SEMANTIC

End-to-End Slicing and Data-Driven Automation of Next Generation Cellular Networks with Mobile Edge Clouds

Marie Skłodowska-Curie Actions (MSCA)
Innovative Training Networks (ITN)
H2020-MSCA-ITN-2019
861165 - SEMANTIC

WP3 – Optimizations for Integrated Access/X-haul and End-to-End Slicing

D3.3: Performance Evaluation of E2E- Slicing and Traffic Steering towards IAB

Contractual Date of Delivery:	M49
Actual Date of Delivery:	19/1/2024
Responsible Beneficiary:	IQU
Contributing Beneficiaries:	CTTC, EUR, IQU
Security:	Public
Nature:	Report
Version:	V1

Document Information

Version Date: 19/1/2024
Total Number of Pages: 50

Authors

Name	Organization	Email
Swastika Roy	CTTC	swastika.roy.roy@upc.edu
Suvidha Sudhakar Mhatre	Iquadrat Informatica S.L.	suvidha.sudhakar.mhatre@upc.edu
Pavlos Doanis	EURECOM	pavlos.doanis@eurecom.fr

Document History

Revision	Date	Modification	Contact Person
V0.0	30/10/2023 17/11/2023	First draft of the contributions of ESRs	Navideh Ghafouri n.ghafoori@iquadrat.com
V0.1	30/11/2023	First edition	Navideh Ghafouri
V0.2	10/12/2023	Revisions by ESRs	Navideh Ghafouri
V0.3	15/12/2023	Revisions by editor	Navideh Ghafouri
V1.0	19/1/2024	Released version	Navideh Ghafouri

Table of Contents

List of Acronyms and Abbreviations 5
1 Executive Summary 8
2 Introduction 9
3 Joint Explainability and Sensitivity-Aware Federated Deep Learning for Transparent 6G RAN Slicing 11
3.1 Introduction 11
3.2 Related Works 11
3.3 Explainable FDL For Transport Traffic Drop classification 12
3.3.1 Proposed Network Model 13
3.3.2 Network Configuration 14
3.4 Research Methodology 14
3.5 Results 17
3.5.1 Parameter Settings and Baseline 17
3.5.2 Results Analysis 17
3.6 Conclusion 20
4 Intelligent QoS aware slice resource allocation with user association parameterization for beyond 5G ORAN based architecture using DRL 21
4.1 Introduction 21
4.2 Related Work 22
4.3 Network Architecture 24
4.4 System Model 25
4.4.1 Intelligent qoS aware Resource Allocation (IQRA) 27
4.4.2 Low Complexity Intelligent QoS aware Resource Allocation (LIQRA) 28
4.4.3 Key Performance Indicator 28
4.5 Proposed DRL based Intelligent Agents 29
4.5.1 Markov Decision Process 29
4.6 Simulation Setup 29
4.7 Results and analysis 30
4.8 Conclusion 32
5 Multi-agent DQN with sample-efficient updates for large inter-slice orchestration problems 33
5.1 Introduction 33
5.2 RL framework for inter-slice orchestration 34

5.2.1 Environment: A B5G system 34
5.2.2 States, actions, and rewards 36
5.3 Approximate RL schemes 37
5.3.1 DQN 37
5.3.2 iDQN 38
5.3.3 DQN+/iDQN+ 38
5.4 Simulation results 39
5.4.1 Scalability of tabular vs approximate RL schemes 40
5.4.2 Performance gains of DQN+/iDQN+ 42
5.4.3 Validation in large scale scenario 43
5.5 Conclusion 44
6 Summary 44
7 References 45

List of Acronyms and Abbreviations

Acronym	*Description*
3GPP	Third Generation Partnership Project
ETSI	European Telecommunications Standards Institute
5G	5th Generation Mobile Networks
B5G	Beyond 5G
6G	6th Generation Mobile Networks
AI	Artificial Intelligence
XAI	Explainable AI
BBU	Base Band Unit
vBBU	Virtual BBU
CN	Core Network
MEC	Multi Access Edge Computing
RAN	Radio Access Network
C-RAN	Cloud RAN
CPU	Central Processing Unit
CU	Centralized Unit
DU	Distributed unit
eMBB	Enhanced Mobile Broadband
mMTC	Massive Machine Type Communication
uRLLC	Ultra-Reliable Low Latency Communication
muRLLC	Massive uRLLC
KPI	Key Performance Indicator
QoS	Quality of Service
NS	Network Slicing
NFV	Network Function Virtualization
SDN	Software-Defined Networking
vSDNC	Virtual SDN Controller
SLA	Service Level Agreement
VNF	Virtual Network Function
VL	Virtual Link
IAB	Integrated Access and Backhaul
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
QL	Q-Learning
PI	Policy Iteration
DQN	Deep Q-Network
iDQN	Multi-agent DQN (with independent agents)
TD	Temporal Difference
NN	Neural Network
DNN	Deep NN
CNN	Convolutional NN
MANO	Management and Orchestration
ML	Machine Learning
FL	Federated Learning
E2E	End-to-End

ZSM	Zero Touch Network and Service Management
NSI	Network Slice Instance
NSSI	Network Slice Subnet Instance
SON	Self-Organizing Network
TRP	Transmission / Reception Point
MS	Monitoring System
AE	Analytic Engine
CL	Closed Loop
OTT	Over The Top service
MILP	Mixed Integer Linear Programming
PRB	Physical Resource Block
AN	Access Network
OAI	Open Air Interface
MNO	Mobile Network Operator
RIC	RAN Intelligence Controller
O-RAN	Open RAN
UE	User Equipment
ERAB	Enhanced Radio Access Bearer
vDFE	Virtual Digital Front End
O-RU	Open RAN Radio Unit
O-DU	Open RAN Distributed Unit
RBG	Resource Block Group

