must process large volumes of network telemetry data in real-time.

Energy efficiency considerations are paramount in MEC deployment strategies, as edge computing infrastructure must balance computational capabilities with power consumption constraints. The dynamic nature of edge computing workloads requires

intelligent resource management mechanisms that can adapt to varying computational demands while optimizing energy efficiency. This creates opportunities for AI-driven orchestration systems that can learn optimal resource allocation policies based on application requirements and energy availability.

2.1.3 End-to-End Network Slicing in 5G

Network slicing represents the cornerstone technology enabling 5G networks to support diverse applications with heterogeneous performance requirements on a unified infrastructure. Network slicing is a promising technology for 5G networks to provide services tailored for users' specific QoS demands, driven by increased massive wireless data traffic from different application scenarios. The concept of network slicing extends beyond simple resource partitioning to encompass the creation of logically isolated, end-to-end network services that can be customized to meet specific application requirements while sharing underlying physical infrastructure[9, 11].

The 3GPP standards define three primary service categories that are typically implemented as dedicated network slices, each with distinct performance characteristics and optimization objectives:

- **Enhanced Mobile Broadband (eMBB)** services focus on delivering high data throughput and extensive coverage for bandwidth-intensive applications. eMBB slices are optimized for applications such as 4K/8K video streaming, virtual and augmented reality experiences, and cloud gaming services that require sustained high data rates and moderate latency constraints. The resource allocation strategies for eMBB slices prioritize spectrum efficiency and capacity maximization while maintaining acceptable quality of experience for multimedia applications.

- **Ultra-Reliable Low-Latency Communications (URLLC)** services are designed for latency-critical and reliability-sensitive applications including autonomous vehicles, industrial automation, remote surgery, and mission-critical control systems. URLLC slices must guarantee extremely stringent delay bounds (typically less than 1 millisecond) and near-perfect reliability (99.999% availability). The resource management for URLLC slices requires dedicated spectrum allocation, priority-based scheduling, and redundant resource provisioning to ensure consistent performance under all operating conditions.

- **Massive Machine-Type Communications (mMTC)** services target massive-scale, low-power communication among vast numbers of IoT devices including sensors, smart meters, and monitoring equipment. mMTC slices prioritize energy efficiency and scalability over high bandwidth requirements, supporting millions of devices per square kilometer while maintaining long battery life for connected devices. The resource allocation for mMTC slices focuses on efficient spectrum utilization through advanced access techniques and optimized signaling procedures.

The implementation of end-to-end network slicing requires sophisticated orchestration mechanisms that can instantiate, configure, and manage slice resources across multiple network domains. These orchestration systems must coordinate resource allocation decisions across RAN, transport, and core network segments while maintaining slice isolation and performance guarantees. The dynamic nature of the slice requirements, driven by varying traffic patterns and application demands, necessitates intelligent management systems capable of real-time resource optimization.

2.1.4 Service Assurance in 5G Networks

Service assurance in 5G networks encompasses a comprehensive framework of policies, procedures, and mechanisms designed to ensure that network services are delivered according to predefined quality standards and Service Level Agreement (SLA) specifications. Service assurance is required to collaborate with orchestration and management to automate the slice provisioning process, forming closed loops between assurance and orchestration to enable automation through cooperation between monitoring, analytics, and orchestration. The evolution from reactive to proactive service assurance represents a fundamental shift in network operations, enabled by advanced monitoring capabilities and AI-driven analytics.

SLAs establish contractual commitments between service providers and customers, specifying performance targets, availability guarantees, and penalty clauses for service violations. In 5G networks, SLAs must accommodate the diverse requirements of different application domains while providing measurable metrics for service quality evaluation [8, 5]. The translation of high-level SLA commitments into technical Quality of Service (QoS) parameters requires sophisticated mapping mechanisms that can account for the complex relationships between user-perceived quality and network-level performance indicators.

QoS mechanisms in 5G networks operate through the QoS Flow framework, which provides fine-grained control over packet forwarding treatment for different traffic types within network slices. The 5G QoS model relies on QoS flows that enable both guaranteed and non-guaranteed flow bit rate services while providing reflective QoS capabilities. QoS flows are managed by Session Management Functions and configured during PDU session establishment, creating end-to-end service guarantees that span from user equipment to application servers.

Key Performance Indicators(KPIs) serve as the foundation for monitoring and

managing service quality in 5G networks. Primary KPIs include throughput metrics that measure data transfer rates across network links and slices, latency metrics that quantify end-to-end delay for different service types, jitter measurements that assess delay variation and its impact on real-time applications, and packet loss ratios that indicate network congestion and performance degradation. These KPIs are continuously monitored across multiple network layers and aggregated to provide comprehensive views of service quality and network performance.

The implementation of effective service assurance requires integration with network orchestration and management systems, creating closed-loop control mechanisms that can automatically respond to performance issues and optimization opportunities. Intent-driven network management aims at building autonomous systems with minimal expert-driven policies, which may be applied to slice assurance through intelligent agents during prediction, assurance, evaluation, and actuation phases. This integration enables automated service lifecycle management, from initial slice provisioning through ongoing optimization and eventual decommissioning.

2.1.5 AI/ML-Driven Orchestration and Network Automation

The integration of artificial intelligence (AI) and machine learning (ML) into 5G network orchestration represents a transformative approach to managing the complexity and scale of modern telecommunications infrastructure. AI and ML can be seen as potential drivers in the automation and optimization of network performances and management complexities, with shifting network behaviors and complicated modern applications presenting diverse network performance traffic. AI-driven orchestration systems enable autonomous network management capabilities that can adapt to dynamic conditions, predict future requirements, and optimize resource allocation without human intervention [3, 37].

ML algorithms are particularly well-suited for addressing the high-dimensional, non-linear optimization problems that characterize 5G network management. ML techniques in RAN slicing enable resource management to instantiate and operate network slices while meeting performance and functional requirements, addressing challenges due to network dynamics and specific application requirements. These algorithms can learn complex patterns from network telemetry data, identify optimal resource allocation strategies, and adapt to changing network conditions through continuous learning and policy refinement.

The deployment of AI/ML-driven orchestration introduces several key capabilities that are essential for effective service assurance. Predictive analytics enable proactive identification of potential service degradation events, allowing preventive actions to be taken before performance issues impact end users. Anomaly detection algorithms can identify unusual network behavior patterns that may indicate security threats, equipment failures, or configuration errors. Automated optimization engines can continuously adjust network parameters to maintain optimal performance while minimizing resource consumption and operational costs.

The distributed nature of 5G networks, with virtualized functions deployed across edge, aggregation, and core locations, requires orchestration systems that can coordinate decisions across multiple domains and timescales. AI-driven orchestration frameworks must balance local optimization decisions with global network objectives while maintaining real-time responsiveness to changing conditions. This requires sophisticated coordination mechanisms that can handle the complexity of multi-layer, multi-domain resource allocation while ensuring consistent service quality across all network slices.

2.2 Artificial Intelligence in 5G Networking

Artificial intelligence (AI) and machine learning (ML) have emerged as key enablers for addressing these challenges in 5G and beyond networks. By leveraging data-driven algorithms, AI/ML can learn complex patterns, make predictions, and automate decision-making in ways that traditional fixed or reactive algorithms cannot. Indeed, academia and industry are actively exploring AI/ML techniques to realize the vision of the intelligent network capable of self-planning, self-optimization, and self-healing without requiring any human intervention. Concretely, AI/ML is being applied to multiple aspects of 5G network management.

AI/ML techniques are being applied across multiple layers of 5G network management. In radio access networks, machine learning algorithms assist with spectrum allocation and resource partitioning among network slices. Deep reinforcement learning approaches have been explored for dynamic resource allocation in scenarios with multiple traffic classes and varying service priorities [15, 33].

For network slice management, AI/ML methods are being investigated for bandwidth allocation among slices [16], capacity adjustment in response to changing latency requirements [34], and coordination of compute and network resources. Energy-aware deployment strategies using reinforcement learning have been proposed to incorporate power consumption considerations alongside service quality objectives [41].

Multi-agent reinforcement learning architectures are being developed for distributed network control, where multiple agents manage different network segments or functions. Approaches include coordinated control of admission and slicing decisions [36], joint optimization across multiple slices [43], and the use of graph neural networks to model relationships between network elements [35]. These distributed methods aim to manage networks where control decisions must be made across numerous geographically separated components.

The 5G PPP white paper on "AI and ML – Enablers for B5G Networks" [12] underscores that as B5G networks grow in complexity and scale, automation through AI/ML becomes indispensable for guiding network decisions in operation, planning, and optimization. In essence, AI/ML techniques offer a path to manage the multi-dimensional, multi-layer complexity of the 5G networks in a cost-effective and proactive manner.

2.2.1 Reinforcement Learning Overview

Among the various AI/ML approaches, reinforcement learning (RL) has gained a lot of traction in the networking domain because it learns to make sequences of effective control decisions from data. In an RL paradigm, problems are formulated as Markov decision process (MDP), where a controller called the agent interacts with an environment [29, 21]. At discrete decision times $t = 0, 1, 2, ...$, the environment produces a **state** denoted $s_t \in \mathcal{S}$, then the agent selects an **action**, denoted $a_t \in \mathcal{A}$, the environment yields a **reward** $r_{t+1} \in \mathcal{R}$, and the system transitions to the next state s_{t+1} according to the dynamics induced by the action. For network management problems, these interactions, often referred to as actions, are performed on the network, which here is treated as the environment for the agent. We write $\pi(a|b)$ for a **policy**, a conditional disctribution or rule that assigns probabilities to actions given the present state, and we use the shorthand $(s_t, a_t, r_{t+1}, s_{t+1})$ for the interaction tuple at time t. At each decision step, the agent observes a state that represents the full configuration that matters for control. The agent using the observed state then takes an action based on the observed state. This generates a new state in the network along with a reward that reflects the agent's immediate performance. The goal of an RL agent is to learn a policy which is a rule mapping the available information to actions that maximizes expected performance over time. Deep reinforcement learning

combines RL with neural networks, enabling the agent to handle high-dimensional inputs and complex state spaces such as network state information.

In theory, the Markov decision process assumes that the agent sees the full state. In practice, however, modern networks produce enormous volumes of telemetry, therefore, the measurements are aggregated and summarized before control decisions. These measurements are delayed and noisy and many variables are not directly visible. As a result, the agent does not observe the full network state, which makes the problem a partially observable Markov decision process. For readability, we keep the usual MDP notation and interpret the state as a belief or learned summary of recent observations and actions. This convention preserves clarity while remaining faithful to how network control actually operates.

Within this view, RL can generally be categorized into model-based and model-free approaches. In model-based RL, the core idea is to build an explicit description of how the system evolves and of how immediate performance is scored. Concretely, it specifies or learns a transition operator that gives the probability of the next state given the current state and the chosen action, along with a reward function that evaluates the short term consequences of that state-action pair. With these two ingredients, an agent can plan ahead by rolling the model forward over several decision steps to compute summaries of long-term return (cumulative reward over time), and then choose the action that performs best under those predictions. Modern communication networks make model-based reinforcement learning hard. The state spans radio, queues, scheduling, transport, and service chains, while measurements are delayed, noisy, and incomplete. Cross layer feedback yields nonlinear and sometimes abrupt responses, so rigid models miss key interactions and highly flexible ones require large and shifting datasets as policies change, which weakens identification and

generalization. Planning on such models accumulates small errors over long look-ahead and must run under tight control budgets, so horizons are short and pruning is aggressive. At the same time, the system drifts as software, mobility, and traffic evolve, which forces continual retraining and creates periods where predictions lag reality. The approach remains attractive for data efficiency and counterfactual analysis, yet a robust and faithful model at scale is uncommon in practice.

