UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Lifelong AI-driven zero-touch network slicing

Farhad Rezazadeh

Centre Tecnològic de Telecomunicacions de Catalunya

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Department of Signal Theory and Communications

Technical University of Catalonia

Department of Signal Theory and Communications

Doctoral thesis

Lifelong AI-Driven Zero-Touch Network Slicing

Author:
Farhad Rezazadeh

Supervisor:
Prof. Christos Verikoukis, ISI/ATHENA and University of Patras

Tutor:
Prof. Luis Alonso, Technical University of Catalonia (UPC)

Barcelona, July 2023

Abstract

To streamline the implementation of sixth-generation (6G) network slicing, algorithmic and architectural innovations are needed to transition from being AI-native to becoming intrinsic trustworthy automation-native. In this context, network slicing is viewed as a disruptive technology and the backbone of future communication systems, creating an all-encompassing environment that extends tenancy to the end consumer through advanced digital use cases. Specifically, in the upcoming 6G networks, network slicing will play a crucial role in almost every aspect of life, society, and industry, enabling network operators and service providers to meet the communication needs of humans and intelligent machines at any time and place. Indeed, network slicing is a powerful technology that allows for the creation of multiple virtual networks tailored to meet diverse service requirements, such as low latency, high throughput, and high reliability, across different network deployments. Vertical tenants can rent and manage isolated logical networks, or slices, on top of a physical operator's network, either fully or partially. This is made possible by employing softwarization and virtualization technologies, including software-defined networking (SDN) and network functions virtualization (NFV), to enable the necessary flexibility and programmability required for network slicing. Network slicing holds immense potential to boost revenue while minimizing the capital and operational expenditure (CAPEX and OPEX) of service providers.

While the advent of fifth-generation (5G)/6G technologies with network slicing promise to revolutionize digital society, one of the main challenges of this deployment for multi-tenancy is radio access network (RAN) slicing. It poses a significant complexity in configuring and managing various RAN operations; therefore, legacy solutions cannot handle this dynamic environment and satisfy stringent service level agreements (SLAs). To close this gap, zero-touch network slicing, as a fully-automated management and orchestration (MANO) scheme, supports fully automated operations and on-demand configuration without needing fixed contractual agreements and manual intervention. One of the main components of this technology is the decision engine, to determine the action that needs to be taken based on the issues detected by other components. In this context, algorithmic innovation can be employed to optimize the allocation of network resources to different network slices to address the fundamental challenges associated with RAN slicing, such as i) energy-aware, ii) latency-aware, iii) scalability, and iv) trustworthiness issues. In this regard, lifelong reinforcement learning is paramount to learning continuously and accumulating past knowledge to assist future learning and problem-solving because it is tough to collect a large number of training examples in each complex RAN interactive environment. This thesis will address these challenges by designing, developing, and assessing innovative and intelligent resource allocation schemes.

In fact, efficient allocation of resources, such as computing and network bandwidth, to network slices is necessary to minimize energy consumption or latency, while maintaining acceptable performance and meeting quality of service (QoS) requirements. Adopting a joint optimization approach to energy/latency and resource allocation considering a trade-off to optimize both factors simultaneously. Concretely, we propose a stochastic Actor-Critic approach to support continuous state and action spaces in telecommunication while stabilizing the learning procedure, improving time efficiency, and reducing the need for hyperparameter tuning in an energy-aware network slicing setup. Moreover, we investigate the feasibility of a multi-objective and multi-action approach where model-free agents learn to jointly allocate optimal power and computing resources to minimize the latency of service provisioning under long-term statistical SLA, namely, Q-th delay percentile. In particular, we propose a massive deep reinforcement learning (DRL)-based actor-learner framework dubbed CS-AC. The CS-AC is a software framework for designing and training DRL agents that attempts to address complexity issues. To cope with control challenges in network slicing, such as increased dynamism, heterogeneity, and extended training time of slice instances, we separate the actor from the learner where the proposed approach can be scaled up to several thousand parallel actors-learners across a large collection of tasks without sacrificing data efficiency.

On the other hand, the temporal variations of the traffic demand deeply complicate resource planning and allocation tasks, especially in the RAN domain where resource allocation decisions, e.g., in terms of bandwidth, must cope with the additional variability inherent of the wireless channel and end-user's mobility. To solve this, we propose a distributed architecture for RAN slice resource orchestration based on DRL, composed of multiple AI-enabled decision agents that perform local radio allocation decisions without needing a centralized control entity. We design a federated learning (FL) scheme consisting of multiple parallel layers, one for each slice, to enhance the capabilities of the local decision-making process following the recent development of the Open RAN (O-RAN) architecture and solve scalability issues in network slicing. We further improve the decision process by dynamically defining the subset of decision agents to be involved in the federation process based on long-term slice traffic demand variations and their temporal similarities.

Despite the impressive performance of AI/machine learning (ML) solutions, there is a growing concern about the lack of transparency in deep neural networks (DNNs), which are often viewed as opaque models. This issue is particularly critical when reliability and security are crucial in real-world network scenarios, and it undermines users' trust in the trained agents and predicted results in the network slicing ecosystem. Following the European Commission's (EC) technical report on ethics guidelines for trustworthy AI, proposed AI solutions should prioritize trustworthiness. However, due to the lack of transparency and trust in AI models, telecommunication operators are hesitant to widely deploy AI models in their networks, especially when such decisions have financial and service quality implications. To address this challenge, we introduce the architecture of SliceOps, where explainable ML operations are consolidated in a standalone slice that provides AI services to other slices. This continuous delivery (CD) and continuous integration (CI) of ML models enhances reliability and interpretability while quickly deploying AI models in the network with greater consistency. By transitioning from an AI-native zero-touch to an automation-native zero-touch approach, the framework can manage not only service functions but also the underlying AI functions, allowing the full potential of slicing in RAN to be harnessed and reducing the complexities involved.

Resumen

Para agilizar la implementación del slicing de redes de sexta generación (6G), se necesitan innovaciones algorítmicas y arquitectónicas que habiliten una transición de redes nativas en inteligencia artificial (IA) a redes nativas en automatización confiable intrínseca. En este contexto, el slicing de redes se considera una tecnología disruptiva y la columna vertebral de los futuros sistemas de comunicación, habilitando un entorno integral que extiende la tenencia (en inglés, tenancy) al consumidor final a través de casos de uso digitales avanzados. Específicamente, en las próximas redes 6G, el slicing de redes jugará un papel crucial en casi todos los aspectos de la vida, la sociedad y la industria, permitiendo a los operadores y proveedores de servicios de red satisfacer las necesidades de comunicación de humanos y máquinas inteligentes en cualquier momento y lugar. De hecho, el slicing de redes es una tecnología que permite la creación de múltiples redes virtuales adaptadas para satisfacer diversos requisitos de servicio, como baja latencia, alta capacidad y alta confiabilidad, en diferentes implementaciones de red. Los tenants verticales pueden alquilar y administrar redes lógicas aisladas, o ”slices”, en la red física del operador, ya sea completa o parcialmente. Esto es posible mediante el empleo de tecnologías de virtualización y software, como la red definida por software (SDN) y la virtualización de funciones de red (NFV), que dotan a las redes de flexibilidad y programabilidad. A su vez, el slicing de redes tiene un inmenso potencial para aumentar los ingresos al tiempo que minimiza el gasto de capital y operativo (CAPEX y OPEX) de los proveedores de servicios.

Si bien el advenimiento de las tecnologías de quinta generación (5G)/6G con slicing de redes promete revolucionar la sociedad digital, uno de los principales desafíos de esta implementación en cuanto a multi-tenancy es el slicing de la red de acceso por radio (RAN). Esto plantea una complejidad significativa en la configuración y gestión de diversas operaciones de RAN; las soluciones heredadas no pueden manejar este entorno dinámico y cumplir con los estrictos acuerdos de nivel de servicio (SLAs). Para cerrar esta brecha, el slicing de redes sin intervención, como un esquema de gestión y orquestación completamente automatizado (MANO), respalda operaciones completamente automatizadas y una configuración bajo demanda sin necesidad de acuerdos contractuales fijos ni intervención manual. Uno de los principales componentes de esta tecnología es el motor de decisión, que determina la acción que debe tomarse en función de los problemas detectados por otros componentes. En este contexto, la innovación algorítmica se puede emplear para optimizar la asignación de recursos de red a diferentes slices con tal de abordar los desafíos fundamentales asociados con el slicing de RAN, como i) energía, ii) latencia, iii) escalabilidad y iv) confiabilidad. En este sentido, el aprendizaje de refuerzo (en inglés, reinforcement learning) es fundamental a lo largo de la vida útil de la red para aprender de manera continua y acumular conocimientos pasados para ayudar al aprendizaje y la resolución de problemas futuros. Esta tesis abordará estos desafíos mediante el diseño, desarrollo y evaluación de esquemas de asignación de recursos innovadores e inteligentes.

En este sentido, la asignación eficiente de recursos de red a slices, como la computación y el ancho de banda de red, es necesaria para minimizar el consumo de energía o la latencia, al tiempo que se mantiene un rendimiento aceptable y se cumplen los requisitos de calidad de servicio (QoS). Adoptar un enfoque de optimización conjunta de consumo energético/latencia y asignación de recursos considerando el equilibrio entre ellos para optimizar ambos factores simultáneamente. Concretamente, proponemos un enfoque estocástico Actor-Critic para soportar espacios continuos de estados y acciones en escenarios de telecomunicaciones con tal de estabilizar el procedimiento de aprendizaje, mejorando la eficiencia temporal y reduciendo la necesidad de ajustar hiperparámetros en una configuración de slicing de red enfocada al ahorro energético. Además, investigamos la viabilidad de un enfoque multiobjetivo y de múltiples acciones donde los agentes sin modelo aprenden a asignar conjuntamente recursos óptimos de potencia y computación para minimizar la latencia de aprovisionamiento de servicios bajo un SLA estadístico a largo plazo, es decir, relativo al percentil de demora Q. En particular, proponemos un marco de trabajo basado en el aprendizaje profundo masivo de refuerzo (DRL) denominado CS-AC. El CS-AC es un marco de trabajo de software para diseñar y entrenar agentes de DRL que intenta abordar problemas complejos. Para enfrentar los desafíos de control en el slicing de redes, como el aumento de la dinamicidad, heterogeneidad y tiempo de entrenamiento extendido de instancias de slices, separamos el actor del aprendiz, donde el enfoque propuesto puede ampliarse a varios miles de actores-aprendices paralelos en una gran colección de tareas sin sacrificar la eficiencia de los datos.

Por otro lado, las variaciones temporales de la demanda de tráfico complican profundamente las tareas de planificación y asignación de recursos, especialmente en el dominio de RAN donde las decisiones de asignación de recursos, por ejemplo, en términos de ancho de banda, deben lidiar con la variabilidad adicional inherente al canal inalámbrico y la movilidad del usuario final. Para resolver esto, proponemos una arquitectura distribuida para la orquestación de recursos de slices de RAN basada en DRL, compuesta por múltiples agentes de decisión habilitados para IA que realizan decisiones de asignación de radio locales sin necesidad de una entidad de control centralizada. A este respecto, diseñamos un esquema de aprendizaje federado (FL) que consta de múltiples capas paralelas, una para cada slice, para mejorar las capacidades del proceso de toma de decisiones local, siguiendo el desarrollo reciente de la arquitectura Open RAN (O-RAN), y para resolver problemas de escalabilidad en el slicing de redes. Mejoramos aún más el proceso de toma de decisiones definiendo dinámicamente el subconjunto de agentes de decisión que participarán en el proceso de federación en función de las variaciones de la demanda de tráfico de slices a largo plazo y sus similitudes temporales.

Por último, a pesar del rendimiento demostrado por las soluciones de IA/aprendizaje automático (ML), existe una creciente preocupación por la falta de transparencia en las redes neuronales profundas (DNN), que a menudo se consideran modelos opacos. Este problema es particularmente crítico cuando la confiabilidad y la seguridad son cruciales en escenarios de red del mundo real, y socava la confianza de los usuarios en los agentes entrenados y los resultados predichos en el ecosistema de slicing de redes. Siguiendo el informe técnico de la Comisión Europea sobre pautas éticas para la IA confiable, las soluciones de IA propuestas deben priorizar la confiabilidad. Sin embargo, debido a la falta de transparencia y confianza en los modelos de IA, los operadores de telecomunicaciones dudan en implementar ampliamente modelos de IA en sus redes, especialmente cuando tales decisiones tienen implicaciones financieras y de calidad de servicio. Para abordar este desafío, presentamos la arquitectura de SliceOps, donde las operaciones de ML explicables se consolidan en un slice independiente que proporciona servicios de IA a otros slices. Esta entrega continua (CD) e integración continua (CI) de modelos de ML mejora la confiabilidad y la interpretabilidad al implementar rápidamente modelos de IA en la red con mayor consistencia. Al pasar de un enfoque nativo IA sin intervención (en inglés, zero-touch) a un enfoque zero-touch nativo de automatización, el marco propuesto puede gestionar no solo las funciones de servicio sino también las funciones de IA subyacentes, permitiendo aprovechar todo el potencial del slicing en RAN y reducir las complejidades involucradas.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Prof. Christos Verikukis, and my tutor Prof. Luis Alonso for their guidance, support, and encouragement throughout my Ph.D. studies. Their invaluable advice and feedback have shaped my research and contributed significantly to the completion of this thesis.

I am also grateful to my esteemed thesis committee members for their insightful comments and suggestions, which have improved the quality of my research and challenged me to think critically.

I extend my heartfelt thanks to Dr. Hatim Chergui who helpt me as a co-advisor, and my colleagues and friends at Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), Services as NetworkS (SaS) research unit, and NEC Laboratories Europe for their support and encouragement during this journey and also Marie Skłodowska-Curie fellowship that supported my Ph.D. Their presence has made the Ph.D. experience more enjoyable and less daunting.

I am grateful to my wife and family for their unwavering love and support. Their belief in me has sustained me through the ups and downs of this challenging process.

Finally, I would like to express my appreciation to Prof. Josep Solé-Pareta, and all the participants who took part in my study. Without their cooperation and generosity, this research would not have been possible. Thank you all for your contributions, guidance, and support.

This thesis is a testament to your kindness and generosity.

Contents

List of Figures

List of Tables

Abbreviations

4G	fourth-generation
5G	fifth-generation
5GC	fifth-generation core
6G	sixth-generation
A2C	advantage Actor-Critic
AI	artificial intelligence
AIoT	artificial intelligence of things
API	application programming interface
AR/VR	augmented/virtual reality
B5G	beyond 5G
BS	base station
CAPEX	capital expenditure
CCDF	complementary cumulative distribution function
CI	continuous integration
CN	core network
CNF	cloud-native network functions
CP	control plane
CS-AC	collaborative statistical actor-critic
CU	central unit
D-QoS	differentiated quality of service
DA	decision agent
DDPG	deep deterministic policy gradient
DDQN	double deep Q-networks
DE	decision engine
DL	deep learning

DN	data network
DU	distributed unit
E2E	end-to-end
ECP	enhanced control plane
EC	European commission
EGL	explanation-guided learning
EPC	enhanced packet core
ETSI-ENI	European telecom standards institute experiential networked intelligence
ETSI-ZSM	European telecom standards institute zero-touch service management
FDRL	federated deep reinforcement learning
FL	Federated Learning
FFT	fast Fourier transform
Gbps	gigabits per second
gNB	gNodeB
IIoT	industrial IoT
IoT	Internet of things
IID	independent and identically distributed
KP	knowledge plane
LCM	lifecycle management
LSTM	long short-term memory
MANO	management and orchestration
MDP	Markov decision process
MEC	multi-access edge computing
MLOps	machine learning operations
MGEN	multi-generator
MNO	mobile network operator
MOPTS	million operations per time slot
NAS	non-access stratum
Near-RT RIC	near-real time RAN intelligent controller
NFV	network function virtualization
NFVO	virtual network function orchestration
NR	new radio
NSSAI	network slice selection assistance information

O-CU	O-RAN central unit
O-DU	O-RAN distributed unit
OPEX	operational expenditure
OSS	operations support system
PDU	protocol data unit
PFCP	packet forwarding control protocol
PoC	proof of concept
PRB	physical resource block
QoS	quality of service
RAN	radio access network
RIC	radio intelligent controller
RR	random-representative
RL	reinforcement learning
SAC	soft actor-critic
SDN	software-defined networking
SDO	standard development organization
SDLC	software development life cycle
SDR	software-defined radio
SGD	stochastic gradient descent
SINR	signal-to-interference-plus-noise
SLA	service-level agreement
SMF	session management function
SON	self-organizing network
SST	Slice/Service Type
TD-SAC	twin-delayed double-Q soft actor-critic
UE	user equipment
UP	user plane
UPF	user plane function
URLLC	ultra reliable low latency communication
vRAN	virtualized radio access network
V2V	vehicle-to-vehicle
V2X	vehicle-to-everything
VNF	virtual network function

xURLLC	ultra-reliable and low-latency communication
XAI	eXplainable artificial intelligence
XRL	eXplainable deep reinforcement learning
ZSM	zero-touch network and service management

Symbols

H_t	History of actions, rewards and observation in time step t
$\mathcal{S}$	State Space
$\mathcal{A}$	Action space
$\mathcal{R}$	Reward
γ	Discount factor
V_π	State-value function under a given policy π
Q_π	Action-value function under a given policy π
N	Set of APs
M	Set of single-antenna users
$\mathcal{B}$	Set of BSs
$\mathcal{I}$	Set of slices
C_b	Capacity of BS $b \in \mathcal{B}$
Λ_i	Latency requirement of slice $i \in \mathcal{I}$
λ_i	Throughput requirement of slice $i \in \mathcal{I}$
$\mathcal{T}$	Set of decision intervals
ϵ	Duration of decision intervals $t \in \mathcal{T}$
$a_{i,b}^{(t)}$	PRB allocation for slice $i \in \mathcal{I}$ at BS $b \in \mathcal{B}$ taken at time interval $t \in \mathcal{T}$
$\sigma_{i,b}^{(t)}$	Average SNR experienced by the users of slice $i \in \mathcal{I}$ at BS $b \in \mathcal{B}$ in time interval $t \in \mathcal{T}$
$\lambda_{i,b}^{(t)}$	Aggregated traffic demand of the users of slice $i \in \mathcal{I}$ at BS $b \in \mathcal{B}$ in time interval $t \in \mathcal{T}$
$\varphi_{i,b}^{(t)}$	Instantaneous traffic demand of the users of slice $i \in \mathcal{I}$ at BS $b \in \mathcal{B}$ in time interval $t \in \mathcal{T}$
$d_{i,b}^{(t)}$	Traffic of slice $i \in \mathcal{I}$ at BS $b \in \mathcal{B}$

	dropped in time interval $t \in \mathcal{T}$
ι	Minimum PRB allocation
$E_{i,b}^{(t)}$	Expected transmission latency
$\nu_i^{(t)}$	Amount of available capacity left by previous decisions of other agents
$\alpha_i^{(t)}$	Allocation gap
$\rho_{\text{up}}^{(t)}$	Upper boundary of allocation gap
$\rho_{\text{lower}}^{(t)}$	Lower boundary of allocation gap
$r_i^{(t)}$	Instantaneous reward of the i-th agent
$P_i^{(t)}$	Penalty of i-th agent
η_i	Penalty coefficient
ξ	Learning rate
β_i	Experience buffer
$\theta_i^{(t)}$	Online network parameter
$\tilde{\theta}_i^{(t)}$	Target network parameter
$\Omega_i^{(t+1)}$	Global updated model
Ψ	Set of clusters

Chapter 1

Introduction

Digital cellular data communications refer to data traffic exchange over a cellular telecommunications network and require the presence of an underlying physical data communications infrastructure layered upon a cellular network, such as that first evidenced by 4G cellular communications and, more recently, by the substantially more robust and reliable 5G networks. In 5G, the network architecture supports the connectivity of UE to different base stations (gNBs) clustered in different RANs, with each RAN coupled to the CN. Whereas 4G represented a giant leap in performance over 2G and 3G networks, 5G represents an enormous improvement over 4G. Capitalizing on Massive MIMO antenna arrays in each base station, the utilization of millimeter wave radio communications, beamforming for direct wireless communications with individual UE, and a bifurcated CU and DU architecture, 5G can achieve a data exchange capacity of nearly thirteen terabytes–almost a twenty times improvement over 4G LTE. The CN of the 5G architecture reflects a substantial change over the EPC of 4G. In the CN of 5G, the changes have been reduced, abstractly, into what has been referred to as the "Four Modernizations". The first is "information technology" or "IT", the second is the "Internet", the third is "extremely simplified", and the fourth is "service-based". The most typical change in the network architecture of the CN is the service-based network architecture to separate the control plane from the user plane. Other technologies support network slicing and edge computing.

As to IT modernization, the essential characteristic of the 5G architecture is the notion of NFV. Indeed, NFV decouples software from hardware by replacing various network functions such as firewalls, load balancers, and routers with virtualized instances running as software. This eliminates the need to invest in many expensive hardware elements and can also accelerate installation times, thereby providing revenue-generating services to the customer faster. NFV enables the 5G infrastructure by virtualizing appliances within the 5G network. This includes the network slicing technology that enables multiple virtual networks to run simultaneously. NFV can address other 5G challenges through virtualized computing, storage, and customized network resources based on the applications and customer segments.

Network slicing adds an extra dimension to the NFV domain by allowing multiple logical networks to run simultaneously on top of a shared physical network infrastructure. As such, network slicing becomes integral to 5G architecture by creating end-to-end virtual networks that include both networking and storage functions. Operators of a 5G network then can effectively manage diverse 5G use cases with differing throughput, latency, and availability demands by partitioning network resources to multiple users or "tenants". With strategically tuned network slicing and optimized allocation of VNF instances, the cost of operating a 5G architected network can be optimized.

This thesis aims to tackle the implementation of network slicing in the RAN through the development of customized mechanisms and AI/ML solutions for allocating resources to slices in a zero-touch manner. However, during the design process of these policies and mechanisms, it became clear that there is a requirement for more precise definitions of the slicing concepts and a more accurate problem description and formulation. As a result, this thesis also incorporates an elaboration on these matters.

Further details on the objectives and contributions covered in this thesis are provided in the subsequent sections.

1.1 Objectives

The incorporation of AI into 5G/6G networks can substantially boost network performance, elevate dependability, and facilitate novel applications and services that were previously unattainable. Nevertheless, further research and development are required to completely comprehend the possibilities of AI in 5G/6G networks and establish the requisite technologies and protocols to facilitate these advancements. Since B5G/6G natively underpins AI, the main objective of this thesis is to introduce new technology and a revolutionary approach to transform from AI-native to automation-native network slicing. The network slicing ecosystem needs algorithmic and architectural innovations and adjustments to embed AI solutions into the network and maintain AI/ML pipelines in production. This thesis design and develop zero-touch automation techniques (in decision engine), frameworks, and mechanisms to pave the way for implementing fully-automated network slicing in 5G/6G. The developed resource allocation mechanisms are expected to automate procedures efficiently and guarantee different performance requirements while bringing openness, transparency, reliability, and robustness into network slices.

In pursuit of this objective, we follow a feasibility study, including theoretical and empirical parts, to answer the following question: *Is it possible to design and implement a trustable lifelong zero-touch mechanism for controlling RAN slicing resources in 5G/6G?*

In this regard, we pursue a set of steps based on the literature survey and study SoA, modeling, designing DRL algorithms, simulation, and performance analysis. The obtained results are both qualitative and quantitative. The research started with a comprehensive study and analysis of the slicing concepts and main challenges. Then it follows with a thorough study of the SoA, emphasizing algorithmic innovation. Afterward, novel DRL approaches and frameworks are designed and developed for studying the specific problems according to energy-aware, low-latency, scalability, and explainability issues in RAN slicing. Later, the strategy consists of developing new techniques and mechanisms with complementary approaches to test in simulation and frameworks. A reproducible process is followed to design, implement, and analyze the solutions to attain SoA performance. For the assessment of the proposed solutions, specific evaluation scenarios are defined. Moreover, the evaluation and validation of AI and network metrics are considered to analyze the results.

1.2 Contributions

This thesis delves into algorithmic innovations, focusing on deep reinforcement learning (DRL) techniques to enhance the decision engine in the closed-loop of zero-touch network slicing. The primary objective is to control resource allocation in a manner that prioritizes energy-efficiency, low-latency, scalability, trustworthiness, and interpretability. This research aims to drive automation-native transformation within the context of 5G and forthcoming 6G mobile networks. The main ideas, proposed algorithms, figures, tables, and presented results in this thesis are derived from several scientific publications, listed in **Chapter 6**, published or submitted in international peer-reviewed conferences and journals. The following is a summary of the main contributions of this thesis:

- **Continuous DRL approach for multi-objective resource allocation in RAN slicing setup**. The DRL typically benefits from continuous action spaces due to increasing flexibility, precision, and providing robust performance compared to discrete action spaces. Discrete action spaces limit the agent to a fixed set of actions, resulting in suboptimal behavior if the optimal action falls between the available options. In contrast, continuous action spaces allow the agent to choose any action within a range of values (e.g., CPU allocation), resulting in higher precision and the ability to select the optimal action, even if it falls between discrete actions. Additionally, continuous action spaces are better suited for complex tasks such as resource allocation in network slicing because they capture a more comprehensive range of behaviors and experiences. By leveraging this technique and complementary approaches such as reward shaping and parallelism, we solved resource allocation optimization while minimizing energy consumption and incurred latency within network slices in **Chapter 3**.

- **An FDRL framework for managing and orchestrating scalable and distributed RAN slicing in 6G.** The early stages of creating a scalable and distributed 6G network involve designing the frameworks to manage substantial data traffic and dynamically allocate resources to meet changing demand in real-time. Planning and allocating resources in the RAN domain, particularly regarding bandwidth, is complex due to fluctuating traffic demand over time. This is further compounded by the additional variability inherent in the wireless channel and end-user mobility, which adds another layer of variability to resource allocation decisions. To cope with this challenge in **Chapter 4**, we proposed a distributed and hierarchical scheme for managing RAN slice resource orchestration using DRL. It involves deploying several decision agents, each enabled with AI, to carry out localized radio allocation decisions. By adopting this approach, a centralized control entity will not be required. To improve the effectiveness of the decision-making process, we have designed an FL solution comprising several layers where each layer corresponds to a specific slice, following the advancements made in the O-RAN architecture. This FL scheme will help to address scalability challenges experienced in network slicing. Besides, the framework dynamically determines which subset of decision agents should participate in the federation process based on long-term variations in slice traffic demand and their temporal similarities that further enhance the decision-making process.

- **A framework for RAN slices based on OpenAI Gym towards openness B5G/6G**

technology. In the context of B5G/6G, openness pertains to the ability of diverse technologies, devices, and services to interoperate without closed or proprietary interfaces seamlessly. An open B5G/6G framework would enable various devices, applications, and services to communicate with each other irrespective of their origin or brand. In this intent and following open-source and fulfill reproducible experiments for algorithmic innovation in network slicing, we designed and implemented a software platform based on OpenAI Gym and in compliance with O-RAN in **Chapter 4**, which includes virtual transmission queues and main PHY/MAC/RLC functionalities and enables slice networking statistics to be collected.

- **A revolutionary approach for transforming from AI-native to Automation-native B5G/6G networks**. Since B5G/6G natively supports AI, the stakeholders in structuring B5G/6G network slices need innovations and adjustments to embed AI solutions into the network. In **Chapter 5**, we propose a revolutionary approach for B5G/6G networks, called SliceOps to concentrate AI/ML operations (MLOps) in standalone slices, which manages the whole lifecycle of AI models separately beside the underlying service functions. It pushes AI models into production, i.e., helps telecom companies or service providers to maintain ML pipelines in a production environment to proactively monitor, unveil, and measure ML models' quality to improve the network's automation while providing AI services to the main B5G/6G slices.

- **An explanation-guided framework for transparent, trustworthy, and interpretable DRL-based resource allocation in the B5G/6G network slicing**. Transparency and trustworthiness in B5G/6G networks are essential to maintain users of network slicing ecosystem confidence in the applied AI technologies. To this end, we proposed an explanation-guided DRL scheme to enable the agent to scrutinize each feature and its impact on the output of the DNN model and enable to observe the factors that either positively or negatively impact the performance of DRL for resource allocation. The Sparse rewards and a lack of interpretability hamper the accuracy and reliability of estimating state-action pairs in DRL. To cope with this problem, we proposed an Explainer in **Chapter 5** that uses the SHAP importance values to generate a probability distribution over a batch of state-action data by applying the softmax function to the SHAP values. An Entropy Mapper then calculates the entropy, which reflects the uncertainty of the selected action given the input state. The inverse of the maximum entropy value is then used as the XAI reward. We validated that this approach can reduce the uncertainty of state-action pairs and encourages the agent to choose the best actions for specific network states. It helps to clarify the learning process while guiding the learning towards making explainable decisions for a given state. This framework is integrated into the training phase of SliceOps with an attention-based submodule to inspect the contribution and impact of network slice states on decision-making and choice of particular actions.

1.3 Thesis Outline

Fig. 1.1 illustrates the general organization and interconnections between chapters within this thesis. The remaining outline of the thesis is organized as follows:

Chapter 2 contains an overview of the relevant SoA on zero-touch 5G/6G network slicing and algorithmic innovation to fulfill zero-touch decision engine. We target energy, low-latency, scalability, trustworthiness, and explanation scopes in the future generation of mobile communication networks. It details DRL solutions and complementary approaches to solving the problems and challenges of the considered scopes in this thesis.

Chapter 3 targets novel energy-aware and latency-aware resource allocation solutions based on actor-critic algorithms. Optimizing energy consumption and reducing latency based on intelligent resource allocation mechanisms is crucial for ensuring energy efficiency and supporting low-latency services of 5G/6G networks. One effective method for achieving this goal is using actor-critic techniques that support continuous action while providing more reliable and robust results. Actor-Critic methods involve an actor network that takes actions based on the current state of the environment and a critic network that evaluates the actions taken by the actor. The critic provides feedback to the actor to improve its future actions. To apply actor-critic methods in energy-aware and latency-aware resource allocation, an actor network can be utilized to allocate resources such as frequency bands, transmission power, and CPU resources to each user or slice, while a critic network evaluates the energy consumption or incurred latency associated with each allocation. We have developed an approach that separates the actor from the learner to address the control challenges that arise with network slicing, such as greater dynamism, heterogeneity, and longer training times for slice instances. This approach allows us to scale up to thousands of parallel actors-learners across various tasks while maintaining data efficiency.

Chapter 4 deals with scalability and distributed challenges in the next generation of network slicing in 6G. We investigate the application of FDRL in 5G/6G network slicing and elaborate on the proposed framework that deploys a set of traffic-aware local decision agents within the RAN, which adapt their resource allocation strategy based on the long-term trends in traffic. These decentralized DEs establish specialized clusters facilitating faster training and reducing communication overhead. With the assistance of a traffic-aware agent selection algorithm, our FDRL approach can respond rapidly to changes in end-user mobility patterns, resulting in higher resource efficiency compared to benchmark solutions while reducing the need for costly interactions with centralized controllers.

Chapter 5 delves into proposing a new technology to transform from AI-native to automation-native 6G network aims to, besides handling the service functions, enable the network to control underlying AI functions, i.e., push AI/ML models into production. Moreover, since 6G networks are anticipated to serve as the foundation of critical services such as healthcare, transportation, and energy management, ensuring high reliability, availability, and security with trustworthy solutions is vital. Consequently, any breakdowns or disturbances in the 6G network could have serious ramifications, from financial losses to jeopardizing people's lives. To solve this, we proposed an explanation-guided AI framework in this chapter to enhance the reliability and interpretability of ML models by implementing CI and CD methodologies.

Chapter 6 aims to provide results to those who can best benefit from them, including the scientific community, industry, policymakers, and other commercial players. Dissemination serves several purposes, including demonstrating the broad relevance of science to society, garnering support for future research and innovation funding, ensuring that findings are utilized within the scientific community, and creating opportunities for new products or services. Furthermore, the goal is to consider the utilization of results and findings in the development, enhancement, marketing, or refinement of a product, process, or service, as well as in standardization efforts or policy formation. This can involve commercial, societal, political, or educational purposes and promote public knowledge and action. Exploitation concentrates on transforming research ideas into tangible solutions that benefit people's quality of life.

In conclusion, **Chapter 7** of this thesis summarizes the presented work and debates the limitations of the contributions. Additionally, potential areas and directions for future research and expansion of the topics covered in this thesis are discussed.

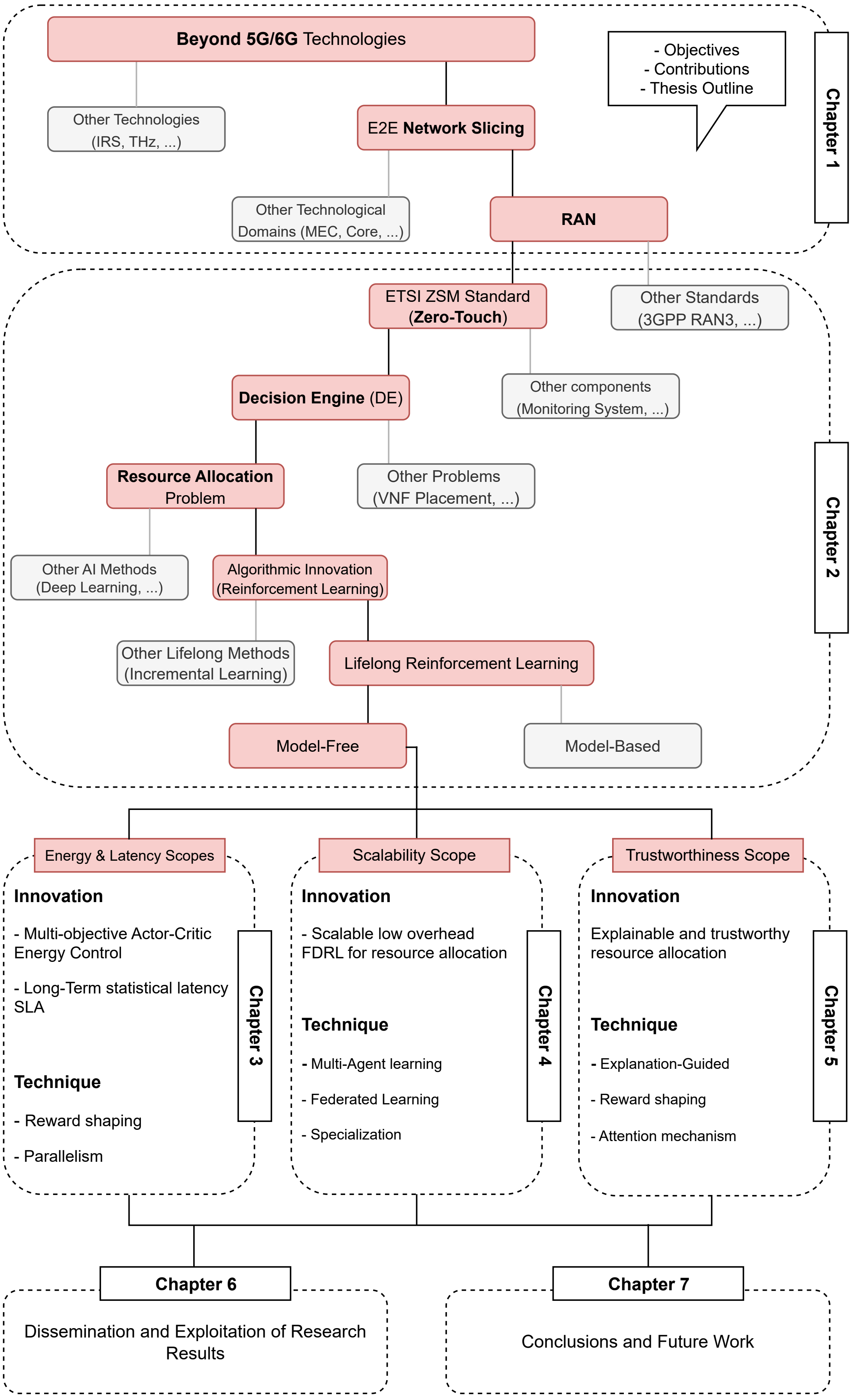

Beyond 5G/6G Technologies
Other Technologies (IRS, THz, ...)
E2E Network Slicing
- Objectives
- Contributions
- Thesis Outline
Chapter 1
Other Technological Domains (MEC, Core, ...)
RAN
ETSI ZSM Standard (Zero-Touch)
Other Standards (3GPP RAN3, ...)
Decision Engine (DE)
Other components (Monitoring System, ...)
Resource Allocation Problem
Other Problems (VNF Placement, ...)
Chapter 2
Other AI Methods (Deep Learning, ...)
Algorithmic Innovation (Reinforcement Learning)
Other Lifelong Methods (Incremental Learning)
Lifelong Reinforcement Learning
Model-Free
Model-Based
Energy & Latency Scopes
Innovation
- Multi-objective Actor-Critic Energy Control
- Long-Term statistical latency SLA
Technique
- Reward shaping
- Parallelism
Chapter 3
Scalability Scope
Innovation
- Scalable low overhead FDRL for resource allocation
Technique
- Multi-Agent learning
- Federated Learning
- Specialization
Chapter 4
Trustworthiness Scope
Innovation
Explainable and trustworthy resource allocation
Technique
- Explanation-Guided
- Reward shaping
- Attention mechanism
Chapter 5
Chapter 6
Dissemination and Exploitation of Research Results
Chapter 7
Conclusions and Future Work

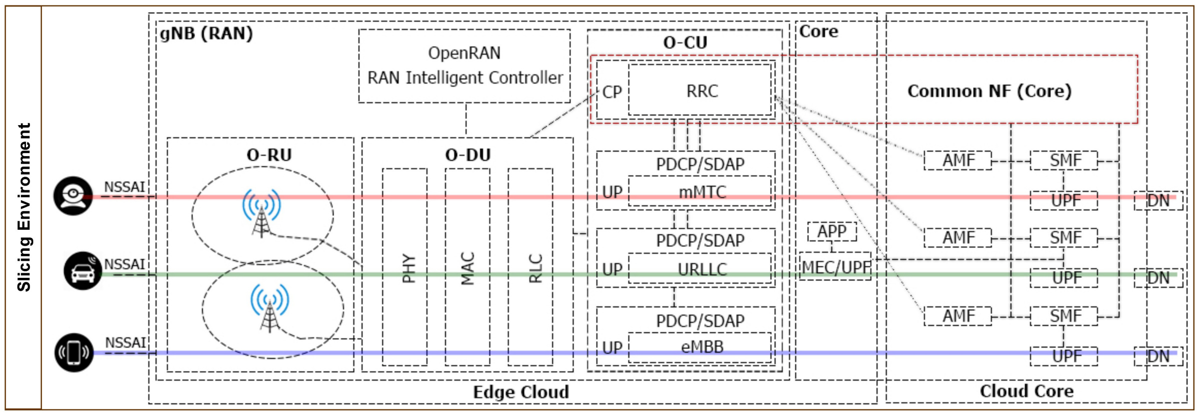
FIGURE 1.1: Thesis organization.