List of Figures

Figure 3-1: RAN federated traffic drop classification in NS 13
Figure 3-2: Explainable FDL building blocks..... 14
Figure 3-3:Analysis of FL training loss vs FL rounds of Proposed EFL with Lower bound of Recall score, α =[0.9, 0.95, 0.95] and Upper bound of log-odds score, β =[−0.01,−0.01,−0.01] 18
Figure 3-4:Analysis of Recall score with Lower bound of Recall score, α = [0.9, 0.95, 0.95] and Upper bound of log-odds score, β = [−0.01,−0.01,−0.01] 18
Figure 3-5:Analysis of log-odds score with Lower bound of Recall score, α = [0.9, 0.95, 0.95] and Upper bound of log odds score, β = [−0.01,−0.01,−0.01]..... 19
Figure 3-6:Correlation heatmap of eMBB slices based on attribution scores of features generated by XAI 20
Figure 4-1 ORAN based Network Architecture..... 24
Figure 4-2 System Model..... 25
Figure 4-3 DRL Agent at Near-RT RIC (Remote Edge) 26
Figure 4-4 reward performance for DRL Intelligent agent 31
Figure 4-5 URLLC reward performance for DRL intelligent agent 31
Figure 4-6 ECCDF of Throughput for eMBB Slice..... 31
Figure 4-7 ECCDF of Throughput for URLLC Slice 31
Figure 4-8 ECCDF of Latency for eMBB Slice..... 31
Figure 4-9 ECCDF of Latency for URLLC Slice 31
Figure 5-1. Toy example exhibiting the main components of our environment. 34
Figure 5-2. Slice embedding example. 35
Figure 5-3. Main algorithmic steps of DQN. 38
Figure 5-4. Modifications of iDQN algorithm with respect to DQN. 38
Figure 5-5. Modification 1 of DQN+ algorithm with respect to DQN. 39
Figure 5-6. Modification 2 of DQN+ algorithm with respect to DQN. 39
Figure 5-7. Convergence plot (Scenario 1: Markovian traffic – small problem size)..... 41
Figure 5-8. Box plot of cost in the testing dataset (Scenario 1: Markovian traffic – small problem size). 41
Figure 5-9. Convergence plot (Scenario 2: Markovian traffic – larger problem size). 42
Figure 5-10. Convergence plot in real traffic small scale scenario. 43
Figure 5-11. Box plot of cost in the testing dataset (large scale real traffic scenario). 43

List of Tables

Table 1: Dataset Features and Output 13
Table 2: Settings..... 17

1 Executive Summary

This deliverable is the follow-up of deliverable 3.2, which was the initial proposed optimization frameworks and algorithms towards optimal network slicing, contributed by the SEMANTIC ESRs, in the framework of WP3 (Optimizations for integrated access/X-haul and end-to-end slicing). The proposed solutions considered an end-to-end slicing perspective as well as novel technologies, like integrated access/X-haul and traffic steering. This document presents the progress of ESRs, including the final results and numeric simulation details. Each Chapter of this report focuses on different aspects of network slicing and corresponds to the work of a different ESR. It includes an introduction section for all the contributions, followed by a detailed description of the work of each ESR in each section. In chapter 3, the slice resource allocation problem at the RAN-Edge domain is examined, and a zero-touch Federated Learning solution combined with Explainable AI techniques is proposed. The chapter 4 provides a framework for slice resource allocation and management in the RAN-Edge domain, proposing distributed deep Reinforcement Learning solutions at different timescales to optimize resource utilization and distribution. Finally, the Chapter 0 tackles the dynamic slice embedding problem, introducing a generic framework suitable for multi-domain networks and end-to-end slice KPIs, while a multi-agent deep Reinforcement Learning approach is proposed.

2 Introduction

The standardization and deployment of the fifth generation (5G) of mobile networks is ongoing during the last few years. Enhanced mobile broadband (eMBB), massive machine-type communications (mMTC), and ultra-reliable and low latency communications (uRLLC) are three major communication scenarios. Concurrently, both academia and industry have focused on research for beyond 5G (B5G) and towards 6G networks. These networks are not limited to the three major use cases of 5G [1], they are envisioned to support more vertical application scenarios, having unique features and specific capabilities (e.g., latency, peak date rate, etc.). The parallel provisioning of such heterogeneous vertical services urges a highly flexible, adaptive, and intelligent network architecture, directly contradicting the "one-size-fits-all" network design paradigm. 6G networks are expected to intelligently support a massive number of simultaneous and heterogeneous slices. Consequently, the challenges of scalability and sustainability will affect the deployment of artificial intelligence (AI)-driven zero-touch management and orchestration (MANO) of end-to-end (E2E) slices. In this respect, ETSI has standardized the zero-touch network and service management (ZSM) framework, where a reference architecture and AI-based closed-loop management automation have been proposed [2]. However, the traditional centralized approach for monitoring, analyzing, and controlling the underlying raw data will be problematic, because it suffers from significant overhead, delay, and a single point of failure. On the other hand, decentralized approaches ensure scalability, low data exchange and, therefore, more security. In this view, distributed artificial intelligence (AI) approaches, particularly Federated Learning (FL) techniques can play a vital role in monitoring scattered data across the network while reducing the computational costs and enabling fast local analysis and decision. Nonetheless, both the convergence delay and computation cost often limit FL capability under non-IID real network data. These aspects are going to be considered in the FL approach proposed in chapter 3. Furthermore, in real deployment, both the operator and the slice tenant need to understand the behavior of the FL model, in order to trust AI's decisions. Thus, Explainable Artificial Intelligence (XAI) empowered Federated Learning (FL) is getting a lot of attention due to the end-user trust and secured operation. This approach, which can build an advanced AI-based trust model, ensure hassle-free processes, and improve security to the 6G heterogeneous networks, will be also examined in chapter 3.

The diverse requirements of beyond 5G services increase design complexity and demand dynamic adjustments to the network parameters. This can be achieved with slicing and programmable network architectures such as the open radio access network (ORAN). It facilitates the tuning of the network components exactly to the demands of future-envisioned applications as well as intelligence at the edge of the network.
Artificial intelligence (AI) has recently drawn a lot of interest for its potential to solve challenging issues in wireless communication. Due to the non-deterministic, random, and complex behavior of models and parameters involved in the process, radio resource management is one of the topics that needs to be addressed with such techniques. The study presented in Chapter 4 proposes quality of service (QoS)-aware intra-slice resource allocation that provides superior performance compared to baseline and state-of-the-art strategies. The slice-dedicated intelligent agents learn how to handle resources at near-RT RIC level time granularities while optimizing various key performance indicators (KPIs) and meeting QoS requirements for each end user. In order to improve KPIs and system performance with various reward functions, the study discusses Markov's decision process (MDP) and deep reinforcement learning (DRL) techniques, notably deep Q network (DQN). The simulation evaluates the efficacy of the algorithm under dynamic conditions and various network characteristics. Results and analysis demonstrate the

improvement in the performance of the network for enhanced mobile broadband (eMBB) and ultra-reliable low latency (URLLC) slice categories.

Data-driven network slicing has recently been explored as a major driver for networks beyond 5G. Nevertheless, we are still a long way before such solutions are practically applicable to real problems. Most solutions addressing the problem of dynamically placing virtual network function chains ("slices") on top of a physical topology still face one or more of the following hurdles: (i) they focus on simple slicing setups (e.g., single domain, single slice, simple VNF chains, and performance metrics); (ii) solutions based on modern reinforcement learning theory have to deal with astronomically high action spaces when considering multi-VNF, multi-domain, multi-slice problems; (iii) the training of the algorithms is not particularly data-efficient, which can hinder their practical application given the scarce(r) availability of cellular network related data (as opposed to standard machine learning problems). To this end, in Chapter 0, we attempt to tackle all the above shortcomings in one common framework. For (i), we propose a generic, queuing network-based model that captures the inter-slice orchestration setting, supporting complex VNF chain topologies and end-to-end performance metrics. For (ii), we explore multi-agent DQN algorithms that can reduce action space complexity by orders of magnitude compared to standard DQN. For (iii), we investigate two mechanisms to store and select from the experience replay buffer to speed up the training of DQN agents. The convergence speed gains of the proposed scheme are validated using real traffic data.