2.2.2 Model-free Reinforcement Learning

Model-free reinforcement learning learns control directly from interaction data, by turning experience into numerical targets that shape a decision rule. At each decision step (indexed by $t \in 0, 1, ...$), the agent observes a **state** s_t in state space $\mathcal{S}$, selects an **action** a_t in an action space $\mathcal{A}$ according to a **policy** $\pi(a|s)$, receives a **reward** $r_{t+1} \in \mathbb{R}$, and the environment moves to a new state $s_{t+1} \sim P(\cdot|s_t, a_t)$ under a transition kernel P. We retain MDP notation and interpret s_t as the information state used for control. Consider an infinite-horizon control problem with discount $\gamma \in [0, 1)$. The fundamental quantity is the **discounted return** G_t, a weighted sum of future rewards that formalizes long-term performance and ensures that distant outcomes count less than immediate ones.

$$G_t = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k+1},$$

where $\gamma \in [0, 1)$ is the **discount factor** that geometrically attenuates the influence of distant outcomes and r_{t+k+1} is the reward realized k steps ahead. This return brings about two value functions. The **state-value** function $V^\pi(s)$ quantifies the expected return from a state $s \in \mathcal{S}$ when following π, whereas the **action-value** function $Q^\pi(s, a)$ quantifies the expected return from a state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$. Both

are conditional expectations under the policy π.

$$V^{\pi}(s) = \mathbb{E}_{\pi}[\, G_t \mid s_t = s \,],$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[\, G_t \mid s_t = s,\ a_t = a \,].$$

These value functions are not arbitrary. For a fixed policy π, each must satisfy a Bellman self-consistency relation. The Bellman operator $\mathcal{T}^{\pi}$ maps a candidate value function to its one-step lookahead under π.

$$\mathcal{T}^{\pi}V(s) = \mathbb{E}_{\pi}[\, r_{t+1} + \gamma\, V(s_{t+1}) \mid s_t = s \,],$$

The fixed point of this operator is the true value, $V^{\pi} = \mathcal{T}^{\pi}V^{\pi}$. An action-value form is defined analogously by taking the expectation over the next action $a_{t+1} \sim \pi(\cdot|s_{t+1}$:

$$\mathcal{T}^{\pi}Q(s,a) \;=\; \mathbb{E}_{\pi}\big[\, r_{t+1} + \gamma\, Q(s_{t+1}, a_{t+1}) \,\big|\, s_t = s, a_t = a \,\big], \qquad Q^{\pi} = \mathcal{T}^{\pi}Q^{\pi}.$$

For $0 \le \gamma \le 1$, the operator $\mathcal{T}^{\pi}$ is a contraction in $||\cdot||_{\infty}$ and admits a unique fixed point, which ensures the existence and uniqueness of V^{π} and Q^{π}. This property motivates bootstrapped estimation: a target for the current step can combine an observed reward with a prediction of the next step built from the current estimate.

Temporal-difference (TD) learning realizes this idea by regressing toward the Bellman fixed point using sample-level discrepancies. With a differentiable approximator $V_{\phi} : \mathcal{S} \to \mathbb{R}$, the **temporal-difference error** at time t is

$$\delta_t \;=\; r_{t+1} + \gamma\, V_{\phi}(s_{t+1}) - V_{\phi}(s_t),$$

and the parameters update by stochastic approximation

$$\phi \leftarrow \phi + \alpha\, \delta_t\, \nabla_\phi V_\phi(s_t).$$

Here $\alpha > 0$ is the step size, also known as the learning rate and $\nabla_\phi V_\phi(s_t)$ is the gradient of the approximated value with respect to its parameters ϕ. Conditioning on $s_t = s$ under the policy π shows that the TD error is a sample of the Bellman residual:

$$\mathbb{E}_\pi[\, \delta_t \mid s_t = s \,] \;=\; \mathcal{T}^\pi V_\phi(s) - V_\phi(s).$$

Averaging this residual under the stationary state distribution d^π induced by π yields the mean-squared Bellman error,

$$\mathrm{MSBE}(V_\phi) \;=\; \mathbb{E}_{s\sim d^\pi}\Big[\big(\mathcal{T}^\pi V_\phi(s) - V_\phi(s)\big)^2\Big],$$

so each stochastic step moves V_ϕ towards the projection of the fixed point in the function class.

The choice of target inside δ_t determines the bias–variance profile of the update. Here, **bias** means the gap between the expected target and the true value $V^\pi(s)$ and **variance** means the dispersion of that target across trajectories with the same present state. Two canonical targets delimit the spectrum. The **Monte Carlo** target uses only realized rewards and is unbiased in expectation but typically high-variance:

$$\hat{G}_t^{\mathrm{MC}} \;=\; \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}, \qquad \mathbb{E}_\pi\Big[\hat{G}_t^{\mathrm{MC}} \mid s_t = s\Big] \;=\; V^\pi(s).$$

The **one-step bootstrap** (TD(0)) replaces the random tail with a prediction, reducing variance but introducing bias whenever $V_\phi \neq V^\pi$:

$$\hat{G}_t^{(1)} \;=\; r_{t+1} + \gamma\, V_\phi(s_{t+1}), \qquad \mathbb{E}_\pi\Big[\hat{G}_t^{(1)} \mid s_t = s\Big] \;=\; \mathcal{T}^\pi V_\phi(s).$$

Extending the lookahead before bootstrapping interpolates between these ends. The n**-step return** delays the first bootstrap to step n,

$$\hat{G}_t^{(n)} \;=\; \sum_{k=0}^{n-1} \gamma^k r_{t+k+1} \;+\; \gamma^n V_\phi(s_{t+n}), \qquad n \geq 1,$$

and the residual $\hat{G}_t^{(n)} - V_\phi(s_t)$ replaces δ_t in the same update rule. Larger n reduces bootstrap bias by admitting more realized reward, while smaller n curbs variance and computational cost.

A single horizon n is rarely uniformly appropriate across states or time scales, which motivates a target that adapts effective depth by averaging horizons with principled weights. The λ**-return** forms an exponentially weighted mixture of all n-step returns,

$$\hat{G}_t^{\lambda} \;=\; (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1}\, \hat{G}_t^{(n)}, \qquad \lambda \in [0,1],$$

and admits an equivalent recursive form convenient for online computation,

$$\hat{G}_t^{\lambda} \;=\; r_{t+1} + \gamma\Big((1-\lambda)\, V_\phi(s_{t+1}) + \lambda\, \hat{G}_{t+1}^{\lambda}\Big).$$

Substituting $\hat{G}_t^{\lambda}$ into the same stochastic step yields TD(λ):

$$\phi \leftarrow \phi + \alpha\, \big(\hat{G}_t^{\lambda} - V_\phi(s_t)\big)\, \nabla_\phi V_\phi(s_t).$$

The parameter λ, therefore, balances bias and variance through the composition of

the target itself: as $\lambda \to 0$, the update becomes TD(0) with minimal variance and maximal bootstrap bias, whereas as $\lambda \to 1$, it approaches Monte Carlo with vanishing bias in expectation and maximal variance. Intermediate λ aligns the effective horizon with the environment's mixing time, transmitting sufficient delayed credit without amplifying noise from far-future randomness.

The same mechanism can be expressed in the backward view using an eligibility trace e_t that accumulates recent sensitivities and decays geometrically with $\gamma\lambda$:

$$e_t \;=\; \gamma\lambda\, e_{t-1} \;+\; \nabla_\phi V_\phi(s_t), \qquad e_{-1} = 0,$$

with the local update

$$\phi \leftarrow \phi + \alpha\, \delta_t\, e_t.$$

This backward formulation reveals how current TD errors distribute credit to recently visited states in proportion to their eligibility, tying together the return G_t, the value functions V^π, Q^π, the Bellman operator $\mathcal{T}^\pi$, and the sample-level residual δ_t within a single pipeline governed by γ and λ. The action-value case follows identically by replacing $V_\phi(s)$ with $Q_\psi(s,a)$ and conditioning on actions where appropriate:

$$\delta_t^Q \;=\; r_{t+1} \;+\; \gamma\, Q_\psi(s_{t+1}, a_{t+1}) \;-\; Q_\psi(s_t, a_t).$$

Chapter 3

Key Papers

This section examines key papers that represent significant contributions to the understanding and implementation of AI-driven network management and service assurance in 5G networks. These papers were selected based on their relevance to our research contributions, their methodological innovations, and their influence on current approaches to intelligent network optimization.

3.1 Deep Reinforcement Learning for Online Resource Allocation in Network Slicing

The first key paper, published in IEEE Transactions on Mobile Computing, by Cai et al. [15], addresses the fundamental challenge of dynamic resource allocation in 5G network slicing environments. The authors propose a comprehensive framework that formulates resource allocation as a time-sequential dynamic optimization problem, incorporating system stability constraints, resource limitations, different timescales, and long-term system performance objectives. The paper's primary contribution lies in developing a dynamic RAN slicing model that accommodates multiple traffic

distributions with diverse priorities within the same slice, while accounting for time-varying resource availability.

The methodological approach employs deep reinforcement learning with an actor-critic architecture to handle the complex state spaces and continuous action domains inherent in network resource allocation. The authors demonstrate that their approach significantly outperforms traditional static allocation methods and reactive resource management strategies. The paper's treatment of heterogeneous user request types and varying priorities provides valuable insights into managing the complexity of multi-service 5G environments. The experimental validation using both synthetic and real-world traffic data strengthens the practical applicability of their findings.

The significance of this work lies in its comprehensive treatment of temporal dynamics in resource allocation and its consideration of system stability as a fundamental constraint. The authors' formulation of the problem as a sequential decision-making process under uncertainty provides a solid mathematical foundation for understanding how reinforcement learning can be applied to network management problems. Their results demonstrate substantial improvements in resource utilization efficiency and system performance compared to conventional approaches.

3.2 Energy-Aware Design Policy for Network Slicing Using Deep Reinforcement Learning

The second key paper, published in IEEE Transactions on Services Computing [41], tackles the critical challenge of energy-efficient network slice deployment and management. The authors recognize that while network slicing enables diverse service provisioning, the energy consumption implications of slice deployment strategies have received limited attention in existing literature. Their work addresses this gap by

proposing an energy-aware design policy that optimizes both energy consumption and deployment capacity through deep reinforcement learning.

The paper introduces an innovative actor-critic architecture enhanced with pointer networks and attention mechanisms to handle the combinatorial nature of slice placement decisions. The attention mechanism enables the learning agent to focus on the most relevant network elements when making deployment decisions, while the pointer network structure facilitates handling variable-length input sequences corresponding to different network configurations. The authors demonstrate that their approach achieves significant energy savings compared to conventional deployment strategies while maintaining service quality requirements.

The experimental evaluation encompasses diverse network scenarios and traffic patterns, validating the approach's effectiveness across different operational conditions. The paper's treatment of the trade-off between energy efficiency and service performance provides practical insights for network operators seeking to minimize environmental impact while meeting customer expectations. The authors' consideration of both short-term operational costs and long-term sustainability objectives reflects the growing importance of environmental responsibility in network operations.

This work contributes to our understanding by demonstrating how reinforcement learning can be extended beyond traditional performance optimization to incorporate sustainability objectives. The integration of energy awareness into slice management decisions represents an important step toward green networking technologies that will be essential for future telecommunications infrastructure.