Chapter 2

Zero-Touch Network Slicing

2.1 5G/6G RAN Slicing Overview

Due to the exponential growth in the demand for service provisioning and immense challenges in 5G/6G, it is necessary to adopt network slicing technology as a key enabler. Specifically, zero-touch RAN slicing is a significant step towards the evolution of 5G/6G and future networks, and it holds great promise for mobile network operators (MNOs) operating in the 5G/6G network environment. By utilizing RAN slicing technology, MNOs can create several virtual RANs on a single physical infrastructure, allowing them to offer personalized connectivity services and optimize their network resources. A slice instance is self-contained in terms of traffic flow and operation which has its architecture and specific characteristics that lead to support manifold use case realization. It provisions the functionality of the whole C-RAN sites and cloud-native core that is bolstered up through multifarious technological advances and platforms such as cloudification. In this regard, network slicing can provide better performance, flexibility, and scalability compared to one-size-fit-all networks. This technology enables MNOs to optimize network resources, reduce costs, and enhance operational efficiency. To implement RAN slicing, MNOs require SDN and NFV technologies to segment the RAN infrastructure into virtual RANs. Once segmented, the virtual RANs can be independently managed and configured with different policies and performance characteristics. RAN slicing can be implemented through various methods such as dedicated physical resources, time or frequency resources, or network slicing over the same physical resources.



NSSA	Network Slice Selection Assistance Information	**MAC**	Medium Access Control	**SDAP**	Service Data Adaptation Protocol
O-RAN	Open Radio Access Network	**RLC**	Radio Link Control	**MEC**	Multi-access Edge Computing
O-RU	O-RAN Radio Unit	**UP**	User Plane	**NF**	Network Function
O-DU	O-RAN Distributed Unit	**CP**	Control Plane	**UPF**	User Plane Function
O-CU	O-RAN Central Unit	**RRC**	Radio Resource Control	**AMF**	Access and Mobility Management Function
PHY	Physical	**PDCP**	Packet Data Convergence Protocol	**SMF**	Session Management Function

FIGURE 2.1: Network slicing background.

Fig. 2.1 presents the E2E network slicing scheme based on O-RAN slicing related use cases, requirements and architecture [2], which encompasses the RAN slicing and the core network slicing with CP and UP network functions within the physical infrastructure of a mobile network. A network slice spans the network domains on shared physical infrastructure resources. We consider slice-enabling RAN (gNB) with 3GPP CU-DU functional split. The network slice identification is made through the single-NSSAI, where NSSAI signaling message is allowed to send a collection of up to eight S-NSSAIs between the user and the network[1]. The S-NSSAI consists of a SST and an optional SD, where SST refers to expected specific features and services. The SST includes URLLC, eMBB, and mMTC. The SD is an additional differentiator for multiple network slices with the same SST value. Indeed, standardized S-NSSAI has only SST value.

The UE sends configured or allowed NSSAI to the RRC and NAS signaling as part of registration. NSSAIs are managed at the tracking area level (RAN) and registration area level (core). According to NSSAI, the CU-CP selects a specific AMF during the PDU setup procedure. Notice that UE can use multiple network slices; therefore, AMF can be shared between slices. For example, the AMF for an eMBB user selects a specific SMF, and thereby SMF chooses a specific UPF according to location and load information. The UPF connects mobile infrastructure to the DN or application server. The common core is a cost-efficient approach for 4G to 5G migration where LTE and 5G NR and non-3GPP accesses (e.g., Wi-Fi and fixed broadband)

[1]`https://www.etsi.org/deliver/etsi_ts/124500_124599/124501/16.11.00_60/ts_124501v161100p.pdf`

are integrated via a common interface to underpin multi-RAT. The RAN and core maintain and deploy a set of VNFs and CNFs to serve the users of distributed RUs. They host agents to facilitate the training process to learn the best policies and actions for scaling the resources vertically and consequently scaling horizontally for VNFs and CNFs instances according to system states.

To fully automate network slicing management, ETSI has introduced the ZSM reference framework [3]. It is paired with the use of intent-based interfaces, closed-loop operation, and AI techniques to empower the full-automation of network slicing, which are cutting-edge technologies that streamline creating and managing network slices without requiring manual intervention. By leveraging zero-touch technology, virtual networks can be created and managed automatically, making deploying new applications and services more straightforward and quicker. The main principle underlying zero-touch is to automate network slice creation and management. It leads to cost and time savings; a critical requirement in industries such as telecommunications, where service providers need to quickly deploy new services and applications to meet evolving customer needs. Zero-touch technology can also enhance network efficiency by dynamically allocating resources, ensuring that each network slice has the resources it needs to operate efficiently. As demand for new services and applications continues to grow, zero-touch is poised to become increasingly crucial for service providers and other organizations seeking to deploy and manage complex network environments.

To this end, the ZSM architecture supports a set of architectural design principles, including:

- Modularity aspects for creating self-contained and loosely-coupled services to prevent monoliths and tight coupling.
- Extensibility enables the network to extend new services and service capabilities.
- Scalability fulfills increasing or decreasing demands to deploy managed entities, and modules can be independently scaled.
- Resilience aspects cope with the degradation of the infrastructure and other management services as well as simplicity makes minimal complexity while still meeting the functional and non-functional requirements.

The modular characteristic is paired with intent-based interfaces, closed-loop operation, and AI/ML techniques to empower the full automation of management operations.

The architectural building blocks consist of i) management services, ii) management functions, iii) management domains, iv) integration fabric, and v) data service. The management services are main pivotal in ZSM framework reference architecture that federate together management domains, including a set of capabilities for communication purposes, automation orchestration, and managing one or more entities such as infrastructure resources that can be physical (e.g., physical network functions), virtual (e.g., VNFs or software-based services) and/or cloud-based (e.g., "X-as-a-service" (XaaS) resources). Each management domain is split into sub-domain to consider different management concerns. The integration fabric is a special management function that enables the communication between management functions within management domains while offering a set of communication capabilities, such as synchronous and asynchronous communication. The data service allows us to separate data storage from data processing and support different types of storage mechanisms and database technologies to provide current management data such as configuration data, performance/fault alarm events, and topology data for AI-based closed-loop automation procedures.

Within the ZSM architecture, closed-loop may exist in each management domain. The functional scheme of the closed-loop depicted in Fig. 2.2, highlights the importance of the 'Decision' stage. Fig. 2.2 consists of four stages in addition to the *Knowledge* functional block. The *moni-*

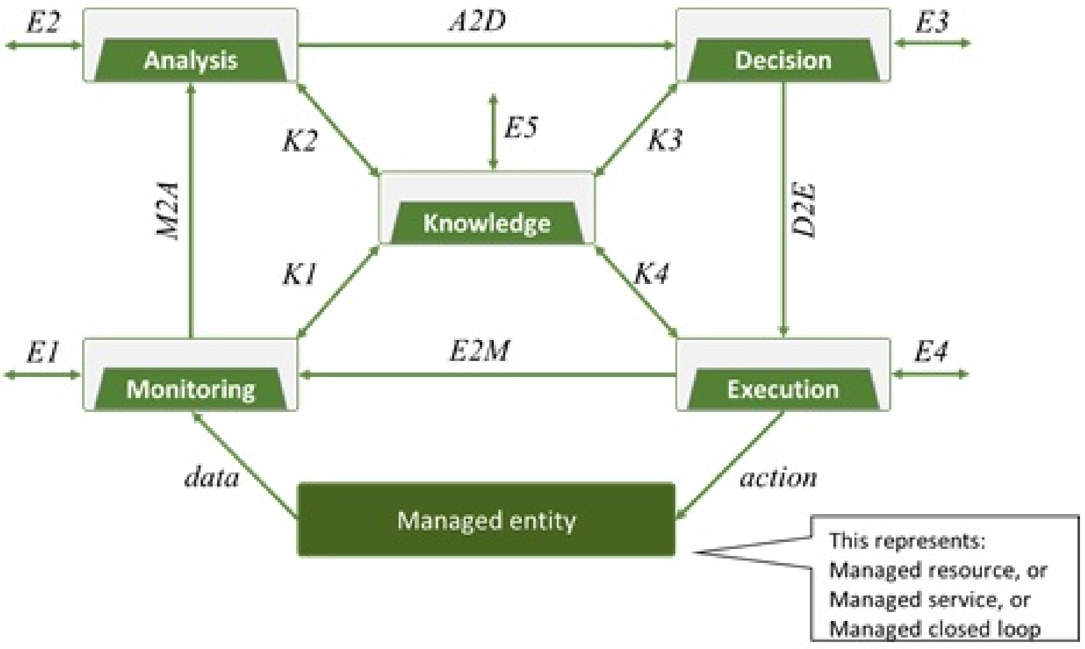


FIGURE 2.2: Functional view of a closed-loop [1].

toring/collection stage collects and preprocesses raw data from various sources such as services, managed resources, or closed-loops. Since the raw data can have different formats, it is transformed to enable its analysis with data from other sources. The *Analysis* stage provides insights from the available data obtained from the Monitoring stage. The *Decision* stage determines the action that needs to be taken based on the issues detected by the analysis block, which can be reactive, proactive, or predictive. The Decision stage only decides which actions are necessary, while the *Execution* stage is responsible for executing the necessary workflows to implement the actions determined by the Decision stage. This execution may involve other management domains and interactions with other closed-loops. Therefore, multiple distributed closed-loops are required for end-to-end service management. The *Knowledge* block in the Fig. 2.2 is not technically a stage of the closed-loop. It refers to the storage and retrieval of historical, configuration, and operational data shared between the stages of a closed-loop and between different closed-loops in the network. The Decision stage encompasses several key flows or interfaces:

- A2D interface: Connects the Analysis function to the Decision stage, providing historical and/or real-time information from the collection/monitoring stage to inform the analytic models and decision processes.
- D2E interface: Used by the Decision stage to generate action plans through workflows, such as configuring changes or onboarding services and resources. This interface can also be used to fine-tune decision models and initiate or terminate decision processes.
- E3 external interface: Manages data and control inputs and outputs from/to external entities or closed-loops. This interface can start or stop decision processes, adjust Decision stage settings and model attributes, retrieve historical or real-time data from the function (such as logs or outcomes), and export resulting data to authorized external management systems or other closed-loops within the ZSM framework.
- K3 interface: Represents a knowledge-enabled flow for data-related inputs and outputs from the Decision stage, allowing data from the *Knowledge* functional block (historical or real-time workflows) to be incorporated into decision-making processes. Data from the Knowledge block may further supplement these primary interfaces to enhance decision-making capabilities.

To attain closed-loop operation, a management framework must offer a way to systematically invoke the different steps or phases of the closed-loop (such as observe and decide) in a specific sequence; therefore, we should boost and tune the current dominant AI paradigm to improve learning in complex telecommunications environments for intelligent control and decision. This thesis focuses on the decision in closed-loop and algorithmic innovation for RAN slicing.

2.1.1 Algorithmic Innovation

Network slicing has a dynamic, heterogeneous, and decentralized nature. Emerging use cases and applications with different stringent requirements pose new challenges to algorithm designing in network slicing. Implementing 5G/6G network slicing is anticipated to facilitate various use cases that demand different network performance characteristics, such as URLLC, mMTC, and eMB). To accommodate these use cases, network operators must develop and deploy network slices tailored to each use case's needs. Algorithmic innovation can significantly contribute to efficient and effective network slicing in 5G/6G. By utilizing advanced algorithms and techniques, network operators can tailor network slicing to accommodate the diverse requirements of various use cases and applications. Examples of how algorithmic innovation can be employed in network slicing in 5G/6G include:

- Resource allocation algorithms: These algorithms can optimize the allocation of network resources to different network slices based on factors such as network load, available bandwidth, and quality of service requirements of each network slice.
- Virtual network function placement algorithms: These algorithms can determine the optimal placement of network functions, such as firewalls, routers, and load balancers, in the network to minimize latency and maximize the throughput of the network slices.
- Dynamic slice management algorithms: These algorithms can facilitate the automatic scaling of network slices based on traffic demand and support the creation and deletion of network slices based on application requirements.
- Predictive analytics algorithms: These algorithms can aid network operators in forecasting traffic patterns and performance characteristics of different network slices based on historical data, allowing proactive optimization of the network slices and early detection of potential issues.

ML algorithms can optimize resource allocation, network function placement, and dynamic slice management in network slicing. They can also detect anomalies and security threats in network slices for faster response times and better network protection. Algorithmic innovation can unleash the potential of network slicing and supports progressive change across all stakeholders. Designing intelligent algorithms can potentially yield benefits in terms of reliability, convergence, and network performance optimization in a large-scale network slicing environment. Initial results have shown that AI can solve problems that are difficult to address and not be easily modeled concerning wireless communication uncertainties. AI/ML methods exploit the huge volumes of data that are produced by the network monitoring entities (such as centralized OSS platforms and virtual probes) to control and manage the sophisticated and complex nature of the 5G/6G environment through an analytics platform. This big data creates unprecedented opportunities for MNOs to discern the requirements and predict the behavior of demands. In a challenging situation where prediction modeling faces several limitations, RL provides a promising technique to be incorporated in mobile communication systems as an elegant and SON solution for solving many resource management and other optimization issues. Indeed, RL acquires new behaviors and skills cumulatively, so its agent does not require complete view or control of the network.

In this thesis, we leverage proposed DRL solutions to enhance network slicing in 5G/6G, explicitly focusing on resource allocation algorithms to address the subsequent challenges.

2.1.1.1 Energy Scope

Managing energy consumption is crucial in network slicing as it is a critical factor in network infrastructure operations. Energy scope in network slicing refers to managing energy consumption at various levels of the network slice in the entire network. One way to optimize energy consumption is to leverage SDN and NFV technologies, which allocate network resources dynamically based on demand, reducing energy consumption during low utilization. Another approach is to use energy-efficient hardware and equipment, such as low-power processors, energy-efficient switches, servers, and renewable energy sources. Using energy-efficient technologies and equipment can help reduce the environmental impact of network operations while also improving the overall efficiency and profitability of the network slice. In this context, [4]-[5] have presented softwarization approaches in network slicing. In [6], the authors have proposed vrAIn as a dynamic resource controller based on DRL for optimal allocation of computing and radio resources. Li *et al.* have proposed a DDPG-based solution to enhance energy efficiency and obtain the optimal power control scheme [3]. Correspondingly, [7] has proposed a method to learn the optimum solution for demand-aware resource management in C-RAN network slicing.

They have developed a DRL method as GAN-DDQN to handle resource management in network slicing. In [8], it has leveraged A2C and incorporated the LSTM to track the user mobility and improve the system utility. More recently, Liu *et al.* have proposed a DRL-based method called, DeepSlicing where they decompose network slicing problem into a master problem and several slave problems wherein DDPG agents learn the optimal resource allocation policy [9].

In **Chapter 3**, we propose a novel approach to optimizing energy consumption and VNF instantiation cost using continuous model-free DRL. We introduce a new method called TDSAC, which is an actor-critic-based network slicing approach that enables continuous learning to minimize future NS costs. The TDSAC method is designed to stabilize learning, allowing the CU to accumulate knowledge learned in the past and apply it to future network slicing decisions. The use of DRL in this approach means that the system does not require a model of the environment to learn, making it more flexible and adaptable to changes in the network. This approach can be particularly useful in dynamic network environments, where the system needs to adapt quickly to changes in traffic or network conditions.

2.1.1.2 Low-Latency Scope

In network slicing, a low-latency slice pertains to a network slice specifically engineered to offer low-latency connectivity for particular applications or services. Real-time and low-latency communication is indispensable for applications that require it, such as augmented reality, virtual reality, gaming, and self-driving vehicles. Creating a low-latency slicing involves assigning network resources and prioritizing low-latency traffic over other traffic types. This can be accomplished by utilizing specialized networking protocols, optimized equipment to lessen latency, and latency-aware resource allocation. Latency-aware resource allocation is a technique utilized in network slicing to guarantee that low-latency applications and services have the necessary network resources to minimize latency. By applying latency-aware resource allocation, service providers can prioritize low-latency traffic and ensure that the essential resources are assigned to support these applications and services. Koo *et al.,* have proposed a DRL-based network slicing method and improved resource utilization and latency performance with time-varying traffic [10]. In [9], the authors have proposed a scheme to allocate network resources effectively. The authors have integrated the alternating direction method of multipliers (ADMM) and DRL, where they exploit the DDPG [11] as a SoA actor-critic technique to learn the optimal policy.

Pujol Roig *et al.* have proposed an actor-critic, called parameterized action twin (PAT) deterministic policy gradient algorithm where automated MANO allows a CU to learn to re-configure resources autonomously [12]. Liu *et al.* have studied a new decentralized DRL-based resource orchestration system to automate dynamic network slicing in wireless edge computing networks [13]. In [8], the authors have investigated a demand-aware inter-slice resource management solution based on A2C as a DRL algorithm. In [14], the authors have proposed two centralized scheduling algorithms that take into account latency and SLA requirements in terms of minimal demand and allocate resources in network slicing systems.

From this SoA overview, it turns out that there is no actor-critic-driven network slicing resource allocation strategy that integrates long-term practical SLA constraints with respect to some KPIs while also optimizing service cost. Without loss of generality, **Chapter 3** investigates a multi-tenant network scenario with a developed variant of mMIMO [15] and edge computing approach as promising B5G/6G wireless access technologies.

2.1.1.3 Scalability Scope

AI-driven approaches applied to mobile networks have recently gained momentum in distributed resource control and management tasks. In this context, DRL and FDRL stand out among a multitude of different approaches and are at the center of a strong research interest, especially in the field of automated resource orchestration.

The authors of [16] consider the sum power minimization problem based on jointly optimizing resource allocation, user association, and power control in a MEC system. In this intent, they propose a multi-agent federated RL algorithm to solve centralized method limitations and privacy concerns. The simulation results shown that the proposed approach provides lower maximal latency, lower maximal computation capacity, higher CPU cycles for the tasks, and higher data rate. Due to enhancing spectrum utilization in new generation wireless communication technologies, the authors in [17] invoke an FDRL approach to accelerate learning convergence in edge nodes.

In [18], the authors investigate the decentralized joint optimization of channel selection and power control for V2V communication, proposing a federated multi-agent DRL (Fed-MARL) approach to satisfy the reliability and latency requirements of V2V communication, and maximize the transmit rates of cellular links. The results have shown how the federation of local

DRL models coming from different V2V agents can tackle the limitations of partial observability of the entire network, resulting in superior performances over baseline approaches in terms of communication rate and packet delivery rate.

Targeting at implementing the O-RAN with virtualized network components, the authors of [19] proposed an FDRL-based with a global model server installed in the RIC to update the DQN parameters. This approach can mitigate load balancing and frequent handovers in the massive base station deployment. Following O-RAN standardization, the numerical results have demonstrated the proposed method enables UE to maximize the long-term throughput and avoids frequent handovers. The authors of [20] develop a DRL algorithm for resource allocation in a mobility-aware FL network, optimizing the number of successful transmissions while minimizing energy and channel costs. In [21], the authors propose a federated network slicing scheme based on DRL techniques for channels and bandwidth allocation in the context of IIoT, highlighting significant performance improvements when compared against centralized strategies.

In [22], the authors model the network utility maximization problem and exploit DRL techniques such as deep Q-learning to solve the decision-making task. Their work highlights significant improvements over KPIs and networking metrics, such as throughput and latency. This solution exploits a centralized approach that aims to solve a global optimization, therefore limiting the individual network slices in the management of their own resources. A similar problem has been addressed by [13], which proposes a decentralized resource orchestration system to automate dynamic end-to-end network slicing resource management in wireless edge computing networks. The proposed architecture makes use of a central performance coordinator entity and multiple orchestration agents. This work provides limited details on inter-agent information exchange aspects. Also [23] proposes a DRL approach for the orchestration of service function chains in NFV-enabled networks, addressing both placement-error-rate-based and reward-based federated weighed strategies, showing significant convergence performance, higher average reward, and smaller average resource consumption in a variety of networking scenarios.

From the viewpoint of real-time inter-slice resource management and yield an intelligent strategy, the authors of [24] design a graph attention multi-agent RL to cope with frequent BS handover. The simulation results have demonstrated that the proposed approach effectively enhances the cooperation for the multi-BS system in RAN while satisfying the strict SLA requirements. In [25], the authors propose a multi-agent RL approach for RAN capacity sharing, showcasing better scalability and faster learning in comparison to single-agent approaches. More recently, the authors of [26] developed an FL framework in the context of fog computing, focusing on the distribution of training tasks. The numerical results show that the proposed network-aware scheme significantly improves network resource utilization while achieving comparable accuracy.

Following this SoA overview, the key novelty of our approach in **Chapter 4** relies on the exploitation of distributed RAN information to design a new class of specialized agents that collaborate in homogeneous clusters via a federation layer, which leads to scalable and stable decision under highly dynamic traffic conditions and then the proposed framework is also mapped to O-RAN. To the best of our knowledge, this is the first work to propose an FDRL framework in the context of distributed radio resource management, by adopting dynamic agent selection to improve the specialization of agents and reduce communication overhead.

2.1.1.4 Automation-Native and Trustworthiness Scope

The MLOps practice is an iterative process to push the best ML models into software solution production by bridging ML applications with DevOps principles. The authors in [27] developed an edge MLOps framework for automating and more efficient AIoT operations and decision-making. The proposed framework aims to operationalize the CD and CI of ML models to the nodes as an essential part of DevOps. They successfully implemented the scheme in a real-life scenario by automating workloads assisted an MLOps approach. Samaras *et al.* [28] proposed a cloud-native MLOps automation platform for automated and optimized network slice LCM. This QoS monitoring and prediction platform (QMP) proactively detect and mitigate QoS violation in network slice operation. This event-based framework is a zero-touch MLOps platform that automatically analyzes its accuracy performance and adapts itself automatically through the training job. In [29], the authors presented an ML-as-a-Service (MLaaS) approach for 5G IoT based on a TinyMLaaS (TMLaaS) architecture to improve future IoT deployments in terms of energy consumption, security, and privacy. They leveraged an MLOps framework for unifying ML systems to implement, deploy, and maintain operations. The authors in [30] described two levels of MLOps in O-RAN. They considered a DL-based MLOPs and leveraged the DevOps principles to ease ML system development (Dev) and ML system operation (Ops) for O-RAN automation. Correspondingly, [31] proposed an MLOps lifecycle based on RL aiming to automate and reproducible model development process in the O-RAN deployment. The proposed scheme introduces principles and practices for developing data-derived optimal decision-making strategies integrated with the digital twins and the network analytics platform. Tsourdinis *et* *al.* [32] developed a service-aware dynamic slice allocation scheme. The proposed AI/ML unit in the pipeline follows an MLOps-based distributed ML architecture where the validity and efficiency of the entire framework are evaluated by LSTM model assessment. The results have shown the solution can allocate the proper resources to the network in real time through open APIs.

DECODER[2] is an H2020-EU project that aims to develop a methodology and tools to increase the productivity of the software development process for IoT, cloud computing, and operating systems. The project focuses on the automate the transformation steps using existing techniques from big data (knowledge extraction), model-driven engineering (knowledge representation and refinement), and formal methods. The project revisits the software development lifecycle to propose practical tools for a systematic approach to develop and safely reuse components and their associated knowledge in software production. Different use cases have been defined targeting the IoT and embedded systems, AI, and enterprise computing based on a cloud environment. InSecTT[3] is an ECSEL-JU project that creates trust in AI-based intelligent systems aiming to design AI and ML-based systems trustable, explainable, and not just a black box. The project identified 16 use cases based on AI-supported embedded processing for industrial tasks and AI-enhanced wireless transmission and also worked on an MLOps framework for managing intelligent algorithms. AITHENA [4] researches on three main AI aspects: data (real/synthetic data management), models (data fusion, hybrid AI approaches), and testing. The project methodology is defined in four critical use cases for an XAI approach in 5G connected and cooperative automotive mobility, namely: (i) perception (what does the AI perceive, and why), (ii) situational awareness (what is the AI understanding about the current driving environment), (iii) decision (why a specific decision is taken), and (iv) traffic management (interoperate with AI-enabled systems). The project defines a scalable MLOps approach for testing the above use cases.

Alongside research contributions, some organization and industry initiatives are relevant for MLOps components in telecommunications. The ETSI-ENI[5] is defining a cognitive network management architecture leveraging AI techniques. The architecture targets all networks, including 5G networks aims to provide fully-automated service provision, operation, and assurance. ENI has also launched PoCs to demonstrate the potential of AI solutions to assist network

[2] https://www.decoder-project.eu
[3] https://www.insectt.eu
[4] https://cordis.europa.eu/project/id/101076754
[5] https://www.etsi.org/technologies/experiential-networked-intelligence

operation, such as optimized slice management and resource orchestration. ENI focuses on using closed-loop AI mechanisms based on context-aware, metadata-driven policies to pave the way for making actionable intelligent decisions. It can adopt the closed-loop AI mechanisms to the feedback system in the MLOps lifecycle. Unlike ETSI ENI, which focuses on AI techniques, ETSI-ZSM[6] is investigating automation challenges faced by operators and vertical industries. The group was formed in December 2017 to work on E2E architecture and services automation to solve radical changes in the way networks are managed and orchestrated in the pivotal deployment of 5G and network slicing. To fulfill the unprecedented operational agility required of network slicing and control network without further human intervention, the group is working on a new horizontal and vertical E2E architecture framework based on closed-loop automation and optimized for AI/ML algorithms. In this intent, an MLOps approach with lifecycle management of ML models for reproducible ML pipelines can be necessary. 3GPP RAN3[7] has studied a functional framework to guide how different AI/ML functions interwork to solve the challenge of leveraging AI and ML in current 5G architecture. The framework consists of data collection, model training, model inference, and actor. This RAN-centric data collection and utilization approach identified three prominent use cases based on AI/ML solutions, including network energy saving, load balancing, and mobility optimization. The outcomes influence work in similar activities such as 3GPP SA2[8]. The system architecture and services specification group has specified the network data analytics function (NWDAF). NWDAF can be distributed or centrally located, aiming to guide how data analytics can be derived from the industry's collected data of 5G core network functions, applications, and the OSS. This standalone core network feature can solve the absence of an effective network analytics solution in MLOps to utilize optimized data collection with training and ML model retrieval. ITU-T SG13 ML5G[9] targeted an architectural framework with ten technical specifications for ML in future networks, including interfaces, network architectures, protocols, algorithms, and data formats. This framework aims to abstract ML workflow (ML pipeline) and required functionality. NGMN alliance[10] has published an E2E architecture framework for 6G use cases, and analysis leverages an autonomous system, based on cognitive awareness and AI/ML closed-loop feedback learning models. This framework aims to fulfill self-CHOP (configuration, healing, optimizing, and protecting) behaviors in E2E network slicing. O-RAN Alliance[11] emphasizes

[6]https://www.etsi.org/technologies/zero-touch-network-service-management
[7]https://www.3gpp.org/3gpp-groups/radio-access-networks-ran/ran-wg3
[8]https://www.3gpp.org/3gpp-groups/service-system-aspects-sa/sa-wg2
[9]https://www.itu.int/en/ITU-T/focusgroups/ml5g/Pages/default.aspx
[10]https://www.ngmn.org/
[11]https://www.o-ran.org/

evolving 3GPP access with openness, virtualized, fully interoperable, and AI-enabled RAN. The O-RAN describes RIC as non-realtime RIC and near-realtime RIC to provide the foundations to control and manage 3GPP-defined RANs intelligently.

In this intent, we propose a revolutionary scheme in **Chapter 5** to break ground for adopting SDLC and MLOps approaches in telecommunication to enable fully-automated service provisioning from management to maintenance while transforming AI-native 6G to automation-native 6G networks.

2.2 Reinforcement learning (RL)

RL technique is evaluative feedback-based learning and a branch of ML to optimize the accumulated long-term reward (average return). It automates complex sequential decision-making tasks of an agent without a priori knowledge of the system through interaction with the environment. To enable the agent to take the right actions, it should spend time in a certain part of the environment. Generally, all the sequences of actions, rewards and observation time up to time step t is called the history

$$H_t = [O_1, A_1, R_1, ..., O_t, A_t, R_t]. \tag{2.1}$$

MDP's mathematical context can be considered a formal framework to formalize diverse RL methods for learning optimum decision-making policies in a fully or partially observable environment. One of the important issues is the intelligent agent can dynamically make decisions based on actions, states, and rewards where the states refer to possible network configurations, and reward (or penalty) stands for feedback signal from the network (environment) that implies the agent's performance. The agent observes the network state at each time step t and acts on that network to transit from one state to another. Typically, an MDP is defined by a 5-tuple (S, A, P, γ, R) where S is a set of state (state space), A refers to a set of action (action space), P denotes the transition probability from current state s to the next state s' where they govern rules of state transitions and define our dynamics. The R notation stands for the reward function. The agent obtains an immediate reward by taking action a_t in state s_t. Unlike finite or episodic tasks, the total reward can be infinite in continuing tasks. The γ defines a reward discounting hyperparameter that results in significant deviations in the agent's performance where the future agent values rewards will turn to less and less. Indeed, it is a real-valued discount factor weighting to determine the importance of future rewards as a short-sighted or myopic agent ($\gamma = 0$), i.e., agent aims to maximize its current/immediate rewards, or far-sighted agent ($\gamma = 1$) that strives to accumulate long-term higher rewards. The γ is often chosen in $[0.95, 0.99]$. The total discounted rewards from time step t is given by

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \sum_{n=0}^{\infty} (\gamma^n R_{t+n+1}) \,. \tag{2.2}$$

We can derive useful recursive relation between rewards and subsequent time step, $G_t = R_{t+1} + \gamma G_{t+1}$ that it can attenuate computational complexity and memory requirements. However, there is one more piece we need to complete the MDP. The key term of MDP is the decision. Indeed, the way we make decisions for what actions to do in what states is called a policy which denotes with the symbol π that it is usually a probabilistic function as mapping states to actions.

Technically, the policy is not part of the MDP itself but part of the solution along with the value function. Unlike supervised and unsupervised learning, the agent in RL banks on reward signals to evaluate the effectiveness of actions and enhance the improvement of the policy. The excellence of each state or state-action pair can be determined by how large the future reward of the agent is. The state value function for the policy π is an explicit measure of how good the state is or how much reward to expect

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{n=0}^{\infty} (\gamma^n R_{t+n+1} | S_t = s) \right] , \tag{2.3}$$

and expectation value of the reward at time t is defined as the action value function

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{n=0}^{\infty} (\gamma^n R_{t+n+1} | S_t = s, A_t = a) \right] . \tag{2.4}$$

According to total expectation in reverse and nested expected value, we can define Bellman equation by the assumption of MDP

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V_\pi(s')]. \tag{2.5}$$

The above recursive relationship between each value function and the successor state's value function is considered the heart of RL to solve stochastic and non-deterministic search problems where agents encounter random events. In deterministic policy, there is just one action with

$\pi(a|s) = 1$, and the rest receive 0. Some algorithms specifically apply to only the state-value or action-value functions. Let suppose we have two policies π_1 and π_2 and π_1 is better than π_2, if expected return of π_1, $V_{\pi_1}(s)$ greater than or equal to the expected return of π_2, $V_{\pi_2}(s)$ for all states

$$\forall \quad s \in S, \quad if \quad V_{\pi_1}(s) \geq V_{\pi_2}(s) \quad \rightarrow \quad \pi_1 \geq \pi_2 \tag{2.6}$$

The optimal policy in RL is the best one for which no greater value function exists. The optimal value functions are optimal but optimal policies are not necessarily unique. Due to designing the RL program, there are two different problems we should try to solve. The finding $V(s)$ given a policy is called the prediction problem, and finding the optimal policy for a given environment refers to control problems. The prediction problem is easier than the control problem, and the job is single thing to find the value function based on the algorithm.

In the parlance of RL, the agent engages in exploration and exploitation, where exploitation or greed refers to taking the best-known action while taking sub-optimal action is called exploration (i.e., the way of exploring the environment). This process of exploration and exploitation enables the agent to refine its model and gradually find the true estimate of the value function. The control algorithms with a model-based (planning) approach constitute a set of distributed computations in the network to build a model of how the environment works, which is referred to the decision-making of agents in the control theory domain. The agents don't need to learn network dynamics to adopt this scheme, but assuming some network parameters and configurations (e.g., channel information) is necessary. In contrast, black-box network optimization/control works without a priori network model, namely model-free methods where it relies on a trial and error approach to learn the state transition probabilities to solve the Bellman equation and try to figure out a policy of how to behave optimally while the aim of this trade-off of the time is ease computational complexity and yield controller convergence. The tendency towards fulfilling fully automated operations and limited computational resources in wireless networks has aroused intensive research interest in working on model-free problems.

Q-learning is a model-free approach in the class of temporal difference (TD) learning algorithms where it has an online nature and updates the agents' estimate of value function at each time step. To update the value function concerning TD, we have

$$V(s_t) = V(s_t) + \alpha[R_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{2.7}$$

where α denotes exponentially smooth mean approach to deal with non-stationary distributions instead of calculating the sample mean inefficiently, and $R_{t+1} + \gamma V(s_{t+1})$ is the target of TD method and $V(S_t)$ refers to old estimate. Moreover, the quantity in the brackets $R_{t+1} + \gamma V(s_{t+1}) - V(S_t)$ is called TD error, and therefore we have squared error $E = [Target - Prediction]^2$ with pursuing a gradient descent approach. The aim is the prediction value to be closer to the target value. This approach that uses one estimate to update another estimate is called bootstrapping. The updated Q-value (quantifiable value for more lucrative action) in Q-learning as the off-policy method is given by

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[R_{t+1} + \gamma \max_a Q(s_{t+1}, a_{max}) - Q(s_t, a_t)]. \quad (2.8)$$

Q-learning alternates between policy evaluation and, thereby, policy improvement where select an action greedily concerning the current value estimate. Unlike on-policy methods in which one policy generates actions and updates the value function, Q-learning is an off-policy method that uses one policy to generate actions and another to update the value function. For example, Q-learning uses $\epsilon - greedy$ policy to choose an action while it updates the action-value function according to purely greedy action.

2.2.1 Deep Q-Learning (DQN)

The Q-learning works in a very simplistic environment. In practice, communication network problems have complicated system models with large and continuous state spaces that can have infinite states. Therefore, the tabular representation of the action-value function in the previous section leads to computational complexity and training time limitation problems. Unlike the tabular and non-parametric approach, RL is assisted with DNN in DRL to surmount the curse of dimensionality concerning inordinate large state spaces. Indeed, It is possible to model $Q(s, a)$ based on simple linear regression, but another new approach is the DNN that benefits from crafting inductive biases to overcome the curse of dimensionality effectively and thereby enables RL to scale the decision-making problems for intractable high-dimensional state and action spaces through approximate the Q-value function. The DQN takes the state as input and returns approximated Q-functions of all actions under the input state. In this regard, finding a convincing solution for the instability of function approximation techniques in RL-based agents is necessary. We parameterize the value function with linear regression or a neural network. Let define $V(s) = \theta^T s$ where s is a feature vector representing the state and in linear regression for calculating the value function prediction θ refers to the weights. According to the previous section, we apply gradient descent using squared error between the target and prediction. Instead of updating $V(s)$ directly, the goal is to update the parameters of V that according to the chain rule and multiply to the gradient of $V(s)$ we have

$$\theta \leftarrow \theta + \alpha[r + \gamma V(s') - V(s)]\frac{\partial V(s)}{\partial \theta} \tag{2.9}$$

and in linear regression, we have a simple update rule

$$\theta \leftarrow \theta + \alpha[r + \gamma V(s') - V(s)]s \tag{2.10}$$

with the same logic for $Q-updating$, we consider s as input, and for infinite action, the number of outputs is equal to the number of possible actions

$$\theta \leftarrow \theta + \alpha[r + \gamma \max_a Q(s', a_{max}) - Q(s,a)]\frac{\partial Q(s,a)}{\partial \theta}. \tag{2.11}$$

This approach is unable to support infinite or continuous action space. However, DRL can solve problems with gradient-free approaches.