3 Joint Explainability and Sensitivity-Aware Federated Deep Learning for Transparent 6G RAN Slicing

3.1 Introduction

In recent years, wireless networks are evolving complex, which upsurges the use of zero-touch artificial intelligence (AI)-driven network automation within the telecommunication industry. In particular, network slicing, the most promising technology beyond 5G, would embrace AI models to manage the complex communication network. Besides, it is also essential to build the trustworthiness of the AI black boxes in actual deployment when AI makes complex resource management and anomaly detection. Inspired by closed-loop automation and Explainable Artificial intelligence (XAI), we design an Explainable Federated deep learning (FDL) model to predict per-slice RAN dropped traffic probability while jointly considering the sensitivity and explainability-aware metrics as constraints in such non-IID setup. In precise, we quantitatively validate the faithfulness of the explanations via the so-called attribution-based log-odds metric that is included as a constraint in the run-time FL optimization task. Simulation results confirm its superiority over an unconstrained integrated-gradient (IG) post-hoc FDL baseline.

3.2 Related Works

The most promising 6G network slicing technology insists on adopting autonomous management and orchestration of the end-to-end (E2E) network resources at the network domains because the isolation of slices may induce a high cost in terms of efficiency [3], [4].So, ETSI standardized zero-touch network and service management (ZSM) framework has been considered [2].Here, zero-touch refers to the automation and management of resources without human interference.

Besides, developing cognitive slice management solutions in 6G networks is essential to automatically orchestrate and manage network slices, particularly network resources across different technological domains (TDs), along with ensuring the end-user's QoE and QoS [5], [6]. Hence, the [7] has proposed an AI-native network slicing management solution of 6G networks to support emerging AI services.

Also, AI algorithms should be driven by the distributed nature of datasets to acquire the full potential of network slicing automation, which will solve the problematic behavior of the cloud-centric traditional ML schemes. Thus, a decentralized learning approach is required to handle distributed network slices efficiently.

For this, we choose Federated learning (FL) [8] to handle distributed network slices efficiently like our another research work [9].Besides, even if DNN hold the state-of-the-art [10], [11] in solving resource allocation and orchestration problems of network slicing, the black-box nature of such ML models impedes understanding of their decisions, any flaws in the datasets or the model's performance behavior. Moreover, the 6G network is going to be "machine-centric" technology which signifies that all the corresponding "smart things" in the 6G network will operate intelligently but as a smart black box [12]. Here, the smart black box is not transparent in its action or decision-making processes and could have adverse effects on the network's operations of the 6G technology. In this concern, XAI provides human interpretable methods to adequately explain the AI system and its decisions for gaining the human's trust in the loop.

Also, [13] indicates that it is a prerequisite of any ZSM-based AI models in 6G to enrich translucency of their models. Viewing this fact, zero-touch XAI-driven FL will be fetching a particular emphasis for its automation and unique advantages, which are essential for end-user

trust and secured procedure. In contrast, the conventional XAI focuses only on the interpretability and transparency of any ML system.

Some works of XAI [14], [15], [16] indicate the importance of explainability and present some research works on handover and resource allocation, etc., in the beyond 5G networks. In [17], XAI for physical/MAC layers in 6G networks are focused. In comparison, the authors of [18] present a trust-aware federated deep reinforcement learning-based device selection technique in an autonomous driving scenario.

And, to evaluate the performance of XAI models, the paper [19] introduces some essential metrics. So, in this work, we will present a novel zero-touch Explainable Federated learning (FL) as the decentralized approach for traffic drop classification in 6G network slices.

3.3 Explainable FDL for transport traffic drop classification

In this Section, we will present an Explainable Federated learning approach to achieve transparent zero-touch service management of 6G network slices at RAN in a non-IID setup.

The main contributions of this paperwork are:

- We introduce a novel iterative explainable federated learning approach, where a constrained traffic drop detection classifier and an explainer exchange---in a closed loop way--- attributions of the features as well as predictions to achieve a transparent zero-touch service management of 6G network slices at RAN in a non-IID setup.
- We adopt the integrated gradients XAI method to showcase features attributions.
- The generated attributions are then used to quantitatively validate the faithfulness of the explanations via the so-called log-odds metric which is included as a constraint in the FL optimization task.
- We formulate the corresponding joint recall and log-odds-constrained FL optimization problem under the proxy-Lagrangian framework and solve it via a non-zero sum two-player game strategy [19], while comparing with the unconstrained integrated-gradient post-hoc FL baseline.