3.3 Deep Reinforcement Learning for Online Resource Allocation in Network Slicing

The third key paper, published in IEEE Transactions on Vehicular Technology [35], addresses the challenge of coordinated resource management in dense cellular deployments where spatial relationships and inter-cell dependencies significantly impact system performance. The authors develop a multi-agent reinforcement learning framework enhanced with graph attention networks to capture the complex interdependencies between network elements in 5G slicing scenarios. Their approach recognizes that resource allocation decisions at one base station affect neighboring cells through interference patterns and shared backhaul constraints, necessitating coordination mechanisms that account for network topology. The methodological innovation centers on integrating graph attention mechanisms into the multi-agent learning process, where each agent corresponds to a base station or network element managing local resources. The graph structure encodes the physical network topology, with nodes representing network elements and edges capturing interference relationships or resource sharing constraints. The attention mechanism enables agents to dynamically weight the influence of neighboring agents' states and actions based on current network conditions, allowing the system to adapt its coordination patterns as traffic distributions shift. This architecture addresses the challenge of credit assignment in multi-agent systems by explicitly modeling how local decisions propagate through the network structure. The experimental framework demonstrates the approach's effectiveness in dense urban scenarios with varying traffic loads and mobility patterns. The authors evaluate performance across multiple slice types with distinct QoS requirements, showing that the graph attention mechanism improves resource utilization efficiency compared to independent learning agents. The framework's ability to maintain slice isolation while

enabling beneficial coordination represents a practical advancement for dense 5G deployments. However, the focus on radio resource allocation leaves questions about integration with compute resource management and end-to-end service orchestration. This work contributes to our understanding by demonstrating how structural information about network topology can be incorporated into multi-agent learning architectures. The graph attention mechanism provides a principled approach to managing the locality of interactions in distributed systems, where full coordination is computationally prohibitive but complete independence sacrifices performance. The paper's treatment of dense cellular scenarios highlights scalability challenges that become critical as 5G networks evolve toward ultra-dense deployments with thousands of small cells per square kilometer.

Chapter 4

Multi-Agent Reinforcement Learning for Automated Service Assurance and Dynamic Resource Allocation in 5G Network Slicing

This chapter has been submitted as a Journal paper titled "Multi-Agent Reinforcement Learning for Automated Service Assurance and Dynamic Resource Allocation in 5G Network Slicing" written by M. Angrish, B. Jaumard, and O. Collao.

4.1 Abstract

Service assurance in 5G networks requires dynamic resource management to meet diverse Quality of Service requirements across heterogeneous network slices. This paper proposes a reinforcement learning approach for automated service assurance that dynamically allocates compute and network resources while maintaining Key

Performance Indicators within specified bounds. We formulate the problem as a Partially Observable Markov Decision Process and employ Proximal Policy Optimization with reward engineering that balances service quality against resource efficiency. Our framework simultaneously manages network slices with varying performance requirements, from ultra-low latency services to massive-scale IoT deployments, each with distinct KPI thresholds for throughput, delay, jitter, and packet loss. The algorithm dynamically adjusts CPU, RAM, disk, and link bandwidth capacities based on real-time telemetry, utilizing a novel utilization transformation function that aligns resource usage with operator-defined targets. We implement both single-agent and truly parallel multi-agent configurations, where independent agents control per-slice resources concurrently. Validation using OMNET++ with the Simu5G library and refactored Montreal urban traffic patterns demonstrates that the multi-agent approach effectively manages resource allocation for each slice while reducing computational time compared to single-agent control. Results show robust KPI adherence under dynamic traffic conditions, with the ranged transformation function achieving highest resource utilization gains while the intersection transformation provides conservative operation with stronger Service Level Agreement protection. The framework generalizes from single-slice to multi-slice scenarios without parameter retuning, validating its practical applicability for production 5G networks.

4.2 Introduction

In the context of 5G and B5G-area networks, service assurance becomes highly critical. The complexity of these networks, with their diverse services and intricate architecture, needs seamless interoperability among service assurance processes. Communications Service Providers (CSPs) have put much of their developmental focus on 5G in the radio in a first stage, to prove the access technology will work. Now they

are turning their focus to building 5G access and core networks at scale, meaning they worry about 5G E2E service assurance. In doing so, they are concerned with the economics of 5G operations, with service assurance becoming a particularly hot topic because it lies at the heart of network-oriented operations and services offered to users.

Service assurance can be broadly defined as a set of policies and processes applied to ensure that services are delivered in a manner that meets a predefined level of quality, to meet growing customer expectations for high quality of service in a landscape where the speed and scale of services is increasing significantly and very quickly. The evolution of service assurance in 5G and B5G environments today involves a data-centric approach to move from traditional reactive to proactive policies, even if data is not readily available and there is still poor end-to-end visibility between layers, especially in today's cloud-native world, including, for example, hardware, virtualized resources, services and client layers.

This study aims to propose a reinforcement learning model and algorithm, which provides automated service assurance in the context of dynamic 5G traffic with 4 different slices, each of them targeting a distinct 5G traffic category: eMBB, URLLC and mMTC and the last associated with VoIP. Design of the algorithm is thought to be adaptive to slices with different soft or hard requirements, and to E2E 5G networks. Validation is done with a simulation framework using OMNET++ and the Simu5G [30] library. 5G traffic results from a refactoring of the pedestrian and vehicle traffic of the city of Montreal [28] around slices with different service assurance characteristics and peak hours.

Results show a very good adherence of the required network and compute resource prediction for each slice, in order to be able to guarantee service assurance guarantees that meet the requirements of 5G network operators. It therefore validate the

excellent adaptive behavior of the proposed reinforcement learning algorithm, and the superiority of the multi-agent one with respect to the reduction of the computational times.

The paper is organized as follows. We review the state of the art with respect to automated service assurance in 5G/B5G networks. Key concepts and definitions are recalled in Section 4.4, together with the service assurance problem statement of our study. We then describe in Section 4.5 the foundation of our Reinforcement Learning approach, tailored to be adaptive to network sizes and services with different service assurance requirements. Numerical results are presented in Section 4.6, together with the description of our simulation environment relying on OMNET++. Conclusions and future work are drawn in the last section.

4.3 Literature Review

Network slicing enables customized network services by virtualizing multiple logical networks on shared physical infrastructure. Managing these slices for service assurance requires addressing challenges in dynamic resource allocation, heterogeneous performance requirements, and operational efficiency. We review recent work in Reinforcement Learning (RL) and Multi-Agent Reinforcement Learning (MARL) as applied to network slicing and service assurance, organizing the discussion into four areas: general RL algorithms, Multi-Agent Reinforcement Learning (MARL) paradigms, RL for 5G network slicing, and MARL for 5G network slicing.

4.3.1 Reinforcement Learning

Reinforcement Learning involves an agent learning to make sequential decisions via interaction with an environment. Modern RL algorithms combine neural networks

with different learning strategies [29, 21]. Value-based methods such as Deep Q-Network (DQN) and its variants including Double DQN and Dueling DQN learn to estimate expected returns from state-action pairs. Policy gradient methods including Deep Deterministic Policy Gradient (DDPG), Twin Delayed DDPG, and Soft Actor-Critic (SAC) directly optimize policy parameters and can handle continuous action spaces. Actor-critic methods such as Asynchronous Advantage Actor-Critic and Proximal Policy Optimization (PPO) maintain both value functions and policies to reduce gradient variance during learning.

Algorithm selection in network resource management depends on whether resource allocations are discrete or continuous, whether online learning from sequential trajectories is required, and the available computational budget for training. On-policy methods like Proximal Policy Optimization (PPO) learn from current policy interactions, while off-policy methods like Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG) can reuse historical experience through replay buffers.

4.3.2 Multi-Agent Reinforcement Learning

Many network management problems involve multiple decision-making entities that must coordinate their actions. Multi-Agent Reinforcement Learning (MARL) addresses scenarios where several agents learn and operate concurrently [31]. Training approaches range from centralized architectures where a single network produces actions for all agents, to decentralized approaches where each agent maintains independent parameters and learns from local observations. Centralized training with decentralized execution represents an intermediate approach that uses shared networks during learning but deploys independent policies during operation.

For network slicing, different architectural choices affect computational requirements and operational characteristics. The degree of agent independence influences

how well the system scales with the number of slices and how coordination is achieved across distributed network resources.

4.3.3 Reinforcement Learning in 5G Networks

RL has been applied to various aspects of 5G network control and optimization. Cai et al. [15] study online Radio Access Network (RAN)-slicing decisions under time-varying resource availability using a Deep Reinforcement Learning (DRL) policy that couples admission and per-slice provisioning. Their evidence shows the learned policy adapts to traffic dynamics and outperforms static baselines. However, the study targets RAN-level control and does not validate end-to-end, packet-level Service Assurance (SA) KPIs, which we address in our closed-loop design.

Cheng et al. [16] reallocate Open Radio Access Network (O-RAN) midhaul bandwidth with an RL agent to boost target-slice throughput and latency proxies in a simulator. This does not model packet-level dynamics or verify KPI compliance, making it complementary to our packet-level SA focus. Qi et al. [33] introduce discrete normalized advantage functions to stabilize learning over discrete actions.

A common limitation pertaining to prior solutions is that they do not incorporate the operation side SA requirements in the decision making process. Raftopoulos et al. [34] trained a DRL agent to adapt to changing maximum tolerable latency and reduce SLA violations with respect to baseline algorithms. Aslan et al. [13] worked on optimizing allocated percentage bandwidth among slices by accepting or blocking user requests while incorporating SLA requirements in the reward function to the RL agent.

Beyond performance optimization, energy efficiency has emerged as a critical constraint in network slicing. Wang et al. [41] propose a DRL actor-critic with pointer and attention mechanisms to design energy-aware slice deployments. While effective

for energy and placement trade-offs, it does not evaluate packet-level SA KPIs, which our study brings into the loop. Still, many solutions are evaluated in simplified simulators or randomly generated data rather than a testbed or packet level 5G network simulator such as ns-3 [23] or OMNet++ [39].

4.3.4 Multi-Agent Reinforcement Learning in 5G Networks

Given the distributed nature of network control problems, researchers have explored MARL in context of 5G networks. Many problems in networks are inherently multi-agent such as controlling multiple slices, orchestrating resources across several network functions, and coordinating resource management across users. MARL is well suited to these scenarios since multiple agents can learn to perform distinct but related tasks in parallel, as illustrated by some recent works.

Sulaiman et al. [36] cast joint slicing and admission control as a MARL problem optimized for long-term revenue and acceptance rate in simulated Cloud Radio Access Network (C-RAN). Their evaluation is simulation-based and does not test packet-level effects or KPI compliance, which is the gap we target with a SA-aware controller. Zhou et al. [43] proposed a Multi-Agent DDPG algorithm to optimize performance in multiple slices.

Shao et al. [35] use graph attention to coordinate MARL agents across dense cells, improving inter-slice resource decisions. The work focuses on coordination efficacy rather than end-to-end SA verification, which we incorporate explicitly. Vila et al. [40] frame capacity-sharing among tenants as MARL. We adopt a related multi-agent perspective but extend it with explicit SA objectives and packet-level telemetry integration.

Implementation details regarding agent independence vary across these works. Some employ single policy networks that output joint actions for all agents, while

others maintain separate policy networks for each agent. For closed-loop service assurance, this distinction affects computational scaling and the ability to handle different types of agents with varying observation and action spaces.