2.2.2 Policy Gradient

Another approach to solving the control problem is called the policy gradient method, where we also parameterize the policy. Q-learning is a value-based method where the policy is just to take the action which gives the best value

$$a^* = \arg\max_a Q(s,a). \tag{2.12}$$

In this case, the return is a probability distribution and determines what is the best action for a given state $\pi(a|s)$. Indeed, we just sample from $\pi(a|s)$ because the policy has probabilistic nature, and there is no need to pursue the strategies such as $\epsilon-greedy$. It makes more flexibility and chance for exploration with sampling from policy distribution and more control over the amount of randomness. In deep Q-learning, the neural network has one output for each action, and thereby it can support infinite or discrete action spaces. In contrast, the output of policy networks is a probability distribution. Unlike the value-based method, in the policy-based method, instead of directly considering objective function, the main objective is the integral of gradient called policy gradient. The aim is to maximize the sum of total rewards, but there is no functional dependence on the policy parameters θ, and thereby there is no formula where reward depends on neural network weights. We pursue this approach to improve policy gradients while reducing variance with a baseline to increase stability because we don't care about the absolute reward. Due to calculating $V(s)$, we use another neural network.

2.2.3 Actor-Critic

The actor-critic method is a DRL algorithm that blends value-based and policy-based approaches to enhance decision-making in a specific environment. It comprises two components: the actor, which chooses actions based on the current policy, and the critic, which evaluates the value of state and action pairs. This on-policy algorithm learns directly from the executed policy, as the actor takes actions based on the current policy and receives rewards as feedback. The critic utilizes this feedback to update its estimate of the value function, which the actor then uses to refine its policy. Various variations of the actor-critic method exist, which differ in how they update the policy and value function and incorporate DNNs for function approximation. Actor-Critic methods balance exploration and exploitation, handle continuous action spaces, and converge more quickly than other methods. The TD3 [33] is one of the main actor-critic algorithms that consists of a set of techniques, and we leverage them in the proposed methods in **Chapter 3**.

The basic idea behind policy-based algorithms is to adjust the parameters ϕ of the policy in the direction of the performance gradient $\nabla_\phi J(\pi_\phi)$. The fundamental result underlying these algorithms is the policy gradient theorem [34]: $\nabla_\phi J(\pi_\phi) = \int_S p_\pi(s) \int_A \nabla_\phi \pi_\phi(a|s) Q^\pi(s,a) dads$.

We can parameterize policy like value function, and the goal is to find the optimal policy π_ϕ where ϕ includes updating the weight of the policy. The expected return can be approximated in many ways. We calculate the gradient of expected return according to parameters of ϕ as $\nabla_\phi J(\phi)$. We use gradient ascent as opposed to gradient descent for updating the parameters, $\phi_{t+1} = \phi_t + \alpha \nabla_\phi J(\pi_\phi)|\phi_t$.

In the actor-Critic method, we have two models that work concurrently where the actor is a policy taking the state as input and delivering actions as output, while the critic takes states and actions concatenated together and return the Q-value and a policy that can be updated

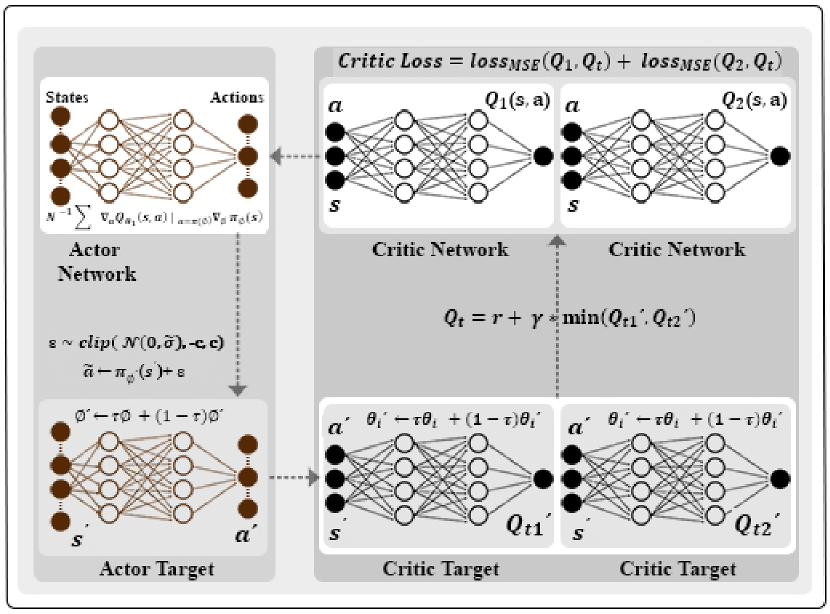

Critic Loss = $loss_{MSE}(Q_1, Q_t) + loss_{MSE}(Q_2, Q_t)$
States
Actions
$N^{-1}\sum \nabla_a Q_{\theta_1}(s,a)|_{a=\pi(\phi)} \nabla_\phi \pi_\phi(s)$
Actor Network
$Q_1(s, a)$
$Q_2(s, a)$
Critic Network
Critic Network
$Q_t = r + \gamma * \min(Q_{t1}', Q_{t2}')$
$\varepsilon \sim clip(\mathcal{N}(0,\tilde{\sigma}), -c, c)$
$\tilde{a} \leftarrow \pi_{\phi'}(s') + \varepsilon$
$\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
Q_{t1}'
Q_{t2}'
Actor Target
Critic Target
Critic Target

FIGURE 2.3: Training flow process between different DNNs.

through the deterministic policy gradient[33], $\nabla_\phi J(\phi) = \mathbb{E}_{s\sim p_\pi}\left[\nabla_a Q^\pi(s,a)|_{a=\pi(s)}\nabla_\phi \pi_\phi(s)\right]$. where $Q^\pi(s,a) = \mathbb{E}_{s_f\sim p_\pi \sim a_f \sim \pi}\left[R_t|s,a\right]$ is known as value function or critic.

Initially, we should store random experiences in the buffer β. In the other words, we store (s_t, a_t, r_t, s_{t+1}) to train Deep Q-Network. We take a random batch B, and for all transitions $(s_{t_B}, a_{t_B}, r_{t_B}, s_{t_B+1})$ of β, the predictions are $Q(s_{t_B}, a_{t_B})$ and the targets consider as optimal immediate return that are the exactly first part of temporal difference learning (TD) error as $R(s_{t_B}, a_{t_B}) + \gamma \max_a(Q_{(s_{t_B+1}, a)})$. Over the whole batch B, we calculate the loss between predictions and the targets in the batch B. Another target network is used instead of using Q-network to calculate the target to fulfill more stability for the learning algorithm. As shown in Fig. 2.3, the TD3 is based on the actor-critic model that leverages three tricks to improve the algorithm:

1. **Clipped double Q-learning with pair of critic networks:** We denote two DNNs as two actor networks by ϕ as the actor network and ϕ' as the actor target. In addition, we create two pairs of critic networks and denote them by θ_1,θ_2 for parameterization of value network and θ_1',θ_2' as critic targets. Indeed, two learnings happen simultaneously, namely, Q-learning and Policy learning, and they address approximation error, reduce the bias, and find the highest Q-value. This was inspired by the technique seen in [35] as Double-Q Learning. For each element and transition of the batch, the actor target plays a' based on s' while we add Gaussian noise to this a'. The critic targets takes the couple (s', a')

and return two Q-values Q'_{t1} and Q'_{t2} as output. Then, the $(\min Q'_{t1}, Q'_{t2})$ is considered as an approximated value for critic networks. In [36] has proposed using the target network as one of the value estimates. Given that we calculate the final target of the two value networks, we have the following:

$$Q_t = r + \gamma * \min(Q'_{t1}, Q'_{t2}) \tag{2.13}$$

then the two critic networks return two Q-values as Q_1(s,a) and Q_2(s,a). Next, we calculate the loss based on two critic networks and with mean squared error (MSE). To minimize the loss over iterations via the back-propagation technique, we use an efficient optimizer called Adaptive Moment Estimation (Adam) [37] in our code:

$$L = l_{MSE}(Q_1, Q_t) + l_{MSE}(Q_2, Q_t) \tag{2.14a}$$

$$\nabla_\phi J(\phi) = N^{-1} \sum \left[\nabla_a Q_{\theta_1}(s,a)|_{a=\pi(\phi)} \nabla_\phi \pi_\phi(s) \right] \tag{2.14b}$$

In the next step, we explain how we update the target networks.

2. **Delayed policy updates and target networks:** The main idea is to update the policy network less frequently than the value network since we need to estimate the value with lower variance [38]. Polyak Averaging gives the update rule, so we update parameters by:

$$\theta'_f \longleftarrow \tau\theta_f + (1-\tau)\theta'_f \tag{2.15a}$$

$$\phi' \longleftarrow \tau\phi + (1-\tau)\phi' \tag{2.15b}$$

where $\tau \leq 1$ is a hyperparameter to tune the updating speed.

3. **Target policy smoothing and noise regularisation:** When updating the critic, a learning target using a deterministic policy is highly susceptible to inaccuracies induced by function approximation error, increasing the variance of the target. This induced variance can be reduced through regularization [33] to be sure of the exploration of all possible continuous parameters. We add Gaussian noise to the next action a' to prevent two large actions played and disturbing the state of the environment:

$$\tilde{a} \longleftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim clip(\mathcal{N}(0, \tilde{\sigma}), -c, c) \tag{2.16}$$

, where the noise ϵ is sampled from a Gaussian distribution with zero and certain standard deviation and clipped in a certain range of value between $-c$ and c to encourage exploration. Due to avoid the error of using the impossible value of actions, we clip the added noise to the range of possible actions (min_action, max_action). The TD3-based NS method is summarized in Algorithm 1.

Algorithm 1: TD3-based network slicing with OpenAI Gym

```
Initialize actor network $\phi$ and critic networks $\theta_1$, $\theta_2$
Initialize (copy parameters) target networks $\phi'$, $\theta'_1$, $\theta'_2$
Initialize replay buffer $\beta$
Import network slicing environment (‘smartech–v0’)
while $t$ ¡ $max\_timesteps$ do
    if $t$ ¡ $start\_timesteps$ then
        $a$ = env.action_space.sample()
    else
        $a \longleftarrow \pi_{\phi}(s) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma)$
    end
    next_state, reward, done, _ = env.step($a$)
    store the new transition $(s_t, a_t, r_t, s_{t+1})$ into $\beta$
    if $t \geq start\_timesteps$ then
        sample batch of transitions $(s_{t_B}, a_{t_B}, r_{t_B}, s_{t_B+1})$
        $\tilde{a} \longleftarrow \pi_{\phi}\prime(s') + \epsilon, \quad \epsilon \sim clip(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
        $Q_t = r + \gamma * \min(Q'_{t1}, Q'_{t2})$
        $L = l_{MSE}(Q_1, Q_t) + l_{MSE}(Q_2, Q_t)$
        $\theta_f \longleftarrow argmin_{\theta_f} N^{-1} \sum (L - Q_{\theta_f(s,a)})^2$
        if $t\%policy\_freq == 0$ then
            $\nabla_{\phi} J(\phi) = N^{-1} \sum \left[ \nabla_a Q_{\theta_1}(s,a)|_{a=\pi(\phi)} \nabla_{\phi} \pi_{\phi}(s) \right]$
            $\theta'_f \longleftarrow \tau\theta_f + (1-\tau)\theta'_f$
            $\phi' \longleftarrow \tau\phi + (1-\tau)\phi'$
        end
    end
    if $done$ then
        obs, done = env.reset(), False
    end
    t=t+1
end
```

2.2.4 Lifelong Reinforcement Learning

Lifelong deep reinforcement learning involves employing reinforcement learning algorithms to enable an agent to continuously learn and adapt to new tasks over its entire lifespan. Unlike conventional reinforcement learning, which focuses on training an agent for a specific task, lifelong reinforcement learning equips the agent to explore new environments and acquire new skills while leveraging its existing knowledge to perform well in previously learned tasks. A significant challenge in lifelong reinforcement learning is balancing exploration and exploitation. The agent needs to continually explore new tasks and environments while utilizing existing knowledge to excel in tasks it has already learned. In the context of dynamic environments like network slicing, lifelong reinforcement learning faces various challenges related to the allocation of RAN resources. These challenges encompass dealing with sparse rewards, efficiently transferring knowledge from prior tasks to new ones, optimizing the use of computational resources for faster learning, the ability to prioritize important or relevant experiences or tasks, and diagnosing of undesired behaviors. Moreover, the limited availability of a large number of samples further amplifies the complexity of the problem. Overcoming these obstacles is essential for enhancing the effectiveness and efficiency of lifelong reinforcement learning in the context of RAN slicing. As shown in Fig. 2.4, we proposed innovative and complementary approaches to address these challenges and discussed them in subsequent sections.

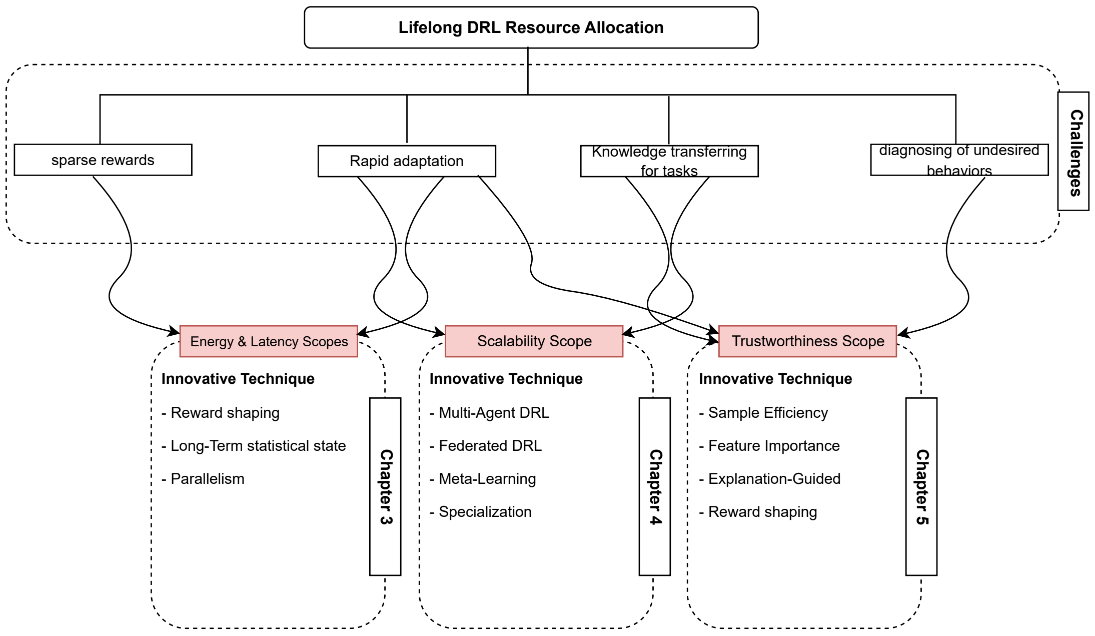


FIGURE 2.4: Lifelong DRL resource allocation challenges and solutions.

2.2.4.1 Statistical and Parallel

Reward shaping is used in lifelong DRL by modifying the reward function that the agent receives during training to improve the learning process and speed up convergence to optimal policies. The traditional approach is to provide a scalar reward signal based on the state of the environment and actions taken by the agent. However, this signal may need to provide more guidance, particularly when it is sparse or delayed. Reward shaping involves introducing additional rewards into the environment to guide the agent toward better behavior. These rewards can be based on various metrics relevant to the task, such as heuristics or expert knowledge. For instance, for an agent in network slicing, the standard reward may only be given for reaching the objective, but reward shaping may involve: i) Adding rewards for avoiding SLA degradation, ii) Exploring new areas, and iii) Moving closer to the goal. To improve learning performance and accelerate convergence to optimal policies, reward shaping techniques can be used in combination with lifelong DRL algorithms, such as DQN, policy gradient, and methods. However, designing the reward-shaping functions carefully is crucial to prevent unintended biases and distortions in the learning problem. The parallelization involves executing multiple computations simultaneously, leading to a substantial acceleration of lifelong DRL training and inference processes. The agents continuously acquire knowledge from various tasks by training on multiple experiences concurrently. This efficient utilization of computational resources enables faster learning and better retention of previous knowledge, facilitating quick adaptation to new tasks without forgetting the knowledge gained from past experiences.

In **Chapter 3**, we proposed a reward shaping approach specifically based on statistical latency states in the network. Statistical lifelong DRL aims to create algorithms that can use the experience to make the best decisions in an environment that is not entirely predictable. This requires the algorithms to understand the inherent uncertainty in the data and to develop models of the relationship between the agent's actions and the rewards it receives. Additionally, it involves modeling the distribution of data that arises from the environment. Furthermore, we introduced a parallelized actor-learner framework that harnesses the power of distributed computing or multi-core processing to train the agent concurrently on multiple tasks or experiences. This approach facilitates faster and more efficient learning in lifelong DRL.

2.2.4.2 Specialization of Multi-Agent and Federated Learning

Multi-agent DRL addresses scenarios where multiple agents interact with both the environment and each other, learning individual policies to enhance their overall collective performance. These agents might engage in competition, cooperation, or even both within a complex environment. FDRL is an ML approach incorporating three fundamental techniques: FL, DL, and RL. In FDRL, DRL trains agents to make decisions that optimize a shared reward signal, usually a measure of global system performance. FDRL offers several benefits compared to conventional

DRL methods. For instance, it allows agents to learn from diverse datasets, leading to better generalization and improved performance. Furthermore, FDRL addresses data privacy and distribution issues, as agents only require access to their own local data and do not need to share it with other learners.

In this regard, we propose a framework based on Multi-agent DRL and FDRL in **Chapter** 4. In this framework, each agent trains a local DRL model and shares its experience, in the form of model hyperparameters, with the entities belonging to the corresponding federation layer within the FDRL context. This iterative training approach allows each federation layer to combine the collected knowledge of individual agents into a global updated model, which is usually stored in a cloud platform or a nearby edge platform to facilitate faster feedback. To enhance efficiency and avoid communication overhead, we allow the federation layer to collect and share the local models (and updated models) only every $\hat{T}$ decision interval, which we refer to as a *federation episode*. Different strategies can be used to derive the global federated model, each implementing a predefined federation strategy function $f_{strategy}(\cdot)$. For example, the *Average* federation strategy computes the collective federation model for the next time interval as the simple average of the incoming model weights from all the agents. Aggregated mobile traffic demands exhibit repetitive spatio-temporal trends due to human activities. It is necessary to characterize these processes comprehensively to achieve more accurate forecasting of network utilization and enable efficient resource allocation planning. However, more than leveraging the base stations' geographical locations and spatial proximity is required to obtain a comprehensive view of traffic demands, as the land usage of the slice resources may differ even within base stations belonging to the same geographical area. This introduces an additional issue in our framework, as only some of the federated agents should exchange knowledge with each other, nor should this be restricted to only nearby entities. To address this issue, we propose a clustering algorithm based on network monitoring traces and their similarity to guide the definition of DA subsets.

The integration of Multi-agent learning and federated learning in lifelong DRL enables agents to collectively enhance their performance in their respective environments. Leveraging the iterative process of federated learning, agents can continuously update their policies and exchange knowledge across various domains. As a result, the agents become more adaptive and skilled at generalizing to new tasks or domains, exemplifying the essence of meta-learning. This collaborative approach fosters improved cooperation and knowledge sharing among agents, ultimately leading to enhanced performance in complex and diverse environments.

2.2.4.3 Explanation-Guided

The concept of XAI involves creating artificial intelligence systems that are transparent, interpretable, and capable of explaining their decisions and actions. Unlike traditional AI systems that often operate with opaque algorithms and models that are difficult for humans to comprehend, XAI seeks to address the growing concerns around accountability and transparency of AI systems that are increasingly integrated into our daily lives. XAI is crucial in high-stakes applications, such as healthcare, finance, or legal systems, where the lack of transparency can lead to distrust and potential errors. XAI techniques enable humans to better comprehend AI systems by visualizing how decisions are made, providing scores on feature importance, and generating natural language explanations of the decision-making process. This allows humans to identify potential biases or errors in the system, leading to greater trust and confidence in the technology.

In this intent, XRL is considered a trustworthy and responsible ML approach that can be integrated into the AI lifecycle in network slicing environment. Lifelong DRL agent interacts with the network to generate the dataset on the fly. This evaluative and feedback-based learning method optimizes long-term rewards to solve complex control tasks in network slicing. The recent advancements in deep learning have helped DRL overcome the challenges of dealing with large state spaces, but the opacity of DNNs raises concerns regarding the reliability and security issues for the trained agents and predicted results. Additionally, the complex relation between state and action selection can hinder the effectiveness of RL solutions in automated network slicing. To address these issues, we propose an interpretability approach based on EGL in the training phase of our proposed SliceOps agent in **Chapter 5**. The interpretability solution is categorized into reward, state, action, model, and task, which should be examined before deployment. By utilizing XAI, we can extract more relevant state-action pairs, which can help make prompt and robust decisions, especially in time-sensitive applications such as URLLC.

Through the integration of reward shaping, feature importance, and attention mechanisms, the lifelong DRL agent gains the ability to prioritize learning from specific experiences or tasks based on their significance and relevance. This targeted approach helps overcome catastrophic forgetting, where learning new tasks can erase knowledge of previously acquired ones. The agent effectively retains knowledge across tasks by selectively attending to task-specific information. Moreover, these explanations play a crucial role in diagnosing potential biases or undesired behaviors in the agent's decision-making process, thereby enhancing trust in lifelong DRL.

Chapter 3

Energy-Aware and Latency-Aware Zero-Touch Network Slicing

This chapter begins by showcasing how network slicing allows MNOs to manage energy usage with precision. Next, we introduce a DRL framework that emphasizes network slicing solutions that prioritize low-latency and high-capacity services for end-users in the next generation of mobile networks. To this end, we propose a novel KP-based MANO framework that leverages recent network slicing technologies, named KB5G. Our focus is on algorithmic innovation and AI in KB5G. We apply a continuous model-free DRL method to minimize energy consumption and VNF instantiation costs. Additionally, we present a new Actor-Critic-based network slicing method, known as twin-delayed double-Q soft Actor-Critic (TDSAC), to stabilize learning. Specifically, TDSAC enables CU to learn continuously to accumulate the knowledge learned in the past to minimize future network slicing costs. Finally, we present numerical results to showcase the gain of the adopted approach and verify the performance in terms of energy consumption, CPU utilization, and time efficiency. Furthermore, proposed a novel model-free DRL framework, called $CS\text{-}AC$ that enables a scalable and farsighted slice performance management in a 6G-like RAN scenario that is built upon mobile MEC and mMIMO. In this regard, the proposed CS-AC targets the optimization of the latency cost under a long-term statistical SLA. In particular, we consider the *Q-th delay percentile* SLA metric and enforce some slice-specific preset constraints on it. Moreover, we propose a developed variant of SAC to implement distributed learners with less hyperparameter sensitivity. Finally, we present numerical results to showcase the gain of the adopted approach on our built OpenAI-based network slicing environment and verify the performance in terms of latency, SLA Q-th percentile, and time efficiency.

3.1 Energy Management in Network Slicing

The management of energy in network slicing involves optimizing the energy consumption of the physical infrastructure while guaranteeing the performance and QoS of the different virtual networks. Several techniques can be utilized to achieve energy management in network slicing, including efficient resource allocation, dynamic scaling, energy-efficient hardware, energy-aware routing, and traffic management. Efficient resource allocation involves mapping virtual resources to physical resources to minimize energy consumption. Dynamic scaling allows for resources to be added or removed based on the traffic load of the virtual network, thereby avoiding over-provisioning of resources that leads to energy wastage.

Furthermore, energy-efficient hardware should be utilized in the physical infrastructure for network slicing, such as low-power processors, reducing cooling requirements, and using renewable energy sources. Energy-aware routing algorithms should be designed to consider energy consumption, which can reduce overall energy consumption by routing traffic through paths that consume less energy. Traffic management techniques, such as reducing unnecessary traffic or using compression techniques, can also reduce energy consumption. Network operators must adopt a comprehensive approach to manage energy in network slicing that considers the various layers of the network architecture. By implementing these techniques, network operators can minimize energy consumption while maintaining the virtual networks' performance and QoS. In the following, we highlight the efficient resource allocation solution.

To automate network slicing orchestration, ZSM framework reference architecture [39] has been designed by ETSI, but the closed-loop operation of its building blocks is still an open research problem to fulfill an efficient and robust zero-touch management. Indeed, we still need a KP that plays the role of a pervasive system within the network by building and maintaining high-level models of what the network is supposed to do, in order to provide services and advice to other elements of the network [40]-[41]. Specifically, the quest of automation and optimal control in dynamic telecommunication environments has aroused intensive research on the applications of DRL. The DRL can provide a promising technique to be incorporated in network slicing and solve the control and optimization issues, specially in energy management.

3.1.1 Knowledge-Based Slicing

As depicted in Fig. 3.1, the KP encompasses ML and intelligent decisions to handle knowledge discovery, data analysis, and optimization. In what follows, we elaborate on the steps in KB5G:

3.1.1.1 Network Twin → Analytics Platform

The analytics platform is gathering enough information to offer a complete view of the network (slicing environment) and provide current and historical data for feeding learning algorithms. This data is categorized into two types, namely, users' data and operators' data, where both can be either local data or global data. This platform can rely on protocols and functions, such as NETCONF[1] and NWDAF [42].

3.1.1.2 Analytics Platform → ML → Intelligent Decision

Cloud computing platforms utilize the collected data to feed learning algorithms for knowledge discovery, data analysis, optimization, and generally manage and control the network to facilitate inferencing. The data analysis represents hidden patterns in big data and can predict and model the future behavior of the network. The KB5G benefits from some indicative abilities of intelligent behavior like learning from experience.

3.1.1.3 Intelligent Decision → SDN Controller(s)

In SDN, the northbound APIs present an abstraction of network functions with a programmable interface to dictate the behavior of the network at the top of the SDN stack. Using declarative languages for the SDN northbound interface and translating intelligent decisions to specific control directives is an open research question yet. The SDN controller receives the declarative primitives through its northbound interface and then renders the intent-driven language into specific imperative control actions [41].

[1]https://wiki.onap.org/display/DW/5G+-+Configuration+with+NETCONF

3.1.1.4 SDN Controller(s) $\rightarrow$ Network Twin

Due to robustness issues, we can consider the distributed control logic [43]. The distributed SDN consists of multiple interconnected network domains. In network slicing, we have SDR for C-RAN, transport controllers, and NFVO for cloud-native core. The SDN concept can be applied in the KB5G through PFCP instead of OpenFlow-enabled protocol. Indeed, OpenFlow does not support all aspects of QoS issues and it is also packet-based, while PFCP is session-based [44]-[45].

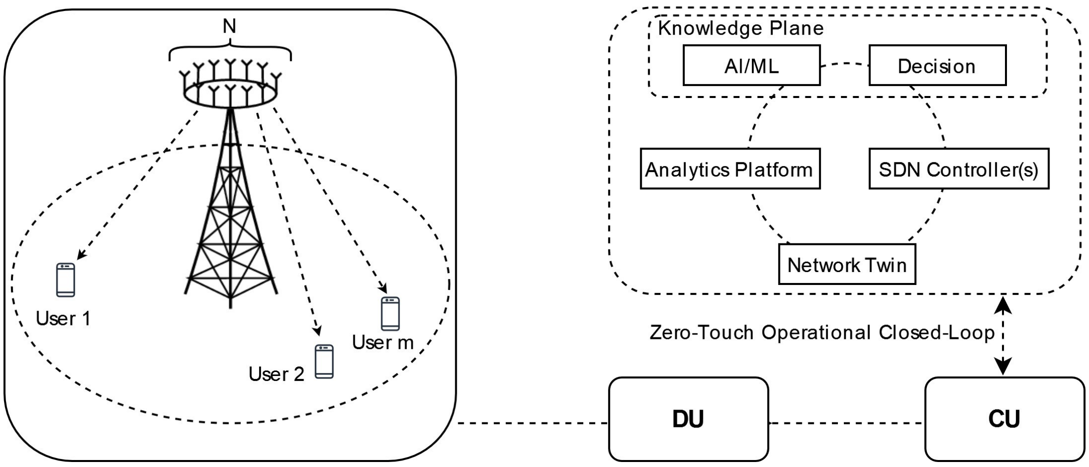


FIGURE 3.1: Proposed zero-touch KB5G network slicing.

3.1.2 Energy-Aware Network slicing Setup

Fig. 3.1 shows the considered C-RAN CU-DU split-based network architecture. A total of N APs are covering M single-antenna users in a downlink setup, and are connected to CU hosting control agents and running as a set of VNFs of the same type. We define $L \in \mathbb{N}$ as the number of slices in the network, and assume that the MNO collects the free and unused resources from the tenants and allocates them to the needy slices in a periodic fashion to avoid over-heading. A maximum of $X \in \mathbb{N}$ VNFs can be deployed on top of the cloud, endowed with $Z\,(z = 1, \ldots, Z)$ active CPUs having a processing capability of P_z MOPTS [46]. Let us denote $\mathbf{h}_m = [h_{1,m}, h_{2,m}, ..., h_{N,M}]^H \in \mathbb{C}^{N\times 1}$ as vector of channel gains from the N APs to the M users, where $(\cdot)^H$ is the conjugate transpose and $\mathbb{C}$ represents the complex set. Moreover, we consider the optimal beamforming vector $\mathbf{v}_m = [v_{1,m}, v_{2,m}, ..., v_{N,M}]^H \in \mathbb{C}^{N\times 1}$ associated with user m and whose expression is given by [47] as $\mathbf{v}_m = \sqrt{p_m}\frac{\left(\mathbf{I}_N+\sum_{j=1}^{M}\frac{1}{\hat{\sigma}^2}\mathbf{h}_j\mathbf{h}_j^H\right)^{-1}\mathbf{h}_m}{\left(\mathbf{I}_N+\sum_{j=1}^{M}\frac{1}{\hat{\sigma}^2}\mathbf{h}_j\mathbf{h}_j^H\right)^{-1}\mathbf{h}_m}$, where p_m is beamforming power, $\mathbf{I}_N$ denotes the $N \times N$ identity matrix and $\hat{\sigma}^2$ is the noise variance. Therefore we model the received signal $r_m \in \mathbb{C}$ at user m as $r_m = \mathbf{h}_m^H \mathbf{v}_m s_m + \sum_{j \neq m}^{M} \mathbf{h}_m^H \mathbf{v}_j s_j + n_m$, where $s_j \in \mathbb{C}$ is data signal to user m and received noise n_m is the white Gaussian noise with zero mean and variance σ^2. Let define the following channel model [48], $h_{n,m} = 10^{-L^*(d_{n,m})/20} \sqrt{\vartheta_{n,m} \Theta_{n,m}} g_{n,m}$, where $L^*(d_{n,m})$ denotes the path loss with a distance of $d_{n,m}$. Moreover, $\vartheta_{n,m}$ is the antenna gain, $\Theta_{n,m}$ is the shadowing coefficient and $g_{n,m}$ is the small-scale fading coefficient. Then the achievable rate for user m is given by $R_m = \log(1 + \underbrace{\frac{\mathbf{h}_m^H \mathbf{v}_m{}^2}{\sum_{j \neq m}^{M} \mathbf{h}_m^H \mathbf{v}_j{}^2 + \sigma^2}}_{SINR_m})$, where $SINR$ stands for signal-to-interference-plus-noise ratio. Let define $\mathbf{G}_n \in \mathbb{C}^{1 \times N}$ as $\mathbf{G}_n = [\underbrace{0,...,0,}_{\text{n-1}} 1, 0, ..., 0], \quad n > 0$. Then the power consumption for AP n serving all potential m users can be written as [49],

$$\mathcal{E}_w^n = \sum_{m=1}^{M} \mathbf{v}_m^H \mathbf{G}_n^H \mathbf{G}_n \mathbf{v}_m. \tag{3.1}$$

Note that the circuit power and fronthaul power consumption can be neglected because they are small compared with transmit power. Moreover, we consider energy consumption incurred by the running processors. The computing resource model follows that in [50]. We suppose Δ_m is a fraction of a CPU core,

$$\Delta_m = \underbrace{\hat{\theta} R_m + C_0}_{\text{baseband}} + \underbrace{\delta \sum_{n=1}^{N} \Upsilon \mathbf{v}_{n,m}}_{\text{transmission}}, \tag{3.2}$$

where baseband processing refers to coding, modulation and FFT. Furthermore, $\hat{\theta}$ is experimental value, C_0 denotes constant complexity for FFT, $\delta > 0$ is slope parameter and $\Upsilon(\cdot)$ denotes the step function. The energy consumed by processor z in Watts is given by ιP_z^3, where ι parameter denotes the processor structure [51]. We define constant value ψ for VNF deployment and then compute energy consumption in CU with respect to total Δ_m. Therefore, the whole energy consumption in network is given by

$$\mathcal{E}_{Net}^{(t)} = \underbrace{\sum_{z=1}^{Z} \iota P_z^3 + \sum_{x=1}^{X} \psi_x}_{\text{baseband}} + \underbrace{\sum_{n=1}^{N} \sum_{m=1}^{M} \mathbf{v}_m^H \mathbf{G}_n^H \mathbf{G}_n \mathbf{v}_m}_{\text{transmission}}. \tag{3.3}$$

The objective is to minimize the overall network cost with respect to the incurred computing resources and energy consumption at each decision time step, and thereby the continuous model-free DRL optimization is given by

$$\begin{aligned}
\min \quad & \frac{1}{M^{(t)}}(\mathcal{E}_{Net}^{(t)}) && (3.4a)\\
\text{subject to} \quad & p_m \leq \mathcal{P}_{max}, \quad m \in M, && (3.4b)\\
& SINR_m \geq SINR_{th,l}, \quad m \in M, l \in L, && (3.4c)\\
& \Delta_m \leq \Delta_{th,l}, \quad m \in M, l \in L. && (3.4d)
\end{aligned}$$

Note that higher traffic can induce higher costs. We consider the number of users ($M^{(t)}$) at each decision time step to normalize and balance network cost with respect to heavy and low traffic periods. Moreover, $\mathcal{P}_{max}$ is an experimental value while $SINR_{th,l}$ and also $\Delta_{th,l}$ are predefined thresholds for slice l.

3.1.2.1 MDP Problem Formulation

Problem (3.4) can be formulated from an MDP perspective, where the objective is to achieve lower total costs under user QoS, predefined thresholds, and computing resource constraints. This reflects the correlation between energy consumption and CPU usage, where beamforming power p_m for each user affects SINR, that in turn influences computing resource consumption. The MDP can be solved by finding an optimal policy for selecting the best actions with respect to beamforming power and computing resource allocation. Indeed, the MDP is mathematically characterized by a 5-tuple (S, A, P, γ, R) where S is the state space, A refers to the action space, P denotes the transition probability from current state s to the next state s', γ is the reward discounting hyperparameter, and R stands for the reward function. The state value function for the policy π is an explicit measure of how much reward to expect, $V_\pi(s) = \mathbb{E}_\pi\left[\sum_{n=0}^{\infty}(\gamma^n R_{t+n+1}|S_t = s)\right]$ and is defined as action-value function (referred as Q-Function) $Q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{n=0}^{\infty}(\gamma^n R_{t+n+1}|S_t = s, A_t = a)\right]$. The concerned MPD problem is defined as follows:

-State space: The state space provides input data about possible network configurations for agent via interaction with network slicing environment parameters. In our scenario, the state transits to the next state at each time step t by $S^{(t)} = \{S_1^{(t)}, S_2^{(t)}, S_3^{(t)}, S_4^{(t)}\}$, where ($S_1^{(t)}$) is the number of arrival requests for each slice corresponding to each VNF, ($S_2^{(t)}$) refers to computing resources allocated to each VNF, ($S_3^{(t)}$) shows energy status, ($S_4^{(t)}$) refers to number of users being served in each slice.

-Action space: We consider vertical scaling for computing resources consists of either scaling up or down procedure. The CU selects continuous value action with respect to traffic fluctuations to learn how to scale up/down a VNF properly and thereby according to time step, we have $\mathcal{A}_{CPU}^{(t)} \in \{o|o \in \mathbb{R}, -\mathcal{C}_{Net}^{(t)} \leq o \leq \mathcal{C}_{Z}^{(t)} - \mathcal{C}_{Net}^{(t)}\}$, where $\mathcal{A}_{CPU}^{(t)}$ is vertical scaling action for CPU resources, $\mathcal{C}_{Z}^{(t)}$ is CPU capacity and $\mathcal{C}_{Net}^{(t)}$ denotes the total CPU requirements. Moreover, we assign beamforming power according to SINR constraint, $\mathcal{A}_{P}^{(t)} \in \{o|o \in \mathbb{R}, 0 \leq o \leq \mathcal{P}_{max}^{(t)}\}$ the complete continuous multi-action space is given by $\mathcal{A} \triangleq \mathcal{A}_{CPU}^{(t)} \cup \mathcal{A}_{P}^{(t)}$.

-Reward: Due to to guide the agent for learning good results, we define $\chi_{T}^{(t)}$ as constraints function that is given by the following piecewise function,

$$\chi_{T}^{(t)} = \begin{cases} \mathbf{0}, & \textbf{if} \quad SINR_m \geq SINR_{th,l} \quad \textbf{and} \quad \Delta_m \leq \Delta_{th,l} \\ \mathbf{1}, & \textbf{otherwise} \end{cases} \tag{3.5}$$

Accordingly, we define the penalty function as $\varepsilon_{m}^{(t)} = -\varrho_m 1\left(\chi_{T}^{(t)} = 1\right)$, where ϱ_m is the penalty coefficient for not fulfilling constraints and $\varrho_{m}^{SINR} > \varrho_{m}^{CPU}$. The objective is maximize the total return $R^{(t)}$,

$$R^{(t)} = \frac{\frac{1}{\frac{1}{M^{(t)}}(\mathcal{E}_{Net}^{(t)})} + \sum_{m=1}^{M} \varepsilon_{m}^{(t)}}{\hat{\omega}} \tag{3.6}$$

where $\hat{\omega}$ is a hyperparameter that guarantees $R^{(t)} \in [-1, 1]$. DNN uses this return function for training while satisfying the main goal of overall objective function.