3.3.1 Proposed Network Model

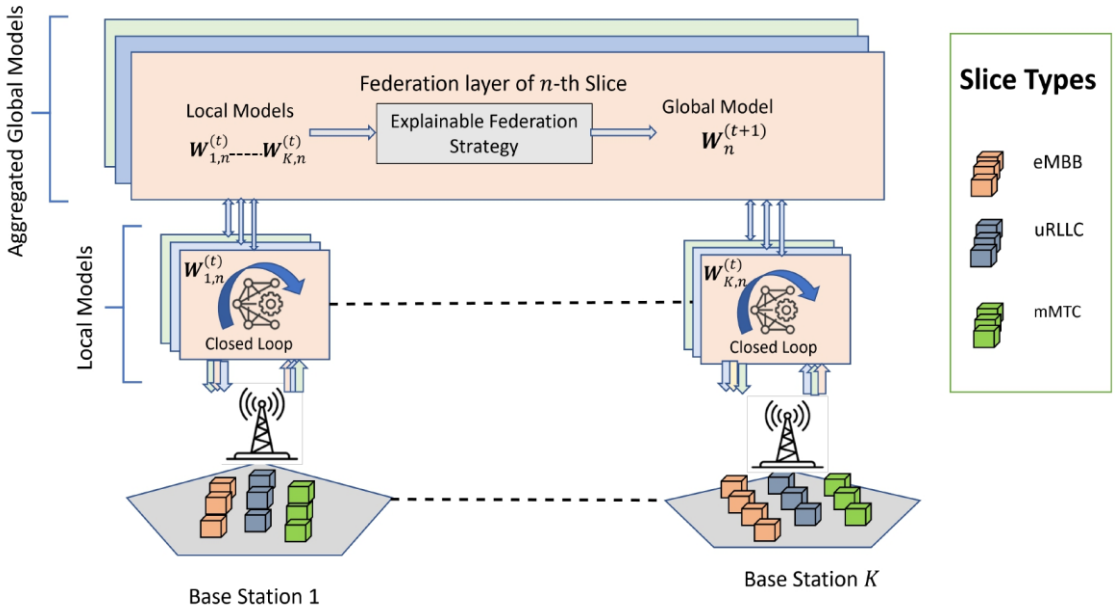


Figure 3-1: RAN federated traffic drop classification in NS

As shown in Fig. 3-1, we consider a radio access network (RAN), which is composed of a set of K the base station (BSs), wherein a set of N parallel slices are deployed. Each BS runs a local control closed loop (CL) which collects monitoring data and performs traffic drop prediction. Specifically, the collected data serves to build local datasets for slice n ($n = 1, \ldots\ldots, n$) i.e., $D_{k,n} = \{x_{k,n}^{(i)}, y_{k,n}^{(i)}\}_{i=1}^{D_{k,n}}$, where $x_{k,n}^{(i)}$ stands for the input features vector while $y_{k,n}^{(i)}$ represents the corresponding output.

In this respect, Table I summarizes the features and the output of the local datasets. These accumulated datasets are non-IID due to the different traffic profiles induced by the heterogeneous users' distribution and channel conditions. Moreover, since the collected datasets are generally non-exhaustive to train accurate anomaly detection classifiers, the local CLs take part in a federated learning task wherein an E2E slice-level federation layer plays the role of a model aggregator.

Feature	**Description**
Average PRB	Average Physical Resource Block
Latency	Average transmission latency
Channel quality	SNR value expressing the wireless channel quality
Output	**Description**
Dropped Traffic	Probability of dropped traffic (%)

Table 1: Dataset Features and Output

3.3.2 Network Configuration

We consider below three primary slices to analyse the proposed Explainable FL policy, defined as follows:

- eMBB: Netflix, Youtube and Facebook Video,
- Social Media: Facebook, WhatsApp and Instagram,
- Browsing: Apple, HTTP and QUIC

Here, the traffic associated with each mentioned slice is the sum of the underlying OTTs' traffics that collects from the hourly traffics of the slices for five days, and the overall summary of those datasets are presented in Table 1.

3.4 Research Methodology

Here, we describe the different stages of the joint explainability and sensitivity aware FDL as summarized in Fig. 2.

A. Closed-loop Description

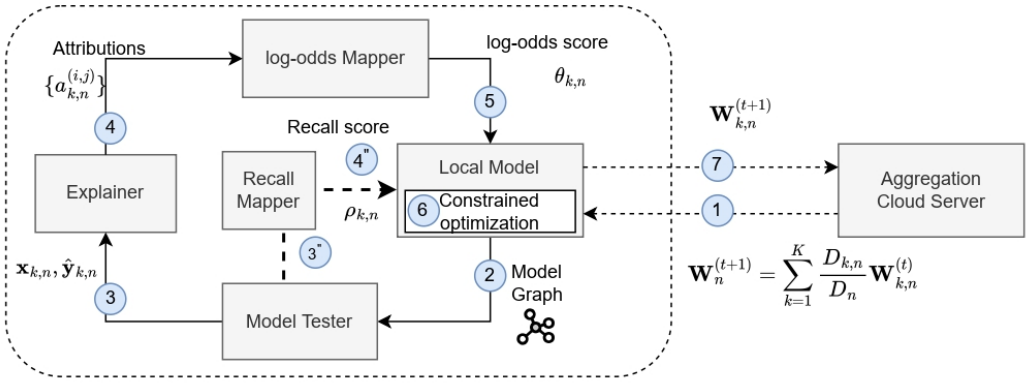


Figure 3-2: Explainable FDL building blocks

We propose a federated deep learning architecture where the local learning is performed iteratively with run-time explanation in a closed loop way as shown in Fig. 2. We design a deep neural network FL model. For each local epoch, the Learner module feeds the posterior symbolic model graph to the Tester block which yields the test features and the corresponding predictions $\hat{y}_{k,n}^{(i)}$ to the Explainer. The latter first generates the features attributions using integrated gradients XAI method. The Log-odds Mapper then uses these attributions to select the top p features that are then masked. The corresponding soft probability outputs are afterward used to calculate the the log-odds (LO) metric that is fed back to the Learner to include it in the local constrained optimization in step 6. Similarly, the Recall Mapper calculate the recall score $\rho_{k,n}$ based on the predicated and true positive values at stage 3 and 4 to include it in the local constrained optimization in step 6.

Indeed, for each local CL(k, n), the predicted traffic drop class $\hat{y}_{k,n}^{(i)}$, $(i = 1, \ldots \ldots, D_{k,n})$, should minimize the main loss function with respect to the ground truth $y_{k,n}^{(i)}$, while jointly respecting some long-term statistical constraints defined over its $D_{k,n}$ samples and jointly corresponding to recall and explainability log-odds.

As shown in steps 1 and 7 of Fig. 2, the optimized local weights at round t, $W_{k,n}^{(t)}$, are sent to the server which generates a global FL model for slice n as,

$$W_n^{(t+1)} = \sum_{k=1} \frac{D_{k,n}}{D_n} W_{k,n}^{(t)} \qquad (1)$$

where $D_n = \sum_{k=1}^{K} D_{k,n}$ is the total data samples of all datasets related to slice n. The server then broadcasts the global model to all the k CLs that use it to start the next round of iterative local optimization. Specifically, it leverages a two-player game strategy to jointly optimize over the objective and original constraints as well as their smoothed surrogates and detailed in the sequel.

B. Model Testing and Explanation

As depicted in stage 2 of Fig. 2, upon the reception of the updated model graph, the Tester uses a batch drawn from the local dataset to reconstruct the test predictions $\hat{y}_{k,n}^{(i)}$. All the graph, test dataset and the predictions are fed to the Explainer at stage 3. After that, at stage 4, Explainer generates the attributions.

by leveraging the low-complexity Integrated Gradient (IG) scheme [20], which is based on the gradient variation when sampling the neighborhood of a feature.

Attributions are a quantified impact of each single feature on the predicted output. Let, $a_{k,n}^{(i)} \in R^Q$ denote the attribution vector of sample i, which can be generated by any attribution based XAI method.

C. Log-odds Mapping

To characterize the trustworthiness of the local model, we calculate the log-odds metric, $\theta_{k,n}$ [21]. It measures the influence of the top-attributed features on the model's prediction. Specifically, the log-odds score is defined as the average difference of the negative logarithmic probabilities on the predicted class before and after masking the top p% features with zero padding [21].