Our work addresses closed-loop service assurance using multi-agent architectures where independent agents control compute and network resources for individual slices. We employ PPO with discrete action spaces for its stable on-policy updates and ability to handle both continuous and discrete domains. The approach is validated in a packet-level 5G simulator using traffic patterns derived from urban mobility data, incorporating both network and compute resource management within a unified RL framework that explicitly targets KPI compliance.

4.4 Closed Loop 5G Service Assurance Problem

We describe here the key elements pertaining to 5G service assurance, before stating the service assurance problem statement as we address it in this study. The system model we use is a 5G network model $\mathcal{N} = (V, L)$ where V is the set of nodes, and L is the set of bidirectional links, with both network and compute resources to be described below, as well as an E2E perspective. Details of the network model used in our simulation are summarized in Section 4.6.1 with details available in [18].

4.4.1 E2E 5G Network Slicing, Service Assurance and Automated Closed Loop

Network slicing is a key enabler of 5G, i.e., a network architecture that enables the multiplexing of virtualized and independent logical networks on the same physical network infrastructure. Each network slice is an isolated end-to-end network tailored to fulfill diverse requirements requested by a specific service. As traffic varies over

the time, both in terms of intensity and of service distributions, it is required to orchestrate service assurance, i.e., a process of continuous improvement that involves monitoring and analyzing KPI to identify areas for improvement, i.e., satisfying KPI bounds. This involves the design of a closed loop assurance (CLA) algorithm which, based on the prediction of network events, (e.g., congestion/traffic variations identified by the KPI prediction as in [38]), that are highly probable of causing service degradation, and automatically take preventive actions to avert congestion side effects. While generic closed loop assurance also includes interruptions due to failures, in this study, we only focus on the congestion/traffic variation disruptions.

Increasingly, 5G services are being supported by chaining capabilities across networks that span multiple domains and technologies, as well as a mix of physical, virtual, and cloud network functions that spread from the network access and edge to the core, with both telco and public clouds. As a consequence, closed loop assurance automation needs to be defined as an end-to-end service orchestration integrated with network and compute resource in order to meet the customer needs.

In the following paragraphs, we describe the KPIs that will be used by our closed-loop assurance algorithm, as well as the network and compute resources of our 5G network model that impact these KPIs.

4.4.2 KPIs: the heart of the closed loop assurance

Our E2E network model produces 4 Key Performance Indicators (KPIs) for each logical link (denoted by ℓ) of each network slice in operation, each over a given time window T (index is omitted in the sequel of this section for alleviating the notations), made of a given sequence of time slots.

The first KPI is the throughput, denoted by $\text{KPI}^{\text{TH}}_{\ell s}$ (or $\text{TH}_{\ell s}$ for short). Values vary depending on the logical link in each slice s and the UL/DL direction of these

links. It is defined as follows:

$$\text{KPI}_{\ell s}^{\text{TH}} = \#\text{ packets sent in time window } T \text{ (Mbps)}$$
$$\ell \in L_s = L_s^{\text{UL}} \cup L_s^{\text{DL}}, s \in S, \quad (1)$$

where L_s^{UL} and L_s^{DL} denote the set of uplink and downlink logical links in slice s, respectively.

The second KPI is associated with delay, denoted by $\text{KPI}_{\ell s}^{\delta}$ (or $\delta_{\ell s}$ for short). We distinguish the average and maximum delay:

$$\text{KPI}_{\ell s}^{\delta,\text{mean}} = \frac{1}{|T|}\sum_{t\in T}\delta_t, \qquad \text{KPI}_{\ell s}^{\delta,\text{max}} = \max_{t\in T}\delta_t$$
$$\ell \in L_s, s \in S. \quad (2)$$

Here, $|T|$ denotes the number of time slots in a given time window T with t being the time slot index. Hence, δ_t is the delay during time slot t.

The third KPI is the jitter, i.e., the mean deviation over delays δ_t in time window T. It is written as follows:

$$\text{KPI}_{\ell s}^{\text{JIT}} = \frac{1}{|T|-1}\sum_{t\in T}(|\delta_{t+1} - \delta_t|) \qquad \ell \in L_s, s \in S. \quad (3)$$

The fourth KPI is the packet loss, estimated as the ratio of difference of total packets sent, received and carried over to packets sent in the time window T. It is

written as follows:

$$\text{KPI}_{\ell s}^{\text{LOSS}} = \begin{cases} 0 & \text{if } P_{\ell s}^{\text{R}} + P_{sd}^{\text{C}} \geq P_{\ell s}^{\text{S}} \\ 0 & \text{if } P_{\ell s}^{\text{S}} = 0, \\ \dfrac{P_{\ell s}^{\text{S}} - P_{\ell s}^{\text{R}} + P_{\ell s}^{\text{C}}}{P_{\ell s}^{\text{S}}} & \text{otherwise,} \end{cases}$$

$$\ell \in L_s, s \in S, \quad (4)$$

where $P_{\ell s}^{\text{R}}$, $P_{\ell s}^{\text{C}}$ and $P_{\ell s}^{\text{S}}$ are the received, carried over and sent packets for slice s and direction d, respectively. The packet loss is non-zero when the number of sent packets in a time window T is greater than the sum of carried over and received packets. The number of carried over packets for time window $T+1$ is updated as follows:

$$P^{\text{C},T+1} = \begin{cases} P^{\text{R},T} + P^{\text{C},T} - P^{\text{S},T} & \text{if } P^{\text{R},T} + P^{\text{C},T} > P^{\text{S},T} \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

The packets are only carried over to the next time window $T+1$ if the sum of the number of received and carried over packets is larger than the number of sent packets.

In the sequel, we will assume that all KPI values are normalized, meaning we performed the following operation on each of them:

$$\text{KPI}_{\ell s}^{\square} \leftarrow \min\{\frac{\text{KPI}_{\ell s}^{\square}}{\text{KPI}_{\ell s}^{\square,\max}}, 1\}$$

$$\square \in \{\text{TH}, \text{JIT}, \delta, \text{pLOSS}\}, \ell \in L_s, s \in S, \quad (6)$$

where the KPI threshold values are described in Table 1 (Section 4.6), based on the 5G 3GPP 5QI [17] threshold values.

In the sequel, we will denote the KPI vector as follows, with the assumption that

all components are normalized:

$$\text{KPI\_vec} = (\text{KPI}^{\text{TH}}, \text{KPI}^{\delta}, \text{KPI}^{\text{JIT}}, \text{KPI}^{\text{PLOSS}}).$$

4.4.3 Network (Link) Resources

As part of our study, network resources are associated with the bandwidth available on the logical links (denoted by ℓ); we can modify their transport capacity as long as we do not reach the transport capacity of the physical links which contain them.

We denote by $C_{\ell s}^{T\,\max}$ the transport capacity of virtual link ℓ in slice s at time window T, and by $\text{U}_{\ell s}^{T}$ the bandwidth usage of virtual link ℓ in slice s during time window T. The relative bandwidth usage over T can be written as follows:

$$\text{U}_{\ell s}^{T}(\%) = \frac{\text{U}_{\ell s}^{T}}{C_{\ell s}^{T\,\max}} \qquad \ell \in L_s, s \in S, \tag{7}$$

where $\text{U}_{\ell s}^{T}$ can be defined as the maximum or average bandwidth usage during the time window T:

$$\text{U}_{\ell s}^{T,\text{mean}} = \sum_{t \in T} \text{U}_{\ell s}^{t}/|T| \qquad \text{or} \qquad \text{U}_{\ell s}^{T,\max} = \max_{t \in T} \text{U}_{\ell s}^{t}$$

$$\ell \in L_s, s \in S. \tag{8}$$

From now on, $\text{U}_{\ell s}^{T}$ will denote the relative average bandwidth usage over T, and

$$\text{U}^{\text{NET}}\text{\_vec} = (\text{U}_{\ell s}^{t})_{\ell \in L_s, s \in S}. \tag{9}$$

4.4.4 Compute (Cloud) Resources

We consider the three common compute resources $R = \{\text{CPU}, \text{RAM}, \text{DISK}\}$ for each VNF instance, e.g., the equivalent a virtual machine (VM) or container. Let VNF_set denote the set of all VNF instances. We denote by $C_{\square}^{\max}$ the capacity of compute resource $\square$ and by $\text{U}_{\square}^{T}$ the usage of compute resource $\square$ during time window T. Each compute resource is dynamically managed over a given time window T, and its relative usage value can be described as follows:

$$\text{U}_{\square}^{T,\text{VNF}}(\%) = \frac{\text{U}_{\square}^{T}}{C_{\max}^{\square}} \qquad \square \in \{\text{CPU}, \text{RAM}, \text{DISK}\}, \text{VNF} \in \text{VNF\_set}. \tag{10}$$

Again, $\text{U}_{\square}^{T}$ can be computed as either the max or the average over time window T, as for the KPI values or the link usage in (2) and (8), respectively.

In the sequel, for each VNF instance, we will denote the compute resource utilization vector as follows, assuming values are relative as defined in (10):

$$\text{U}^{\text{CLOUD}}\text{\_vec} = (\text{U}_{\text{VNF}}^{\text{CPU}}, \text{U}_{\text{VNF}}^{\text{RAM}}, \text{U}_{\text{VNF}}^{\text{DISK}}). \tag{11}$$

4.4.5 Closed Loop 5G Service Assurance: Problem Statement

The closed-loop 5G service assurance problem can be defined as the design and validation of an algorithm capable of continuously providing the expected quality of service (QoS), by automating the process of reconfiguring network and computational resources so that at any given time, each 5G service can meet its requirements. It should be noted that in a 5G network, slices share a given pool of network and computational resources which, when traffic varies, must be reallocated in order to minimize

the overall amount of resources required and, therefore, save energy consumption to manage the network and computing resources.

An efficient closed-loop 5G service assurance algorithm is one that can anticipate on the network or compute performance degradation impacts by proactive resource re-allocation, in order to maintain the proper 5G QoS requirements for each 5G service.

In the current study, QoS requirements will be expressed through selective KPI requirements for each service/slice, network and compute resources will be the virtual link capacities and the compute (cloud) resources (RAM, CPU, DISK), with the indicators as defined in the previous paragraphs of this section.

4.5 Reinforcement Learning Algorithm: Service Assurance Automation

The dynamic nature of 5G networks, characterized by diverse slice requirements and fluctuating traffic patterns and cost constraints, demands an intelligent control mechanism that can adapt in real-time without explicit programming for every scenario. This section presents our RL-based proactive closed-loop assurance for automated service assurance in 5G networks, designed to maintain KPI compliance while optimizing resource utilization across heterogeneous network slices.

4.5.1 Problem Formulation and Approach

The challenge of maintaining service assurance (SA) while minimizing operating expenses (OpEx) in network slicing environments requires continuous adaptation of resource allocations in response to varying traffic patterns. We formulate this challenge as a Partially Observable Markov Decision Process (POMDP), recognizing that

network controllers operate with delayed, aggregated telemetry rather than complete system state information. This formulation naturally captures the inherent uncertainty in network measurements and the temporal dynamics of resource demands.

Our solution employs on-policy RL using policy-optimization methods (e.g., Proximal Policy Optimization (PPO)), which provide stable learning through conservative policy updates. The algorithm learns from episodic interactions with the network environment, where each interaction captures the relationship between resource allocation decisions and their impact on service quality. The learning process balances exploration of new resource configurations against exploitation of proven strategies, guided by three critical measurement categories:

- **Compute Resource Utilization**: The fractional usage of dynamic maximum capacity for each compute resource type, denoted as $\mathrm{U}_{\mathrm{VNF}}^{\mathrm{Comp\_Res}}$ for Virtual Network Function (VNF) (VNF) instance VNF. This encompasses CPU, RAM, and Disk resources that support network slice operations.