3.1.2.2 Proposed Soft Actor-Critic

Actor-Critic methods are a combination of policy optimization and Q-Learning. We use $\rho_\pi(s_t)$ and $\rho_\pi(s_t, a_t)$ to denote the state and state-action distribution respectively that induced by policy π in network slicing environment $\pi(a_t|s_t)$. Unlike the DDPG [11] and TD3 [33], the TDSAC benefits from stochastic policy gradient. The basic idea behind policy-based algorithms is to adjust the parameters ϕ of the policy in the direction of the performance gradient $\nabla_\phi J(\pi_\phi)$ concerning the policy gradient theorem [52]. The goal in standard RL is to learn a policy $\pi(a_t, s_t)$, which maximizes the expected sum of rewards. We consider a more general entropy-augmented objective concerning stochastic policies approach where augments the objective with a policy entropy term $\mathcal{H}$ over $\rho_\pi(s_t)$. Maximum entropy RL can optimize the expected return and entropy of the policy, thereby improving the policy's exploration efficiency. The objective for finite-horizon MDPs is given by, $J_\pi = \mathbb{E}\left[\sum_{i=t}^{T} \gamma^{i-t}[r_i + \alpha\mathcal{H}(\pi(\cdot|s_i))]\right]$. As we mentioned before, γ is the discount factor. The temperature parameter α determines the relative importance of the $\mathcal{H}$ against the reward, thereby handling the stochasticity of the optimal policy. Maximum entropy RL gradually proceeds toward the conventional RL $\alpha \rightarrow 0$.

Let us define entropy-augmented accumulated return or soft return as $G_t = \sum_{i=t}^{T} \gamma^{i-t}[r_i - \alpha \log \pi(a_i|s_i)]$. Then we can define soft Q-value with respect to policy π as $Q_\pi(s_t, a_t) = \mathbb{E}[r] + \gamma \mathbb{E}[G_{t+1}]$. We use soft policy iteration method for learning optimal maximum entropy policies that alternates between soft policy evaluation and soft policy improvement. In the soft policy iteration, we wish to compute the value of a policy π according to the maximum entropy objective [53], thus the soft Q-value can be learned by applying a Bellman operator $\mathcal{T}^\pi$ under policy π repeatedly as, $\mathcal{T}^\pi Q_\pi(s, a) = \mathbb{E}[r] + \gamma \mathbb{E}[Q_\pi(s', a') - \alpha \log \pi(a'|s')]$, the optimality and convergence of soft policy iteration have been verified in [54]. The main goal is to find a new policy π_{new} that is better than the current policy π_{old} and thereby $J_{\pi_{new}} \geq J_{\pi_{old}}$. This particular choice of update can be accomplished by maximizing the entropy-augmented objective (J_π) with respect to soft Q-value, $\pi_{new} = \arg\max_{\pi} \mathbb{E}[Q_{\pi_{old}}(s, a) - \alpha \log \pi(a|s)]$.

Our method (TDSAC) incorporates the following key approaches. The main aim is to stabilize the learning and improve time efficiency while mitigating very high sample complexity and meticulous hyperparameter tuning: 1) The (clipped) double Q-learning technique [33] parameterizes critic networks and critic targets by θ_1, θ_2 and θ'_1,θ'_2 respectively. Unlike the TD3 in TDSAC, the next state-actions used in the target come from the current policy (ϕ) instead of a target policy. 2) The target in Q-learning depends on the model's prediction, so cannot be considered as a true target. To address this problem, we use another target network instead of using Q-network to calculate the target. 3) In TDSAC, the delayed strategy updates the policy, temperature, and target networks less frequently than the value network to estimate the value with a lower variance to have a better policy [33]. 4) Experience replay enables RL to reuse and also memorize past experiences to solve the catastrophic interference problem. In our method, we store (s_t, a_t, r_t, s_{t+1}) to train deep Q-Network and sample random many batches from the experience replay β (buffer/queue) as training data. We take a random batch B for all transitions $(s_{t_B}, a_{t_B}, r_{t_B}, s_{t_B+1})$.

Let us define $Q_\theta(s, a)$ and $\pi_\phi(a|s)$ as parameterized functions to approximate the soft Q-value and policy, respectively. We consider a pair of soft Q-value functions $(Q_{\theta_1}, Q_{\theta_2})$ and separate target soft Q-value functions $(Q_{\theta'_1}, Q_{\theta'_2})$. We calculate the update targets of Q_{θ_1}, Q_{θ_2} according to $y = r + \gamma(\min_{i=1,2} Q_{\theta'_i}(s', a')) - \alpha \log \pi_\phi(a'|s'), \quad a' \sim \pi_\phi$ we can train soft Q-value by directly

Algorithm 2: TDSAC-based Network slicing

```
Initialize actor network $\phi$ and critic networks $\theta_1$, $\theta_2$
Initialize (copy parameters) target networks $\theta'_1$, $\theta'_2$
Initialize learning rate $\ell_\alpha, \ell_Q, \ell_\pi$
Initialize replay buffer $\beta$
Import custom gym network slicing environment (‘smartech–v2’)
while $t$ ¡ $max\_timesteps$ do
    if $t$ ¡ $start\_timesteps$ then
        $a$ = env.action_space.sample()
    else
        Select action $a \sim \pi_\phi(a|s)$
    end
    next_state, reward, done, _ = env.step($a$)
    store the new transition $(s_t, a_t, r_t, s_{t+1})$ into $\beta$
    if $t \geq start\_timesteps$ then
        sample batch of transitions $(s_{t_B}, a_{t_B}, r_{t_B}, s_{t_B+1})$
        $\theta_i \longleftarrow \theta_i - \ell_Q \nabla_{\theta_i} J_Q(\theta_i)$,    i=1,2    #Update soft Q-function
        if $t \mod freq$ then
            $\phi \longleftarrow \phi + \ell_\pi \nabla_\phi J_\pi(\phi)$    #Update policy weights
            $\alpha \longleftarrow \alpha - \ell_\alpha \nabla_\alpha J(\alpha)$    #Adjust temperature
            $\theta'_i \longleftarrow \tau\theta_i + (1-\tau)\theta'_i$    i=1,2    #Update target network
        end
    end
    if $done$ then
        obs, done = env.reset(), False
    end
    t=t+1
end
```

minimizing,

$$J_Q(\theta_i) = \mathbb{E}[(y - Q_{\theta_i}(s,a))^2], \quad i = 1, 2 \tag{3.7}$$

To obtain lower variance estimates, we use the reparameterization trick [53] and reparameterize the policy using a neural network transformation where $a = f_\phi(\xi; s)$. Therefore, the policy update gradients with respect to experience replay (β) is given by

$$\nabla_\phi J_\pi(\phi) = \mathbb{E}[-\nabla_\phi \alpha \log(\pi_\phi(a|s)) + (\nabla_a Q_\theta(s,a) - \alpha\nabla_a \log(\pi_\phi(a|s))\nabla_\phi f_\phi(\xi; s))] \tag{3.8}$$

We can update temperature α by minimizing the following objective

$$J(\alpha) = \mathbb{E}[-\alpha \log \pi_\phi(a|s) - \alpha\mathcal{H}] \tag{3.9}$$

To enforce action bounds in algorithms with stochastic policy, we use an unbounded Gaussian as the action distribution [54]. The proposed approach is summarized in Algorithm 2.

TABLE 3.1: Comparison of hyperparameters tuning in simulation.

Architecture	**SAC**	**TD3**	**our Method (TDSAC)**
Method	Actor-Critic	Actor-Critic	Actor-Critic
Model Type	Multilayer perceptron	Multilayer perceptron	Multilayer perceptron
Policy Type	Stochastic	Deterministic	Stochastic
Policy Evaluation	Double Q-learning	Clipped double Q-learning	Clipped double Q-learning
No. of DNNs	6	6	5
No. of Policy DNNs	1	1	1
No. of Value DNNs	2	2	2
No. of Target DNNs	3	3	2
No. of hidden layers	2	2	5
No. of hidden units/layer	256	400/300	128
No. of Time Steps	$2e6$	$2e6$	$2e6$
Batch Size	256	100	128
Optimizer	ADAM	ADAM	ADAM
ADAM Parameters (β_1, β_2)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
Nonlinearity	ReLU	ReLU	GELU [55]
Target Smoothing (τ)	0.005	0.005	0.001
Exploration Noise	None	$\mathcal{N}(0, 0.1)$	None
Update Interval ($freq$)	None	2	2
Policy Smoothing	None	$\epsilon \sim clip(\mathcal{N}(0, 0.2), -0.5, 0.5)$	None
Expected Entropy($\mathcal{H}$)	-dim(Action)	None	-dim(Action)
Actor Learning Rate	0.0001	0.001	0.001
Critic Learning Rate	0.0001	0.001	0.001
Discount Factor	0.99	0.99	0.99
Replay Buffer Size	$1e6$	$1e6$	$1e6$

3.1.3 Network Performance

We use a PyTorch[2] custom environment for network setup interfaced through OpenAI Gym as the most famous simulation environment in the DRL community and evaluate our method described in Sec. 3.1.2.2 against other SoA DRL approaches, namely, TD3 [33], DDPG [11], and SAC [54] with a minor change to keep all algorithms consistent. PyTorch is a popular open-source ML framework used extensively for training and developing DNNs. We consider three slices (A, B, and C) with different constraints where the number of new service requests for VNFs follows a distributed homogeneous Poisson process. There exist 20 APs and a maximum of 50 registered subscribers assigned to different slices randomly, and the algorithm computes the computing requirements to allocate to the relevant VNF. The dedicated subscribers to Slice-A are less than Slice-B and Slice-C. Table 3.1 provides a comparison of architectures and hyperparameters, while Table 3.2 presents network parameters. The DNNs structure for the actor-critic networks and target networks are the same. We have set the hyperparameters following extensive experiments [56]. The evaluation computes every 20000 iterations concerning the average return over the best 3 of 5 episodes. The *start_timesteps* denotes the initial time steps for random policy to fill the buffer with enough samples. Moreover, the curves are smoothed for visual clarity in terms of the confidence interval.

[2]https://pytorch.org

TABLE 3.2: Network parameters in simulation.

Network Parameter	**Value**
Channel bandwidth	10 MHz
Background noise (σ^2)	-102 dBm
Antenna gain ($\vartheta_{n,m}$)	9 dBi
Log-normal shadowing ($\Theta_{n,m}$)	8 dB
Small-scale fading distribution ($g_{n,m}$)	$\mathcal{CN}(0, I)$
Path-loss at distance $d_{n,m}$ **(km)**	148.1+37.6 $\log_2(d_{m,n})$ dB
Distance $d_{m,n}$	distributed uniformly [0, 600]
(ι, P_z)	$(10^{-26}, 10^9)$

3.1.3.1 Learning Curve

As shown in Fig. 3.2, the learning curve of TDSAC outperforms all other algorithms in the final performance. Note that the scenario has a big and complex state space. The learning procedures are based on interaction with the network slicing environment. The network slicing has different network configurations and parameters (states), and thereby the curves experience high fluctuation during learning. We use a reward-penalty approach for constraints and thresholds in Problem (3.4). Indeed, this experimental approach can lead the agent to good results because the problem formulation (3.4) is general.

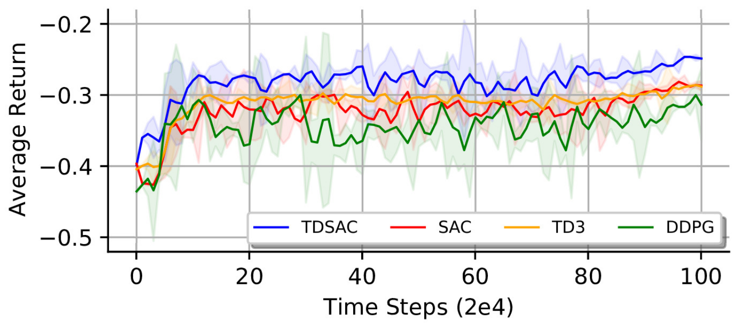


FIGURE 3.2: Learning curves of the smartech-v2 network slicing environment and continuous control benchmarks.

3.1.3.2 Time Efficiency

Fig. 3.3 demonstrates the time efficiency of the different algorithms in terms of the wall-clock time consumption on the custom network slicing environment (smartech-v2). The results show that the TDSAC method yields performance improvement, and it has comparable performance to TD3 and lower than SAC. Note that DDPG uses four DNNs in its architecture and this results in lower wall-clock time consumption compared to other methods, but it has the lowest average return between methods, and thereby we should compare wall-clock time with average

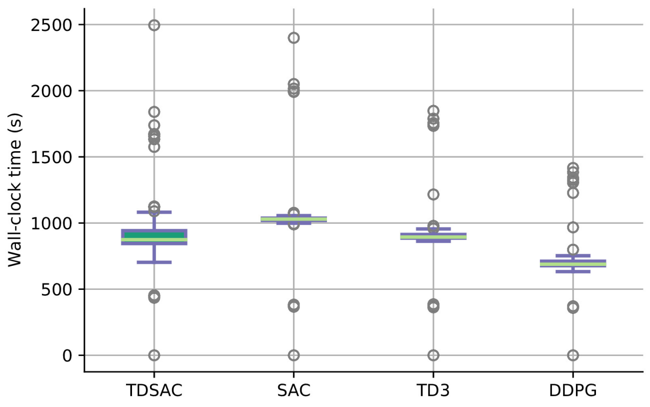
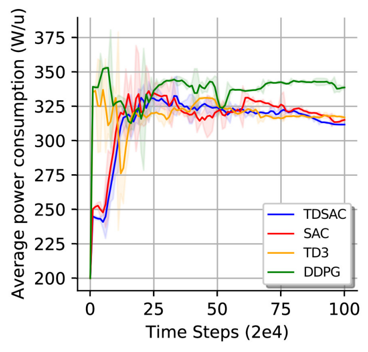


FIGURE 3.3: Time efficiency comparison of different algorithms on the custom environment (smartech-v2).

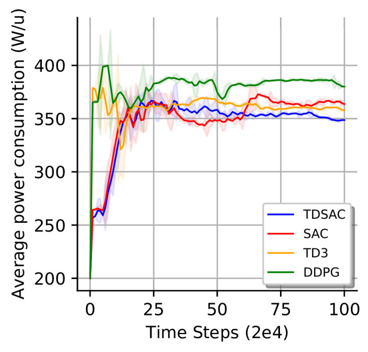
return. All evaluations were run on a single computer with a 3.40 GHz 5-core Intel CPU and evaluation is according to the average per 50 time steps and based on 100 evaluations.

3.1.3.3 Energy Consumption and CPU Utilization

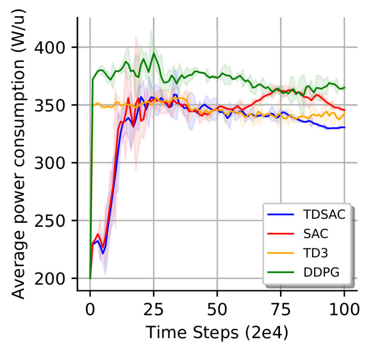
By defining a cross-layer and correlated cost function, we consider the trade-off between CPU resource usage and energy consumption. Fig. 3.4a, 3.4b, and 3.4c show that the performance of TDSAC is better than other approaches. The agent learns to decrease VNFs instantiation and thereby reduce energy in the baseband part while tuning optimal wireless transmission power. Indeed TDSAC has better performance for dynamic resource allocation and adaptive scaling. In some scenarios, DDPG cannot learn perfectly because of some issues, such as overestimation and lack of stable learning behavior. In contrast, the TDSAC, TD3, and SAC used the referred techniques (see Sec. 3.1.2.2) to reduce overestimation, stabilize the training, surmount the curse of dimensionality, solve gradient explosion, and mitigate catastrophic forgetting problems. Note that a large part of energy consumption is constant, and the agent cannot minimize these values. In Fig. 3.4d and 3.4e, we consider MNO and slices (tenants) as a unified network where slices are isolated and trade-off computing resources with MNO. As shown in Fig. 3.4d and 3.4e, the TDSAC performs better than other methods. The TDSAC has better resource control between MNO and tenants. We consider CPU utilization efficiency as the ratio of exploited computing resources concerning the total available CPU for the execution of a VNF.



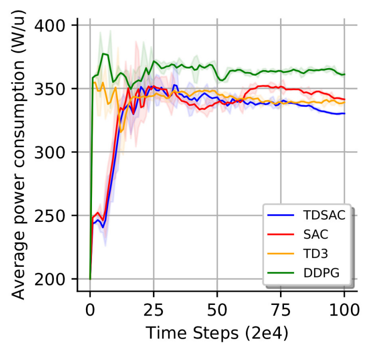
(A) Energy (Slice A)



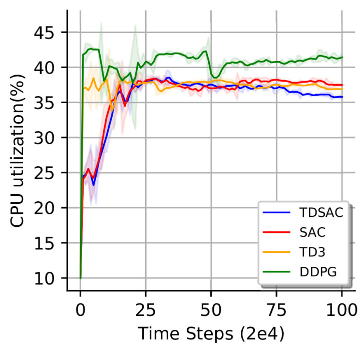
(B) Energy (Slice B)



(C) Energy (Slice C)



(D) Energy (Network)



(E) CPU utilization

FIGURE 3.4: Network performance and costs comparison between TDSAC and other DRL benchmarks. The curves are smoothed for visual clarity. The solid lines demonstrate the mean and the shaded regions correspond to the confidence interval over three trials.

3.2 Network Slicing for Low-Latency Services

In the context of 5G/6G networks, a low-latency slice refers to a virtual network that facilitates prompt and low-latency communication. Low-latency slices play a crucial role in 5G/6G networks by providing real-time applications that require instantaneous communication, such as remote surgery, autonomous vehicles, and industrial automation. These applications necessitate high reliability and ultra-low latency to guarantee the speedy and accurate execution of commands. Several optimization techniques are employed to minimize latency in a network slice, such as minimizing packet loss, reducing network congestion, optimizing routing, and optimal resource allocation. Additionally, edge computing technologies like edge servers and edge caches are deployed to reduce latency by bringing computing resources closer to the end-users. In this section, we focus on the algorithmic innovation and solution aspects of the ZSM standard. The SLA guarantees that slice-level QoS is fulfilled by automating the control of underlying performance metrics [57]. Automated MANO operations require a flexible and scalable design that considers long-term performance to account for the plethora of user patterns over different slices and handle such a heterogeneous and complex network.

This tendency towards fully automated MANO has aroused intensive research interest in applying AI and DRL to tackle challenging NP-hard tasks. We propose a novel model-free DRL framework, called *collaborative statistical Actor-Critic* (CS-AC), that enables a scalable and farsighted slice performance management in a 6G-like RAN scenario. The proposed CS-AC aims to optimize latency cost under a long-term statistical SLA. Specifically, the SLA metric used is the *Q-th delay percentile*, enforcing some slice-specific preset constraints on it. To implement distributed learning, we proposed a developed variant of SAC that is less sensitive to hyperparameters. Finally, we present numerical results that demonstrate the effectiveness of our approach in an OpenAI-based network slicing environment, validating its performance concerning latency, SLA Q-th percentile, and time efficiency.

3.2.1 Slice-Enabling Cell-Free

As shown in Fig. 3.5, we consider a slice-enabling cell-free mMIMO network scheme with edge-cloud computing (central unit/central processing units) capability and distributed APs connected to a central server via serial fronthaul links. Let us define the RAN environment consisting of N APs that cover M single-antenna users in a downlink setup. The network slicing architecture consists of $L \in \mathbb{N}$ slice instances where each slice accommodates M_l users with $\sum_{l=1}^{L} M_l = M$. Each user M sends NSSAI to assist the network in selecting a particular network slice where it may be served by a maximum of eight network slices simultaneously [58]. We suppose each user just requests one type of service and thereby one slice instance at each decision time step.

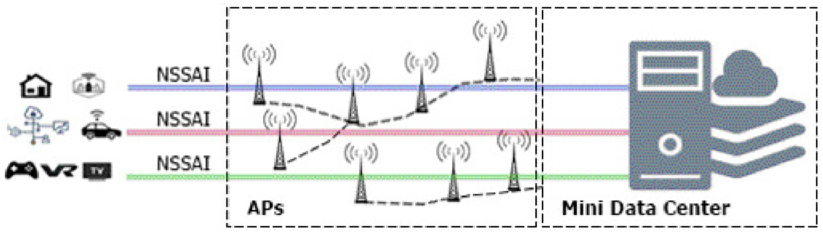


FIGURE 3.5: The slice-enabling cell-free mMIMO scheme.

We consider resource allocation (in NFV) tasks where the MNO collects the free and unused resources from the tenants and allocate them to the needy slices. It is done either periodically to avoid over-heading or based on requests of tenants. Moreover, we consider another action concurrently to allocate power to different users. We assume that all the APs are connected to a central server that maintains and deploys a set of VNFs to serve the users of distributed APs and also hosts agents for the training process to learn the best policies and actions for scaling the computing resources vertically and consequently scale horizontally for VNFs instantiation to minimize the latency according to system states.

We follow a slotted resource allocation scheme, where the central server allocates resources to new arriving users at the start of the next slot. Indeed, the time horizon is discretized into the decision time step where $t \in \mathbb{N}^{+}$. We define $\jmath_l^{(t)}$ as number of new service requests from all APs for l-th slice at time step t where it follows an independent and identically distributed Poisson process with parameter $\lambda_l^{(t)}$. Therefore, the probability of new demands to arrive at the central server for time-slot of duration T is given by, $P(\jmath_l^{(t)} = \jmath) = \frac{(\lambda_l^{(t)} T)^{\jmath}}{\jmath!} e^{-\lambda_l^{(t)} T}$, where $\lambda_l^{(t)} = \max\{x \sim \mathcal{N}(\mu_l, \sigma_l,), 0\}$ is time-varying value to capture slow variations of network traffic over time by sampling a Gaussian distribution with parameters μ_l and σ_l [12]. Let us define vector of channel gains from the all N APs to the user m as $\mathbf{h}_m = [h_{1,m}, h_{2,m}, ..., h_{N,m}]^H \in \mathbb{C}^{N\times 1}$, where $(\cdot)^H$ is the conjugate transpose and $\mathbb{C}$ represents the complex set. Let consider the following channel model [48], $h_{n,m} = 10^{-L^*(d_{n,m})/20} \sqrt{\vartheta_{n,m} \Theta_{n,m}} g_{n,m}$, where $L^*(d_{n,m})$ denotes the path loss with a distance of $d_{n,m}$. Moreover, $\vartheta_{n,m}$ is the antenna gain, $\Theta_{n,m}$ is the shadowing coefficient, and $g_{n,m}$ is the small-scale fading coefficient. The beamforming vector $\mathbf{v}_m = [v_{1,m}, v_{2,m}, ..., v_{N,m}]^H \in \mathbb{C}^{N\times 1}$ is associated with user m and whose expression is given by [47],

$$\mathbf{v}_m = \sqrt{p_m} \frac{\left(\mathbf{I}_N + \sum_{j=1}^{M} \frac{1}{\sigma_v^2} \mathbf{h}_j \mathbf{h}_j^H\right)^{-1} \mathbf{h}_m}{\left(\mathbf{I}_N + \sum_{j=1}^{M} \frac{1}{\sigma_v^2} \mathbf{h}_j \mathbf{h}_j^H\right)^{-1} \mathbf{h}_m}, \tag{3.10}$$

where p_m is beamforming power, $\mathbf{I}_N$ denotes the $N \times N$ identity matrix and σ_v^2 is the noise variance. Then we define approximate data rate for user m with respect to channel bandwidth $\hat{B}$ and signal-to-interference-plus-noise ratio as follow,

$$R_m^{(t)} = \hat{B} \log_2 \left(1 + \frac{\mathbf{h}_m^H \mathbf{v}_m{}^2}{\sum_{j\neq m}^{M} \mathbf{h}_m^H \mathbf{v}_j{}^2 + \sigma^2}\right). \tag{3.11}$$

We suppose each user m has a task to be executed. Let define data size of task m as $d_m^{(t)} = R_m^{(t)} k_m^{(t)}$, where $k_m^{(t)}$ denotes the transmission time and we consider the coefficient ζ_m to compute the required computing CPU frequency cycles $\Delta_m^{(t)}$ as proportional to data size of corresponding task, $\Delta_m^{(t)} = \zeta_m d_m^{(t)}$. Then the computing delay is given by,

$$\mathcal{D}_{m,1}^{(t)} = \frac{\zeta_m d_m}{\mathcal{F}_m}, \tag{3.12}$$

where $\mathcal{F}_m$ is computing speed of the edge central server. The n-th fronthaul satisfies, $\chi_{n,m} = \sum_{m \in M} R_m \left[1(\hat{\chi}_{n,m} = 1)\right] \leq \varphi_{n,th}$ capacity constraint, where $\hat{\chi}_{n,m} \in \{0, 1\}$ is a binary variable to determine the association between the n-th AP and the m-th user. Let us define g-th flow as a competitive flow for f-th flow, and then the f-th flow should wait for transmission of the g-th flow. To compute the queuing delay n-th link we have,

$$\mathcal{D}_{n,2}^{(t)} = \frac{\psi \lambda_{\mathcal{D},n}}{\varphi_{n,th}}, \tag{3.13}$$

where ψ denotes the maximum burst size in a fronthaul network [59], and $\lambda_{\mathcal{D},n}$ denotes the number of competitive flows at n-th link. Therefore, the total delay for each task m and corresponding fronthaul link n is given by,

$$\mathcal{D}_{n,m}^{(t)} = \mathcal{D}_{m,1}^{(t)} + \mathcal{D}_{n,2}^{(t)}, \tag{3.14}$$

and thereby, the optimization problem is defined as,

$$\begin{aligned}
\min \quad & \sum_{n=1}^{N} \sum_{m=1}^{M} \mathcal{D}_{n,m}^{(t)} && \text{(3.15a)}\\
\text{subject to} \quad & p_m^{(t)} \leq \mathcal{P}_{max}, \quad \forall m \in M, && \text{(3.15b)}\\
& \Delta_m^{(t)} \leq \Delta_{th,l}, \quad \forall m \in M, \forall l \in L, && \text{(3.15c)}\\
& \chi_{n,m}^{(t)} \leq \varphi_{n,th}, \quad \forall n \in N, \forall m \in M && \text{(3.15d)}\\
& f_Q^l\left(\mathcal{D}_{n,m}^{(1)}, ..., \mathcal{D}_{n,m}^{(t)}\right) \leq \eta_l, \quad \forall l \in L && \text{(3.15e)}
\end{aligned}$$

where $\mathcal{P}_{max}$ is the maximum allowable power level and $\{\Delta_{th,l}\}$ is the maximum CPU cycles threshold which can be set based on MNO's preferences and policies.

3.2.2 Long-Term Statistical SLA

A typical latency SLA between slice l tenant and the MNO consists on imposing a long-term statistical constraint on the distribution of latency values. In this regard, we invoke the Q-th percentile metric $f_Q^{(t)}$ that captures, at each step t, the actual latency value below which $Q\%$

of latency samples over the measurement interval $i = 1, \ldots, t$ are located. We then enforce a slice-specific upper bound η_l on it. To calculate $f_Q^{(t)}(\cdot)$ over set $\mathcal{Y}_l = \left\{\mathcal{D}_{n,m}^{(1)}, ..., \mathcal{D}_{n,m}^{(t)}\right\}$, the elements thereof are sorted in the ascending order, i.e., $\mathcal{Y}_l = \{z_1, \ldots, z_t \mid z_i < z_{i+1}\}$, and then the Q-th percentile is derived as,

$$f_Q^l\left(\mathcal{D}_{n,m}^{(1)}, ..., \mathcal{D}_{n,m}^{(t)}\right) = z_j,\, j = *\frac{Q \times (t+1)}{100}. \tag{3.16}$$

In this constrained optimization problem, the system latency closely depends on date rate $R_m^{(t)}$, and the underlying computing resources delay $\mathcal{D}_{m,1}^{(t)}$, as well as transmission delay $\mathcal{D}_{n,2}^{(t)}$. Specifically, user m allocated power p_m has a direct impact on the achieved data rate, thereby the required computing resources for VNFs as well as the resulting computing delay. This approach presents correlated models where the main aim is to find the best policy for jointly allocating power and computing resources to minimize the service provisioning latency while respecting the long-term statistical SLA. Following model-free DRL-based approaches [12]-[13], we formulate the problem from an MDP perspective and develop a new DRL scheme to cope with the underlying high dimensional state and action spaces as detailed in the sequel.

3.2.3 Collaborative Statistical Actor-Critic

Fig. 3.6 presents the proposed cross-platform framework (CS-AC) concerning the RAN data center. The architecture consists of six main components: i) The network slicing environment (*smartech-v4*), ii) The slice-level SLA buffer that stores the historical SLA-related metrics, namely, latency states in the current scenario, to instantly calculate their empirical distribution, such as the Q-th percentile, and feed it to the DRL block, iii) The parallel actors typically run on CPUs and interact with network environment to generate new experiences and behaviors $(s_t : state, a_t : action, r_t : reward, s_{t+1} : next - s_t)$ asynchronously and enforce the best actions, iv) The experience replay (buffers) to store past experiences while coping with catastrophic interference, v) The parallel learners to train and optimize the model on GPUs where they sample a random batch $\mathbb{B}_i, i \in \mathbb{N}$ for all transitions $(s_{t_B}, a_{t_B}, r_{t_B}, s_{t_B+1})$ of β_i, and vi) The memories $(\mathbb{M}_i)$ for sharing the parameters and models where mitigate the load on learners for model update request and also lessens the average read latency related to the actors. The memories share the policy model of learners (ϕ) with actors to update their policy (μ_ϕ). Moreover, the memories can solve the problem of parameter synchronization of actors and learners because they are asynchronous. Indeed, the policy gradient parallelization approach that initially has proposed in A3C [60] can reduce computation time and stabilizes the learning.

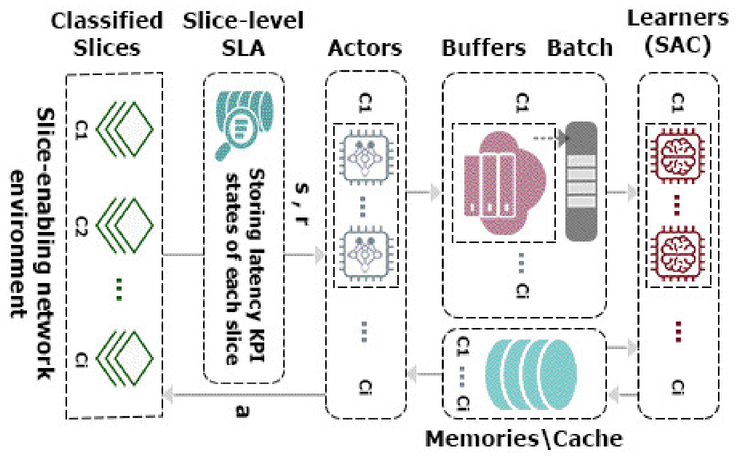


FIGURE 3.6: The proposed software-based framework for massive network slicing.

Separating actors from learners in a network slicing is motivated by improving the learning efficiency by referring to other high-throughput learning frameworks, such as Gorila [61] and Impala [62]. Unlike the previous works, the inference model in CS-AC is executed centrally by the learners. This approach can reduce bandwidth requirements for transferring updated model parameters from learners to actors while the multiple buffers and memories mitigate read latency. We can classify slices according to different scenarios and metrics (e.g., QoS, priority, and tenant ID) and assign each class (c_i) to a collection of actors and learners. The CS-AC can support unbounded limit actors for a massive network, such as network slicing. To reduce waiting time, the CS-AC ignores the slowest actors. Following MDP in RL parlance, the state (s_t), action (a_t) and proposed reward function (r_t) are defined as follows:

1) State space: The state space provides some information about different possible network configurations. Indeed, it helps to learn the best policy (mapping states to actions) through interaction with network slicing parameters. In our scenario, the state transits to the next state at each time step t as input can be characterized by $S^{(t)} = \{S_1^{(t)}, S_2^{(t)}, S_3^{(t)}, S_4^{(t)}\}$, where ($S_1^{(t)}$) is the number of arrival requests for each slice, ($S_2^{(t)}$) is data rate status, ($S_3^{(t)}$) refers to computing resources allocated to each slice, and ($S_4^{(t)}$) is latency status with respect to latency cost for each slice.

2) Action space: We define a continuous multi-action space in telecommunication environment and pursue an experimental approach aiming to allocate power and computing resources to each slice and scrutinize the learning behaviour of the agent in terms of minimizing latency for service provisioning where the allocated power is given by,

$$\mathcal{A}_P^{(t)} \in \{o | o \in \mathbb{R}, 0 \leq o \leq \mathcal{P}_{max}^{(t)}\}, \tag{3.17}$$

where $\mathcal{P}_{max}^{(t)}$ is an experimental value. Moreover, we consider vertical scaling consists of either scaling up or down, i.e., increasing or decreasing the computing resources respectively. Therefore, the agents allocate computing resource according to each time step,

$$\mathcal{A}_{CPU}^{(t)} \in \{o | o \in \mathbb{R}, -\sum_{m=1}^{M} \Delta_m^{(t)} \leq o \leq \Delta_{max} - \sum_{m=1}^{M} \Delta_m^{(t)}\}, \tag{3.18}$$

where $\mathcal{A}_{CPU}^{(t)}$ is vertical scaling action for CPU resources. Note that vertical scaling is limited with respect to the amount of free computing resources available Δ_{max} on the physical server hosting the virtual machine. The complete action space is given by, $\mathcal{A}^{(t)} \triangleq \mathcal{A}_{CPU}^{(t)} \cup \mathcal{A}_P^{(t)}$. Note that we do not consider horizontal scaling and server selection because it requires another algorithm with discrete action space.

3) Reward: The total network cost (Problem (3.15)) is an imprecise and very general metric to guide the agents to learn the best policy and thereby select the best actions. To enforce both the statistical and punctual constraints, we introduce the piecewise function $\Omega_{l,m}^{(t)}$,

$$\Omega_{l,m}^{(t)} = -\varrho_{l,m} \mathbb{1}\left(\Delta_m^{(t)} > \Delta_{th,l} \cup f_Q^l\left(\mathcal{D}_{n,m}^{(1)}, ..., \mathcal{D}_{n,m}^{(t)}\right) > \eta_l\right), \tag{3.19}$$

where $\varrho_{l,m}$ is the penalty coefficient for violating either the CPU constraint or the Q-th percentile SLA, which can be fine-tuned. Consequently, the total return is given by,

$$r^{(t)} = \frac{1}{\frac{1}{M^{(t)}} \sum_{n=1}^{N} \sum_{m=1}^{M} \mathcal{D}_{n,m}^{(t)}} + \sum_{l=1}^{L} \sum_{m=1}^{M} \Omega_{l,m}^{(t)}. \tag{3.20}$$

We consider the number of users at each decision time step ($M^{(t)}$) to make a balance and normalize the network cost between heavy and low traffic. This return function is used in DNN training. We propose this reward-penalty technique to increase the expected return (reward, in RL parlance) while minimizing the latency cost.

The learner part of CS-AC uses an actor-critic setup based on a developed variant of SAC [54]. The DQN and policy gradient are fundamentals of actor-critic methods where the actor is a DNN to parameterize the policy and critic is another DNN to parameterize the value function.

Note that the actor task in actor-learner is different from actor task in actor-critic method. Unlike the DDPG [11] method, the SAC benefits from stochastic policy gradient based on policy gradient theorem [52]. The main goal in standard RL is to learn a policy $\pi(a_t, s_t)$ to maximize the expected sum of rewards. The SAC method [54] benefits from a policy entropy term $\mathcal{H}$. Maximum entropy RL improves the exploration efficiency of the policy. The objective for finite-horizon MDPs is given by, $J_\pi = \mathbb{E}\left[\sum_{i=t}^{T} \gamma^{i-t}[r_i + \alpha\mathcal{H}(\pi(\cdot|s_i))]\right]$, where γ is the discount factor and α denotes a temperature parameter to determine the relative importance of the $\mathcal{H}$ against the reward and handle the stochasticity of the optimal policy.

We use the soft policy iteration method to learn the optimal maximum entropy policies. The convergence and optimality of this approach have been verified in [53]. Our proposed SAC method incorporates a set of techniques such as double (clipped) Q-learning technique [33], target DNN to compute true target in DQN, the experience replay to memorize past experiences and solve the catastrophic interference issues, and the delayed strategy [12] to update the policy, target networks and temperature less frequently than the value network. The goal is to mitigate very high sample complexity and meticulous hyperparameter tuning and also stabilize the learning.