In this respect, the log-odds Mapper at stage 5 of Fig. 2 starts by selecting top p% features based on their attributions which is collected from stage 4 and replace them with zero padding. That is,

$$\theta_{k,n} = -\frac{1}{D_{k,n}} \sum_{i=1}^{D_{k,n}} \log \frac{\Pr\left(\hat{y}_{k,n}^{(i)} \middle| \hat{x}_{k,n}^{(i)}\right)}{\Pr\left(\hat{y}_{k,n}^{(i)} \middle| x_{k,n}^{(i)}\right)}, \qquad (2)$$

where, $\hat{y}_{k,n}^{(i)}$ is the predicted class, $x_{k,n}^{(i)}$ are the features in the original dataset and $\hat{x}_{k,n}^{(i)}$ denotes the features in the modified dataset with top p% features zero-padded. Finally, the log-odds Mapper reports the log-odds score, which is used as one of the constraints for the constrained FL optimization task.

D. Joint Recall and Explainability-Aware traffic Drop Classification

Besides the log-odds score used for explainability, as shown in steps 3 and 4, we invoke the recall as a measure of the sensitivity of the FL local classifier, which we denote $\rho_{k,n}$, i.e.,

$$\rho_{k,n} = \pi^+\left(D_{k,n}\left[\hat{y}_{k,n}^{(i)} = 1\right]\right) \qquad (3)$$

Where, $\pi^+(D_{k,n})$ defines the proportion of $D_{k,n}$ classified positive, and $D_{k,n}[*]$ is the subset of $D_{k,n}$ satisfying expression *.

In order to trust the traffic drop anomaly detection/classification, a set of AI SLA is established between the slice tenant and the infrastructure provider, where a lower bound α_n is imposed to the recall score, while an upper bound β_n is set for the log-odds score.

This translates into solving a constrained local classification problem in iterations specified by the epochs as well as in FL rounds $t(t = 0, \ldots, T-1)$ i.e.,

$$\min_{W_{k,n}^{(i)}} \frac{1}{D_{k,n}} \sum_{i=1}^{D_{k,n}} l\left(y_{k,n}^{(i)}, \hat{y}_{k,n}^{(i)}\left(W_{k,n}^{(i)}, x_{k,n}\right)\right), \qquad (4a)$$

$$\rho_{k,n} \geq \alpha_{n,} \qquad (4b)$$

$$\theta_{k,n} \leq \beta_n \qquad (4c)$$

which is solved by invoking the so-called proxy Lagrangian framework [22], since the recall is not a smooth constraint. This consists first on constructing two Lagrangians as follows:

$$\mathcal{L}_{W_{k,n}^{(i)}} = \frac{1}{D_{k,n}} \sum_{i=1}^{D_{k,n}} l\left(y_{k,n}^{(i)}, \hat{y}_{k,n}^{(i)}\left(W_{k,n}^{(i)}, x_{k,n}\right)\right) + \lambda_1 \psi_1\left(W_{k,n}^{(t)}\right) + \lambda_2 \psi_2\left(W_{k,n}^{(t)}\right), \qquad (5a)$$

$$\mathcal{L}_{\lambda} = \lambda_1 \varphi_1\left(W_{k,n}^{(t)}\right) + \lambda_2 \varphi_2\left(W_{k,n}^{(t)}\right), \qquad (5b)$$

where $\varphi_{1,2}$ and $\psi_{1,2}$ represent the original constraints and their smooth surrogates, respectively. In this respect, the recall surrogate is given by,

$$\psi_1 = \frac{\sum_{i=1}^{D_{k,n}} y_{k,n}^{(i)} \times \min\{\hat{y}_{k,n}^{(i)}, 1\}}{\sum_{i=1}^{D_{k,n}} y_{k,n}^{(i)}} - \alpha_n \qquad (6)$$

while $\psi_1 = \varphi_1 = \beta_n - \theta_{k,n}$ since the negative logarithm is already a convex function. It also confirms that the solutions of the optimization problem are equivalent to those obtained if only the original constraints were used.

This optimization task turns out to be a non-zero-sum two-player game in which the $W_{k,n}^{(t)}$player aims at minimizing$\mathcal{L}_{W_{k,n}^{(i)}}$, while the λ-player wishes to maximize $\mathcal{L}_{\lambda}$.While optimizing the first Lagrangian w.r.t. $W_{k,n}$ requires differentiating the constraint functions $\psi_1\ (W_{k,n}^{(t)})$ and $\psi_2\ (W_{k,n}^{(t)})$, to differentiate the second Lagrangian w.r.t. λ we only need to evaluate $\phi_1\left(W_{k,n}^{(t)}\right)$ and $\phi_2\left(W_{k,n}^{(t)}\right)$. Hence, a surrogate is only necessary for the $W_{k,n}$-player; the λ-player can continue using the original constraint functions. The local optimization task can be written as,

$$\min_{W_{k,n} \in \Delta} \max_{\lambda, ||\lambda|| \leq R_{\lambda}} \mathcal{L}_{W_{k,n}^{(t)}} \qquad (7a)$$

$$\max_{\lambda, ||\lambda|| \leq R_{\lambda}} \min_{W_{k,n} \in \Delta} \mathcal{L}_{\lambda} \qquad (7b)$$

where thanks to Lagrange multipliers, the λ-player chooses how much to weigh the proxy constraint functions, but does so in such a way as to satisfy the original constraints, and ends up reaching a nearly-optimal nearly-feasible solution [23].

3.5 Results

This section analyzes the proposed Closed loop EFL framework in detail. To build the explainability-aware constrained traffic drop classification model, we use feature attributions which is the pillar of this approach. After that, we present the impact of considering jointly the recall and log-odds metrics as constraints for optimizing the FL classification problem by showing results of FL convergence and log-odds score. Finally, we study the correlation between features attributions, observed predictions, and true predictions and draw some important conclusions. Specifically, to implement the modelTester and Explainer, we invoke DeepExplain framework, which includes state-of-the-art gradient and perturbation-basedattribution methods [24]. It provides an attribution score based on the feature's contribution to the model's output, which we integrate with our proposed constrained traffic drop classification FL framework in a closed-loop iterative way.

3.5.1 Parameter Settings and Baseline

We consider below three primary slices eMBB, uRLLC and mMTC to analyze the proposed Explainable FL policy. Here, the datasets are collected from the BSs and the overall summary of those datasets are presented in Table II. We use vector β for the explainability lower bound threshold and α for the upper bound of recall score corresponding to the different

slices. As a baseline, we adopt vanilla FL with post-hoc integrated gradient explanation.