- **Network Resource Utilization**: The fractional usage of logical network link capacity for each slice, denoted as $\mathrm{U}_{s}^{\mathrm{Link\_Res}}$ for slice s. This captures the bandwidth consumption patterns across different network segments.

- **Key Performance Indicator**: The service quality metrics including throughput ($\mathrm{KPI}^{\mathrm{TH}}$), packet loss ($\mathrm{KPI}^{\mathrm{pLOSS}}$), delay ($\mathrm{KPI}^{\delta}$), and jitter ($\mathrm{KPI}^{\mathrm{JIT}}$) that directly reflect user experience and service level agreement (SLA) compliance.

To enable consistent learning across heterogeneous slices with different performance requirements, we normalize all values to the range [0,1]. Resource utilization values are normalized relative to their allocated capacities, while KPI values are normalized (and clipped) relative to their SLA-defined thresholds. This normalization

creates a uniform representation space where the learning algorithm can effectively compare and optimize across diverse metrics.

We describe the components of our algorithm next: state (Section 4.5.2), actions (Section 4.5.3), rewards (Section 4.5.4), policies (Section 4.5.5).

4.5.2 State Representation

At each decision window T, the agent observes a compact summary of the network's condition. This summary, or *state*, denoted by STATE_T, captures the partially observable network conditions that inform resource allocation decisions. We develop two complementary configurations that balance centralized coordination against distributed efficiency.

Single-Agent Configuration

In the centralized approach, a single agent observes the comprehensive network state encompassing all slices and resources. At each time window T, the state combines three information streams: the service quality measurements across all slices in the network $\mathcal{N}$ described by the Key Performance Indicator vector KPI_vec_T, the compute resource utilization across all VNF instances descibed by $\text{U}^{\text{CLOUD}}\text{\_vec}_T$, and the network resource utilization across all logical links described by $\text{U}^{\text{NET}}\text{\_vec}_T$. Formally, the state representation becomes:

$$\text{STATE}_T = (\text{KPI\_vec}_T, \text{U}^{\text{CLOUD}}\text{\_vec}_T, \text{U}^{\text{NET}}\text{\_vec}_T) \tag{12}$$

The KPI vector $\text{U}^{\text{NET}}\text{\_vec}_T$ aggregates measurements across all active slices:

$$\text{KPI\_vec}_T = \bigcup_{s \in S} \left(\text{KPI}_{s,T}^{\text{TH}}, \text{KPI}_{s,T}^{\text{PLOSS}}, \text{KPI}_{s,T}^{\delta}, \text{KPI}_{s,T}^{\text{JIT}}\right) \tag{13}$$

The network utilization vector is formed per slice by averaging the utilizations of

all logical links associated with that slice. Let $\mathcal{L}_s$ denote the set of logical links for slice s; then

$$\bar{\mathrm{U}}_{s,T}^{\text{Link\_Res}} = \frac{1}{|\mathcal{L}_s|} \sum_{\ell \in \mathcal{L}_s} \mathrm{U}_{\ell,T}^{\text{Link\_Res}},$$
$$\mathrm{U}^{\text{NET}}\_\mathrm{vec}_T = \bigcup_{s \in S} \bar{\mathrm{U}}_{s,T}^{\text{Link\_Res}}. \tag{14}$$

The compute utilization vector is constructed per slice by averaging, across all VNF instances assigned to that slice, the fractional usage of each compute resource type $\text{Comp\_Res} \in \{\text{cpu}, \text{ram}, \text{disk}\}$. Let vnf_set_s be the set of VNF instances for slice s; then

$$\bar{\mathrm{U}}_{s,T}^{\text{Comp\_Res}} = \frac{1}{|\text{VNF\_set}_s|} \sum_{\text{VNF} \in \text{VNF\_set}_s} \mathrm{U}_{\text{VNF},T}^{\text{Comp\_Res}},$$
$$\mathrm{U}^{\text{CLOUD}}\_\mathrm{vec}_T = \bigcup_{s \in S} \left(\bar{\mathrm{U}}_{s,T}^{\text{CPU}},\ \bar{\mathrm{U}}_{s,T}^{\text{RAM}},\ \bar{\mathrm{U}}_{s,T}^{\text{DISK}} \right). \tag{15}$$

This comprehensive view enables the agent to identify resource bottlenecks and coordinate allocations across the entire network.

Multi-Agent Configuration

The distributed approach decomposes control by assigning specialized agents to individual resources within each slice. Each compute-resource agent in slice s manages a specific resource type $\text{Comp\_Res} \in \{\text{cpu}, \text{ram}, \text{disk}\}$ and observes the slice's service quality together with the mean consumption of that compute resource across its VNF footprint:

$$\text{STATE}_{s,T}^{\text{Comp\_Res}} = \left(\text{KPI\_vec}_{s,T},\ \bar{\mathrm{U}}_{s,T}^{\text{Comp\_Res}} \right) \tag{16}$$

The network-resource agent managing the logical links of slice s observes the

averaged link utilization together with a *throughput-omitted* KPI vector, since the normalized throughput equals the per-slice averaged link utilization and is therefore redundant:

$$\text{KPI\_vec}'_{s,T} = \left(\text{KPI}^{\text{PLOSS}}_{s,T},\ \text{KPI}^{\delta}_{s,T},\ \text{KPI}^{\text{JIT}}_{s,T}\right), \tag{17}$$

$$\text{STATE}^{\text{LINK\_RES}}_{s,T} = \left(\text{KPI\_vec}'_{s,T},\ \bar{\text{U}}^{\text{LINK\_RES}}_{s,T}\right), \tag{18}$$

In both configurations, the transition to the next state STATE_{T+1} preserves the same structure, reflecting updated KPI and utilization measurements after the applied actions take effect.

4.5.3 Action Space Design

The action space defines the control mechanisms available for resource adjustment. We design three categories of actions that provide fine-grained control while maintaining operational simplicity:

1. **Scale Up Actions**: When the agent identifies resource constraints, it can increase capacity using proportional steps. For a compute resource $\text{COMP\_RES} \in \{\text{CPU}, \text{RAM}, \text{DISK}\}$ assigned to VNF instance VNF,

$$C^{\text{COMP\_RES}}_{\text{VNF},T+1} = C^{\text{COMP\_RES}}_{\text{VNF},T}\left(1 + p\%^{+}\right). \tag{19}$$

For the logical link capacity available to slice s,

$$C^{\text{LINK\_RES}}_{s,T+1} = C^{\text{LINK\_RES}}_{s,T}\left(1 + p\%^{+}\right). \tag{20}$$

Here $p\%^{+}$ represents the proportional expansion factor. When physical limits

are reached, the system automatically triggers scale-out operations, instantiating new VNF instances.

2. **Scale Down Actions**: For underutilized resources, the agent can scale down capacities to improve efficiency. For compute,

$$C_{\text{VNF},T+1}^{\text{Comp\_Res}} = C_{\text{VNF},T}^{\text{Comp\_Res}} \left(1 - p\%^{-}\right), \tag{21}$$

and for the slice's logical link,

$$C_{s,T+1}^{\text{Link\_Res}} = C_{s,T}^{\text{Link\_Res}} \left(1 - p\%^{-}\right). \tag{22}$$

Here $p\%^{-}$ ensures gradual reduction to avoid service disruption. Extended underutilization triggers scale-in operations, removing redundant VNF instances.

3. **Allocation Hold**: The agent can choose to maintain current allocations when the system operates within acceptable bounds, avoiding unnecessary reconfiguration overhead.

The action-space dimensionality differs between configurations. The single-agent configuration, the controller manages $|\text{VNF\_set} \times 3| + |S|$ actions covering all resources across the network, while each agent in the multi-agent configuration, each specialized agent (compute type per slice or link per slice) controls only three actions {scale up, allocation hold, scale down} for its assigned resource, reducing learning complexity while aligning decisions with resource roles.

4.5.4 Reward Engineering

Having established how agents observe network conditions through state representations and execute control through the action space, we now address the critical

challenge of guiding learning toward effective resource management policies. The reward signal serves as the primary feedback mechanism that shapes agent behavior, encoding both our SA objectives and resource efficiency goals into a unified scalar signal that drives policy improvement.

Our reward engineering approach addresses a fundamental tension in network resource management: maintaining strict KPI compliance while minimizing resource consumption. Traditional approaches often require manual tuning of coefficients to balance these competing objectives, making them brittle when network conditions or service requirements change. We develop a coefficient-free reward structure that automatically adapts to heterogeneous slice requirements through three interconnected mechanisms.

Utilization Transformation

The first challenge in reward design stems from the asymmetric nature of resource utilization objectives. While KPI values have clear targets (their SLA thresholds), optimal resource utilization depends on operator preferences and varies across deployment scenarios. We address this through transformation functions that map raw utilization measurements onto a normalized scale aligned with KPI severity.

Consider that both KPI values and resource utilizations arrive normalized to [0,1] as described in Section 4.5.1. For KPIs, this normalization directly encodes performance quality: 0 represents excellent service while 1 indicates SLA violation. Resource utilizations, however, lack this inherent quality mapping since optimal operating points vary based on operator objectives.

We developed three transformation functions that encode different operational philosophies, each mapping utilization to a severity scale [0,1] where 0 represents optimal operation and 1 indicates undesirable conditions.

Intersection Transformation (Conservative Operation): This V-shaped transformation penalizes deviations from a target utilization TARGET in both directions, suitable for operators seeking balanced resource usage:

$$Trans(\mathrm{U}_T^{\mathrm{RES}}) = \begin{cases} \mathrm{SLOPE}_1 \cdot \mathrm{U}_T^{\mathrm{RES}} + c_1 & \text{if } \mathrm{U}_T^{\mathrm{RES}} < \mathrm{TARGET} \\ \mathrm{SLOPE}_2 \cdot \mathrm{U}_T^{\mathrm{RES}} + c_2 & \text{otherwise} \end{cases} \tag{23}$$

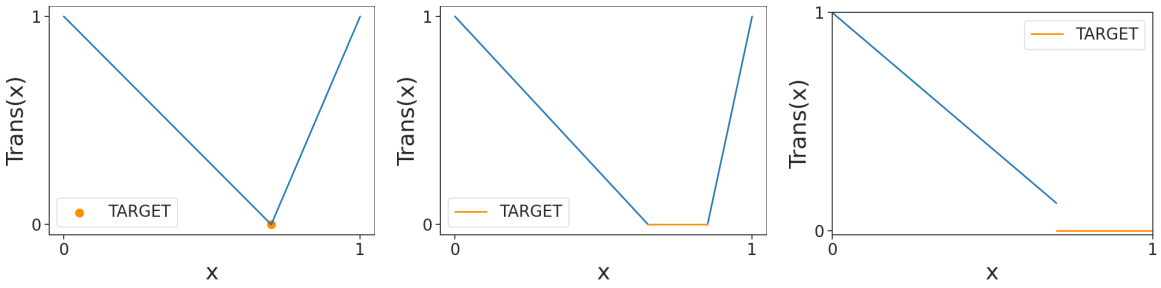
where slopes are chosen such that the transformed value equals 0 at TARGET (optimal) and approaches 1 as utilization moves toward either extreme. This creates symmetric penalties for both under-utilization (waste) and over-utilization (risk).