We parameterize functions $Q_\theta(s, a)$ and $\pi_\phi(a|s)$ to approximate the soft Q-value and policy where $(Q_{\theta_1}, Q_{\theta_2})$ are soft Q-value functions and $(Q_{\theta'_1}, Q_{\theta'_2})$ are target soft Q-value functions. The updates of Q_{θ_1}, Q_{θ_2} based on targets is given by,

$$y = r + \gamma(\min_{i=1,2} Q_{\theta'_i}(s', a')) - \alpha \log \pi_\phi(a'|s'), \quad a' \sim \pi_\phi. \tag{3.21}$$

To train the soft Q-value, we can directly minimize,

$$J_Q(\theta_i) = \mathbb{E}[(y - Q_{\theta_i}(s, a))^2], \quad i = 1, 2. \tag{3.22}$$

The SAC method leverages a reparameterization trick [53] to reduce variance estimates where reparameterizes the policy using a neural network transformation $a = f_\phi(\xi; s)$. The policy update gradients based on experience replay (β) is given by,

$$\begin{aligned}\nabla_\phi J_\pi(\phi) = \mathbb{E}[-\nabla_\phi \alpha \log(\pi_\phi(a|s)) + (\nabla_a Q_\theta(s, a) \\ -\alpha \nabla_a \log(\pi_\phi(a|s)) \nabla_\phi f_\phi(\xi; s))]\end{aligned} \tag{3.23}$$

The temperature α can be updated through the following objective $J(\alpha) = \mathbb{E}[-\alpha \log \pi_\phi(a|s) - \alpha\mathcal{H}]$. The proposed approach for a single agent of network slicing is summarized in Algorithm 3 (actor) and Algorithm 4 (Learner).

Algorithm 3: Actor

```
Initialize replay buffer $\beta_c$
Import network slicing environment (‘smartech–v4’)
Initialize action space $\mathcal{A}$ and state space S
t=0
while $t < max\_timesteps$ do
    if $t < start\_timesteps$ then
        Initial action $a$ = env.action_space.sample() to fill buffer
    else
        Select action using the updated network parameters $a \sim \mu_\phi(a|s)$ $w.r.t.$ Algorithm 2
    end
    Apply the action in the network slicing
    Observe next_state, reward, done, _ = env.step($a$)
    store the new transition $(s_t, a_t, r_t, s_{t+1})$ into $\beta_c$
    if $done$ then
        obs, done = env.reset(), False
    end
    Obtain latest network parameters from $\mathbb{M}_c$ periodically
    t=t+1
end
```

Algorithm 4: Learner

```
Initialize actor network $\phi$, critic network $\theta$, and temperature $\alpha$
Initialize (copy parameters) target networks $\theta_1'$, $\theta_2'$
Initialize learning rate $\ell_\mathcal{Z}, \ell_\pi, \ell_\alpha$
Initialize memory $\mathbb{M}_c$
while $t$ ¡ $max\_timesteps$ do
    if $t \geq start\_timesteps$ then
        sample batch of transitions $(s_{t_B}, a_{t_B}, r_{t_B}, s_{t_B+1})$
        $\theta_i \longleftarrow \theta_i - \ell_Q \nabla_{\theta_i} J_Q(\theta_i)$,    i=1,2    #Update soft Q-function
        if $t$ mod $freq$ then
            $\phi \longleftarrow \phi + \ell_\pi \nabla_\phi J_\pi(\phi)$    #Update policy weights
            $\alpha \longleftarrow \alpha - \ell_\alpha \nabla_\alpha J(\alpha)$    #Adjust temperature
            $\theta_i' \longleftarrow \tau\theta_i + (1-\tau)\theta_i'$   i=1,2   #Update target network
        end
        Update memory $\mathbb{M}_c$ periodically
        Obtain updated parameters from $\mathbb{M}_c$ periodically
    end
end
```

3.2.4 Performance Evaluation

To evaluate our method (CS-AC) described in Sec. 3.2.3, we use our PyTorch-based custom environment (*smartech-v4*) with a multi-processing approach interfaced through OpenAI Gym [63] and compare this method against other SoA DRL approaches, namely, SAC [54] and DDPG

[11]. Note that the benchmarks have a minor change to make algorithms consistent and all of them support continuous action space and state space. We consider a class of slices with three different slices (A, B, and C). The corresponding class consists of three actors, three learners, and two buffers. Moreover, the size of each task is in the range $[2, 20]$ Mb that is generated uniformly. There exist 10 APs and a maximum of 17 registered subscribers that are assigned randomly to different slices in each decision time step, where the number of subscribers of slice-A is less than slice-B and slice-C. The adopted percentile value during the training is $Q = 95\%$.

Table. 3.2 presents the network parameters. We set hyperparameters of DNNs through extensive experiments [56]-[64] and adopt a similar architecture for both actor-critic and target DNN models. We use five hidden layers and 128 units per layer with batch size 128. Unlike SAC and DDPG methods that use ReLU, CS-AC leverages GELU [55] for non-linearity. We compute the performance of the algorithms based on the total of 25×10^4 decision time steps and evaluate the average over each 10^4 iterations with regard to the best three of the five average return episodes. In algorithm. 4, $freq = 2$ refers to update interval for updating policy and target soft Q-value networks. Moreover, $\tau = 0.001$ denotes the target smoothing coefficient [33]. Note that the curves are smoothed for visual clarity based on confidence interval over three trials.

3.2.4.1 Convergence Performance

In Fig. 3.7, the learning curve of CS-AC outperforms other approaches in the final performance. Indeed, the CS-AC leverages a parallel rollouts technique via following parallel multiple samples or batch gradient descent simultaneously. As we mentioned in Sec. 3.2.3, the reward function benefits from a reward-penalty technique, and agents learn to maximize the average reward while minimizing the constrained latency optimization task.

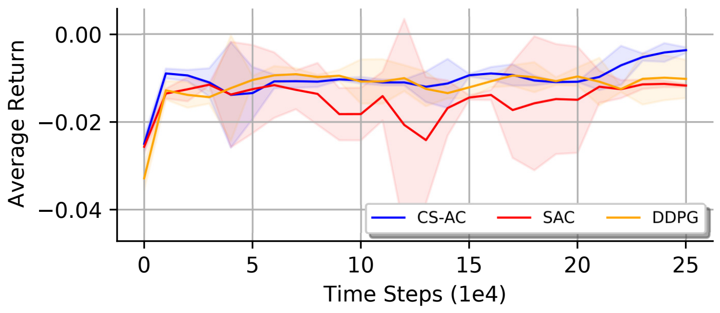


FIGURE 3.7: Learning curves of smartech-v4 network slicing based on continuous control benchmarks.

3.2.4.2 Time Efficiency

As shown in Fig. 3.8, the parallelization approach in CS-AC yields performance improvement significantly compared to SAC and DDPG methods in terms of wall-clock time consumption on the network slicing environment. Note that the actor-critic architecture in CS-AC consists of

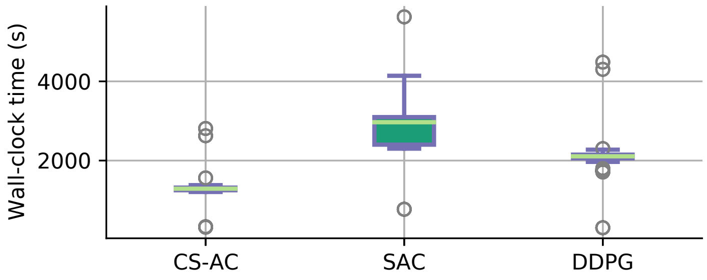


FIGURE 3.8: Time efficiency comparison.

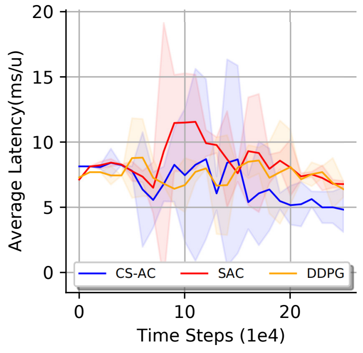
five DNNs, while SAC and DDPG have six and four DNNs, respectively, which is the reason for the lower wall-clock time of DDPG compared to SAC. This evaluation was carried out 100 times over averaging of 50 time steps. To analyze the final performance of algorithms, we should consider both average return and wall-clock time.

3.2.4.3 Network Latency

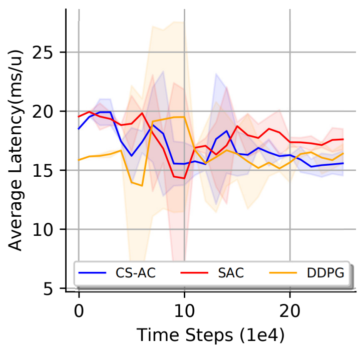
Fig. 3.9 demonstrates the performance of CS-AC in terms of latency. The agents learn to tune optimal power and computing resources to minimize the latency concerning different traffic demands and network configurations (states). Fig. 3.9a, 3.9b, and 3.9c show that the performance of CS-AS is better than other approaches for slice-A, slice-B, and Slice-C. Indeed, the CS-AS can surmount the curse of dimensionality while coping with the overestimation problem [33] in actor-critic methods and stabilize the learning procedure.

3.2.4.4 Latency Percentile

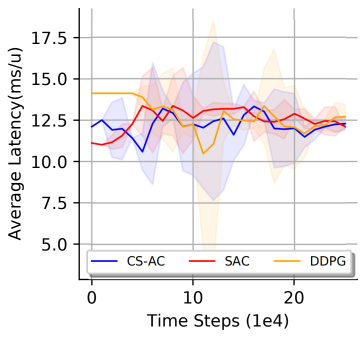
Fig. 3.9d presents the latency percentile vs. Q after the training, i.e., in evaluation mode with $\eta = [10, 20, 15]$ ms latency upper thresholds for slice-A, slice-B, and slice-C, respectively. The Q_{th} latency percentile is a metric for assessing the QoS and performance of the network, as it signifies the highest latency experienced by a specific percentage of requests or transactions. This metric is instrumental in identifying bottlenecks, performance issues, and optimizing the



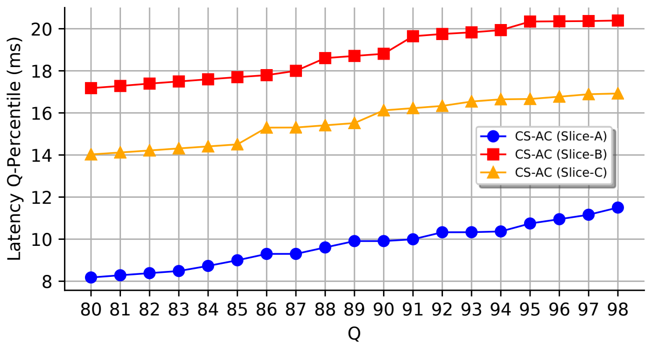
(A) Latency (Slice-A)



(B) Latency (Slice-B)



(C) Latency (Slice-C)



(D) Latency percentile in evaluation mode

FIGURE 3.9: Network latency performance for CS-AC, $\eta = [10, 20, 15]$ ms.

system's QoS. Since the CS-AC has been trained with 95%-perectile statistical constraints, we remark that the three slices are approximately respecting the enforced latency upper bound at $Q = 95$, i.e., the long-term percentile-based SLA is respected.

3.3 Summary

To fulfill zero-touch network slicing, a knowledge-based scheme with an efficient resource provisioning ability should be adopted. We have proposed a KP for B5G network slicing called, KB5G and elaborated on how KP can solve control and optimization problems in network slicing. Specifically, we have deliberated on algorithmic innovation and AI-driven approach and also proposed a continuous model-free DRL method called, TDSAC to minimize energy consumption and VNF instantiation cost. Meanwhile, we have compared the network performance and costs between TDSAC and other DRL benchmarks. We have shown that the proposed solution outperforms other DRL methods. In this chapter, we present the following contributions:

- We propose a KP for B5G network slicing dubbed KB5G and elaborate on how KP can join the architectural aspects of network slicing to make a harmonization in a continuous control setting through revisiting ZSM operational closed-loop building blocks. Specifically, we consider CPU and energy consumption control and optimization.

- We propose TDSAC as an algorithmic innovation in network slicing. This stochastic actor-critic approach supports continuous state and action spaces in telecommunication while stabilizing the learning procedure and improving time efficiency in B5G. Moreover, it benefits from a model-free approach to underpin the dynamism and heterogeneous nature of network slicing while reducing the need for hyperparameter tuning.

- We develop a 5G RAN network slicing environment called *smartech-v2*. It integrates both CPU and energy consumption simulators with an OpenAI Gym-based standardized interface to ensure reproducible comparison of different DRL algorithms.

Also, we have presented a novel AI-driven software-based framework for controlling massive network slicing in B5G/6G, dubbed CS-AC. Specifically, we have considered a slice-level statistical DRL method based on the SAC algorithm for allocating power and computing resources dynamically to minimize a latency-aware cost optimization under Q-th delay percentile SLA metric. The numerical results of the proposed actor-learner approach have shown that the target Q-th percentile is respected while also guaranteeing better performance in terms of latency and time efficiency compared to other SoA DRL benchmarks. Specifically, in this section, we present the following contributions:

- We investigate the feasibility of a multi-objective and multi-action approach where model-free agents learn to jointly allocate optimal power and computing resources to minimize the latency of service provisioning under long-term statistical SLA, namely, *Q-th delay percentile.*

- We propose a massive DRL-based actor-learner framework dubbed CS-AC. The CS-AC is a software framework for designing and training DRL agents that attempts to address complexity and scalability issues. To cope with control challenges in network slicing, such as increased dynamism, heterogeneity, and extended training time of slice instances, we separate the actor from learner where CS-AC can be scaled up to several thousand parallel actors-learners across a large collection of tasks without sacrificing data efficiency.

- To implement distributed learners, we combine DQN and policy gradients in the form of actor-critic [65] approach and propose a developed variant of SAC [54] to reduce the need for hyperparameter tuning and stabilize the learning procedure.
- We develop a 5G RAN network slicing environment called *smartech-v4* where we consider both power and CPU resources in a simulator interfaced through OpenAI Gym [63], which is the most famous toolkit in the DRL community.

Chapter 4

Scalable and Distributed Zero-Touch Network Slicing

Most of the network slicing solutions that are available today face scalability issues when considering massive slices, due to centralized controllers requiring a holistic view of the resource availability and consumption over different networking domains. In this chapter, we tackle this challenge and design a *hierarchical architecture to manage network slices resources in a federated manner*. Driven by the rapid evolution of DRL schemes and the O-RAN paradigm, we propose a set of traffic-aware local DAs dynamically placed in the RAN. These federated decision entities tailor their resource allocation policy according to the long-term dynamics of the underlying traffic, defining *specialized* clusters that enable faster training and communication overhead reduction. Indeed, aided by a traffic-aware agent selection algorithm, the proposed *Federated DRL* approach provides higher resource efficiency than benchmark solutions by quickly reacting to end-user mobility patterns and reducing costly interactions with centralized controllers.

4.1 Distributed Multi-Tenant RAN Slicing

V2X communication, IoT, AR/VR, are just some examples of emerging use-cases in 5G/6G verticals that need to co-exist over a common physical infrastructure. However, the highly heterogeneous performance requirements in terms of bandwidth, latency, and reliability, exacerbate the need for orchestration solutions able to accommodate such services in a resource and cost-efficient manner. Network slicing represents a promising technology able to address such a challenging scenario, by enabling the setup of multiple logical and virtualized network instances, namely *slices*, on top of a common physical mobile network infrastructure [66]. Given the cloud nature of these resources, the networking resources associated to each slice can be dynamically orchestrated and tailored to meet the performance requirements of running services.

In this context, temporal variations of the traffic demand deeply complicate resource planning and allocation tasks, especially in the RAN domain where resource allocation decisions, e.g., in terms of bandwidth, must cope with the additional variability inherent to the wireless channel and end-user's mobility. Traditional RAN slicing solutions envision a centralized controller with a holistic and real-time view of the network, especially about resource utilization, availability, and real-time wireless channel statistics, as depicted in Fig. 4.1. However, similar approaches suffer from scalability issues in real deployments, where the amount of monitoring information to be exchanged, together with the large number of base stations BSs, make it practically impossible to devise optimal resource allocation schemes in a timely and resource-efficient manner [67]. Motivated by the need for a more cost-effective and agile RAN, the O-RAN Alliance recently

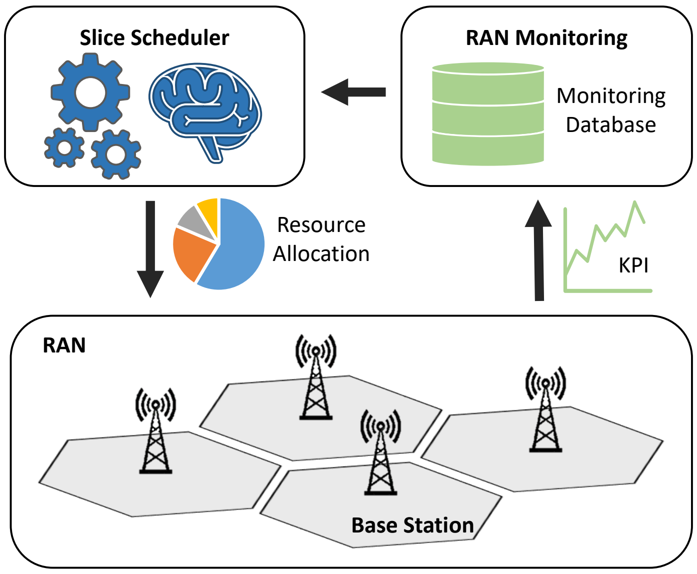


FIGURE 4.1: RAN resource allocation in network slicing.

presented a vendor-neutral alternative way of building mobile networks [68], based on disaggregated hardware and interoperable interfaces that allow secure network sharing by means of virtualization.

Despite the revolutionary approach, it is still not clear how to efficiently support slicing scenarios [69] characterized by a large number of vertical services. Therefore, we take on this challenge and propose a hierarchical architecture for network slice resource orchestration. In particular, given the variable spatio-temporal distribution of mobile traffic demands [70], we envision the dynamic setup of a network of local DAs as virtual software instances co-located within the Near-RT RIC premises, able to access local RAN monitoring information and extract local knowledge without the need of a centralized entity performing decisions on aggregated information. Our framework leverages a dynamic agent selection mechanism based on local traffic conditions similarity, which enables more efficient information exchange and collaboration among groups of local DAs, while specializing their decision policy.

The benefit coming from our approach are several: *i*) it enables resource allocation at the edge of the network, thus accounting for more timely and accurate information, *ii*) the amount of control information that needs to cross the network to reach the central controller dramatically decreases, thus reducing overhead towards the core network, *iii*) by allowing information exchange among local DAs, we enable the provisioning of federated learning schemes to further enrich the capabilities of the DAs. In fact, DAs will not only learn from a local observation space, but also leverage information coming from other (statistically different) RAN nodes, thus improving the generalization of the learning procedure.

The remainder of this Chapter is as follows: Sec. 4.2 formulates our problem and describes the considered scenario. Sec. 4.3 presents the main building blocks of our solution, describing the interaction among the different entities. Sec. 4.4 highlights the compliance of our solution with respect to the O-RAN architecture. Sec. 4.5 validates the design principles of our solution through a comprehensive simulation campaign. Finally, Sec. 4.6 provides the final remarks and concludes this chapter.

4.2 Framework Overview

Our solution builds on the concept of slicing-enabled mobile networks [71], wherein multiple network tenants are sharing a portion, namely a *slice*, of a common mobile network infrastructure, each one with predefined and dedicated networking resources to satisfy an SLA. Within the context of this Chapter, we focus on the RAN domain and consider the SLAs to be expressed in terms of maximum slice throughput and transmission latency. We define transmission latency as the average time the traffic belonging to a certain slice needs to wait within the base station transmission buffers before being served due to inter-slice scheduling procedures. In the following, we overview the main system building blocks and model assumptions, finally introducing the mathematical formulation of the RAN resource allocation problem.

Let us introduce a mobile network infrastructure composed of a set $\mathcal{B}$ of BSs, wherein a set of slices $\mathcal{I}$ is deployed. Each BS $b \in \mathcal{B}$ is characterized by a capacity C_b, expressed in terms of a discrete number of PRBs of a fixed bandwidth. This resource availability must be divided into subsets of PRBs, and dynamically assigned to each network slice according to their real-time traffic demand and SLA requirements.

As part of the SLA between the network operator and the slice owner, we assume each network slice to come with predefined latency and throughput requirements defined by the variables Λ_i and λ_i, respectively. In the context of our work, we focus on the RAN domain, and therefore consider as latency the queueing delay time experienced by the traffic while flowing through the scheduling processes of each base station. Let us consider a time-slotted system where time is divided into *decision intervals* $t \in \mathcal{T} = \{1, 2, \ldots, T\}$. The PRB allocation decisions can be taken only at the beginning of each decision interval, whose duration ϵ may be decided according to the infrastructure provider policies, ranging from few seconds up to several minutes.

We assume the presence of a preliminary admission and control mechanism, e.g., the one presented in [72], to verify the admissibility of the current network slice setup within the available networking capacity, and focus our effort on meeting the resource allocation for the downlink traffic. We envision the allocation of radio resources towards the end-users as a two-step process [73]. Initially, once network slices are admitted into the system, the infrastructure provider schedules the assignment of slots of radio resources for each of the tenants. Then, based on the slice resource availability, each tenant may decide to enforce proprietary scheduling solutions towards its end-users, depending on use-case or business requirements [71].

Given the plethora of user to base station association and scheduling algorithms addressing the end-user resource allocation task [74], we do not address the intra-slice scheduling issue, but rather focus on the correct and fair dimensioning of the inter-slice PRB allocation.

To this aim, we denote with the variable $a_{i,b}^{(t)}$ the PRB allocation decision for the i-th slice under the b-th BS taken at t-th decision time interval, and with $\sigma_{i,b}^{(t)}$ the SNR value expressing the wireless channel quality, averaged over the duration of a decision time interval ϵ, and over the end-users of the i-th slice attached to the b-th BS. Similarly, we introduce $\varphi_{i,b}^{(t)}$ as the aggregated downlink traffic demand generated by the users of the i-th slice under the coverage area of the b-th BS within the t-th time interval.

All together, we can formalize our problem as:

Problem `RAN Resource Allocation`:

$$\min \lim_{T\to\infty} \sum_{t=1}^{T} \mathbb{E}\left[\sum_{i\in\mathcal{I}} d_{i,b}^{(t)}\right] \tag{4.1}$$

subject to:

$$E_{i,b}^{(t)} \leq \Lambda_i, \qquad \forall t \in \mathcal{T}, \forall i \in \mathcal{I}, \forall b \in \mathcal{B}; \tag{4.2}$$

$$\sum_{i\in\mathcal{I}} a_{i,b}^{(t)} \leq C_b, \qquad \forall t \in \mathcal{T}, \forall b \in \mathcal{B}; \tag{4.3}$$

$$a_{i,b}^{(t)} \in \mathbb{Z}_+, d_{i,b}^{(t)} \in \mathbb{R}_+, \qquad \forall t \in \mathcal{T}, \forall i \in \mathcal{I}, \forall b \in \mathcal{B}; \tag{4.4}$$

where $E_{i,b}^{(t)} = \mathbb{E}\left[\frac{\varphi_{i,b}^{(t)}}{\Gamma\left(a_{i,b}^{(t)},\sigma_{i,b}^{(t)}\right)+d_{i,b}^{(t)}}\right]$ defines the expected transmission latency, and $\Gamma(a,\sigma)$ is a function that translates the PRB allocation a in the equivalent transmission capacity, given the experienced channel quality σ. The traffic demand generated within a decision interval might not be fully satisfied due to erroneous PRB allocation estimations, incurring in additional transmission latency due to traffic queuing at the BS. Therefore, we introduce the variable $d_{i,b}^{(t)}$ as a deficit value indicating the volume of traffic not served within the agreed slice latency tolerance Λ_i, and that is therefore dropped.

Due to fast traffic variations, slice resource allocation decisions at the RAN domain should be taken in a dynamic, proactive, and flexible way to avoid service and performance degradation. While advanced admission and control mechanisms could select the set of slices to be admitted to the system, and provide static resource allocation boundaries to satisfy the available capacity, the dynamic nature of the slice's traffic load and wireless channel statistics may lead to suboptimal performances.

Additionally, the optimization problem underlying RAN resource allocation, that is, fitting the requests of the slices maximizing the overall utilization by considering the limited resource availability of a BS, has been proven to be NP-Hard [72]. In fact, this problem can be easily mapped into a knapsack problem instance, wherein the sum of allocated resources is bounded by the capacity of the radio interface, and the experienced latency, i.e., the cost, is minimized. This family of problems is well-known to be NP-Hard [75], resulting in a time complexity of

$O(IC_b)$ in our scenario, where I is the cardinality of the set $\mathcal{I}$, and C_b is the base station resource availability in number of PRBs.

In order to obtain a solution for the overall RAN deployment, the same problem should be solved for all the nodes in the network, therefore introducing scalability issues. Moreover, the centralization of all the necessary up-to-date monitoring information further exacerbates the complexity of this problem, which becomes impractical in real mobile networks characterized by thousands of RAN nodes [76].

4.3 A Multi-Agent Architecture for RAN Resource Allocation

In this chapter, we advocate for the adoption of an FDRL-based architecture to address the RAN slicing scenario. In particular, we rely on local DAs running as software instances within the premises of each BS, as shown in Fig. 4.2. Each agent is in charge of performing slice PRB allocation decisions based on local monitoring information coming from the underlying network monitoring system, or BS *context*. We provide the details of our local decision algorithm later in Sec. 4.3.1.

Nevertheless, the distributed nature of RAN deployments, as well as the varying spatio-temporal behavior of mobile traffic traces [77], make it difficult for an agent trained exclusively on complex and multi-variate monitoring metrics to address unknown statistical distributions of its base station context.

To concurrently address the above issues, we introduce an FL layer that allows inter-agent information exchange, and expedites the learning procedure local knowledge sharing. We provide the details of our FL approach in Sec. 4.3.2.

4.3.1 Local RAN Slicing via DDQN Agent

DQN is a popular RL [78] algorithm that evolves from the well-known concepts of Q-learning and neural network function approximation. DQN represents a model-free approach. It stores the trajectory of experiences for each interaction with the environment in a replay buffer, as to update the network parameters without prior knowledge of the underlying environment statistics. In the following, we will use the i index interchangeably while referring to slices and DAs, assuming a one-to-one mapping of each DA with the corresponding network slice. With focus

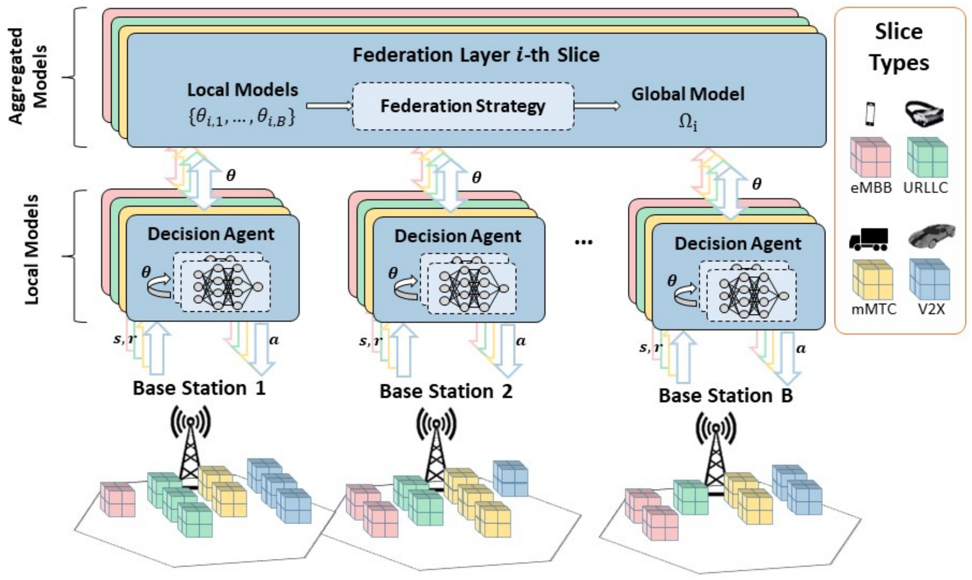


FIGURE 4.2: Generic Federated DRL architecture for RAN slicing.

on a single BS and a single decision interval the design choices of our DQN model are as follows:

State Space $\mathcal{S}$: We define the state of the i-th agent associated to the b-th BS as a tuple of local monitoring information $s_i^{(t)} = \{(\sigma_i^{(t)}, \lambda_i^{(t)}, \nu_i^{(t)}) \mid \forall i \in \mathcal{I}\}$, where $\sigma_i^{(t)}$ is the SNR value, averaged over the duration of a decision time interval experienced by the users of the i-th slice, $\lambda_i^{(t)}$ is the aggregated traffic volume generated by the i-th slice over the time decision duration ϵ, and $\nu_i^{(t)}$ is the amount of available capacity left by the previous decisions of other agents.

Action Space $\mathcal{A}$: Without loss of generality, we define ι as the minimum PRB allocation step, or *chunk size*, and assume that the PRB allocation decision of the i-th agent can only take values that are an integer multiple of ι. It results that $\mathcal{A} = \{\iota \cdot k \mid k = \{0, 1, \ldots, \frac{C}{\iota}\}\}$. Such discrete action space allows controlling the dimensionality of the action space and positively influences the learning process [79].

Reward $\mathcal{R}$: We adopt an iterative reward-penalty approach to guide the agent learning procedure, which translates into maximizing a reward function. An accurate PRB allocation should concurrently guarantee the satisfaction of transmission latency Λ_i and the traffic requirements $\lambda_i^{(t)}$, while avoiding both under-provisioning and over-provisioning of resources. Given the instantaneous slice traffic volume $\varphi_i^{(t)}$, and the corresponding allocation decision $a_i^{(t)} \in \mathcal{A}$, we can identify an *allocation gap* $\alpha_i^{(t)} = \Gamma(a_i^{(t)}, \sigma_i^{(t)}) - \varphi_i^{(t)}$. To measure the goodness of the action, we therefore introduce two variables, namely $\rho_{\text{up}}^{(t)}$ and $\rho_{\text{lower}}^{(t)}$, which characterize the upper and lower boundaries of the allocation gap as $\rho_{\text{up}}^{(t)} = 2 \cdot \Gamma(\iota^{(t)}, \sigma_i^{(t)})$ and $\rho_{\text{lower}}^{(t)} = -\Gamma(\iota^{(t)}, \sigma_i^{(t)})$. Accordingly, we define the instantaneous reward $r_i^{(t)} \in \mathcal{R}$ of the i-th agent as:

$$r_i^{(t)} = \begin{cases} \alpha_i^{(t)} - 4\rho_{\text{lower}}^{(t)} & \text{if} \quad \alpha_i^{(t)} < \rho_{\text{lower}}^{(t)}, \\ (1 - \frac{\alpha_i^{(t)}}{\rho_{\text{up}}^{(t)}})\frac{\alpha_i^{(t)}}{\rho_{\text{up}}^{(t)}} & \text{if} \quad \rho_{\text{lower}}^{(t)} \leq \alpha_i^{(t)} \leq \rho_{\text{up}}^{(t)}, \\ -(\alpha_i^{(t)} - \rho_{\text{up}}^{(t)}) & \text{if} \quad \alpha_i^{(t)} > \rho_{\text{up}}^{(t)}. \end{cases} \tag{4.5}$$

Notably, the first case linearly penalizes the occurrence of under provisioning decisions, while the third case acts in a similar way on the over-provisioning cases. The middle case is the target scenario, which assumes correct PRB allocation decisions in response to the instantaneous slice traffic request.

We envision the multi-agent RAN slicing problem as a sequential procedure, where at the beginning of each decision interval t, the different agents perform local decisions according to a priority value μ_i. Nevertheless, multiple and independent agents may perform inaccurate decisions and leave the subsequent agents with no spare resources, specially in the initial training phase. Therefore, at the end of each training period, we calculate a penalty

$$P_i^{(t)} = -\eta_i 1 \left(a_i^{(t)} > \nu_i^{(t)}\right), \tag{4.6}$$

where η_i is the penalty coefficient of the i-th slice, and 1 denotes the logical operator. This penalty overrides the instantaneous agent reward r_i if the decision $a_i^{(t)}$ is greater than the amount of spare resources left by the previous decisions of the other agents, that in turn prevents the agents to exceed the available resources at the base station. This design choice is justified by the results provided in Sec. 4.5.1.

Training of Agents The training of the local agent implies the characterization of the action-value function $Q\colon \mathcal{S} \rightarrow \mathcal{A}$. Let us define the policy π as a probabilistic function mapping states to actions. The agent makes decisions and selects the corresponding actions based on π, determining the best action for each state. Under a given policy π, the action-value function can be defined as, $Q_\pi(s^{(t)}, a^{(t)}) = \mathbb{E}_\pi \left[\sum_{n=0}^{\infty} \left(\gamma^n r^{(t+n+1)} | (s^{(t)}, a^{(t)})\right)\right]$, where $\gamma \in [0, 1]$ is a discount factor that weights the short-sighted and far-sighted reward, and n is the temporal index. According to Bellman's equation [80], the optimal state-action value function can be expressed as $Q^\star(s^{(t)}, a^{(t)}) = \mathbb{E}\left[r^{(t)} + \gamma \max_{a^{(t+1)}} Q^\star(s^{(t+1)}, a^{(t+1)} | s^{(t)}, a^{(t)})\right]$, and thereby the Q-learning update

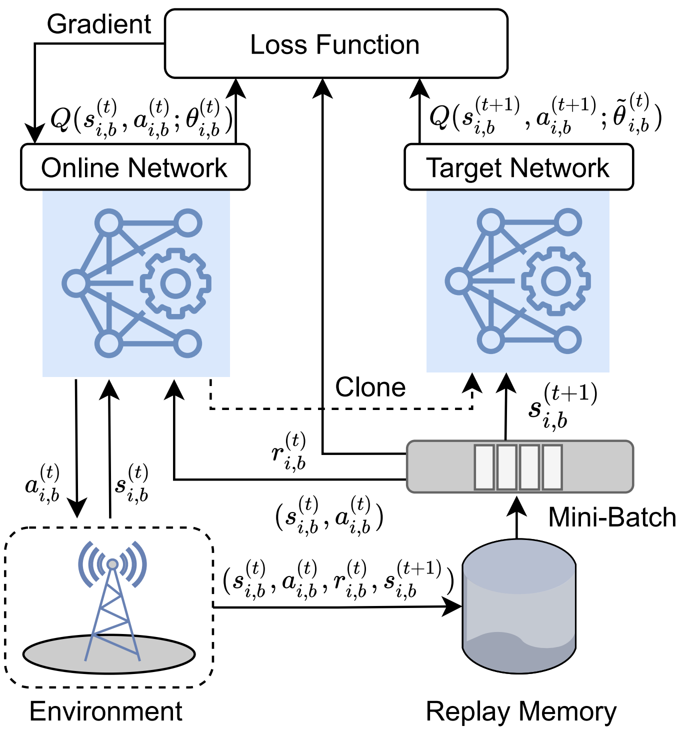

Gradient
Loss Function
$Q(s_{i,b}^{(t)}, a_{i,b}^{(t)}; \theta_{i,b}^{(t)})$
$Q(s_{i,b}^{(t+1)}, a_{i,b}^{(t+1)}; \tilde{\theta}_{i,b}^{(t)})$
Online Network
Target Network
Clone
$s_{i,b}^{(t+1)}$
$r_{i,b}^{(t)}$
$a_{i,b}^{(t)}$
$s_{i,b}^{(t)}$
$(s_{i,b}^{(t)}, a_{i,b}^{(t)})$
Mini-Batch
$(s_{i,b}^{(t)}, a_{i,b}^{(t)}, r_{i,b}^{(t)}, s_{i,b}^{(t+1)})$
Environment
Replay Memory

FIGURE 4.3: An illustration of DDQN workflow.

rule based on temporal difference (TD) [81] is given by,

$$Q(s^{(t)}, a^{(t)}) \leftarrow Q(s^{(t)}, a^{(t)}) + \xi[r^{(t)} + \gamma \max_{a^{(t+1)}} Q(s^{(t+1)}, a^{(t+1)}) - Q(s^{(t)}, a^{(t)})], \quad (4.7)$$

where ξ is the learning rate. DQN adopts DNN to approximate the state-action value and surmount the curse of dimensionality concerning inordinate large state spaces. To limit the catastrophic interference problem [82], which is the tendency of a neural network to forget about previously learned information upon learning new ones, we adopt an experience replay strategy. In particular, let us introduce β_i as the experience buffer. As depicted in Fig. 4.3, in every training interval, we store the tuple $(s_i^{(t)}, a_i^{(t)}, r_i^{(t)}, s_i^{(t+1)})$ describing the instantaneous experience generated by the agent while interacting with the environment, and sample from β_i a random batch of past experiences to regularize the training.