Table 2: Settings

Parameter	Description	Value
N	#Slices	3
K	#BSs	50
DNN	Deep neural network size	2-hidden layers with 10 nodes
$D_{k,n}$	#Local dataset size	800 samples
T	#Max FL rounds	50
L	#Local epochs	100
R_λ	#Lagrange multiplier radius	Constrained: 10^{-5}
η_λ	#Learning rate	0.02
U	#Total users (All BSs)	15000

3.5.2 Results Analysis

In this scenario, resources allocated to slices according to their traffic patterns and radio conditions while ensuring a long-term isolation via the constraints.

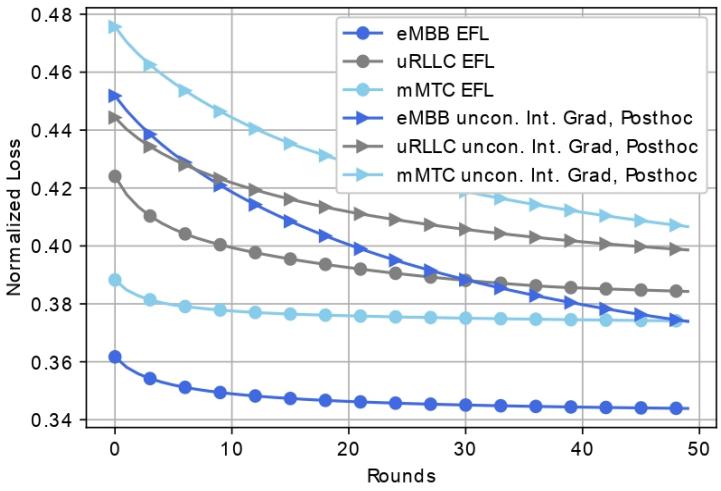


Figure 3-3:Analysis of FL training loss vs FL rounds of Proposed EFL with Lower bound of Recall score, α =[0.9, 0.95, 0.95] and Upper bound of log-odds score, β =[−0.01,−0.01,−0.01]

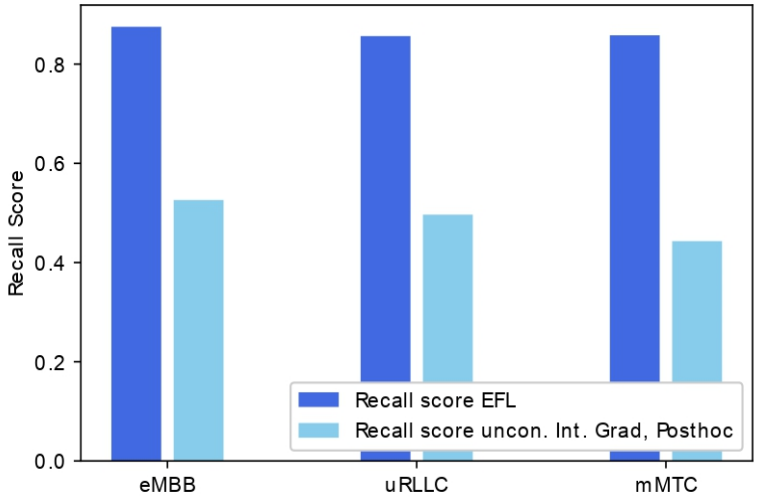
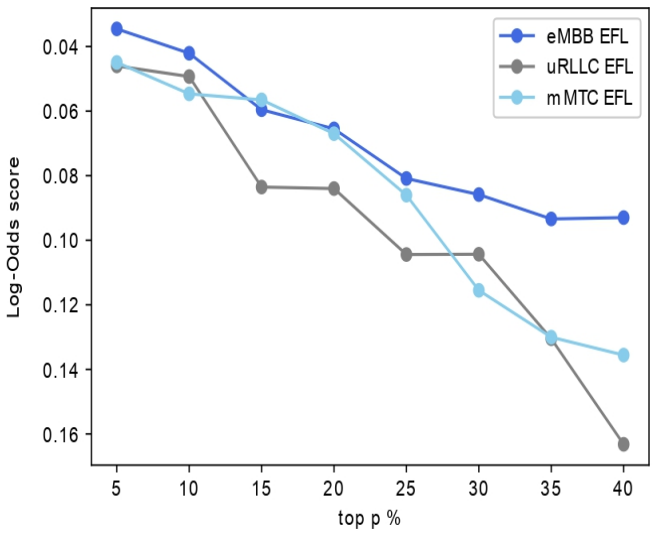


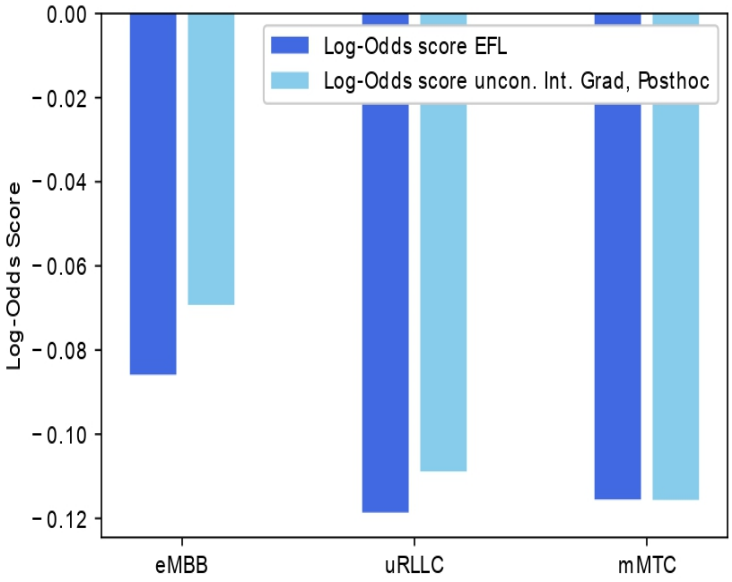
Figure 3-4:Analysis of Recall score with Lower bound of Recall score, α = [0.9, 0.95, 0.95] and Upper bound of log-odds score, β = [−0.01,−0.01,−0.01]

- **Convergence:** As depicted in Fig. 3, we can conclude that the proposed constrained EFL resource allocation model of the different slices has converged faster than the baseline unconstrained IG post-hoc case.

- **Sensitivity analysis:** To analyze our proposed model's sensitivity, we choose the recall metric, which is the rate of actual positive values for measuring the performance of our binary classification model. From Fig. 4, we can observe that the recall score of the proposed one for all slices is near the target threshold γ (i.e., around 0.88%), which is an acceptable value for operators and slices' tenants.



(a) log-odds Score vs. top p %



(b) log-odds Score

Figure 3-5:Analysis of log-odds score with Lower bound of Recall score, α = [0.9, 0.95, 0.95] and Upper bound of log odds score, β = [−0.01,−0.01,−0.01]

- **Trustfulness:** In Fig. 5-(a), we observe the effect of changing the value top p% on the log-odds, considering proposed model for all slices. Also, we present a comparative analysis of log-odds score in Fig.5-(d) for both cases which proof the superiority of proposed constrain EFL model. So, the statistics of the log-odds score give us an approximate idea of our model's reliability and trustworthiness. It shows that the log-odds score is decreasing with respect to the top p% value, which conveys that our model is explainable and trustworthy in the training phase.