Ranged Transformation (Flexible Operation): This transformation defines an acceptable operating range $[\mathrm{TARGET}_{min}, \mathrm{TARGET}_{max}]$ rather than a single point, accommodating natural traffic variations:

$$Trans(\mathrm{U}_T^{\mathrm{RES}}) = \begin{cases} \mathrm{SLOPE}_1 \cdot \mathrm{U}_T^{\mathrm{RES}} + c_1 & \text{if } \mathrm{U}_T^{\mathrm{RES}} < \mathrm{TARGET}_{min} \\ \mathrm{SLOPE}_2 \cdot \mathrm{U}_T^{\mathrm{RES}} + c_2 & \text{if } \mathrm{U}_T^{\mathrm{RES}} > \mathrm{TARGET}_{max} \\ 0 & \text{if } \mathrm{TARGET}_{min} \leq \mathrm{U}_T^{\mathrm{RES}} \leq \mathrm{TARGET}_{max} \end{cases} \tag{24}$$

This provides a stability region where minor fluctuations do not generate penalty signals, reducing unnecessary resource adjustments while maintaining efficiency bounds.

Flat Transformation (Aggressive Utilization): This transformation encourages maximum resource usage by treating all utilization at or above TARGET as equally optimal:



(a) Intersection target (b) Ranged target (c) Flat target

Figure 1: Utilization transformation functions used to set TARGET: (a) intersection, (b) ranged, and (c) flat targets.

$$Trans(\mathrm{U}_T^{\mathrm{RES}}) = \begin{cases} \mathrm{SLOPE} \cdot \mathrm{U}_T^{\mathrm{RES}} + c_1 & \text{if } \mathrm{U}_T^{\mathrm{RES}} < \mathrm{TARGET}_{min} + c_2 \\ 0 & \text{if } \mathrm{U}_T^{\mathrm{RES}} \geq \mathrm{TARGET}_{min} \end{cases} \tag{25}$$

where the linear penalty applies only to under-utilization, with no penalty for high resource usage. This suits scenarios prioritizing maximum efficiency over safety margins, though it risks operating too close to capacity limits.

Figure 1 illustrates all three transformation approaches. The choice of transformation encodes operational preferences: intersection for balanced conservative operation, ranged for flexibility with bounds, and flat for aggressive efficiency maximization. The transformation is applied element-wise to create the transformed utilization vector $Trans(\mathrm{U\_vec}_T)$ that aligns resource efficiency metrics with service quality indicators.

Worst-Case Selection

With both KPI values and resource utilizations mapped to a common severity scale [0,1], we implement a worst-case selection mechanism that focuses learning attention on the most critical performance bottleneck at each time step. This approach ensures that agents prioritize addressing the most severe issues rather than optimizing average

performance at the expense of critical violations.

For the single-agent configuration managing all network resources, the selection identifies the maximum (worst) values across both domains:

$$\text{SELECT}_T^{\text{TYPE\_UTIL}} = \max\left[Trans(\text{U\_vec}_T)\right] \tag{26}$$

$$\text{SELECT}_T^{\text{TYPE\_KPI}} = \max\left[\text{KPI\_vec}_T\right] \tag{27}$$

In the multi-agent configuration, each specialized agent performs this selection within its scope of control:

$$\text{SELECT}_T^{\text{TYPE\_UTIL}} = \max\left[Trans(\text{U}_T^{\text{RES}})\right] \tag{28}$$

$$\text{SELECT}_T^{\text{TYPE\_KPI}} = \max\left[\text{KPI\_vec}_{s,T}\right] \tag{29}$$

This worst-case focus creates a natural prioritization mechanism: agents learn to eliminate severe violations before optimizing moderate deviations, aligning with operational priorities in production networks where preventing SLA breaches takes precedence over marginal efficiency improvements.

Composite Reward Signal

The final reward combines the selected worst-case metrics through carefully designed component functions that encode the relative importance of KPI compliance versus resource efficiency. Rather than using linear combinations with manually tuned coefficients, we employ distinct function types that naturally express different severity characteristics: a linear function for resource efficiency and an exponential function for KPI violations.

For resource utilization deviations, we apply a linear penalty that increases proportionally with distance from the target:

$$\text{REW}_T^{\text{TYPE\_UTIL}} = -6 \cdot \text{SELECT}_T^{\text{TYPE\_UTIL}} + 3 \tag{30}$$

This yields rewards ranging from +3 (optimal utilization according to the chosen transformation) to -3 (extreme deviation), providing steady gradient signals for efficiency improvement.

For KPI violations, we employ an exponential penalty that rapidly escalates as performance approaches SLA boundaries:

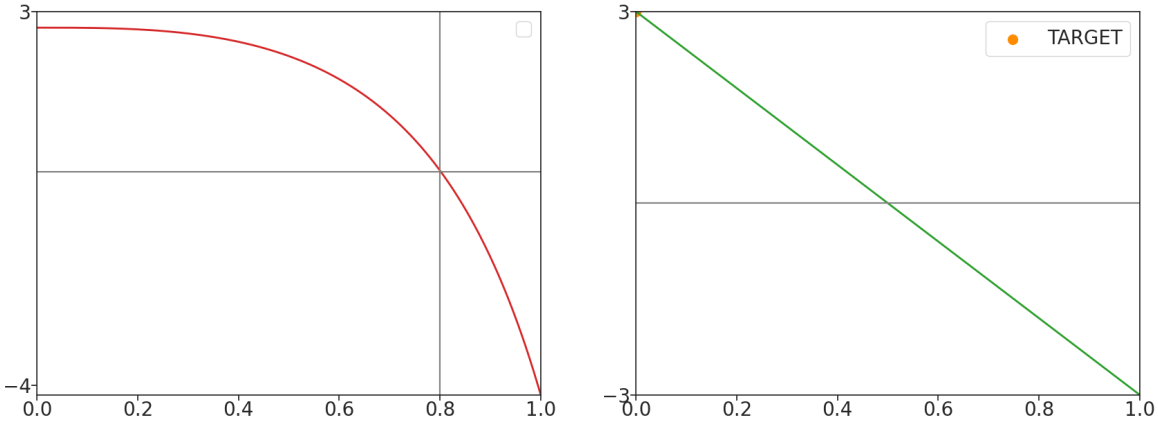
$$\text{REW}_T^{\text{TYPE\_KPI}} = -e^{3 \cdot \text{SELECT}_T^{\text{TYPE\_KPI}}} + 6.7 \tag{31}$$

The exponential structure ensures that near-violations generate learning signals substantially stronger than those from minor deviations, naturally encoding the criticality of maintaining SLA compliance. This contrasts with the linear utilization penalty, creating an asymmetric response where KPI degradation triggers increasingly urgent corrective action while resource efficiency improvements follow a steady optimization path. Figure 2 visualizes these reward components.

The composite reward delivered to the agent combines both components when valid actions are taken, while invalid actions (those violating physical constraints) receive a fixed penalty:

$$\text{REW}_T = \begin{cases} \text{REW}_T^{\text{TYPE\_UTIL}} + \text{REW}_T^{\text{TYPE\_KPI}} & \text{if action is valid} \\ -c_{\text{ACT}} & \text{if action violates constraints} \end{cases} \tag{32}$$

This reward structure eliminates the need for manual coefficient tuning between



(a) TYPE_KPI (b) TYPE_UTIL

Figure 2: Reward functions

SA and efficiency objectives. The exponential KPI component naturally dominates when service quality degrades, forcing corrective action, while the linear utilization component guides optimization when KPIs are satisfied. The self-balancing property, combined with the flexibility to select different transformation strategies, enables the same reward formulation to work across heterogeneous slices with vastly different performance requirements and operational constraints.

4.5.5 Policies and Agents

Having established the state representation, action space, and reward structure, we now describe how these components integrate within our reinforcement learning framework to enable automated service assurance. The agent learns a policy π_θ that maps observed states to resource allocation actions, where the parameters θ are optimized through interaction with the network environment.

At each time window T, the agent observes the current state STATE_T comprising normalized KPI measurements and resource utilizations as defined in Section 4.5.2. The policy network π_θ outputs a probability distribution over the action space described in Section 4.5.3, from which an action ACT_T is sampled. This action modifies

compute or network resource capacities according to the selected scale up, scale down, or hold operation. The environment transitions to state STATE_{T+1} reflecting the impact of the resource adjustment, and the reward function from Section 4.5.4 evaluates this transition by applying utilization transformation, worst-case selection, and composite reward computation.

We employ on-policy optimization methods, specifically PPO, which iteratively collect trajectories of state-action-reward transitions and update the policy to maximize expected cumulative reward. The learning process alternates between trajectory collection, where the current policy interacts with the network environment across multiple time windows, and policy improvement, where the accumulated experience guides parameter updates. A value function V_ϕ estimates expected future rewards from each state, enabling advantage estimation that reduces learning variance while maintaining unbiased policy gradients.

Single-Agent Configuration: In the centralized approach detailed in Algorithm 1, a single agent observes the comprehensive network state encompassing all slices and manages all resource allocation decisions. The agent's state includes KPI measurements and resource utilizations aggregated across all network slices, enabling coordinated resource management that accounts for inter-slice dependencies and shared infrastructure constraints. After each episode of network interaction, the policy π_θ and value function V_ϕ are updated using the collected trajectory data.

Multi-Agent Configuration: In the distributed approach detailed in Algorithm 2, we deploy independent agents $a \in A$, where each agent controls a specific resource type (compute: CPU, RAM, DISK, or network: link capacity) for a particular slice. Each agent a maintains its own policy π_{θ_a} and value function V_{ϕ_a}, observing only the state information relevant to its assigned resource and slice as specified in Section 4.5.2. During each time window, all agents observe their respective states,

Algorithm 1: Single-Agent Reinforcement Learning for Service Assurance

Input: Number of episodes E, time windows per episode T_{max}

Output: Trained policy π_θ and value function V_ϕ

Initialize policy network π_θ and value network V_ϕ

for *episode* $e = 1$ *to* E **do**

  Initialize trajectory buffer $\mathcal{B} \leftarrow \emptyset$

  Reset environment and observe initial state STATE_0

  for *time window* $T = 0$ *to* $T_{max} - 1$ **do**

    `// State Observation`

    Collect KPI measurements KPI_vec_T and resource utilizations $\text{U}^{\text{CLOUD}}\text{\_vec}_T$, $\text{U}^{\text{NET}}\text{\_vec}_T$

    Form state $\text{STATE}_T = (\text{KPI\_vec}_T, \text{U}^{\text{CLOUD}}\text{\_vec}_T, \text{U}^{\text{NET}}\text{\_vec}_T)$

    `// Action Selection`

    Select action $\text{ACT}_T \sim \pi_\theta(\cdot|\text{STATE}_T)$

    Execute action ACT_T in network environment

    `// Reward Computation`

    Observe next state STATE_{T+1}

    if *action* ACT_T *is invalid* **then**

      $\text{REW}_T \leftarrow -c_{\text{ACT}}$

    else

      Apply utilization transformation $Trans(\cdot)$ to map usage to severity scale

      Compute worst-case utilization: $\text{SELECT}_{T+1}^{\text{TYPE\_UTIL}} \leftarrow \max[Trans(\text{U}^{\text{CLOUD}}\text{\_vec}_{T+1}), Trans(\text{U}^{\text{NET}}\text{\_vec}_{T+1})]$

      Compute worst-case KPI: $\text{SELECT}_{T+1}^{\text{TYPE\_KPI}} \leftarrow \max[\text{KPI\_vec}_{T+1}]$

      Compute composite reward using worst-case selection

      $\text{REW}_T \leftarrow \text{REW}_{T+1}^{\text{TYPE\_UTIL}} + \text{REW}_{T+1}^{\text{TYPE\_KPI}}$