Additionally, DQNs are well known to provide an overoptimistic value estimation. We alleviate this problem by leveraging an additional DQN network, in the form of DDQN [83]-[84]. With a slight abuse of notation, let us introduce $Q(s_i^{(t)}, a_i^{(t)}; \theta_i^{(t)})$ and $Q(s_i^{(t)}, a_i^{(t)}; \tilde{\theta}_i^{(t)})$ as the online network and target network respectively, where $\theta_i^{(t)}$ and $\tilde{\theta}_i^{(t)}$ denote the model parameters. To optimize the parameter set $\theta_i^{(t)}$ and approximate the optimal action-value function $Q^\star(s_i^{(t)}, a_i^{(t)})$, we use the following loss function,

$$L(\theta_i^{(t)}) = \mathbb{E}[y_i^{(t)} - Q(s_i^{(t)}, a_i^{(t)}; \theta_i^{(t)})]^2, \tag{4.8}$$

where $y_i^{(t)} = r_i^{(t)} + \gamma \max_{a_i^{(t+1)}} Q(s_i^{(t+1)}, a_i^{(t+1)}; \tilde{\theta}_i^{(t)})$ and $\tilde{\theta}_i^{(t)}$ is copied from $\theta_i^{(t)}$ at the end of each episode. Finally, the objective function of the DDQN model can be written as,

$$y_i^{(t)} = r_i^{(t)} + \gamma Q(s_i^{(t+1)}, \arg\max_{a_i^{(t+1)}} Q(s_i^{(t+1)}, a_i^{(t+1)}; \theta_i^{(t)}); \tilde{\theta}_i^{(t)})), \tag{4.9}$$

where $\theta_i^{(t)}$ is a local training model used for selecting actions, and $\tilde{\theta}_i^{(t)}$ is used to evaluate their values according to a different policy, thus mitigating over-estimations issues and improving the decision agents' performances [83]. The loss function estimates the difference between true action-value and target action-value. As the overall training procedure aims at minimizing this loss function, we adopt SGD approach [85] to pursue this goal. The local agent training procedure is summarized in Algorithm 5. The overall local process is aided by a federation scheme described in details in the following subsection.

Algorithm 5: DRL RAN resource allocation for the i-th slice

```
Input      : $t, T, \hat{T}, i \in \mathcal{I}, \theta_{i,b}^{(t)}, \Omega_k^{(t)}$ ;
Output     : Improved DDQN model $\theta_{i,b}^{(t+1)}$ ;
Initialize: $\theta_{i,b}^{(0)}, \forall b \in \mathcal{B}, t = 0$;
if $mod(t, \hat{T}) == 0 \wedge t > 0$ then
    Upload $\theta_{i,b}^{(t)}$ ;
    Wait for Algorithm 6;
     #Get FL model and update the local one;
    $\theta_{i,b}^{(t+1)} \leftarrow \Omega_k^{(t)}$;
end
for  $b \in \mathcal{B}$, in parallel do
    $\nabla L(\theta_{i,b}^{(t)}) \leftarrow$ Local model training;
    $\theta_{i,b}^{(t+1)} \leftarrow \theta_{i,b}^{(t)} + \nabla L(\theta_{i,b}^{(t)})$;
end
```

4.3.2 Federated DRL for RAN Slicing

FL allows training machine learning models across multiple decentralized entities which have access to a limited set of the overall data available. Conversely to multi-agent reinforcement learning, which defines a set of autonomous agents that observe a global state (or partial state)

Algorithm 6: RAN resource orchestration for the i-th federation layer

```
Input  : $t, T, \theta_{i,b}^{(t)}\ \forall b \in \mathcal{B},\ \tau_{i,b}^{(t)}\ \forall b \in \mathcal{B},\ \epsilon_d,\ n_{min}$
Output: Improved federation models $\Omega_{i,k}^{(t+1)}, \forall \Psi_k \in \Psi$
#Define clusters and send initial/updated model;
Collect $\tau_{i,b}^{(t)}, \forall b \in \mathcal{B}$ ;
Compute $\mathbf{D} = (DTW^{(j,z)}), \forall j, z \in \mathcal{B}$;
$\Psi_k \in \Psi \leftarrow DBSCAN(\mathbf{D}, \epsilon_d, n_{min})$;
while $t < T$ do
    if $mod(t, \hat{T}) == 0 \wedge t > 0$ then
        for each $\Psi_k \in \Psi$, in parallel do
            Collect $\theta_{i,k}^{(t)}, \forall b \in \Psi_k$;
            #Derive FL models based on Fed. strategy;
            $\Omega_{i,k}^{(t+1)} \leftarrow f_{strategy}(\theta_{i,b}^{(t)}, \forall b \in \Psi_k)$;
            $\theta_{i,b}^{(t+1)} \leftarrow \Omega_{i,k}^{(t+1)}\ \forall b \in \Psi_k$;
        end
        #Return updated local models;
        return : $\theta_{i,b}^{(t+1)}, \forall b \in \mathcal{B}$
    end
    Run Algorithm 5;
end
```

of the system, select individual actions and receive individual rewards, FL allows to collaboratively learn a shared prediction model by iteratively aggregating multiple model updates, thus decoupling the learning procedure from the need of centralized data sources. A refined version of the original models, combination of multiple local models according to specific federation strategies, is then shared to the agents allowing to significantly improve the learning rate, ensure privacy [86] and provide better generalization [87].

As depicted in Fig. 4.2, within the context of our FDRL-based framework each agent trains a local DDQN model $\theta_{i,b}^{(t)}$ and shares its experience, under the form of model hyperparameters, to those entities belonging to the corresponding federation layer. This iterative training approach enables each federation layer to aggregate the collected knowledge of single agents into a global updated model $\Omega_i^{(t+1)}$, usually stored into a cloud platform or a nearby edge platform to allow faster feedbacks. In order to enhance efficiency and avoid communication overhead, we allow the federation layer to collect the local models (and share the updated ones) only every $\hat{T}$ decision intervals, defining this time period as *federation episode*. Different strategies can be adopted to derive the global federated model, each one implementing a predefined federation strategy function $f_{strategy}(\cdot)$.

In *Average* federation strategy, dubbed as *FDRL* in the following of this work, the collective federation model for the next time interval $\Omega_i^{(t+1)}$ is derived as the simple average of the incoming model weights belonging to all the agents, as

$$\Omega_i^{(t+1)} = \frac{1}{B} \sum_{b \in \mathcal{B}} \theta_{i,b}^{(t)}. \tag{4.10}$$

Aggregated mobile traffic demands follow repetitive spatio-temporal trends due to human activities [88]. In this context, it is expected that a good characterization of such processes would allow more accurate forecasting of the network utilization and, in turn, enable an efficient and even proactive planning of the resource allocation.

However, as highlighted in [89], it is not enough to leverage the geographical locations and related spatial proximity of the BS to obtain a comprehensive view of traffic demands, as the land usage of the slice resources may differ even within base stations belonging to the same geographical areas. This introduces an additional issue in our framework, as not all the federated agents should exchange knowledge with each other, nor this should be restricted to only nearby entities. To address this fundamental issue, in the following we propose a clustering algorithm to guide DA subsets definition, based on network monitoring traces and their similarity.

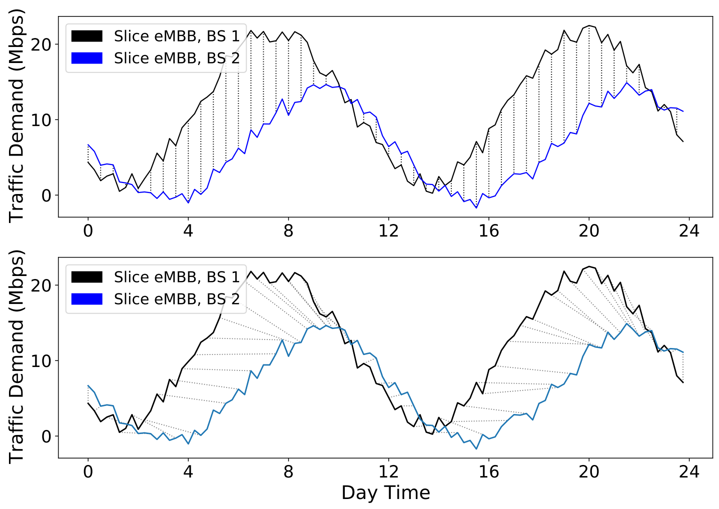


FIGURE 4.4: Example comparison of Euclidean distance against Dynamic Time Warping distance over traffic demand time series.

4.3.3 Dynamic Traffic-Aware Agent Selection

Given the rapid spatio-temporal variation of the traffic demand due to end-user mobility, we advocate for the setup of a clustering algorithm to derive the subset of slice agents that should exchange their local knowledge, while considering both mobility and traffic demand variations. Let us introduce $\tau_{i,b}$ as the time series describing the downlink traffic demand of the i-th slice instantiated over base station b. Then, for each pair $j, z \in \mathcal{B}$, we can compute the similarity of the recorded monitoring information as $DTW^{(j,z)} = f_{DTW}(\tau_{i,j}, \tau_{i,z})$, where $f_{DTW}(\cdot)$ is the Dynamic Time Warping distance [90], a state-of-the-art distance metric for time series analysis[1]. DTW is particularly suitable in our scenario as it allows, conversely to standard distance metrics, e.g., Euclidean distance, to calculate accurate similarity value even in presence of differently sized sequences, and independently of their time shift. An example of DTW distance calculation is depicted in Fig. 4.4, where it can be noticed how maximum and minimum values of the traces are correctly mapped to each other. The pairwise distances are then collected into the distance matrix $\mathbf{D} = (DTW^{(j,z)}) \in \mathbb{R}^{|\mathcal{B}|\times|\mathcal{B}|}$, and provided as input of our clustering algorithm. DTW has linear space complexity, but quadratic time complexity. To reduce the latter, a number of options are available. In our case, we limit the maximal shift by setting a fixed time a window of few hours, thus reducing the complexity even in case of long sequences. Nevertheless, recent work from [91] proposed a novel efficient implementation which breaks the quadratic time complexity to $O(n^2 \log n)$, where n is the length of the sequence. To perform the final classification, we rely on an extended version of the Density-based spatial clustering of applications with noise (DBSCAN) algorithm, introduced in [92]. DBSCAN is a non-parametric density-based clustering algorithm that allows finding the most representative points within a dataset (also known as *core samples*) based on their density in a multi-dimensional space, and expands clusters from them. It expects two inputs: ϵ_d, representing the maximum distance between two samples for one to be considered as in the neighborhood of the other, and n_{min}, which defines the minimum number of samples in a neighborhood of a point to be considered as a core sample.

Given the above, at the end of each federation episode, we can derive in a dynamic way (and based on updated mobile monitoring information) the clusters $\Psi_k \in \Psi$, $k = \{1, \dots, |\mathcal{I}|\}$, where Ψ is the cluster set. Each cluster includes the set of base station $b \in \Psi_k$ that should be involved in the following model update procedure. Therefore, the framework spawns multiple federation models Ω_k, one for each detected cluster k, which evolve in parallel till the next federation episode. The pseudocode of our FDRL-based approach for RAN slicing resource orchestration is listed in Algorithm 6. We remark that in our framework multiple instances of Algorithm 6, i.e., one for each slice $i \in \mathcal{I}$, are deployed to build the corresponding FL domain for a given federation strategy $f_{strategy}(\cdot)$. It follows that the updated federation model, combination of the information coming from the elements of the cluster Ψ_k (described in line 10), can be derived

[1]We refer the reader to [90] for an exhaustive explanation.

following the `Full-Cluster` strategy, namely $f_{FC}(\cdot)$, as:

$$\Omega_{i,k}^{(t+1)} = \frac{1}{|\psi_k|} \sum_{b \in \psi_k} \omega_{i,b}^{(t)} \theta_{i,b}^{(t)}, \qquad \forall \psi_k \in \psi \tag{4.11}$$

where $|\psi_k|$ is the cardinality of ψ_k , and $\omega_{i,b}^{(t)} = \frac{\hat{r}_{i,b}^{(t)}}{\sum\limits_{b \in \psi_k} \hat{r}_{i,b}^{(t)}}$ is a weight parameter. It should be noted that within these settings, the federation step will occur among models with high degree of similarity, thus favoring the *specialization* of the agents towards the most common traffic statistics.

Other complementary approaches can be defined to guide the agent selection and subsequent federation model update. In particular, upon the definition of the cluster set Ψ, we introduce `Random Representative` strategy $f_{RR}(\cdot)$ as a baseline approach, which randomly selects a representative from each cluster:

$$\psi_{random} = \{x | x = rand(\psi_k), \qquad \forall \psi_k \in \psi\} \tag{4.12}$$

and consequently defines the updated federated model as:

$$\Omega_{i}^{(t+1)} = \frac{1}{|\psi_{random}|} \sum_{b \in \psi_{random}} \omega_{i,b}^{(t)} \theta_{i,b}^{(t)}. \tag{4.13}$$

Similarly, let us introduce the `Best Representative` strategy, as a method that derives the updated federation model by selecting a representative agent from each cluster as follows:

$$\psi_{best} = \{x | x = \arg\max_k R_k, \qquad \forall \psi_k \in \psi\} \tag{4.14}$$

where R_k is the cumulative reward within the past federation episode. Thus, the model update strategy `Best Representative` $f_{BR}(\cdot)$, can be defined as:

$$\Omega_{i}^{(t+1)} = \frac{1}{|\psi_{best}|} \sum_{b \in \psi_{best}} \omega_{i,b}^{(t)} \theta_{i,b}^{(t)}. \tag{4.15}$$

By combining single models derived from each cluster, we can pursue higher *generalization* of performances, i.e., aim at a federated model able to deal with heterogeneous traffic statistics.

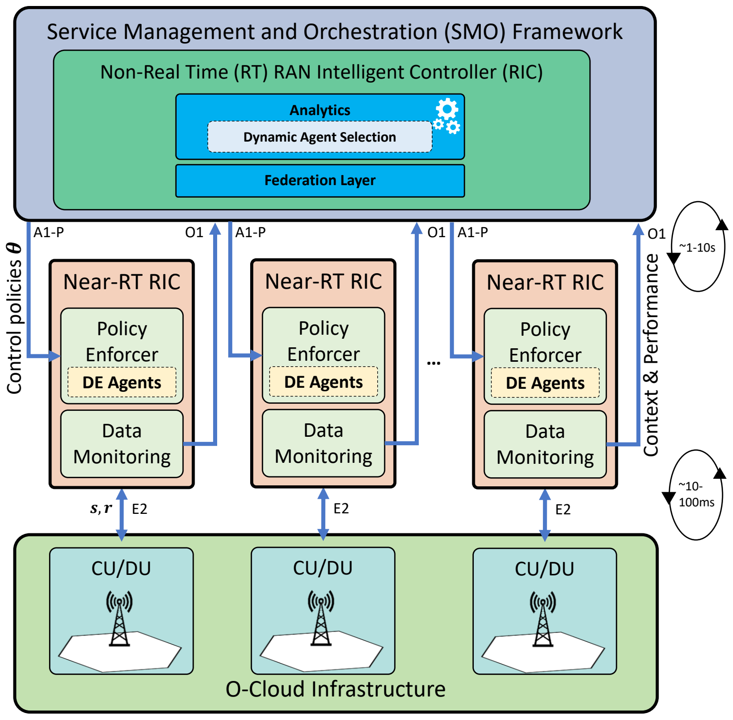


FIGURE 4.5: O-RAN compliant system architecture.

4.4 O-RAN Compliance

The design of our solution closely follows the O-RAN framework [93]. O-RAN represents a worldwide effort to reach new levels of openness in next-generation vRANs. Driven by major carriers, it aims at disrupting the vRAN ecosystem traditionally dominated by a small set of player by breaking vendors' lock-in and opening the business market [94]. The most important functional components introduced by O-RAN are the non-RT RIC and the near-RT RIC [95]. The main functionality provided by the Non-RT RIC it to support RAN optimization over relatively large time scales (e.g., seconds or minutes). This often implies machine ML model training and subsequent control policy definition, to be enforced via the A1 interface towards the distributed Near-RT RICs. The Near-RT RIC is a logical function that enables near-real-time optimization and control, as well as data monitoring of O-CU and O-DU nodes (which support eNBs/gNBs deployment as VNFs) in near-RT timescales (between 10 ms and 1 s). Fig. 4.5 depicts a high-level view of the O-RAN architecture, highlighting the synergies with respect to our proposed approach. In particular, we envision our federated learning and dynamic agent selection module as co-located with Non-RT RIC, which handles the A1's Policy Management Service to enforce radio policies. On the other side, local agents co-located with the Near-RT RICs collect this information, perform local decisions, and exploit the E2 interface to forward

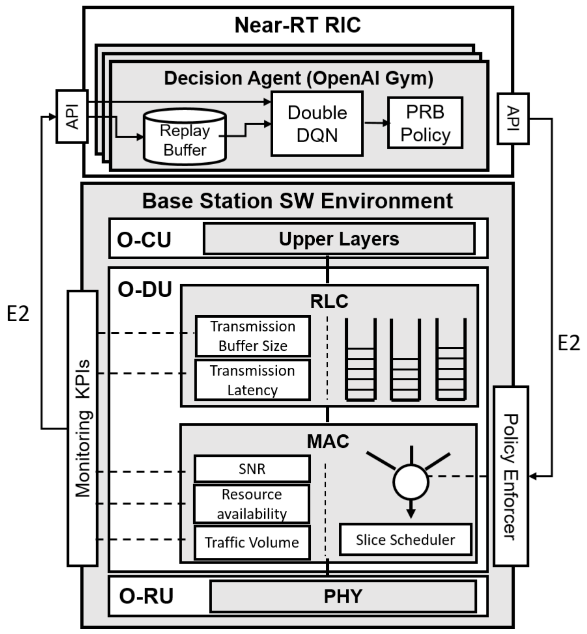


FIGURE 4.6: Software architecture and protocol stack overview.

resulting radio policies to the base station. The same E2 interface would allow the local agent to gather base station KPIs for the purpose of model training and monitoring.

As depicted in Fig. 4.6, we implement our framework in Python programming language, exploiting OpenAI Gym library [96] and interfacing DRL agents with a custom base station simulator environment, which includes virtual transmission queues and main PHY/MAC/RLC functionalities, together with O-RAN E2 interface to allow gathering the slice networking statistics from each O-DU, and to enforce PRB policy decisions in the BS slice scheduler based on defined state space and action space in Sec. 4.3.1. Finally, as described in Sec. 4.3.2, a federation layer connects the DRL agents of the i-th slice to enable inter-agent information exchange and expedite the overall learning procedure. The procedure is summarized in Algorithm 5 and 6.

4.5 Performance Evaluation

In this section, we evaluate our proposed architecture numerical simulations on a dedicated server, equipped with two Intel(R) Xeon(R) Gold 5218 CPUs @ 2.30GHz and two NVIDIA GeForce RTX 2080 Ti GPUs. Moreover, the DNNs are implemented based on TensorFlow-GPU version 2.5.0. In neural network architecture, we use two fully connected layers with 24 neurons activated by ReLU function for each layer where the target network is updated per episode and each episode consists of 5 decision intervals, or epochs. Each decision interval has a duration of $\epsilon = 60$ seconds, during which local monitoring information is collected to build the local agent state. Online and Target networks are characterized by the same DNN structure. The hyperparameter tuning depends highly on capability, scenario, and technology used [56]. The network parameters are updated using the Adam optimizer [37]. The discount factor γ and learning rate ξ are set to be 0.99 and 0.001 respectively. The replay buffer size of each agent $\beta_{i,b}$ is set to 20000 samples, out of which a batch of 32 samples is extracted for each training interval. Without loss of generality, We set $\eta_i = 100$ as penalty value for all the slices. In order to provide a comprehensive overview, we first evaluate single base station settings, focusing on the capabilities of single agents to deal with RAN resource allocation. Then, we address a more realistic scenario considering a multi-slice deployment over several RAN nodes, accounting for end-user mobility and variable traffic demands.

4.5.1 Local Agent Performance Assessment

In our proposed framework, DRL agents optimally allocate radio resources to each slice, while a federation layer enables a periodical exchange of the DRL's parameter values to improve the learning process across multiple agents of the same slice.

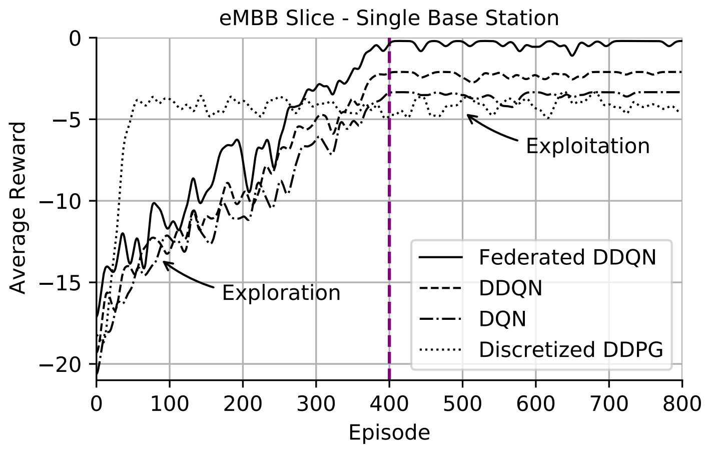


FIGURE 4.7: The convergence performance of different local decision algorithms and an FDRL approach for a single decision agent.

First, we compare the performances of different RL algorithms when dealing with radio resource allocation, without involving federated learning. To this aim, we consider a base station scenario including 3 network slices, i.e., a URLLC kind of slice, an eMBB, and one dedicated to mMTC traffic, each one characterized by the SLA latency values of $\Lambda_i = [10, 40, 20]$ ms, respectively [7]. Regarding the throughput requirements, we do not assume any fixed value as it would depend on the random mobility pattern of the end users and their generated traffic. Instead, we enforce (with a slight abuse of notation), $\lambda_{i,b}^{(t)} = \varphi_{i,b}^{(t)}$ for every decision interval to let the agents adapt their decisions to the instantaneous traffic volumes. We instantiate a DA in every base station for each slice. We model the instantaneous traffic demand of each slice as the realization of a Poisson distribution with mean value λ_i, and emulate the SNR variability extracting its instantaneous values from a Rayleigh distribution with the average value set to 25 dB. Moreover, we set $\iota = 10$ PRBs as the minimum resource allocation step. Fig. 4.7 depicts the training procedure for the eMBB slice, comparing different local decision algorithms. In particular, we consider the single DQN approach, which implements standard Q-Learning procedures, d-DDPG a popular RL algorithm [34], and our DDQN scheme.

We let the scenario run for 800 federation episodes, and depict the results in terms of cumulative reward, as defined in Eq. (4.5). The variability of the network slicing environment leads to experience learning curves with high fluctuations. For visual clarity, results are averaged over 10 simulations. As expected, the DQN approach hardly copes with the definition of suitable PRB allocation policies, providing lower performances both in terms of cumulative reward and convergence time. Similarly, d-DDPG suffers the temporal periodicity of the traffic demand, resulting in a steep learning curve that soon saturates to suboptimal performances. Conversely, after an initial exploration phase, the DDQN approach is able to allocate in a more consistent way correct amount of PRBs to each slice according to the corresponding real-time traffic and latency demands. It is worth highlighting that in terms of convergence time, in general, FDRL schemes do not necessarily provide better performances when compared against standard DRL approaches. In fact, one of the main features of FL is that it allows local DRL agents to indirectly gain knowledge on a wider state space, extending the local experience with that coming from other decision entities deployed within the same environment. This enables the DAs to provide more robust performances when deployed in realistic environments. Nevertheless, the same Fig. 4.7 provides an overview of the local model training procedure, with and without the adoption of FL schemes. In our considered scenario, it can be noticed how DRL curves (dashed lines in the plot) present slower convergence time and higher fluctuations when compared against Federated DDQN approach (solid line in the plot). Additionally, DRL curves present lower cumulative reward after 400 episodes, suggesting a lower capability of the DAs to adapt their decisions at the fast-changing network slicing environment considered in our work.

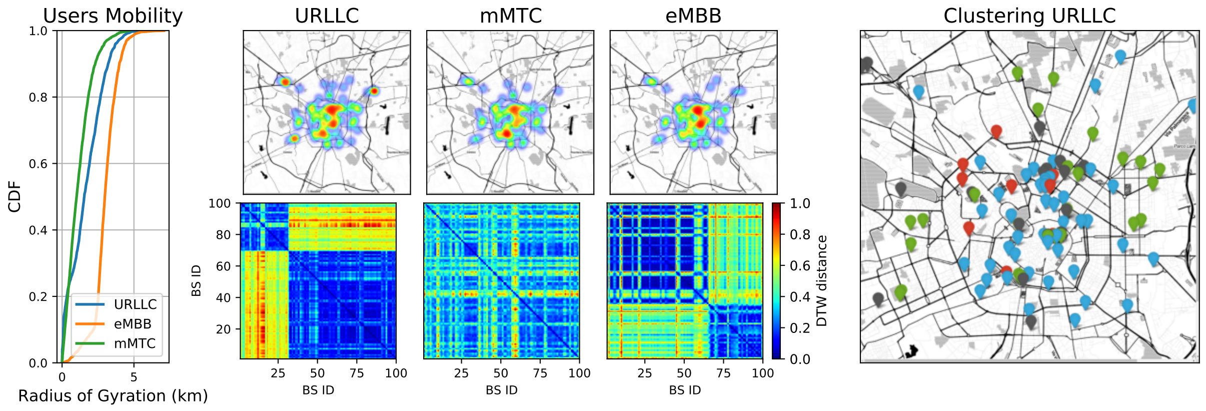


FIGURE 4.8: Mobility Statistics for different network slices. (Left) CDF of end-users radius of gyration, (Center) Average Spatial distribution of slice users over 24h time span, and corresponding DTW distance matrix. (Right) Example of resulting clustering for the URLLC slice case, each color defines a different cluster.

4.5.2 System-level Simulations

4.5.2.1 Mobility and Traffic Demand Characterization

In order to validate our framework in realistic settings, we consider the city of Milan, Italy, as scenario of study. We collect city-wide RAN deployment information including more than 50 BSs from publicly available sources[2], and simulate realistic human mobility patterns leveraging the work of [97]. The d-EPR algorithm allows capturing mobility patterns by specifying as input the geographical position of the base stations together with several probabilistic parameters. We let the model evolve adopting the default parameters described in [98]. By defining the location relevance on the mobility space, we can influence the next-hop selection of each end-user, therefore emulating a higher concentration of mobile devices in specific areas of the city over time, e.g., the daily commuting over the city center during working days. Fig. 4.8 (Left) depicts the CDF of the resulting radius of gyration per slice, aggregating the results over 15000 end-users equally distributed among the different slices. Without loss of generality, we consider the set of BSs characterized by the same radio capacity $C_b = 100$ PRBs, and assume the same 3 slices introduced above simultaneously running over all the BSs. Fig. 4.8 (Center) depicts the resulting spatial distribution of the end-users, accounting for a temporal time span of a full day.

From the picture, it can be noticed how the spatial distribution of slice users is actually similar along with the slice set, and influenced in specific areas of the city by the high density of RAN nodes. This is due to the d-EPR algorithm, which favors the next-hop destination of each user to

[2]https://opencellid.org/

happen towards a nearby point of interest, or, in our settings, the closest base station location. The instantaneous traffic demand of each end-user is derived starting from the values reported in Sec. 4.5.1, weighted by a temporal factor to account for the traffic demand fluctuations typical of mobile network scenarios, as those presented for example by [76] and [99]. In the lower part of Fig. 4.8, we depict the resulting distance matrix **D** of each, i.e., per slice, downlink traffic demand, calculated at the beginning of every federation episode for each base station pair over the past 24 hours. As detailed in Sec. 4.3.3, this matrix is used as input to an instance of the DBSCAN algorithm to derive the set of DAs (belonging to the i-th slice) which should be involved in the next federation episode and model exchange. Fig. 4.8 (Right) shows the resulting output clustering for the URLLC slice case, using $\epsilon_d = 0.06$ and $n_{min} = 2$ as parameters. Such values have been empirically selected following the sensitivity analysis depicted in Fig. 4.9, which certifies that along the evaluation timeline and across the different running slices, the selection algorithm identified on average 3 clusters populated by 15 agents each. The resulting behavior of DAs is heavily affected by the entities participating in the federation process. Therefore, such kind of characterization is fundamental to ensure performances.

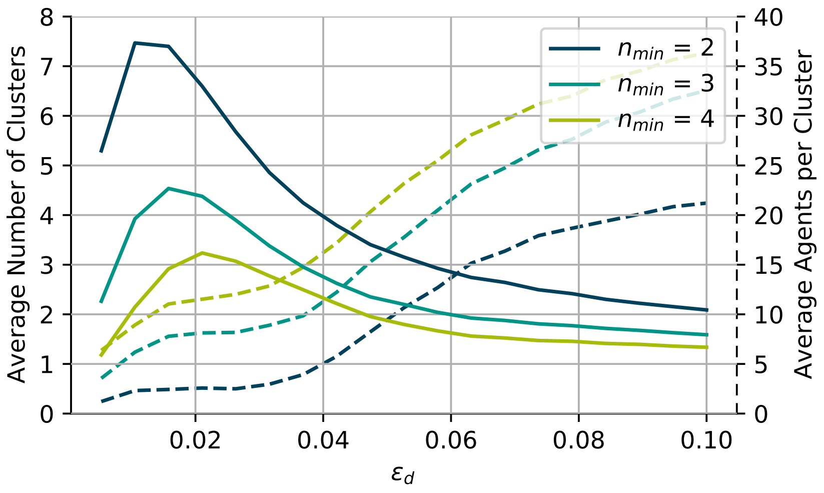


FIGURE 4.9: Sensitivity analysis performed on the clustering parameters and generated traffic traces.

4.5.2.2 Effects of Different PRB Action Space

The size of the action space is well-known to affect the learning curve of any reinforcement learning algorithm. In Fig. 4.10, we investigate this aspect by varying the minimum PRB chunk size $\iota = 2, 5, 10$ of the URLLC slice, while fixing $\hat{T} = 5$ decision epochs per federation episode and adopting the full-cluster federation strategy. The plot shows how increasing the

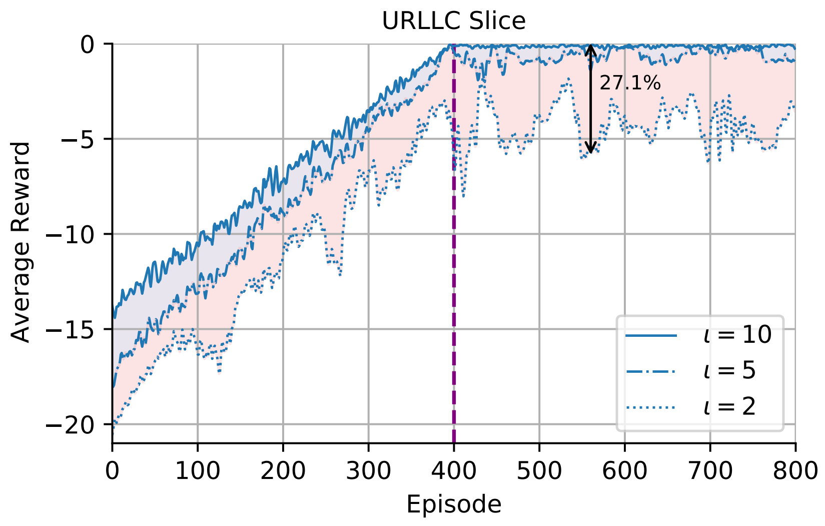


FIGURE 4.10: Learning curve for different action spaces.

PRB chunk size, i.e., adopting smaller action spaces, actually influences the reward of the URLLC slice type and its stringent SLA requirements, with larger PRB chunk values achieving satisfactory performances in a faster way, with about 25% performance gap with respect to $\iota = 2$. Nevertheless, a too broad PRB chunk allocation may result in resource wastage, with portions of the radio resources being under-utilized by the running slices. Such a trade-off should be carefully investigated according to both slice and system requirements. In the following, we will adopt $\iota = 10$ PRBs whenever not specified otherwise.

4.5.2.3 Comparison of Different Federation Strategies

Given the particular nature of the network slicing scenario, in this paper we advocate for a dynamic agent selection method based on the time similarity of traffic demands, dubbed as *DC*. As discussed in Sec. 4.3.3, several strategies can be adopted to combine local models into federated ones, pursuing generalization and performance improvements. In this paper we consider three DC aided approaches, namely *FC*, *BR* and *RR*, and compare their performances against a standard strategy which simply derives a new federated model accounting for all the available local models, without adopting any dynamic agent selection scheme, dubbed as *FDRL*. The benchmark *FDRL* approach exploits all the local trained models and the respective knowledge from the agents, and would theoretically allow for the best generalization of performances [100]. Interestingly instead, from our experiments it turns out that aggregation of widely heterogeneous local models actually limits the capability of the global federated model to converge to a one-fits-all unified model, motivating our dynamic agent selection approach which favors the specialization of federated agents working under similar RAN and mobility contexts.

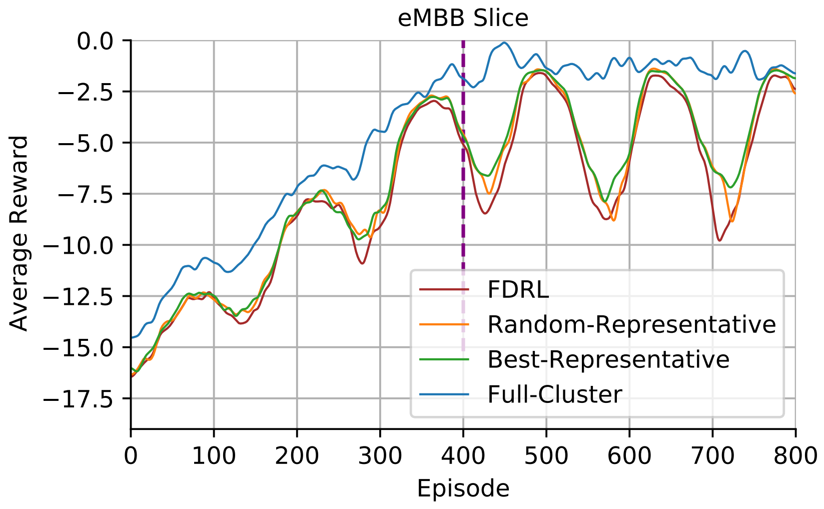


FIGURE 4.11: Comparison of global performances for different dynamic and non-dynamic federation approaches.

Fig. 4.11 provides a comparison of learning performances for different federation strategies in terms of average reward and for $\hat{T} = 5$. The agent's action selection follows a greedy approach which balances exploration of new actions and exploitation of already known decision policies. We gradually limit the exploration capabilities in favour of the adoption of the learned policies, such that around half of the overall simulated time span, the possibility that the agent will explore new actions given a known instantaneous context is in the order of 2%. From the figure, we can observe how *FC* approach achieves better generalization of the learning policies, resulting in stable performances. Conversely, *BR*, *RR* and *FDRL* suffer the dynamic behavior of the underlying traffic conditions, presenting inconsistent reward traces.

4.5.2.4 Latency Analysis for Different Federation Strategies

We continue our performance evaluation by considering the experienced transmission latency. We recall that as mentioned in Sec. 4.2, we define latency as the time spent by the slice traffic within transmission buffer of the base station. Fig. 4.12 depicts the CCDF [101] [102] of the latency experienced by the URLLC and eMBB slices, resulting by different federation strategies. For the both slices, this latency is directly proportional to the traffic demand and the degree of contention of resources among the different slices, as well as to the resource allocation decisions taken by the agents. From the results, it can be noticed that the FDRL strategy leads to the

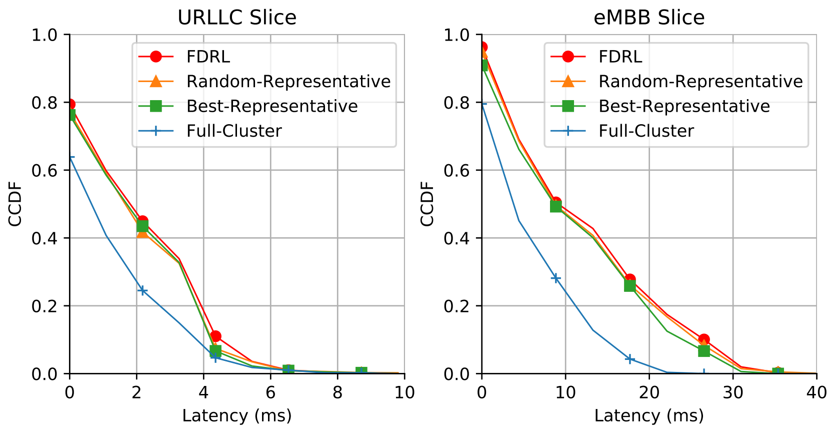


FIGURE 4.12: CCDF of transmission latency for URLLC and eMBB slices.

worst performances, as having all the BSs involved in the learning process results in a slow adaptation of the decisions of the agents to the local traffic conditions, therefore leading to sub-optimal resource allocation and higher latency. In contrast, FC presents a good trade-off in terms of collaboration among agents and specialization to the local traffic conditions, resulting in a more efficient PRB allocation and lower perceived latency. Finally, RR and BR federation strategies achieve performances comparable with the FDRL method, resulting from the limited cooperation in learning that leads these federation approaches to suffer more from the dynamic behavior of the underlying traffic conditions.