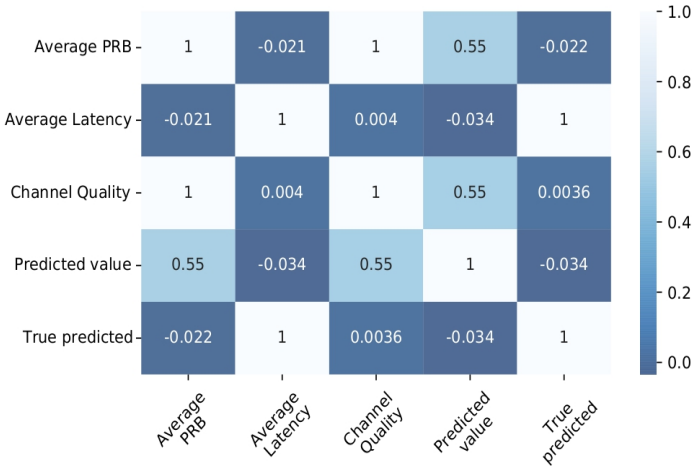


Figure 3-6:Correlation heatmap of eMBB slices based on attribution scores of features generated by XAI

Furthermore, we demonstrate in Fig. 6, the correlation heatmaps of the proposed XAI method of the eMBB slice for further analysis. This approach helps us visualize the strength of relationships between different variables and, in our case, identify which feature variation impacts the most for SLA variation. To plot correlation matrix heatmap, we consider one matrix, $R_{k,n} = [a_{k,n}, \hat{y}_{k,n}, y_{k,n}]$, where, $a_{k,n}$ is the attribution score of features variable with dimensions $D_{k,n} \times Q$ and $\hat{z}_{k,n}$

is the predicted output variable with dimensions $D_{k,n} \times 1$ and $y_{k,n}$ is the true predicted value with dimensions $D_{k,n} \times 1$. From the heatmap we see that the third feature, which is the channel quality, has the most impact on the recall value. If the third feature increases, the recall value will increase and vice versa.

3.6 Conclusion

This paper has presented a novel closed-loop explainable federated learning (EFL) approach to achieve transparent zero-touch service management of 6G network slices at RAN in a non-IID setup. We have jointly considered explainability and sensitivity metrics as constraints in the traffic drop prediction task, which we have solved using a proxy-Lagrangian two-player game strategy. From the results, we conclude that the proposed EFL scheme is reliable and trustful compared to state-of-the-art unconstrained post-hoc FL. Finally, the heatmaps of the attributions correlation matrix are presented to showcase the features whose variation influences more the traffic drop.

Here, we present a 6G RAN-edge network architecture, as well as preliminarily works and results, based on which we will proceed to implement our proposed idea. Our main aim is to include the XAI approach in our current implemented framework. Moreover, to gain more trust and reliability in our proposed solution, we may do a comparative analysis of existing XAI and show some related results. This approach is helpful and trustable for decision-making cases of any kind of critical service in the telecommunication field. Furthermore, it will be beneficial for any telecom operator or service provider to broaden their deployment and services, which is the future goal of the 6G network.

4 Intelligent QoS Aware Slice Resource Allocation with User Association Parameterization for Beyond 5G ORAN-based Architecture using DRL

4.1 Introduction

The 5G verticals can be categorized into 3rd generation partnership project (3GPP) de- fined use cases such as enhanced mobile broadband (eMBB), ultra-reliable
low latency (URLLC), and massive machine type communication (mMTC). With various applications, such as Industry 4.0, smart cities, V2X, video streaming, online gaming, AR/VR, etc., the key performance indicators (KPIs) for the mentioned use cases can change. Such diversity in KPIs and requirements can be achieved with the use of new technology enablers.
Furthermore, it introduces a sophisticated programmable architecture with a radio access network (RAN) intelligent controllers that provide an infrastructure-based abstraction of the networks as well as applications performing closed-loop control for RAN radio resource management (RRM). Network slicing is another important technology enabler, as it essentially allows to customize corresponding services [25] and to adapt to changing traffic requirements [26], [27]
Due to the diversity and complexity of the criteria for future applications under the eMBB and URLLC use cases, this research focuses on these use cases and establishes a slice category for each of them separately. It is crucial to include its awareness [28], [29], [30], while allocating resources to end users. One of the challenges in handing resources to the RAN edge domain is traffic load variation in the wireless network environment. It affects optimal resource allocation, thereby reducing resource utilization and also causing System Level Agreement (SLA) violations as well as degrading the quality of service (QoS) of the end users. Slicing can be tuned to maximize resource utilization and QoS while isolating the performance of individual slices, even under traffic uncertainty. Fine-grained resource reconfiguration has extremely high computational complexity and is a sequential problem when it comes to network slice resource configuration [31], [32]
Another difficult aspects of radio resource management at the RAN edge domain are in terms of connectivity preservation, rate control, offloading, etc.

Moreover, due to the non-deterministic nature of wireless channel conditions, mobility, traffic, etc., as well as their inherent complexity, dynamic RRM is a challenging task.
Thus, it becomes hard to construct models using conventional algorithmic approaches. On the other hand, model-free artificial intelligence (AI) techniques offer effective solutions that are gaining momentum, making them crucial candidates to optimize dynamic RRM [32]. Reinforcement learning (RL), one of the branches of AI, has been proven useful to deal with control problems. One of the most popular algorithms is deep reinforcement learning (DRL) and its variations. The optimization of resource allocation in radio access networks has been successfully accomplished with DRL algorithms, including DQN, DDPG, etc. [32]. DRL techniques have also been proposed to optimize power consumption and other aspects in different RAN architectures before ORAN, such as cloud RAN [28]. The study presented in this paper focuses on integrating the aforementioned technological enablers and proposes deep Q-learning-based techniques to implement effective, dynamic, optimal resource management at the RAN edge domain in order to provide end users that fall into various slice categories with high-Quality of Experience (QoE) and Quality of Service (QoS).