    Store transition $(\text{STATE}_T, \text{ACT}_T, \text{REW}_T, \text{STATE}_{T+1})$ in $\mathcal{B}$

  `// Policy Update`

  Compute returns and advantages from trajectories in $\mathcal{B}$

  Update policy π_θ and value function V_ϕ using on-policy optimization

return π_θ, V_ϕ

Algorithm 2: Multi-Agent Reinforcement Learning with Independent Policies

Input: Number of episodes E, time windows per episode T_{max}, agent set A
Output: Trained agent-specific policies $\{\pi_{\theta_a}\}_{a\in A}$ and value functions $\{V_{\phi_a}\}_{a\in A}$

for *each agent* $a \in A$ **do**
  Initialize agent-specific policy network π_{θ_a}
  Initialize agent-specific value network V_{ϕ_a}
for *episode* $e = 1$ *to* E **do**
  Initialize trajectory buffers $\{\mathcal{B}_a \leftarrow \emptyset\}_{a\in A}$
  Reset environment
  for *time window* $T = 0$ *to* $T_{max} - 1$ **do**
    `// Parallel State Observation and Action Selection`
    for *each agent* $a \in A$ ***in parallel*** **do**
      Collect agent-specific KPI measurements and resource utilizations
      Form agent state $\text{STATE}_{a,T}$
      Select action $\text{ACT}_{a,T} \sim \pi_{\theta_a}(\cdot|\text{STATE}_{a,T})$
    `// Joint Execution`
    Execute all actions $\{\text{ACT}_{a,T}\}_{a\in A}$ simultaneously in network
    `// Parallel Reward Computation`
    for *each agent* $a \in A$ ***in parallel*** **do**
      Observe next state $\text{STATE}_{a,T+1}$
      if *action* $ACT_{a,T}$ *is invalid* **then**
        $\text{REW}_{a,T} \leftarrow -c_{\text{ACT}}$
      else
        Apply utilization transformation $Trans(\cdot)$ to map usage to severity scale
        Compute worst-case utilization: $\text{SELECT}_{a,T+1}^{\text{TYPE\_UTIL}}$ for agent a
        Compute worst-case KPI: $\text{SELECT}_{a,T+1}^{\text{TYPE\_KPI}}$ for agent a
        Compute composite reward using worst-case selection
        $\text{REW}_{a,T} \leftarrow \text{REW}_{a,T+1}^{\text{TYPE\_UTIL}} + \text{REW}_{a,T+1}^{\text{TYPE\_KPI}}$
      Store transition $(\text{STATE}_{a,T}, \text{ACT}_{a,T}, \text{REW}_{a,T}, \text{STATE}_{a,T+1})$ in $\mathcal{B}_a$
  `// Policy Updates`
  for *each agent* $a \in A$ **do**
    Compute returns and advantages for agent a from trajectories in $\mathcal{B}_a$
    Update policy π_{θ_a} using agent's experience in $\mathcal{B}_a$
    Update value function V_{ϕ_a} using agent's experience in $\mathcal{B}_a$
return $\{\pi_{\theta_a}\}_{a\in A}$, $\{V_{\phi_a}\}_{a\in A}$

select actions according to their independent policies, and execute these actions in parallel. The simultaneous execution of agent actions produces a joint effect on the network environment, from which each agent independently computes its reward and updates its policy based solely on its own experience. This truly parallel architecture aligns with the distributed nature of network slice management, where resource control is naturally decomposed across different network functions and administrative domains, while reducing computational complexity compared to centralized control.

The multi-agent formulation enables scalability to large networks with numerous slices, as each agent focuses on a well-defined subproblem rather than the full resource allocation challenge. The reward structure's self-balancing property, combining exponential KPI penalties with linear utilization incentives, ensures that agents learn to prioritize service quality while optimizing efficiency without requiring manual coefficient tuning when extending from single-slice to multi-slice scenarios.

4.6 Validation and Numerical Results

4.6.1 Simulation Environment

For our experiments, we used the enhanced version of the OMNET++ packet level simulator with virtualization and network slicing in the 5G user plane network [18]. The 5G simulation includes four distinct applications running on its own network slice, HD video (eMBB), IoT (mMTC), video gaming (uRLLC), and VoIP (uRLLC) each of which have different requirements and resource demands.

The 5G core enables the possibility to instantiate the UPF as a VNF that can control its resources (CPU, RAM, Disk) in a dynamic manner, and it includes VNF scale up/down and VNF scale in/out. This allows a given slice to accommodate either one VNF or multiple VNFs supported by a load balancer. The load balancer

implements a weighted round-robin algorithm that distributes the incoming packets to the available VNFs proportionally, based on their resource capacity.

Please note that the ability to create/remove new VNFs (scale in/out) combined with the use of the load balancer are used to reduce the action space by automating the VNF instantiating process; for example, if we decide to scale up a VNF, but there are not enough physical resources available, then the simulator can instantiate a new VNF (scale out).The available VNFs receive traffic from the load balancer proportionally to their assigned resource capacity. The simulation was configured to generate traffic with distinctive patterns that are representative of the 5G network traffic. A summary of the parameters used in our simulation is presented in Table 1. Another important feature of the simulator is the ability to update the resource

Table 1: KPI Thresholds

	KPI	HD video	IoT	Gaming	VoIP
τ_{PLOSS}	Packet loss	10^{-6}	10^{-4}	10^{-3}	10^{-2}
τ_{δ}	Delay	300ms	10ms	30ms	100ms
τ_{JIT}	Jitter	100ms	N.A.	5ms	10ms

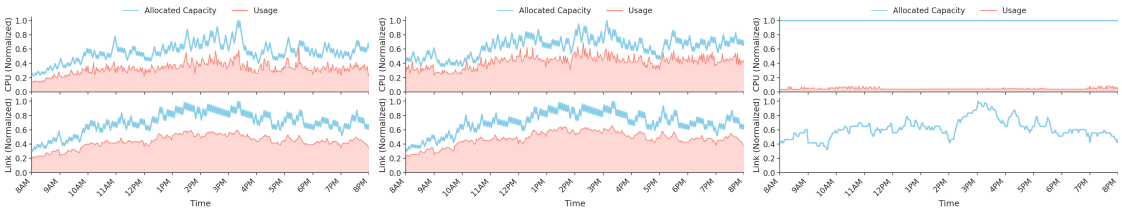
settings within the simulator in almost real time. For example, when we add or remove VNF instances triggered by the recommended action of the RL agent. We next show the results for the URLLC slice.

Our simulation has randomized delays associated with each module which is to reflect real network scenarios. However, as we will discuss later, this did not cause any obstacles in our experiments. Indeed, with the help of the discounted future rewards, the reinforcement learning agent(s) developed near optimal strategies to handle the network and compute resources. We attribute the absence of delay-related obstacles primarily to (i) stochastic policies (via PPO + entropy regularization), which optimize expected return and smooth over small timing variations, and (ii) discounting, which

down-weights far-future effects.

4.6.2 Utilization Transform Comparison: uRRLC

We primarily consider the impact of different utilization transformation functions on the learning and performance of our multi-agent reinforcement algorithm. In this experiment, each agent in the multi-agent configuration controlled CPU, RAM, disk or link respectively. Fig 3a showcases the usage and dynamic capacity for CPU and link when the intersection transformation function (See Fig 1a) is in use. Due to its relative performance, this function can be considered a baseline. Next, Fig 3b highlights the usage and capacity for CPU and link when the ranged transformation function (See Fig 1b) is used. In our analysis, we observed that although the use of ranged transformation improved overall gain, it frequently drove the resource usage close to the system's capacity threshold. And while, in our experiment scenario there was no violation, the likelihood of a violation in case of ranged tranformation function was higher when compared to intersection transformation for the reasons mentioned before. Lastly, when flat transformation function (See Fig 1c) was used some agents (specifically CPU 3c) allocated maximum available capacity while others (including RAM) allocated less causing packet-loss violations. This resulted in comparatively inferior and unsatisfactory performance when compared to our baseline.

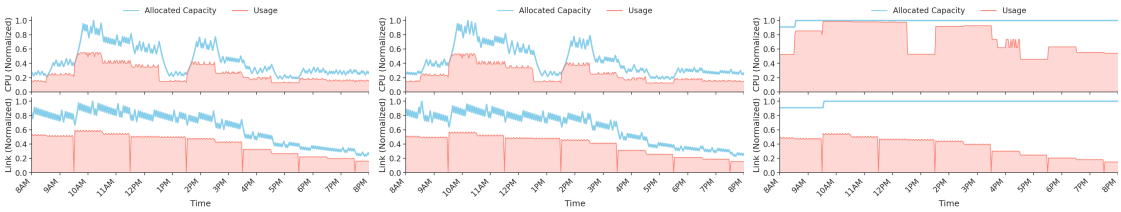


(a) Intersection (b) Ranged (c) Flat

Figure 3: Impact of different utilization transformation functions on CPU & Link Utilization in uRLLC slice.

4.6.3 Utilization Transform Comparison: mMTC

We observed very similar results when the three utilization transformation functions are used on the mMTC slice. Using information transformation function again acts as a baseline for our experiments. The use of ranged transformation function improves the gain of the respective resources but again often results in more instances, on average, of the resource usage closing in on its corresponding capacity when compared to the baseline. An experiment with flat transformation again became an outlier, where it did not learn to allocate appropriate resource capacity in accordance with the dynamic traffic.



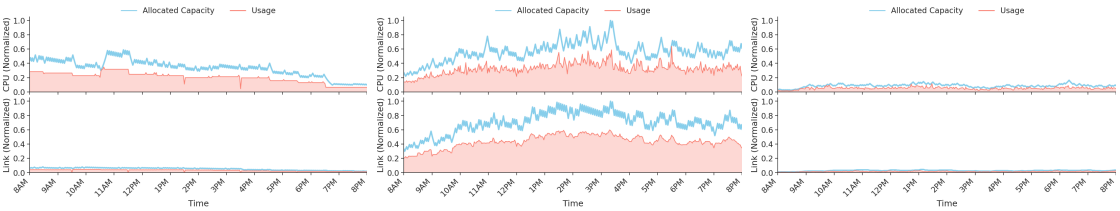
(a) Intersection (b) Ranged (c) Flat

Figure 4: Impact of different utilization transformation functions on CPU & Link Utilization in mMTC slice.

4.6.4 Multi-Slice Experiments

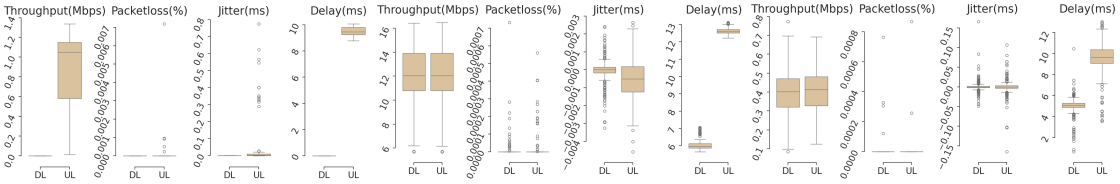
We demonstrate the multi-agent RL algorithm satisfies unique KPI constraints of different slices without the need of any new coefficient learning when extended from one slice to multiple slices. We showcase this by using ranged transformation function with an experiment of 3 unique slices (uRLLC, mMTC and VoIP) in operation. The use of ranged transformation function resulted in the highest total gain over time in the individual experiments for both uRLLC and mMTC slices and hence we selected this function over the others for demonstrating the multi-slice experiment. We note that for both uRLLC and mMTC, the performance is similar to their respective individual

experiments. As for VoIP slice, the use of ranged transformation function helped in following the network demand over time and resulted in a significant amount of gain.