4.5.2.5 Effects of Different Network Loads and Mobility

We continue our analysis investigating the performances of the *FC* method in heterogeneous traffic conditions. To this aim, we generate traffic and mobility dataset for an increasing number of end-users, namely 15k, 20k, and 25k. As highlighted in [103], a non-linear relationship characterizes end-user mobility and throughput performances in crowded scenarios. Clearly, this also affects the communication latency, as a higher number of users will be simultaneously active under the same radio access node. In the context of RAN slicing resource allocation, this translates to finding the best DA logic to efficiently address such variability. In Fig. 4.13, we focus our analysis on the dropped traffic, i.e., the volume of traffic that did not meet the latency requirements due to wrong PRB allocation decisions, measured in percentage of the offered traffic volume of each federation episode. From the picture, we can notice how during the initial exploration phase inexperienced PRB allocation decisions performed by the DAs heavily affect the latency requirements of all network slices, with peaks of dropped traffic that increase with the growing number of end-users. Nevertheless, this trend improves over time as

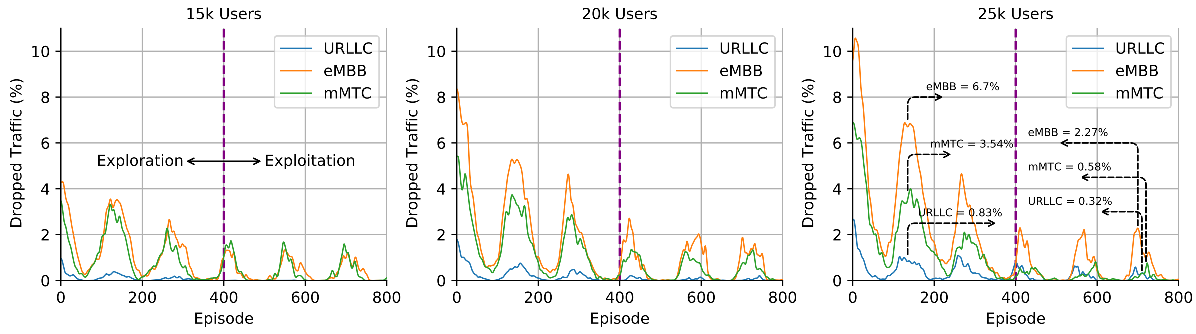


FIGURE 4.13: Performance evaluation for different network loads derived by an increasing number of end-users in Full-Cluster settings.

the agents gain knowledge over the underlying scenario and get trained, finally converging after policy switch, i.e. after episode 400, towards values in the order of 2% for the eMBB slice, and 0,32% for the URLLC slice.

4.5.2.6 Communication Overhead for Different Federation Strategies

Federated Learning aims at building global knowledge from the exchange of multiple locally trained models towards a centralized entity. Such a frequent model exchange however introduces significant communication overhead and synchronization issues, specially in wide scenarios as those considered in our work. Fig. 4.14 compares the model exchange overhead per federation episode resulting from our experiments for a different number of base stations while running the same 3 slices. In the upper part we focus on the overhead statistical distribution. The benchmark *FDRL* approach assumes the exchange of all the locally trained model weights to derive the federated ones, which implies the highest communication overhead. The BR and RR approaches (depicted in the center of the image, and referred to as *Representative*) allow reducing the uplink information exchange by selecting a single representative of each cluster, regardless of the dimensions of the group itself, thus minimizing the communication overhead in each federation episode. It results in less than 800 kBytes in our settings. Finally, the *FC* approach is characterized by an intermediate average value but higher variance. This is due to the variable size of the DAs clusters, which follows the real-time traffic variations, and the need to exchange the local model weights of each element of the cluster, saving communication resources from those base stations that presented peculiar traffic traces and remain unclustered.

On the lower part of the picture, we differentiate between uplink and downlink model exchange overhead. The FDRL approach presents a symmetric behavior matching the model exchange of

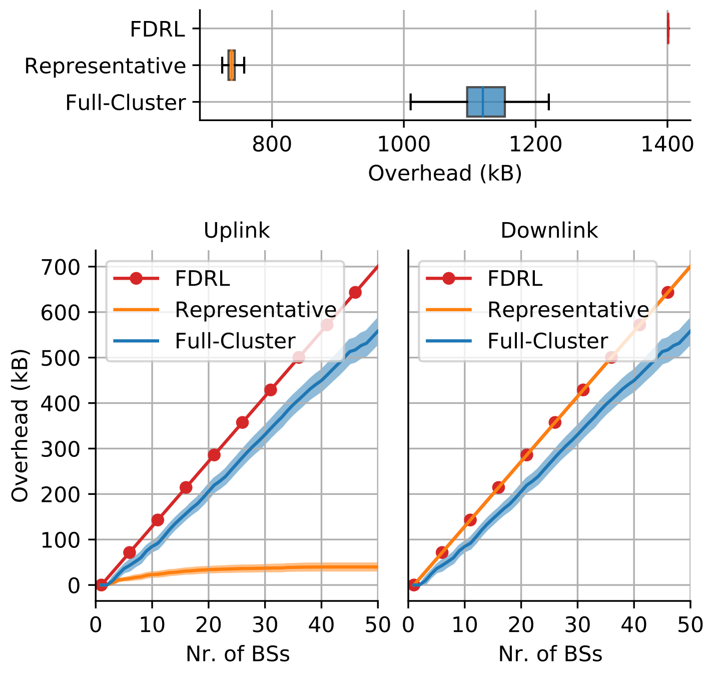


FIGURE 4.14: Communication overhead per federation episode for different federation strategies (top-part) and for different number of BSs deployed (bottom-part). RR and BR federation strategies are referred as Representative.

all the running DAs, in both directions. Conversely, the RR/BR approaches show an asymmetric behavior that favors the upload communication with respect to the downlink one, as only a single DA per cluster shares its local model during the federation process, resulting in a logarithmic trend (with respect to the number of BSs) characterizing the overhead in uplink. This would guarantee better scalability, at the expense of suboptimal performances, as shown in our evaluation. Finally, the FC approach shows a sublinear trend, with a slower growth rate than the benchmark FDRL, but with significant better performances thanks to the specialization of the DAs. It is safe to assert that the proposed dynamic clustering approach enhances the efficiency of the federated learning scheme, limiting the overall communication overhead with respect to traditional approaches, while providing better performances.

4.5.2.7 Power Consumption Comparison

Energy consumption is an important factor in federated learning schemes. In Fig. 4.15, we compare the power consumption of the different DRL strategies during training both in terms of CPU (left-hand side) and GPU (right-hand side), assuming they use the same computational platform as specified at the beginning of Sec. 4.5. We use `nvidia-smi` command line utility[3] to

[3]Part of the NVIDIA management library (NVML). Online available at `https://developer.nvidia.com/nvidia-management-library-nvml`

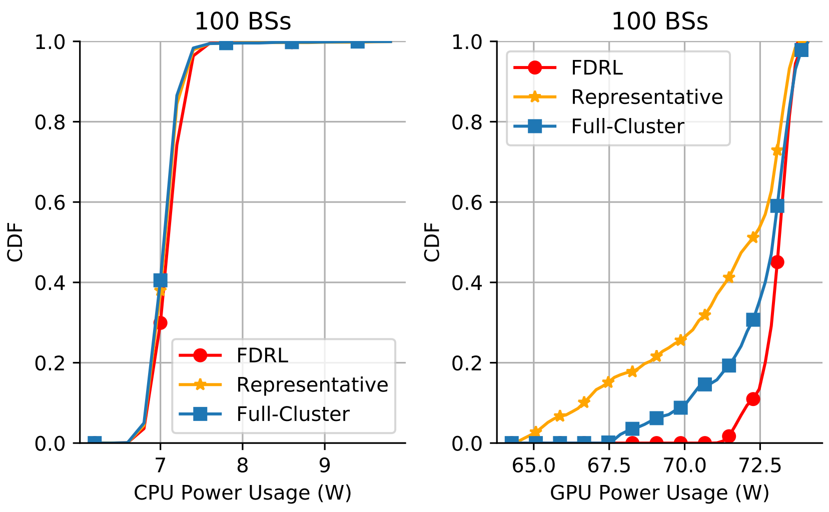


FIGURE 4.15: CPU and GPU power consumption for different federation strategies during agent training.

retrieve in real-time the energy consumption of the device, whereas for the CPU consumption we monitor the CPU utilization during the training, and consider a proportional fraction of the absorbed power at full computational load as declared by the vendor[4]. As we leverage the GPU hardware to train the models, the different federation strategies exhibit a similar impact on the power consumption of the CPU. Therefore, we focus on the GPU power consumption to better highlight their behavior. The obtained CDFs show that RR/BR schemes present lower consumption compared to FC and FDRL. Besides being positively influenced by the communication overhead variation depicted in Fig. 4.14, such reduced power consumption also results from the limited number of RR/BR agents involved in the federation process (selected through accurate clustering procedures, as shown in Sec. 4.3.3), when compared against baseline approaches.

4.6 Summary

Major research efforts in the *network slicing orchestration* area focus on designing solutions able to *concurrently and efficiently deal with both spatial and temporal aspects of users' traffic demand*. Due to the distributed nature of the RANs domain, centralized approaches are doomed to provide suboptimal performance and introduce significant communication overhead towards holistic resource controllers. In this Chapter, we addressed such challenging scenario

[4]https://ark.intel.com/content/www/us/en/ark/products/192437/intel-xeon-gold-6230-processor-27-5m-cache-2-10-ghz.html

and proposed an *FDRL-based architecture for network slice resource orchestration*, where *clusters* of decision agents are dynamically instantiated as virtualized instances with control over base stations radio resources. Enabled by the latest developments in federated learning, our approach allows building specialized knowledge from traffic and mobility patterns by exploiting similarity metrics. Our results show that the proposed *FDRL-based architecture* poses a trade-off involving the minimization of the communication overhead and the specialization of the decision agents, which in turn affects their accuracy along the resource allocation process. The main contributions in this chapter can be summarized as follows:

- We cast the RAN resource allocation problem as an optimization problem, focusing on minimizing the traffic exceeding SLA and assessing its complexity.
- We propose a distributed architecture for RAN slice resource orchestration based on DRL, composed of multiple AI-enabled decision agents that perform local radio allocation decisions without the need for a centralized control entity.
- We design a FL scheme composed of multiple parallel layers, one for each slice, to enhance the capabilities of the local decision-making process, following the recent development of the Open RAN architecture.
- We further improve the decision process by dynamically defining the subset of decision agents to be involved in the federation process, based on long-term slice traffic demands variations and their temporal similarities.
- We validate our hierarchical architecture and assess its capabilities in realistic scenarios by means of an exhaustive simulation campaign, accounting for a wide geographical area and thousands of end-users.

Chapter 5

Explainable Zero-Touch Network Slicing

Since 6G inherently underpins AI, we propose in this chapter a systematic and standalone slice termed SliceOps that is natively embedded in the 6G architecture, and which gathers and manages the whole AI lifecycle through monitoring, re-training, and deploying the ML models as a service for the 6G slices. By MLOps in conjunction with XAI, SliceOps strives to cope with the opaqueness of black-box AI using XRL to fulfill transparency, trustworthiness, and interpretability in the network slicing ecosystem. This chapter starts by elaborating on the architectural and algorithmic aspects of SliceOps. Then, the deployed cloud-native SliceOps working is exemplified via a latency-aware resource allocation problem. The DRL-based SliceOps agents within slices provide AI services aiming to allocate optimal radio resources and impede service quality degradation. Simulation results demonstrate the effectiveness of SliceOps-driven slicing.

5.1 Explainable Machine Learning Operations (MLOps)

6G slicing is envisioned as a disruptive technology to support massive slicing with micro or macro services. Consequently, the complexity of automated MANO operations would rise dramatically. When it comes to catering to the services with strict reliability and latency requirements, 5G NR falls under the URLLC services that have gained traction in industry and academia. The URLLC slice is mandated to address the sporadic nature of this time-critical traffic by relying on Gbps data rates and millisecond latency.

The tendency towards fully automated MANO in B5G/6G has spurred intensive research interest in applying AI and ML as an ideal solution for various nonlinear problems. The main challenge for adopting dominant AI methods in telecommunication environments is the need for AI models tailored to specific services and requirements. Notably, novel practices are required to deploy ML solutions into production, keeping the ML models relevant and re-training them to cope with potential change of conditions, as well as providing a trustworthy, highly accurate, timely, and actionable AI-driven slicing environment. In this intent, the ML model deployed in production should inescapably undergo the combination of DevOps processes. While DevOps is an outgrowth of agile software development accelerated with the cloud-native environment, it is called ML Operations (MLOps) when applied to ML, where telecom companies must maintain ML pipelines in a production environment to proactively monitor, unveil, and measure ML models' quality to improve the network's automation.

On the other hand, following the technical report of the EC on "Ethics guidelines for trustworthy AI" [104], AI solutions should pursue trustworthiness. Due to a lack of transparency and trust in AI models, telecommunication operators are wary of widespread AI model deployment in their networks, especially when the decisions thereof have both financial and service quality implications. Thus, an open challenge is to integrate explainability with MLOps, to ensure explanation-guided learning. Indeed, the XAI approach scrutinizes each feature and its impact on the output of the AI model, enabling to observe the factors that either positively or negatively impact the AI model prediction.

In this regard, XRL is viewed as a responsible and trustful ML approach that can be combined with MLOps lifecycle. Indeed, in the RL method, the agent generates the corresponding dataset on the fly by interacting with the network. This method is an evaluative and feedback-based learning to optimize the accumulated long-term reward. It is assisted with DL in DRL to surmount the curse of dimensionality concerning large state spaces. To fully exploit the potential of the DRL algorithm, we need to streamline the conflict-prone nature of action selection due to the complex relation of state-action. This defect can be highly crippling the promising RL solutions in automated network slicing. In this intent, XAI can assist DRL to extract more relative state-action pairs where it explains which state or input of agent has the most positive impact on action or decision. This solution can pave the way to apply the DRL models for time-sensitive applications such as xURLLC, where the decision should be enforced to the network promptly and robustly.

To incorporate these principles into a single design while ensuring a separation of concerns, this chapter introduces *SliceOps*, an XAI-empowered MLOps framework that is natively embedded in the 6G network architecture as a standalone slice. To illustrate its operation, we consider a latency-aware resource allocation problem where each slice registers to the corresponding SliceOps instance which provides AI services via a SliceOps agent. The main goal is to allocate optimal radio resources to the slices while minimizing the latency to meet the SLAs.

5.2 Explainable Automation-Native Slicing (SliceOps)

The orchestration of large-scale network slices across multiple domains needs AI solutions. Since 6G natively supports AI, we propose—as a separation of matters—to concentrate AI/ML operations (MLOps) in a standalone slice called SliceOps, which manages the whole lifecycle of AI models separately and provides AI services to the main 6G slices.

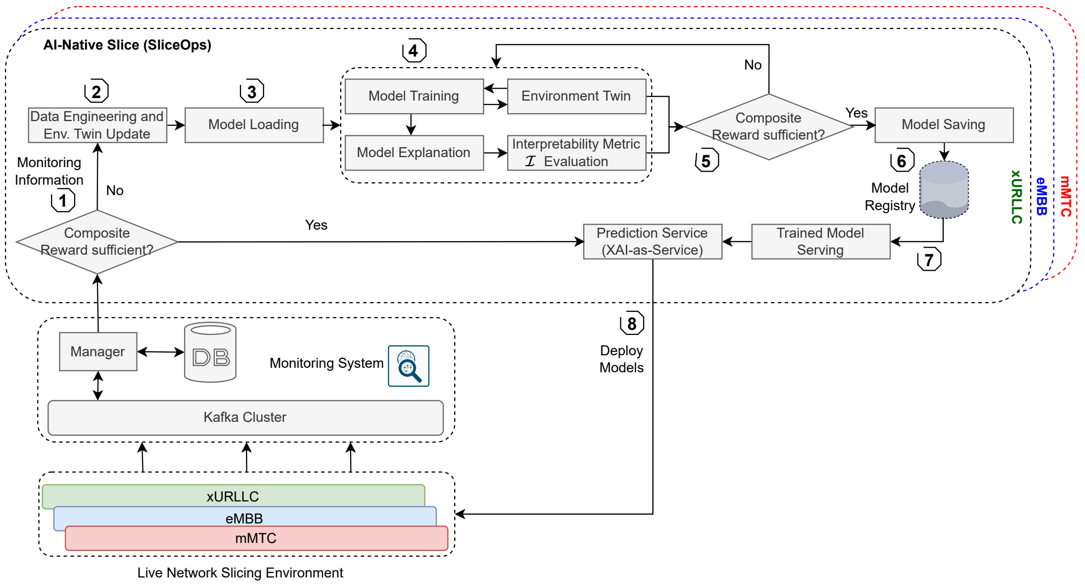

AI-Native Slice (SliceOps)
4
2
3
Data Engineering and Env. Twin Update
Model Loading
Model Training
Environment Twin
Model Explanation
Interpretability Metric $\mathcal{I}$ Evaluation
No
Composite Reward sufficient?
Yes
Model Saving
5
6
Model Registry
Monitoring Information
1
No
Composite Reward sufficient?
Yes
Prediction Service (XAI-as-Service)
Trained Model Serving
7
8
Deploy Models
Manager
DB
Monitoring System
Kafka Cluster
xURLLC
eMBB
mMTC
Live Network Slicing Environment

FIGURE 5.1: The SliceOps workflow comprises of monitoring, re-training, and deploying ML model as a service.

5.2.1 SliceOps Lifecycle

The SliceOps foundation relies on a practice aiming to standardize production methods through incorporating the concept of CI and CD, producing thereby reliable software and AI solutions in short cycles. In 6G AI-native networks, SliceOps modular design allows the evolution, upgrade, and scaling of the MLOps layer and its AI functions separately from the service layers. Fig. 5.1

depicts the pictorial representation of an operationalizing AI-native slice deployed at the top of the network slicing environment with multiple SliceOps instances. The instances provide AI services to the corresponding slice (e.g., URLLC), and they can collaborate for specific tasks like resource allocation, where resources are generally shared between service slices. On the other hand, SliceOps guarantees AI performance isolation through triggering re-training, which means that if a sliceOps instance degrades for one slice, it will have minimum effect on the other slices. This is in line with containerized solutions that decouple the execution of iterative processes which is a necessity in ML pipelines. The SliceOps adheres to the below fundamental pipeline principles based on RL algorithms:

5.2.1.1 Monitoring System and Data Collection (1)

It is considered as a backbone step in ML practices where data acquisition is utilized for data preparation and model design processes. It allows SliceOps agents that run as container instances within slices to collect real-time monitoring data from the gNB platform of multiple KPIs, encompassing available resources, number of connected devices, bandwidth utilization, channel quality, etc. This component can store structured and unstructured data on a very large scale. Such data is streamed through, e.g., a Kafka bus to which the SliceOps AI functions can subscribe and fetch the relevant data under a specific Kafka topic name. The obtained data is used as input of the pipeline to guide the definition of policy in the form of service prediction. The SliceOps RL-based agent collects data on-the-fly (online RL) interacting with the slice environment or initializes the process with a pre-collected dataset (offline RL) stored in an internal database. The online RL can bring additional risks in terms of interacting with live environment and collected data. To solve this, we consider steps 2, 6, and 8.

5.2.1.2 Data Engineering and Model Loading (2, 3)

This component is responsible for data preprocessing or preparation. Different optimization targets and AI services require heterogeneous training data for neural networks. This process takes raw data from the monitoring system and transforms it into an understandable format for RL, such as OpenAI Gym. The raw data contains errors and inconsistencies while having various attributes or patterns. For example, the monitored data of different slice domains can be non-Euclidean or not meet IID dataset features. With the various forms of data in network slicing, pursuing the techniques such as assessment of data quality, data cleaning, data transformation, reduction of data, etc., is a vital step for better learning performance. The next step is to create neural network architecture and hyperparameters tuning before compiling and loading them into a model. The hyperparameter tuning depends highly on capability, scenario, and technology used [56]. Different datasets require setting different hyperparameters to guide the model to predict accurately in the following steps.

5.2.1.3 Model Training, Evaluation and Saving (4, 5, 6)

This module of SliceOps is segregated into a set of processes for the execution of continuous model training automatically with processed data. The ML model training runs a local optimization task, while the model explanation involves an *explainer*, either attribution-based (e.g., Integrated Gradient, Saliency Maps) or perturbation-based (e.g., SHapley Additive exPlanations (SHAP) [105]). Upon the evaluation of RL reward, including interpretability metric, the model retraining is triggered whenever there is a model performance deterioration following new training data arrival (new unexplored states). Note that the reward should also correlate with slice targets (SLA, KPIs, etc.), while the interpretability refers to XAI metrics (attributions-based entropy, confidence, log-odds, fidelity, etc.) depending on the SLA adopted by the slice tenant. In this respect, a feedback loop between the explainer and the model is necessary to feed the model optimizer with the measured XAI metrics, thereby enabling explainability-aware learning. Finally, in case the model fulfills the target performance, it will be registered and stored in the model registry to be promoted into production later on.

5.2.1.4 AI Service Provisioning (7, 8)

The next step after training and evaluation is to retrieve the registered model for encapsulation and move to the production stage in the form of a model prediction service. The trained model is continuously exposed and delivered to slices as REST API assisted with SliceOps agents. The framework considers continuous monitoring in different steps to ensure model performance.

5.2.2 Explainability in RL

The aforementioned agents are assumed to be DRL-based. Unlike the conventional DRL where there is no causal relation between the input state parameters and the output action, this paper introduces an attention mechanism based on SHAP explainer, which quantifies the relevance of

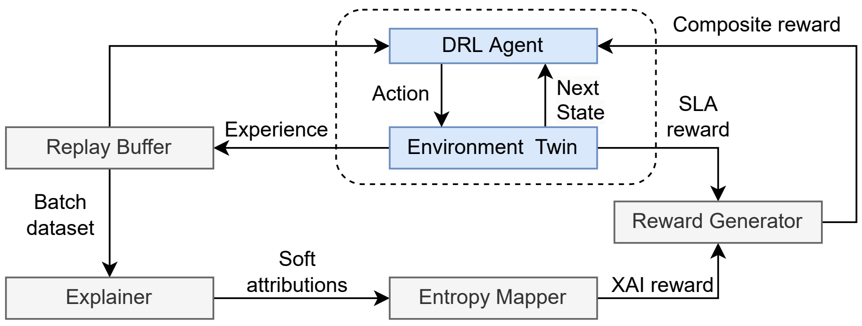


FIGURE 5.2: The workflow of SliceOps model training based on explainable DRL.

a state to the action and guides the XRL agent to perform explainable decisions through XAI-augmented reward shaping. The training workflow of the proposed XRL scheme is illustrated in Fig 5.2, which is composed of the following main components,

5.2.2.1 Explainer

It explains the DRL decision by assigning high scores (in absolute value) to the most influencing input state parameters. The score corresponds to the SHAP value, which is computed using, e.g., a perturbation-based approach. Specifically, each feature is perturbed or modified while keeping other features fixed at their baseline values (e.g., white noise, zero). The model's response is observed by evaluating the perturbed instances and recording the corresponding predictions. The differences between the predictions of the perturbed instances and the baseline prediction are computed to capture the contribution of each feature when changed from the baseline value. In Sec. 5.3.2, we demonstrate these contributions are aggregated across different perturbations to estimate the SHAP values. Following the DRL agent interaction with the Environment Twin, it temporarily saves the experiences and observations in a replay memory/buffer which is steadily updated. Then, it generates the SHAP importance values over an extracted batch dataset of state-action.

5.2.2.2 Entropy Mapper

It applies a *softmax* layer to the SHAP values provided by the *Explainer* and consequently generates a probability distribution, which is used afterward to calculate the entropy that measures uncertainty of the taken action given the input state.

5.2.2.3 Composite Reward Signal

The obtained multiplicative inverse of the maximum entropy value is used as XAI reward. In Sec. 5.3.2, we showcase that the composite reward—which is a sum of the SLA reward (based on meeting or violating the SLA requirements) and the XAI reward—results in minimizing the uncertainty of state-action pairs and encouraging the agent to select the best actions for specific network state values. This approach can elucidate the learning process while directing the learning toward making explainable decisions concerning a specific state.

5.2.3 Benefits of SliceOps

The main objectives of segregating the control plane from the user plane in 5G and beyond 5G networks encompass scalability, flexibility, and agility to streamline the development of new services and use cases. The proposed SliceOps approach aligns with this philosophy by creating an innovative standalone AI plane to independently provide AI services and functions to the rest of the network slices. It eliminates the necessity to modify the underlying AI functions and structure of network slice instances for new services. By a decoupled AI plane from the control plane and user plane, the network can upgrade or add new AI and automation functions without impacting other planes. This flexibility and mentioned explainability features of SliceOps pave the way for faster deployment and trustworthy network optimizations in network slicing.

5.2.4 ETSI ZSM Compliance

The design of SliceOps solution closely adheres to the ZSM framework. The standardization process for ETSI ZSM is still in its early stages, with preliminary specifications based on a high level of abstraction. The core concept of the closed-loop AI system is to utilize context-aware and metadata-driven policies to more efficiently and quickly identify and incorporate new while updating knowledge and making robust and actionable decisions. As shown in Fig. 5.1, SliceOps is a practice toward deploying a more realistic closed-loop scheme to fulfill viable automation solutions for network slicing control. SliceOps manages the lifecycle of AI models in a closed-loop way that includes model performance monitoring, re-training, and delivery. It extends the ZSM framework to manage, besides the service functions, the underlying AI functions, which are natively supported in a standalone slice. Moreover, the SliceOps agents are based on the closed-loop workflow in Fig. 5.2.

5.2.5 O-RAN Slicing Compliance

The SliceOps framework can be adapted to use case 3 of open radio access network (O-RAN) slicing, which is related to resource allocation optimization, requirements, and architecture [2]. Specifically, SliceOps layer can be developed as rApps at the non-real-time (Non-RT) radio intelligent controller (RIC). Thanks to its architecture that brings valuable differentiators by leveraging XAI and CI/CD, SliceOps would implement new use cases with agility and automate network operations. In this respect, rApps need to access abundant monitoring data—such as traffic load, latency, and signal strength—through the O1 interface to efficiently carry out their designated functions in time-demanding training procedures. Then, the model or policy trained by SliceOps can be packaged as an artifact and delivered via the A1 interface to run on the Near-RT RIC interface as xApp. For dynamic network optimization purposes, this xApp would control the underlying O-RAN components, namely, the central unit-control plane (O-CU-CP), different slice open central unit-user plane (O-CU-UP), and open distributed unit plane (O-DU) by using the E2 interface.

5.3 SliceOps Use Case

The next important step after introducing the key elements of the proposed framework is to exemplify the SliceOps approach and evaluate the performance to verify the benefits of XAI in the MLOps pipeline. In this section, we demonstrate the implementation of the small-scale of SliceOps framework for network slicing. Then the effectiveness of the model is validated in terms of ameliorating the long-term revenue (average reward), transmission latency, dropped traffic, and XAI metric.

5.3.1 Network Architecture and Experiment Parameters

For the sake of validating the SliceOps framework in realistic settings, we consider a gNB scenario, wherein a set of slices $\mathcal{I}$ is deployed. The scenario includes *three* slices, i.e., URLLC, eMBB, and mMTC. The slices are characterized by the SLA latency $\Lambda_i = [10, 40, 20]$ ms, respectively. Without loss of generality, the considered gNB is characterized by the radio capacity $C = 100$ PRBs of a fixed bandwidth and assumes the slices running over gNB simultaneously. The slice traffic demand is modeled as the realization of a Poisson distribution with mean value

λ_i and emulates the SNR variability extracting its instantaneous values from a Rayleigh distribution with the average value set to 25 dB. We set $\iota = 10$ PRBs as the minimum resource allocation step. The framework leverages Python programming language, exploiting OpenAI Gym library [69] and interfacing DRL agents with a custom gNB simulator environment [106]. The simulator consists of virtual transmission queues and main PHY/MAC/RLC functionalities.

Each SlicOps agent is endowed with a DDQN[7]. The agents interact with each other and O-DU through the O-RAN E2 interface to gather the slice networking statistics (e.g., channel quality, served traffic, consumed resources, etc.). Then, they enforce PRB policy decisions provided by the corresponding SliceOps layer in the gNB slice scheduler. We use a dedicated server equipped with two Intel(R) Xeon(R) Gold 5218 CPUs @ 2.30GHz, two NVIDIA GeForce RTX 2080 Ti GPUs, and the DNNs are implemented based on TensorFlow-GPU version 2.5.0. The neural network architecture uses two fully connected layers with *24* neurons activated by ReLU function. The network parameters are updated using the Adam optimizer. The discount factor γ and learning rate ξ are set to be *0.99* and *0.001*, respectively. The replay buffer size of each agent β_i is *20000* samples, out of which a batch of *32* samples is extracted for each training interval. To deploy the solution in the cloud-native mode, we leveraged a containerized approach where a cloud server hosts SliceOps instances and corresponding modules responsible for providing AI service for different problems in slices. On the other hand, SliceOps agents of slices run by using the Docker compose tool and communicate with the server through FastAPI as a REST API.

The operational training phases are discussed in Sec. 5.2.2. All together, we formulate the local optimization task as:

Problem `RAN Slice Resource Allocation`:

$$\min \lim_{T\to\infty} \sum_{t=1}^{T} \mathbb{E}\left[\sum_{i\in\mathcal{I}} d_{i,b}^{(t)}\right] \tag{5.1}$$

subject to:

$$E_{i,b}^{(t)} \leq \Lambda_i, \qquad \forall t\in\mathcal{T}, \forall i\in\mathcal{I}, \forall b\in\mathcal{B}; \tag{5.2}$$

$$\sum_{i\in\mathcal{I}} a_{i,b}^{(t)} \leq C_b, \qquad \forall t\in\mathcal{T}, \forall b\in\mathcal{B}; \tag{5.3}$$

$$a_{i,b}^{(t)} \in \mathbb{Z}_+, d_{i,b}^{(t)} \in \mathbb{R}_+, \qquad \forall t\in\mathcal{T}, \forall i\in\mathcal{I}, \forall b\in\mathcal{B}; \tag{5.4}$$

TABLE 5.1: DRL Parameters

Parameter	Type	Description
$s_i^{(t)} = \{(\sigma_i^{(t)}, \lambda_i^{(t)}, \nu_i^{(t)}) \mid \forall i \in \mathcal{I}\}$,	State Space	At a given time t, $\sigma_i^{(t)}$ is the average SNR value experienced by the users in the i-th slice over a decision time interval. $\lambda_i^{(t)}$ is the total traffic volume generated by the i-th slice during this time interval, and $\nu_i^{(t)}$ represents the remaining available capacity after considering previous allocation decisions made by other agents.
$\mathcal{A} = \{\iota \cdot k \mid k = \{0, 1, \ldots, \frac{C}{\iota}\}\}$	Action Space	We set ι as the smallest unit of PRB allocation, also known as the *chunk size.* The PRB allocation decisions made by the i-th agent must be in multiples of ι, creating a discrete action space.
$r_i^{(t)} = \begin{cases} \alpha_i^{(t)} - 4\rho_{\text{lower}}^{(t)} & \text{if} \quad \alpha_i^{(t)} < \rho_{\text{lower}}^{(t)}, \\ (1 - \frac{\alpha_i^{(t)}}{\rho_{\text{up}}^{(t)}})\frac{\alpha_i^{(t)}}{\rho_{\text{up}}^{(t)}} & \text{if} \quad \rho_{\text{lower}}^{(t)} \leq \alpha_i^{(t)} \leq \rho_{\text{up}}^{(t)}, \\ -(\alpha_i^{(t)} - \rho_{\text{up}}^{(t)}) & \text{if} \quad \alpha_i^{(t)} > \rho_{\text{up}}^{(t)}. \end{cases}$	Reward	We assess the quality of the action by introducing two variables, $\rho^{(t)}$up and $\rho^{(t)}$lower, which define the upper and lower bounds of the allocation gap as $\rho^{(t)}\text{up} = 2 \cdot \Gamma(\iota^{(t)}, \sigma_i^{(t)})$ and $\rho^{(t)}\text{lower} = -\Gamma(\iota^{(t)}, \sigma_i^{(t)})$

where $E_{i,b}^{(t)} = \mathbb{E}\left[\frac{\varphi_{i,b}^{(t)}}{\Gamma\left(a_{i,b}^{(t)}, \sigma_{i,b}^{(t)}\right) + d_{i,b}^{(t)}}\right]$ defines the expected transmission latency, and $\Gamma(a, \sigma)$ is a function that translates the PRB allocation a in the equivalent transmission capacity, given the experienced channel quality σ. The traffic demand generated within a decision interval might not be fully satisfied due to erroneous PRB allocation estimations, incurring in additional transmission latency due to traffic queuing at the base station. Therefore, we introduce the variable $d_{i,b}^{(t)}$ as a deficit value indicating the volume of traffic not served within the agreed slice latency tolerance Λ_i, and that is therefore dropped.

The abovementioned optimization task can be solved by invoking the DRL framework, wherein the state and action spaces as well as the reward are summarized in Table 5.1.

A novel approach to measure the confidence of DRL decisions is to observe the distribution of state-features SHAP values in the replay buffer dataset. Specifically, the probability distribution of the states-features is generated as,

$$p_{l,k} = \frac{\exp\left\{\left|\alpha_{l,k}\right|\right\}}{\sum_{l'=1}^{L} \exp\left\{\left|\alpha_{l',k}\right|\right\}}, \; l' = 1, \ldots, L, \tag{5.5}$$

where $\alpha_{l,k}$ stands for the SHAP value corresponding to state l of sample k in the replay buffer dataset. The decision is viewed as certain when high attributions (in absolute value) are more concentrated in some features, thereby minimizing the Shannon entropy,

$$\mathcal{H}_k = -\sum_{l=1}^{L} p_{l,k} \log(p_{l,k}). \tag{5.6}$$

In this respect, we introduce what we call XRL reward, which is defined as the multiplicative inverse of the entropy, i.e.,

$$r_{\mathrm{XRL}}^{(t)} = \frac{1}{\max_k \mathcal{H}_k} \tag{5.7}$$

Finally, the composite reward fed back to the DRL agent is given by,

$$r_{i,b,c}^{(t)} = r_{i,b}^{(t)} + \mu r_{\mathrm{XRL}}^{(t-1)} \tag{5.8}$$

The single agent training procedure is summarized in Algorithm 7.

5.3.2 Latency-Aware Resource Allocation

Acquiring swift and constructive resource allocation in network slicing is precluded due to the lack of dynamic traffic steering. We cast the radio resource allocation problem in gNB as an optimization problem, emphasizing on minimizing allocated resources and latency to meet the SLA. We consider the transmission latency as the average time that traffic of a slice experiences before being served within the gNB transmission buffers due to the inter-slice scheduling process. The radio resource availability for the downlink traffic is divided into subsets of PRBs. The provided AI model by SliceOps instances lets the SliceOps agent dynamically assign the optimum PRBs to each network slice following the real-time traffic and SLA requirements. In this scenario, we consider correct and fair dimensioning of the inter-slice PRB enforcement instead of focusing on the intra-slice scheduling issue.

5.3.2.1 Long-Term Revenue (Average Reward)

As shown in Fig. 5.3, the designed composite reward function (XRL strategy) assisted by the SHAP approach guarantees better learning generalization and robust performance compared

Algorithm 7: Single XRL-Agent Resource Allocation

```
Initialize primary network $\theta$ and target network $\tilde{\theta}$, and replay buffer $\beta$,
Import network slicing environment (‘XRL–v2’),
Initialize action space $\mathcal{A}$ and state space $\mathcal{S}$
t=0
while $t$ ¡ max_timesteps do
    if $t$ ¡ start_timesteps then
        Initial buffer filling: $a_{i,b}^{(t)}$ = env.action_space.sample()
    else
        Observe state $s_{i,b}^{(t)}$ and select $a_{i,b}^{(t)} \sim \pi(s_{i,b}^{(t)}, a_{i,b}^{(t)})$
    end
    Execute $a_{i,b}^{(t)}$ and observe $s_{i,b}^{(t+1)}$ and $r_{i,b}^{(t)} + \mu r_{\mathrm{XRL}}^{(t-1)}$:
      next_state, reward, done,
    Store new transition $(s_{i,b}^{(t)}, a_{i,b}^{(t)}, r_{i,b}^{(t)}, s_{i,b}^{(t+1)})$ into $\beta_{i,b}$
    if $t \geq$ start_timesteps then
        Sample batch of transitions $\tilde{\beta}_{i,b}$
        Calculate XRL reward $r_{\mathrm{XRL}}^{(t)}$ according to (5.7)
        Compute target Q value
        Perform a gradient descent step on:    $(y_{i,b}^{(t)} - Q(s_{i,b}^{(t)}, a_{i,b}^{(t)}, \theta_{i,b}^{(t)}))^2$
        Update target network parameters:    $\tilde{\theta}_{i,b}^{(t)} \longleftarrow \tau\theta_{i,b}^{(t)} + (1-\tau)\tilde{\theta}_{i,b}^{(t)}$
    end
    if done then
        obs, done = env.reset(), False
    end
    t=t+1
end
```

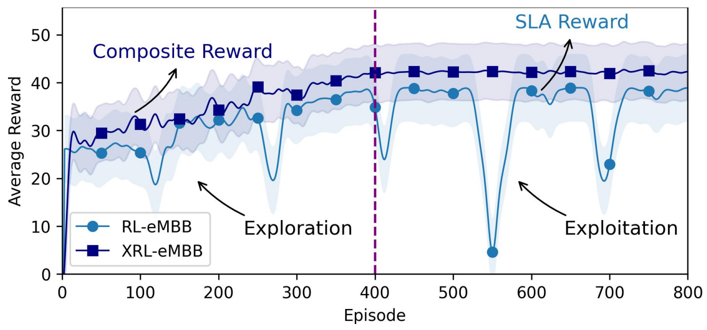


FIGURE 5.3: Convergence performance of the RL and XRL approaches. For the sake of visual clarity, the curves are smoothed concerning confidence bands and standard deviation.

to the RL method (SLA reward) as the baseline. In around half of the overall training, the eMBB SliceOps agent inspects action space (PRBs) that initially leads to high fluctuations in learning curves, i.e., exploration, and then strives to achieve the right trade-off between the learned decision policies and varying network states, i.e., exploitation. It is safe to assert that the composite reward based on the proposed explanation-guided action-selection strategy enhances the performance of SliceOps agents. The SHAP explainer extracts the features and their importance values from the batch dataset for a particular prediction in conjunction with the entropy mapper to provide a reward metric to guide the agent with more relevant state-action pairs.