(a) mMTC Slice (b) uRLLC Slice (c) VoIP Slice

Figure 5: Multi-Slice Experiment: CPU & Link Utilization.



(a) mMTC Slice (b) uRLLC Slice (c) VoIP Slice

Figure 6: Multi-Slice Experiment: KPI performance uplink and downlink

4.7 Conclusion

We presented an end-to-end closed-loop service assurance approach for 5G/B5G networks that uses reinforcement learning to elastically manage both compute (CPU, RAM, DISK) and network (LINK) resources per slice while tracking key performance indicators (KPIs). The problem was cast as a POMDP and addressed with on-policy policy optimization (PPO), aided by (i) KPI normalization to 5QI-inspired thresholds, (ii) a utilization–target transformation that aligns resource usage with operator-set targets, and (iii) a reward that jointly reflects the worst normalized KPI and the worst transformed utilization. We developed both single-agent and truly parallel multi-agent variants, where agents control per-slice compute and link resources, and implemented the closed loop service assurance algorithm in a packet-level OMNET++/Simu5G environment driven by refactored, city-scale traffic.

Numerical results across URLLC, MMTC, and VOIPslices show that the proposed algorithm tracks traffic dynamics and maintains KPI adherence while minimizing over-provisioning. The multi-agent design generalized well from single-slice to multi-slice operation without re-tuning coefficients. Among the utilization transformations, the *intersection* target yielded conservative behavior with strong SLA protection, whereas the *ranged* target delivered the highest aggregate gain but operated closer to capacity limits. The flat target, on the other hand, underperformed. Together, these results validate the practicality of reward engineering plus target-based shaping for balancing quality of service and resource/energy savings in elastic slicing.

Chapter 5

Conclusion & Future Work

5.1 Conclusions

This thesis has presented a comprehensive examination of intelligent service assurance mechanisms for 5G networks, establishing the theoretical foundations and practical frameworks necessary for implementing adaptive, AI-driven network management systems. The background analysis demonstrates that the evolution of telecommunications toward software-defined, service-oriented architectures creates both opportunities and challenges for maintaining service quality while optimizing resource efficiency and energy consumption.

The integration of enabling technologies including SDN, NFV, MEC, and NS provides the architectural foundation for intelligent network management, while reinforcement learning emerges as a particularly suitable approach for handling the complex, dynamic optimization problems inherent in 5G service assurance. The mathematical framework presented for model-free reinforcement learning establishes the theoretical basis for developing adaptive control policies that can learn optimal resource allocation strategies through interaction with network environments.

5.2 Future work

Future research directions building upon this foundation span several critical areas. The development of more sophisticated multi-agent reinforcement learning architectures could enable better coordination between different network management functions while reducing computational overhead and improving scalability. Research into federated learning approaches could address privacy concerns and communication overhead in distributed learning scenarios while enabling collaborative learning across multiple network domains.

The integration of energy efficiency objectives into service assurance mechanisms requires further investigation, particularly in developing reward functions that effectively balance performance requirements with sustainability goals. Advanced prediction techniques that combine reinforcement learning with time-series forecasting could enable more proactive resource management by anticipating future traffic patterns and service demands with greater accuracy.

The application of graph neural networks to network topology-aware resource allocation represents another promising direction, enabling learning algorithms to exploit structural properties of network architectures for improved decision-making. Finally, the development of formal verification and safety frameworks for AI-driven network management systems will be essential for ensuring reliable operation in mission-critical environments and building confidence in autonomous network management capabilities.

Bibliography

[1] 3GPP. NR and NG-RAN Overall Description. Technical Specification (TS) 38.300, 3rd Generation Partnership Project (3GPP), Sophia Antipolis, France, 2018. Release 15, Stage 2.

[2] 3GPP. Service requirements for the 5g system; stage 1. Technical Report TS 22.261, 3rd Generation Partnership Project (3GPP), December 2018.

[3] 3GPP. Study of enablers for network automation for 5G. Technical Report (TR) 23.791, ThreeGPP, June 2019. TR 23.791, Version 16.2.0; Release 16.

[4] 3GPP. Architecture enhancements for 5G System (5GS) to support network data analytics services (NWDAF). Technical Specification (TS) 23.288, 3rd Generation Partnership Project (3GPP), Sophia Antipolis, France, 2022. Release 17, Stage 2.

[5] 3GPP. Architecture enhancements for the 5G system (5GS) to support Network Data Analytics Services (NWDAF). Technical Specification (TS) 23.288, ThreeGPP, May 2022. TS 23.288, Version 17.4.0.

[6] 3GPP. System architecture for the 5G System (5GS). Technical Specification (TS) 23.501, 3rd Generation Partnership Project (3GPP), Sophia Antipolis, France, 2022. Version 17.6.0, Stage 2.

[7] 3GPP. System architecture for the 5G System (5GS). Technical Specification (TS) 23.501, 3rd Generation Partnership Project (3GPP), Sophia Antipolis, France, 2023. Version 18.5.0, Release 18.

[8] 3GPP. Management and orchestration; 5g performance measurements. Technical Specification (TS) 28.552, ThreeGPP, 2024. TS 28.552, Version 18.8.0.

[9] 3GPP. Management and Orchestration; Concepts, Use Cases and Requirements. Technical Specification (TS) 28.530, European Telecommunications Standards Institute (ETSI), Sophia Antipolis, France, Dec. 2024. Version 17.6.0.

[10] 3GPP. Architecture for enabling edge applications. Technical Specification (TS) 23.558, ThreeGPP, April 2025. TS 23.558, Version 17.13.0; Release 17.

[11] 3GPP. Management and orchestration; provisioning (network slicing). Technical Specification (TS) 28.531, ThreeGPP, 2025. TS 28.531, Version 18.5.0.

[12] 5G PPP Technology Board. Ai and ml – enablers for beyond 5g networks. `https://5g-ppp.eu/white-papers/`, May 2021. Version 1.0.

[13] A. Aslan, G. Bal, and C. Toker. Dynamic resource management in next generation networks with dense user traffic. In *IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pages 1–6, 2020.

[14] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui. Software-defined networking (sdn): a survey. *Security and communication networks*, 9(18):5803–5833, 2016.

[15] Y. Cai, P. Cheng, Z. Chen, M. Ding, B. Vucetic, and Y. Li. Deep Reinforcement Learning for Online Resource Allocation in Network Slicing . *IEEE Transactions on Mobile Computing*, 23(06):7099–7116, June 2024.

[16] N. F. Cheng, T. Pamuklu, and M. Erol-Kantarci. Reinforcement learning based resource allocation for network slices in o-ran midhaul. In *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, pages 140–145. IEEE, 2023.

[17] S. Dahmen-Lhuissier. 5G. ETSI. Accessed: 2023-01-26.

[18] O. Delgado, B. Jaumard, Z. Ding, F. Bishay, and V. Bissonnette. Demo: A network simulator for 5G virtualized networks. In *IEEE 8th International Conference on Network Softwarization (NetSoft)*, pages 237–239, 2022.

[19] M. El Rajab, L. Yang, and A. Shami. Zero-touch networks: Towards next-generation network automation. *Computer Networks*, 243:110294, 2024.

[20] Ericsson. Ericsson mobility report: November 2024. Technical report, Ericsson AB, Stockholm, Sweden, November 2024.

[21] M. Ghasemi, A. H. Moosavi, and D. Ebrahimi. A comprehensive survey of reinforcement learning: From algorithms to practical challenges. *arXiv preprint arXiv:2411.18892*, 2024.

[22] GSMA Intelligence. Global Mobile Trends 2023. Industry report, GSMA, 2023.

[23] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 14(14):527, 2008.

[24] Huawei Technologies Co., Ltd. Autonomous Driving Network for Campuses. White paper, Huawei, April 2021. Campus network architecture and transport evolution.

[25] InterDigital, Inc. and Transforma Insights and 6GWorld. Sustainability in New and Emerging Technologies. White paper, InterDigital, April 2021. IoT Technology impact on energy, water and CO2 emissions.

[26] International Energy Agency. Energy Efficiency 2021. Market report, IEA, 2021. License: CC BY 4.0.

[27] International Telecommunication Union. Energy Efficiency for 5G Networks. Technical report, ITU-T Study Group 5, 2021. Environment, EMF and circular economy.

[28] B. Jaumard and J. Ziazet. 5G E2E network slicing predictable traffic generator. In *Int'l Conference on Network and Service Management (CNSM)*, pages 1–7, 2023.

[29] K. Murphy. Reinforcement learning: an overview. *arXiv preprint arXiv:2412.05265*, 2024.

[30] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis. Simu5G – an OMNeT++ library for end-to-end performance evaluation of 5G networks. *IEEE Access*, 8:181176 – 181191, 2020.

[31] Z. Ning and L. Xie. A survey on multi-agent reinforcement learning and its application. *Journal of Automation and Intelligence*, 3(2):73–91, 2024.

[32] Nokia Corporation. Nokia Group Strategy Update: 2026 Operating Margin Target and Strategic Outlook. Corporate strategy report, Nokia, December 2023. Comparable operating margin target of at least 13% by 2026.

[33] C. Qi, Y. Hua, R. Li, Z. Zhao, and H. Zhang. Deep reinforcement learning with discrete normalized advantage functions for resource management in network slicing. *IEEE Communications Letters*, 23(8):1337–1341, 2019.

[34] R. Raftopoulos, S. D'Oro, T. Melodia, and G. Schembra. Drl-based latency-aware network slicing in o-ran with time-varying slas. *arXiv preprint arXiv:2401.05042*, 2024.

[35] Y. Shao, R. Li, B. Hu, Y. Wu, Z. Zhao, and H. Zhang. Graph attention network-based multi-agent reinforcement learning for slicing resource management in dense cellular network. *IEEE Transactions on Vehicular Technology*, 70(10):10792–10803, 2021.

[36] M. Sulaiman, A. Moayyedi, M. A. Salahuddin, R. Boutaba, and A. Saleh. Multi-agent deep reinforcement learning for slicing and admission control in 5g c-ran. In *NOMS 2022-2022 IEEE/IFIP network operations and management symposium*, pages 1–9. IEEE, 2022.

[37] T. Taleb, C. Benzaid, R. A. Addad, and K. Samdanis. Ai/ml for beyond 5g systems: Concepts, technology enablers & solutions. *Computer Networks*, 237:110044, 2023.

[38] N. Tran, O. Delgado, B. Jaumard, and F. Bishay. ML KPI prediction in 5G and B5G networks. In *European Conference on Networks and Communications (EuCNC)*, pages 1–6, Gothenburg, Sweden, 2023.

[39] A. Varga and R. Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–10, 2008.

[40] I. Vilà, J. Pérez-Romero, O. Sallent, and A. Umbert. A multi-agent reinforcement learning approach for capacity sharing in multi-tenant scenarios. *IEEE Transactions on vehicular Technology*, 70(9):9450–9465, 2021.

[41] R. Wang, V. Friderikos, and A. H. Aghvami. Energy-aware design policy for network slicing using deep reinforcement learning. *IEEE Transactions on Services Computing*, 2024.

[42] F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider. Nfv and sdn—key technology enablers for 5g networks. *IEEE Journal on Selected Areas in Communications*, 35(11):2468–2478, 2017.

[43] G. Zhou, L. Zhao, G. Zheng, Z. Xie, S. Song, and K.-C. Chen. Joint multi-objective optimization for radio access network slicing using multi-agent deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 72(9):11828–11843, 2023.