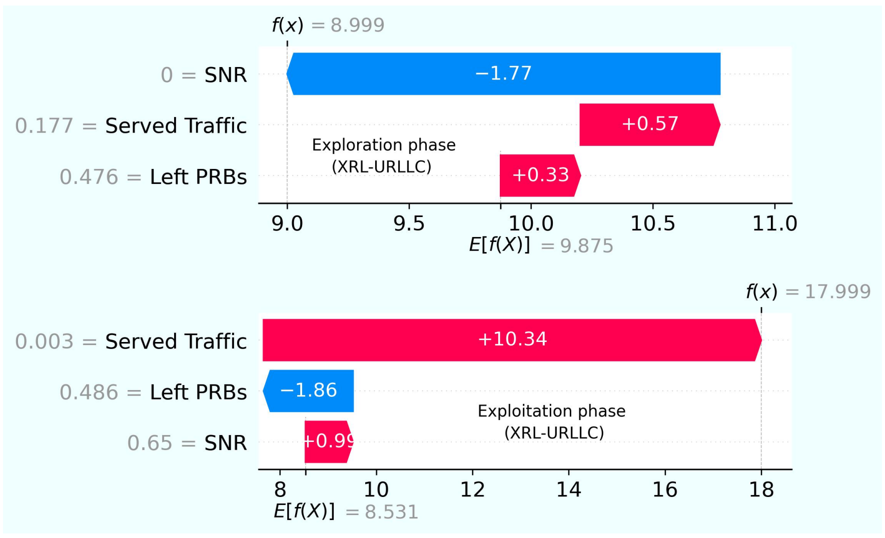


FIGURE 5.4: The XAI waterfall plot illustrates the contribution (either positive or negative) of a given input state parameter to the output decision for URLLC slice. (Up) Exploration phase, (Down) Exploitation phase.

5.3.2.2 Explaining Feature Importance

In Fig. 5.4, we observe that $f(x)$ represents the predicted action taken by the XRL agent, which involves allocating PRBs to the URLLC slice. Meanwhile, $\mathbf{E}[f(x)]$ represents the expected value or the average of all possible actions. The absolute SHAP value provides insights into the influence of a single state on the action taken. During the initial stages of training, the agent behaves as a Max C/I scheduler, resulting in a penalty for URLLC users experiencing a low SNR state (SHAP value $= -1.77$). Consequently, this leads to a relatively low allocation of PRBs per slice, specifically 8.99 PRBs. Contrastingly, in episode 500, which signifies the exploitation phase where the agent has learned the optimal policy, the agent's action primarily depends on the served traffic (SHAP value $= 10.34$). As a result, a higher allocation of PRBs, specifically 17.99 PRBs, is observed. In Fig. 5.4 (Up), the exploration entails engaging in actions with uncertain PRBs allocation policy aims to acquire deep insights about the network environment to learn better optimal strategies. Fig. 5.4 (Down) presents the results in the exploitation phase, where the agent leverages the learned knowledge in the exploration phase and executes PRB actions that are estimated to yield the highest rewards.

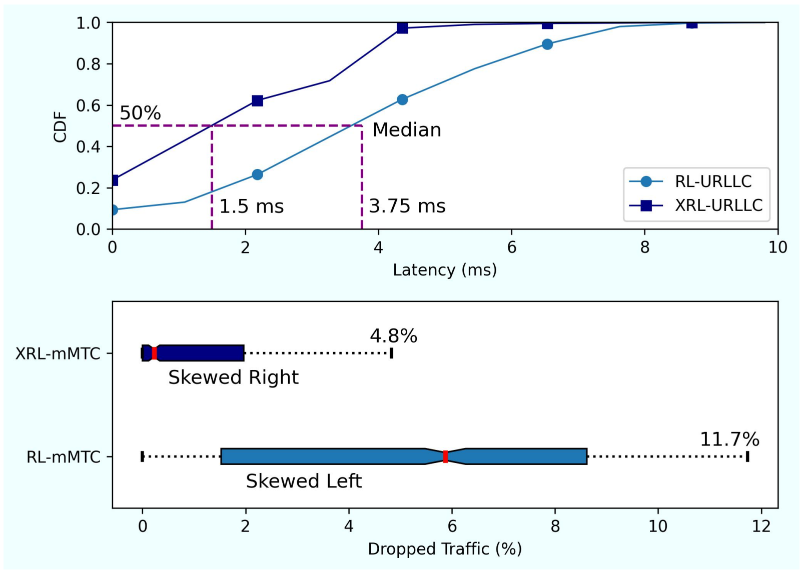


FIGURE 5.5: Network performance comparison for RL and XRL settings. (Up) The transmission latency CDF for URLLC slice, (Down) The performance evaluation in terms of dropped traffic for mMTC slice.

5.3.2.3 Transmission Latency

Fig. 5.5 (Up) depicts the cumulative distribution function (CDF) of the time that URLLC traffic experiences within the gNB transmission buffer, resulting from RL and XRL-based SliceOps resource allocation. From the results, it can be noticed that the XRL approach leads to higher performance where *50%* of perceived latencies in the URLLC slice is less than *1.5 ms*, whereas this value for the RL solution is *3.75 ms*. The performance of XRL-SliceOps agents reveals that they allocate adequate radio resources proportional to the traffic demand while handling varying resource contention among slices. In contrast, the defective collaboration among agents in the RL strategy and the erroneous trade-off between resource allocation and network conditions (state-action pairs) result in higher incurred latency.

5.3.2.4 Dropped Traffic

We continue the performance analysis on the proposed XRL approach by shedding light on the volume of dropped traffic that does not meet latency SLA requirements owing to mistaken radio resource allocation policies, as showcased by Fig. 5.5 (Down). The box plot illustrates the highest value of mMTC dropped traffic, excluding outliers for the XRL scheme, is *4.8%*, whereas this value for the RL method is *11.7%*. Besides, the lopsided box plot of XRL is positively skewed where the mean value is greater than the median, i.e., the majority of the values are located on the left side (lower dropped traffic values). In contrast, mMTC slice experiences higher dropped traffic by RL solution where the box plot indicates a few exceptionally small dropped traffic and most values are large, which results in the mean being pulled to the left.

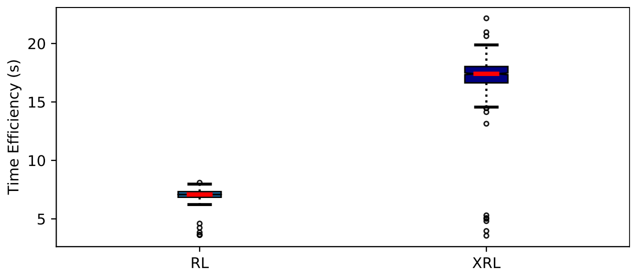


FIGURE 5.6: Comparison of algorithms in terms of time efficiency on the network slicing setup.

5.3.2.5 Time Efficiency

Fig. 5.6 reflects about time efficiency comparison among RL and XRL algorithms during the whole training operation. The box plot illustrates that the median line of the box for RL and XRL is at *7.5s* and *17.5s*, respectively. At a glance, we can explicate that the RL algorithm takes less processing time to complete episodes than XRL. However, as shown in Fig. 5.3, XRL converges faster, which can compensate this higher complexity compared to the RL approach. Note that there are some outliers for each box between *2.5s* and *4s*, which is the period of filling the replay buffer initially as a part of training different DRL algorithms.

5.4 Summary

Algorithmic and architectural innovations are required to streamline 6G slicing automation in future networks. This chapter has introduced SliceOps, a framework for automation-native 6G networks, where the AI operations (MLOps) are gathered in a standalone slice that provides AI service to the rest of slices. This AI slice extends the ZSM closed-loop to the AI lifecycle management. Moreover, explainability-guided learning is adopted in SliceOps to ensure trust in and robustness of the DRL agents. Both AI and network performance results underpin the proposed framework. The following contributions are presented in this chapter:

- We introduce the architecture of SliceOps, where the explainable ML operations are gathered in a standalone slice providing AI services to the rest of the slices. This continuous delivery (CD) and continuous integration (CI) of ML models enhances reliability and interpretability while quickly deploying AI models in the network with higher consistency.

- As a use case of SliceOps, a RAN resource allocation problem is defined, aiming at reducing SLA violations.
- To solve this problem, SliceOps agents are proposed, which are based on a novel explanation-guided DRL (XRL) scheme that is assisted with SHAP importance values and an entropy mapper to guide the agent in reducing uncertainty in its actions across various network states.
- The AI and network analysis demonstrate the superiority and faithfulness of the proposed explanation-guided DRL approach compared to the RL baseline.

Chapter 6

Dissemination and Exploitation of Research Results

6.1 Patent

1) **F. Rezazadeh**, H. Chergui, L. Christofi, and C. Verikoukis, “Continuous Network Slicing in a 5G Cellular Communications Network via a Delayed Deep Deterministic Policy Gradient”, OBI (Greek Patent Office), No. GR1010062B, 2021-08-04.

6.2 Journal

1) **F. Rezazadeh**, H. Chergui, L. Alonso, and C. Verikoukis, “SliceOps: Explainable MLOps for Streamlined Automation-Native 6G Networks”, IEEE Wireless Communications, 2023, under review.

2) **F. Rezazadeh**, L. Zanzi, F. Devoti, H. Chergui, X. Costa-Pérez, and C. Verikoukis, “On the Specialization of FDRL Agents for Scalable and Distributed 6G RAN Slicing Orchestration”, IEEE Transactions on Vehicular Technology, 2022.

3) A. Dalgkitsis, L. A. Garrido, **F. Rezazadeh**, H. Chergui, K. Ramantas, J. S. Vardakas, and C. Verikoukis, “SCHE2MA: Scalable, Energy-Aware, Multi-Domain Orchestration for Beyond-5G URLLC Services”, IEEE Transactions on Intelligent Transportation Systems, 2022.

6.3 Conference

1) S. Roy, **F. Rezazadeh**, H. Chergui, and C. Verikoukis, “Joint Explainability and Sensitivity-Aware Federated Deep Learning for Transparent 6G RAN Slicing”, IEEE ICC, 2023.

2) **F. Rezazadeh**, L. Zanzi, F. Devoti, H. Chergui, and X. Costa-Pérez, “A Federated Deep Reinforcement Learning Approach for Distributed Network Slicing Orchestration”, IEEE Med-itCom, Extended Abstract, 2021.

3) **F. Rezazadeh**, H. Chergui, L. Blanco, L. Alonso, and C. Verikoukis, “A Collaborative Statistical Actor-Critic Learning Approach for 6G Network Slicing Control”, IEEE GLOBE-COM, 2021.

4) **F. Rezazadeh**, H. Chergui, L. Christofi, and C. Verikoukis, “Actor-Critic-Based Learning for Zero-touch Joint Resource and Energy Control in Network Slicing”, IEEE ICC, 2021.

5) **F. Rezazadeh**, H. Chergui, L. Alonso, and C. Verikoukis, “Continuous Multi-objective Zero-touch Network Slicing via Twin Delayed DDPG and OpenAI Gym”, IEEE GLOBECOM, 2020.

6.4 Chapter

1) M. Maule, O. Kohjogh, and **F. Rezazadeh**, “Enabling 6G Mobile Networks, Advanced Cloud-Based Network Management for 5G C-RAN”, Springer Nature, book chapter, 2021.

Chapter 7

Conclusions and Future Work

The technology of network slicing is seen as a significant advancement in the upcoming 5G mobile networks and beyond, which has the potential to introduce new participants into the mobile ecosystem and facilitate innovative business models. It is a promising solution for the challenges faced by modern networks, such as increasing traffic demands, diverse service requirements, and the need for efficient resource utilization. With network slicing, network operators can partition their infrastructure into multiple virtual networks, each tailored to meet the specific needs of different use cases and customers. This enables greater flexibility, agility, and customization in network management and service delivery. Furthermore, network slicing can provide significant benefits to various industries by enabling the deployment of specialized network slices optimized for their unique requirements. However, network slicing also presents several technical and operational challenges, such as the need for effective slice orchestration, security, and scalability. Overall, the potential benefits of network slicing outweigh its challenges, making it a promising technology for the future of networking. As network operators continue to embrace this approach and work towards developing robust network slicing solutions, we can expect to see significant advancements in network performance, reliability, and innovation in the coming years. We summarized the conclusions and contributions of this thesis in **Sec. 3.3**, **Sec. 4.6**, **Sec. 5.4**.

The results of this thesis revealed some challenges and limitations for transforming AI-native to automation-native RAN network slicing while presenting new opportunities that should be considered carefully in future research. The following sections bring attention to various challenges and opportunities.

7.1 Challenges and Limitations

Time-Sensitive Slicing

Network slicing should fulfill innovative services with time-critical characteristics requiring real-time or near-real-time management operations and decision-making. AI enables 5G/6G networks to become functional through predictive and proactive abilities. By fusion of ML models and processes in 6G technology, the telecom domains can make intelligent decisions independently, improving slices' latency and reliability. MLOps can speed up the different steps of AI operations in slices, such as data collection and preparation process, and provide faster response to changed conditions of the network. However, it needs an extensive process and long-time E2E procedure, and this issue becomes more critical for highly dynamic slice environments with big data. Thus, optimizing the training time and ameliorating inference model performance is necessary to harness the full potential of MLOps into a slice instance.

Secure Slicing

Network slicing may introduce new vulnerabilities, while the stringent performance of slices in a shared infrastructure environment requires a secure E2E network. In [107], the paper has identified security issues of network slicing and associated technologies. In this intent, slice isolation play a vital role in guaranteeing safe and accurate operations. In contrast, the AI model in automated slices needs to be protected from unauthorized access and usage. MLOps pipeline collects data from online sources and traffic which increases the risk of data poisoning. This vulnerability can corrupt or manipulate the data used for training ML models while engendering confidentiality damages and changing expected responses to slices. The secure data pipelines in MLOps are challenging and need to consistently set governance rules in each pipeline step to protect against different data attacks.

Collaborative Slicing

The learning procedure for establishing large-scale intelligent network slicing is not trivial since it is time-consuming and computationally complex, especially if each slice is independently trained for the same tasks. Relying on invoking transfer learning with a collaborative approach between slice agents, a partially trained AI model can be distributed to the same deployed slices or different slices with the same tasks. The shared trained model can be re-trained to satisfy particular requirements of slices, accelerate learning of subsequent tasks, and reduce training load and time. Exploiting prior learned knowledge and the weight of different DNN models can improve generalization for different slice settings. The transfer learning for slicing is an open research area while pursuing MLOps-driven slicing is more challenging. MLOps is demanding, and the sheer number of model transfers can raise privacy and isolation concerns.

7.2 Future Research Directions

6G Sustainability

In 6G, we will witness a massive number of slices that deal with significantly more data at faster rates than the current network's deployment. It is of utmost importance to derive more suitable power-optimized solutions to ensure the long-term sustainability of AI-driven slicing. In this respect, challenges of designing sustainable intelligent slices might surface in deploying MLOps pipeline solutions to fulfill the network energy mitigation. Therefore, more research should be conducted to reduce computations and energy consumption in different steps of MLOps-driven slicing.

6G O-RAN

The softwarization and programmability within cellular networks play a vital role in 6G RAN slicing to ensure efficient and flexible sharing of network resources. In this regard, a promising direction to enable this transformation is O-RAN solutions. O-RAN slicing approach can cause greater agility, faster innovation, and support verticals to capture requirements. It represents a considerable challenge due to the current lack of principles and distributed architecture for AI-driven optimal decision-making strategies to underpin massive 6G slicing. However, correlating the MLOps-driven O-RAN with network slicing constitutes a higher challenge, but it is an exciting line of research to achieve reproducible model deployment.

6G Immersive Services

In the ambitious vision of 6G, the coined xURLLC will become conflated with both eMBB and mMTC [108]. The xURLLC paves the way for the implementation of immersive and Metaverse services. Designing a lifelong predictive and proactive solution is necessary to fulfill the requirements of xURLLC services, such as latency, bandwidth, throughput, and QoS. Therefore, feasible MLOps solutions need more study and analysis to satisfy requirements.

Bibliography

[1] ETSI ZSM 009-1, "Zero-touch network and service management (zsm); closed-loop automation; enablers," in *V0.10.5 (2021-01)*.

[2] O-RAN.WG1, "Slicing Architecture Technical Specification 9.0," in *O-RAN Specifications*, March 2023.

[3] H. Li *et al.*, "Deep Deterministic Policy Gradient Based Dynamic Power Control for Self-Powered Ultra-Dense Networks," in *Proc. IEEE Conf. GLOBECOM*, Dec. 2018.

[4] A. NGMN, "5G White Paper," in , Feb. 2015.

[5] G. S. ETSI, "Network Functions Virtualization (NFV): Management and Orchestration," in *V1.1.1*, Dec. 2014.

[6] J. A. Ayala-Romero *et al.*, "vrAIn: A Deep Learning Approach Tailoring Computing and Radio Resources in Virtualized RANs," in *ACM Mobicom*, Oct. 2019.

[7] Y. Hua, R. Li, Z. Zhao, X. Chen, and H. Zhang, "GAN-Powered Deep Distributional Reinforcement Learning for Resource Management in Network Slicing," *IEEE Jour. Selec. Areas Commun.*, vol. 38, no. 2, pp. 334–349, Feb. 2020.

[8] R. Li *et al.*, "The LSTM-Based Advantage Actor-Critic Learning for Resource Management in Network Slicing with User Mobility," *IEEE Communications Letters*, vol. 24, no. 9, pp. 2005–2009, Sep. 2020.

[9] Q. Liu *et al.*, "DeepSlicing: Deep Reinforcement Learning Assisted Resource Allocation for Network Slicing," in *arXiv:2008.07614v2*, Aug. 2020.

[10] J. Koo *et al.*, "Deep Reinforcement Learning for Network Slicing with Heterogeneous Resource Requirements and Time Varying Traffic Dynamics," in *Proc. IEEE Conf. on Network and Service Management (CNSM)*, Oct. 2019.

[11] T. Lillicrap *et al.*, “Continuous Control with Deep Reinforcement Learning,” in *Int. Conf. on Learning Representations (ICLR)*, May 2016.

[12] J. Pujol Roig *et al.*, “Management and Orchestration of Virtual Network Functions via Deep Reinforcement Learning,” *IEEE Jour. Selec. Areas Commun. (JSAC)*, vol. 38, no. 2, pp. 304–317, Feb. 2020.

[13] Q. Liu, T. Han, and E. Moges, “EdgeSlice: Slicing Wireless Edge Computing Network with Decentralized Deep Reinforcement Learning,” in *Proc. IEEE Int. Conf. Distr. Compu. Syst. (ICDCS)*, Dec. 2020.

[14] F. Fossati *et al.*, “Multi-Resource Allocation for Network Slicing under Service Level Agreements,” in *10th Int. Conf. on Networks of the Future (NoF)*, Oct. 2019, pp. 48–53.

[15] J. Zhang *et al.*, “Cell-Free Massive MIMO: A New Next-Generation Paradigm,” *IEEE Access*, vol. 7, pp. 99 878–99 888, July 2019.

[16] Y. Nie, J. Zhao, F. Gao, and Y. F. Richard, “Semi-Distributed Resource Management in UAV-Aided MEC Systems: A Multi-Agent Federated Reinforcement Learning Approach,” *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13 162–13 173, Dec. 2021.

[17] F. Li, B. Shen, J. Guo, K.-Y. Lam, G. Wei, and L. Wang, “Dynamic Spectrum Access for Internet-of-Things Based on Federated Deep Reinforcement Learning,” *IEEE Trans. Veh. Technol.*, vol. 71, no. 7, pp. 7952–7956, Jul. 2022.

[18] X. Li, L. Lu, W. Ni, A. Jamalipour, D. Zhang, and H. Du, “Federated Multi-Agent Deep Reinforcement Learning for Resource Allocation of Vehicle-to-Vehicle Communications ,” *IEEE Trans. Veh. Technol.*, vol. 71, no. 8, pp. 8810–8824, Aug. 2022.

[19] Y. Cao, S.-Y. Lien, Y.-C. Liang, K.-C. Chen, and X. Shen, “User Access Control in Open Radio Access Networks: A Federated Deep Reinforcement Learning Approach,” *IEEE Trans. on Wirel. Commun.*, vol. 21, no. 6, pp. 3721–3736, Jun. 2022.

[20] H. T. Nguyen, N. Cong Luong, J. Zhao, C. Yuen, and D. Niyato, “Resource Allocation in Mobility-Aware Federated Learning Networks: A Deep Reinforcement Learning Approach,” in *Proc. IEEE World Forum Inte. Things (WF-IoT)*, Jun. 2020.

[21] S. Messaoud, A. Bradai, O. B. Ahmed, P. T. A. , Quang, M. Atri, and M. S. Hossain, “Deep Federated Q-Learning-based Network Slicing for Industrial IoT,” *IEEE Trans. Indust. Informat.*, vol. 17, no. 8, pp. 5572–5582, Aug. 2021.

[22] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience-driven Networking: A Deep Reinforcement Learning based Approach," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2018.

[23] H. Huang, C. Zeng, Y. Zhao, G. Min, Y. Zhu, W. Miao, and J. Hu, "Scalable Orchestration of Service Function Chains in NFV-enabled Networks: A Federated Reinforcement Learning Approach," *IEEE Jour. Selec. Areas Commun.*, vol. 39, no. 8, pp. 2558–2571, Aug. 2021.

[24] 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 Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10 792–10 803, Oct. 2021.

[25] I. Vila, J. Perez-Romero, O. Sallent, and A. Umbert, "A Multi-Agent Reinforcement Learning Approach for Capacity Sharing in Multi-Tenant Scenarios," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9450–9465, Sep. 2021.

[26] Y. Tu, Y. Ruan, S. Wagle, C. G. Brinton, and C. Joe-Wong, "Network-Aware Optimization of Distributed Learning for Fog Computing," in *Proc. IEEE Conf. Compu. Commun.*, Jul. 2020.

[27] E. Raj, D. Buffoni, M. Westerlund, and K. Ahola, "Edge MLOps: An Automation Framework for AIoT Applications," in *Proc. IEEE Int. Conf. on Cloud Engineering (IC2E)*, Oct. 2021.

[28] G. Samaras, V. Theodorou, D. Laskaratos, N. Psaromanolakis, M. Mertiri, and A. Valantasis, "QMP: A Cloud-native MLOps Automation Platform for Zero-Touch Service Assurance in 5G Systems," in *Proc. IEEE Int. Mediterranean Conf. on Communications and Networking (MeditCom)*, Sep. 2022.

[29] S. A. R. Zaidi, A. M. Hayajneh, M. Hafeez, and Q. Z. Ahmed, "Unlocking Edge Intelligence Through Tiny Machine Learning (TinyML)," *IEEE Access*, vol. 10, pp. 100 867–100 877, Sep. 2022.

[30] B. Brik, K. Boutiba, and A. Ksentini, "Deep Learning for B5G Open Radio Access Network: Evolution, Survey, Case Studies, and Challenges," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 228–250, Jan. 2022.

[31] P. Li *et al.*, “RLOps: Development Life-Cycle of Reinforcement Learning Aided Open RAN,” *IEEE Access*, vol. 10, pp. 113 808–113 826, Oct. 2022.

[32] T. Tsourdinis, I. Chatzistefanidis, N. Makris, and T. Korakis, “AI-driven Service-aware Real-time Slicing for beyond 5G Networks,” in *Proc. IEEE INFOCOM 2022 - IEEE Conf. on Computer Communications Workshops*, May. 2022.

[33] S. Fujimoto *et al.*, “Addressing Function Approximation Error in Actor-Critic Methods,” in *Int. Conf. on Machine Learning (ICML)*, July 2018.

[34] D. Bega, M. Gramaglia, A. Garcia-Saavedra, M. Fiore, A. Banchs, and X. Costa-Perez, “Network Slicing Meets Artificial Intelligence: An AI-Based Framework for Slice Management,” *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 32–38, 2020.

[35] H. van Hasselt, “Double Q-learning,” in *Advances in Neural Information Processing Systems*, vol. 23, 2010, pp. 2613–2621.

[36] H. van Hasselt *et al.*, “Deep reinforcement learning with double Q-learning,” in *AAAI*, 2016, pp. 2094–2100.

[37] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015.

[38] T. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,” in *ICRL*, 2016.

[39] E.-G. Z. 002, “Zero-touch Network and Service Management (ZSM),” in *Reference Architecture*, Aug. 2019.

[40] D. D.Clark *et al.*, “A Knowledge Plane for the Internet,” in *ACM SIGCOMM*, Aug. 2003, pp. 3–10.

[41] A. Mestres *et al.*, “Knowledge-defined networking,” *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 2–10, July 2017.

[42] S. Barmpounakis *et al.*, “Data Analytics for 5G Networks: A Complete Framework for Network Access Selection and Traffic Steering,” *Int. Journal on Advances in Telecommun.*, vol. 11, no. 3, 2018.

[43] T. Thomas and N. T.Bhuvan, “Study on Distributed SDN Controllers and Failover Mechanisms,” *IJIRCCE*, vol. 5, no. 4, pp. 7183–7185, April 2017.

[44] LTE, "LTE-Interface between the Control plane and the User Plane of EPC Nodes," in *3GPP TS 29.244 version 14.0.0 Release 14*, 2017.

[45] I. Alawe *et al.*, "On evaluating different trends for virtualized and SDN-ready mobile network," in *IEEE CloudNet*, Sep. 2017.

[46] S. Moe *et al.*, "Machine Learning in Control Systems: An Overview of the State of the Art," in *AI XXXV, SGAI 2018*, Nov. 2018, pp. 250–265.

[47] E. Bjornson *et al.*, "Optimal Multiuser Transmit Beamforming: A Difficult Problem with a Simple Solution Structure," *IEEE Signal Processing Magazine*, vol. 31, no. 4, pp. 142–148, July 2014.

[48] Y. Shi *et al.*, "Group Sparse Beamforming for Green Cloud-RAN," *IEEE Trans. on Wireless Commun.*, vol. 13, no. 5, pp. 2809–2823, May 2014.

[49] M. Peng *et al.*, "Energy-efficient resource allocation optimization for multimedia heterogeneous cloud radio access networks," *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 879–892, May 2016.

[50] Y. Liao *et al.*, "How much computing capability is enough to run a cloud radio access network?" *IEEE Comm. Letters*, vol. 21, no. 1, pp. 104–107, Jan. 2017.

[51] J. Tang *et al.*, "System cost minimization in cloud RAN with limited fronthaul capacity," *IEEE Trans. on Wire. Commu. (TWC)*, vol. 16, no. 5, pp. 3371–3384, May 2017.

[52] D. Silver *et al.*, "Deterministic Policy Gradient Algorithms," in *Int. Conf. on Machine Learning (ICML)*, June 2014.

[53] T. Haarnoja *et al.*, "Soft Actor-Critic Algorithms and Applications," in *arXiv:1812.05905*, Jan. 2019.

[54] T. Haarnoja, A. Zhou *et al.*, "Soft Actor-Critic: Off-policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *Int. Conf. on Machine Learning (ICML)*, July 2018.

[55] D. Hendrycks and K. Gimpel, "Gaussian Error Linear Units (gelus)," in *arXiv:1606.08415*, July 2020.

[56] F. Rezazadeh, H. Chergui, L. Alonso, and C. Verikoukis, "Continuous Multi-objective Zero-touch Network Slicing via Twin Delayed DDPG and OpenAI Gym," in *Proc. IEEE Glob. Commun. Conf.*, Dec. 2020.

[57] M. Asif Habibi *et al.*, "The Structure of Service Level Agreement of Slice-Based 5G Network," in *Proc. IEEE Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018.

[58] E. G. NFV-MAN, "Network Functions Virtualisation (NFV)-Management and Orchestration," in *V1.1.1*, 2014.

[59] Y. Nakayama *et al.*, "Low-latency Routing Scheme for a Fronthaul Bridged Network," *IEEE Journal of Optical Commu. and Net.*, vol. 10, no. 1, pp. 14–23, Jan. 2018.

[60] V. Mnih *et al.*, "Asynchronous Methods for Deep Reinforcement Learning," in *arXiv:1602.01783v2*, June 2016.

[61] A. Nair *et al.*, "Massively Parallel Methods for Deep Reinforcement Learning," in *Int. Conf. on Machine Learning (ICML), Deep Learning Workshop*, July 2015.

[62] L. Espeholt *et al.*, "Impala: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner A rchitectures," in *arXiv:1802.01561v3*, June 2018.

[63] G. Brockman *et al.*, "OpenAI Gym," in *arXiv:1606.01540*, June 2016.

[64] F. Rezazadeh, H. Chergui, L. Christofi, and C. Verikoukis, "Actor-Critic-Based Learning for Zero-touch Joint Resource and Energy Control in Network Slicing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2021.

[65] I. Grondman *et al.*, "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 42, no. 6, pp. 1291–1307, Nov. 2012.

[66] W. K. Seah, C. Lee, Y. Lin, and Y. Lai, "Combined Communication and Computing Resource Scheduling in Sliced 5G Multi-Access Edge Computing Systems," *IEEE Trans. Veh. Technol.*, vol. 71, no. 3, pp. 3144–3154, Mar. 2022.

[67] H. Xiang, W. Zhou, M. Daneshmand, and M. Peng, "Network Slicing in Fog Radio Access Networks: Issues and Challenges," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 110–116, Dec. 2017.

[68] O-RAN Alliance, “O-RAN: Towards an Open and Smart RAN,” Tech. Rep., Oct. 2018.

[69] F. Rezazadeh, H. Chergui, L. Blanco, L. Alonso, and C. Verikoukis , “A Collaborative Statistical Actor-Critic Learning Approach for 6G Network Slicing Control,” in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Feb. 2021.

[70] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Pérez, “AZTEC: Anticipatory Capacity Allocation for Zero-Touch Network Slicing,” in *Proc. IEEE Conf. Compu. Commun.*, Jul. 2020.

[71] Y. K. Tun, N. H. Tran, D. T. Ngo, S. R. Pandey, Z. Han, and C. S. Hong, “Wireless Network Slicing: Generalized Kelly Mechanism-Based Resource Allocation,” *IEEE Jour. Selec. Areas Commun.*, vol. 37, no. 8, pp. 1794–1807, Aug. 2019.

[72] V. Sciancalepore, X. Costa-Pérez, and A. Banchs, “RL-NSB: Reinforcement Learning-Based 5G Network Slice Broker,” *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1543–1557, Aug. 2019.

[73] X. Foukas, M. K. Marina, and K. Kontovasilis, “Orion: RAN Slicing for a Flexible and Cost-Effective Multi-Service Mobile Network Architecture,” in *Proc. ACM Int. Conf. Mobile Comput. Netw.*, Oct. 2017.

[74] P. Caballero, A. Banchs, G. de Veciana, and X. Costa-Pérez, “Multi-Tenant Radio Access Network Slicing: Statistical Multiplexing of Spatial Loads,” *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3044–3058, Oct. 2017.

[75] C. H. Papadimitriou, *Computational complexity.* Addison-Wesley, 1994.

[76] A. Okic, L. Zanzi, V. Sciancalepore, A. Redondi, and X. Costa-Pérez, “π-ROAD: a Learn-as-You-Go Framework for On-Demand Emergency Slices in V2X Scenarios,” in *Proc. IEEE Conf. Compu. Commun.*, Jul. 2021.

[77] A. Furno, M. Fiore, R. Stanica, C. Ziemlicki, and Z. Smoreda, “A Tale of Ten Cities: Characterizing Signatures of Mobile Traffic in Urban Areas,” *IEEE Trans. on Mobile Comput.*, vol. 16, no. 10, pp. 2682–2696, Oct. 2017.

[78] Y. Azimi, S. Yousefi, H. Kalbkhani, and T. Kunz, “Energy-Efficient Deep Reinforcement Learning Assisted Resource Allocation for 5G-RAN Slicing,” *IEEE Trans. Veh. Technol.*, vol. 71, no. 1, pp. 856–871, Jan. 2022.

[79] L. Zanzi, V. Sciancalepore, A. Garcia-Saavedra, H. D. Schotten, and X. Costa-Pérez, "LACO: A Latency-Driven Network Slicing Orchestration in Beyond-5G Networks," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 1, pp. 667–682, Oct. 2021.

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

[81] R. S. Sutton, "Learning to Predict by the Methods of Temporal Differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, Aug. 1988.

[82] I. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An Empirical Investigation of Catastrophic Forgetting in Gradient-based Neural Networks," Apr. 2014.

[83] H. V. Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," in *Proc. Assoc. Advanc. Artifi. Intellig. (AAAI)*, Feb. 2016.

[84] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, "Optimal VNF Placement via Deep Reinforcement Learning in SDN/NFV-Enabled Networks," *IEEE Jour. Selec. Areas Commun.*, vol. 38, no. 2, pp. 263–278, Feb. 2020.

[85] X. Lyu, C. Ren, W. Ni, H. Tian, R. Ping Liu, and E. Dutkiewicz, "Optimal Online Data Partitioning for Geo-Distributed Machine Learning in Edge of Wireless Networks," *IEEE Jour. Selec. Areas Commun.*, vol. 37, no. 10, pp. 2393–2406, Oct. 2019.

[86] J. Wang, J. Zhang, W. Bao, X. Zhu, B. Cao, and P. S. Yu, "Not Just Privacy: Improving Performance of Private Deep Learning in Mobile Cloud," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery & Data Min.*, Aug. 2018.

[87] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated Learning: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Access*, vol. 8, pp. 140 699–140 725, Jul. 2020.

[88] C. Zhang and P. Patras, "Long-Term Mobile Traffic Forecasting Using Deep Spatio-Temporal Neural Networks," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jun. 2018.

[89] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Pérez, "DeepCog: Cognitive Network Management in Sliced 5G Networks with Deep Learning," Apr. 2019.

[90] W. Wang, G. Lyu, Y. Shi, and X. Liang, "Time Series Clustering Based on Dynamic Time Warping," in *Proc. IEEE Int. Conf. Soft. Eng. Service Sci. (ICSESS)*, Nov. 2018, pp. 487–490.

[91] O. Gold and M. Sharir, "Dynamic Time Warping and Geometric Edit Distance: Breaking the Quadratic Barrier," *ACM Tran. Algorithms*, vol. 14, no. 4, Aug. 2018.

[92] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proc. Conf. Knowl. Discovery Data Min.*, 1996, pp. 226–231.

[93] O-RAN Alliance, "O-RAN-WG1-O-RAN Architecture Description (ORAN.WG1.O-RAN-Architecture-Description-v04.00)," Technical Specification, 2020.

[94] A. Garcia-Saavedra and X. Costa-Pérez, "O-RAN: Disrupting the Virtualized RAN Ecosystem," *IEEE Commun. Standards Mag.*, pp. 1–8, 2021.

[95] S. D'Oro, L. Bonati, M. Polese, and T. Melodi, "OrchestRAN: Network Automation through Orchestrated Intelligence in the Open RAN," in *Proc. IEEE Conf. Compu. Commun.*, 2022.

[96] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, Jun. 2016.

[97] L. Pappalardo and F. Simini, "Data-driven Generation of Spatio-Temporal Routines in Human Mobility," *Data Min. Knowle. Discovery*, vol. 32, pp. 787–829, Dec. 2017.

[98] L. Pappalardo, F. Simini, S. Rinzivillo, D. Pedreschi, F. Giannotti, and A.-L. Barabaśi, "Returners and Explorers Dichotomy in Human Mobility," *Nature Commun.*, vol. 6, pp. 1–8, Sep. 2015.

[99] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal Modeling and Prediction in Cellular Networks: A Big Data Enabled Deep Learning Approach," in *Proc. IEEE Conf. Comput. Commun.*, May 2017.

[100] N. Bouacida, J. Hou, H. Zang, and X. Liu, "Adaptive Federated Dropout: Improving Communication Efficiency and Generalization for Federated Learning," in *Proc. IEEE Conf. on Comput. Commun. Works. (INFOCOM WKSHPS)*, May 2021, pp. 1–6.

[101] A. Ksentini and N. Nikaein, "Toward Enforcing Network Slicing on RAN: Flexibility and Resources Abstraction," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 102–108, Jun. 2017.

[102] X. Wang and M. C. Gursoy, "Uplink Coverage in Heterogeneous mmWave Cellular Networks With Clustered Users," *IEEE Access*, vol. 9, pp. 69 439–69 455, Apr. 2021.

[103] L. Zanzi, V. Sciancalepore, A. Garcia-Saavedra, X. Costa-Pérez, G. Agapiou, and H. D. Schotten, "ARENA: A Data-Driven Radio Access Networks Analysis of Football Events," *IEEE Trans. Netw. Serv. Manage.*, vol. 17, no. 4, pp. 2634–2647, Dec. 2020.

[104] "Ethics Guidelines for Trustworthy AI,," *High-Level Expert Group on Artificial Intelligence, Technical Report of the European Commission*, Apr. 2019.

[105] S. M. Lundberg and S. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems 30*, 2017.

[106] F. Rezazadeh, L. Zanzi, F. Devoti, H. Chergui, X. Costa-Pérez, and C. Verikoukis, "On the Specialization of FDRL Agents for Scalable and Distributed 6G RAN Slicing Orchestration," *IEEE Transactions on Vehicular Technology*, pp. 1–15, 2022.

[107] Q. Liu, T. Han, and N. Ansari, "Learning-Assisted Secure End-to-End Network Slicing for Cyber-Physical Systems," *IEEE Network*, vol. 34, no. 3, pp. 37–43, June 2020.

[108] J. Park, S. Samarakoon, H. Shiri *et al.*, "Extreme ultra-reliable and low-latency communication," *Nature Electronics*, vol. 5, no. 3, pp. 133–141, March 2022.