4.2 Related Work

The ORAN Alliance is actively introducing a non-proprietary version of the RAN system that allows interoperation between network components by different vendors. In the past, intelligent solutions have also been proposed with proprietary architectures such as [28], [29], [30] but ORAN architecture introduces a sophisticated programmable approach with radio access network (RAN) intelligent controllers that provideinfrastructure-based abstraction of networks. In the ORANarchitecture, there are two separate components: the non-real- time RAN intelligent controller (non-RT RIC) and the near-real-time RIC (near-RT RIC), which reinforces the importanceof the two control levels with different time granularities [33]. The ORAN report and specification include a comprehensive review of a number of use cases that are expected to be integrated with architectural elements to exchange data between various network components. One of the use cases describes how to integrate intelligence at the network's edge in regards to radio resource management (RRM) [34].
These components of the ORAN framework enable the integration of dynamic changes at almost real-time levels for radio resource allocation to end users. Slicing is one ofthe enablers that provide isolated resources. The customized network components are dedicated and isolated specific toeach slice category [35]. There are two different levelsof resource allocation and management: intra- and inter-slice, which help manage RAN edge domain resources effectively. The authors in [8] define an optimization problem that aimsto maximize user perceived throughput while minimizing packet delay violations by modifying the MAC scheduling algorithm's parameters. It comes under the intra-slice resourceallocation category. This specific work is limited to traditionalnetwork architecture, proposing optimization for individual base stations and their underlying users. Such proprietary solutions can be used with ORAN architecture, but they lack the adaptation to utilize programmable ORAN-based infrastructure, which facilitates access to the information availableat a centralized entity from other base stations or ORAN radiounits (ORUs), which plays a crucial role in RRM decisions.
Reinforcement learning (RL) is a very effective tool to achieve optimal decisions in complex environments, such as non-deterministic wireless networks, due to several aspects discussed in Section 1. Traditional table-based RL techniques are infeasible to handle large state and action spaces, as concluded in [5]. Whereas, deep reinforcement learning overcomes these limitations, like deep Q-learning and some of its variations, such as dueling deep Q-learning (Dueling DQN) and deep deterministic policy gradient (DDPG). In the case of large and multidimensional discrete action spaces, branching architecture can be introduced in dueling Q networks, as presented in [11]. The researchers in [5] extend the given neural network (NN) for inherently discrete multi-dimensional and large action spaces to resolve the network slice reconfiguration problem. More computational resources are required to supportsuch solutions.

In state-of-the art work like [36], researchers opt for the approach of using parallel and multiple DNN to make optimal decisions for user association with the base station. The solution proposed in [36] builds multiple DNNs and a number of additional DNNs with random input to increase exploitation. It has K DNNs, N random decisions, and K + N functions for calculating the Q value for a network with a certain number of users. So, DRL goes through all the layers in every run. Furthermore, each DNN is trained to learn the optimal decision for the entire user association (UA) matrix. It requires higher computation capabilities. Instead of this heuristic-style approach, the association decision can be parameterized to learn useful information about the network such as base station or ORU-based parameters. It is possible to establish various network slice resource allocation strategies by utilizing a variety of RL approaches [13], [14]. Dynamic network slice distribution techniques to improve performance for 5G- based bandwidth offerings is evaluated in [14]. It compares different algorithms for slice resource distribution. [13] uses approximation of resources and explores subchannel allocation that impacts differently with changes in multiplexing techniques. The solutions need to be formulated in such a way that they work efficiently and provide the intended outcome, independent of variations in techniques used in the wireless communication network underneath.

Unlike the metrics taken into account in [7], [15], [16], and other papers, the various slices in the network have different KPI thresholds to fulfill the defined QoS requirements. This poses a challenge in terms of achieving optimal system performance that is tackled in this paper by considering the variation in metrics or KPIs for different types of services that come under various RAN slice categories. Hence, the network should support distinct slice KPI-based reward definitions for individual end users to evaluate different service metrics. For instance, previous research [18], [19] considered different types of services based on varying bit rate and delay requirements, each of which took into account a distinct set of metrics for its requirements. Motivated by this work, the research presented in this paper considers the variation of QoS thresholds and KPI metrics as well as their impact while allocating resources to various slices in the formulation. This work does not prioritize only a specific service category as discussed in the above papers, such as constant bit rate. Instead, it defines weights to evaluate metrics of different slice categories included in reward function design of intelligent agents. Furthermore, each slice category has a separate intelligent agent that learns the importance of parameters and receives rewards based on weighted metrics and QoS thresholds for different slice categories.

The proposed work discusses a system model based on ORAN architecture to deal with intra-slice RAN RRM decisions taken in real time for eMBB and URLLC slices. The problem is formulated to achieve optimal system performance and KPIs while satisfying the QoS of individual end users. In contrast to the work talked about in [30], the proposed work intends to learn the importance of UA parameters with a unique approach for slice-based beyond 5G networks with multiple ORUs. The presented research work considers resource availability at all ORUs to serve the buffered traffic at users under specific categories and allocates the resources to achieve a high QoS. As indicated in introduction, traffic uncertainty affects RAN edge domain resource allocation [32] [31]. Hence, the formulation in this paper considers parameters reflecting network traffic load that have a significant impact on resource management. It takes into consideration the most crucial aspects of a wireless communication network in terms of end users and ORUs present in the network, as indicated in the problem formulation section. A resource allocation or reconfiguration problem can be written in such a way that the action space is constrained to a limited number of actions to which DQN can converge. It is also vital for intra-slice RRM in terms of time granularity of 1–10 ms for the near-RT RIC component of ORAN.

The contributions of this work are as follows:

1. The proposed algorithms deal with the association of user-ORU and serve the traffic load at individual users. In contrast to techniques discussed in the state of the art, the algorithm and intelligent agent sit at the remote edge, near-RT RIC component of the ORAN architecture. It is completely adapted to the ORAN architecture, which facilitates dynamic configurations of resources within a smaller timescale of 1–10 ms. It benefits from centralized access to the end-user and ORU information for under- standing the factors affecting the respective decisions.

2. The DQN approach iterates through all actions, and hence the proposed formulation ensures the limited number of action spaces that impact performance, reducing the convergence time. In addition to this, as it avoids large action spaces, the desired results are achieved without additional NN architectures or DQN variants, unlike [37], [38]. The computation capabilities required at the edge server, processing time, and training time are reduced compared to other approaches. This is vital to achieving smaller time scales for near-real-time decisions. Furthermore, the action space defined in proposed MDP is independent of number of users in the network.

3. Instead of learning the whole user association matrix as done in the state of the art, the proposed work distinguishes itself by parameterizing the user association. The agents learn the weights for each of the parameter crucial for UA decision. With this, the computational and memory

requirements can be reduced, as can the con- vergence time. A problem formulation is proposed that utilizes the intelligently selected weights and concludes the actual decision. It avoids the parallel and multiple DNN approaches used in several state-of-the-art works for UA.

4. Based on how well individual users perform, each intelligent agent receives rewards. The reward function is de-signed to include deviation of the user performance from the QoS threshold value. Further these KPIs are weighted based on the slice category. The combination of these helps to get better performance compared to baseline approaches for KPIs like throughput, delay, BER, and overall system performance.

4.3 Network Architecture

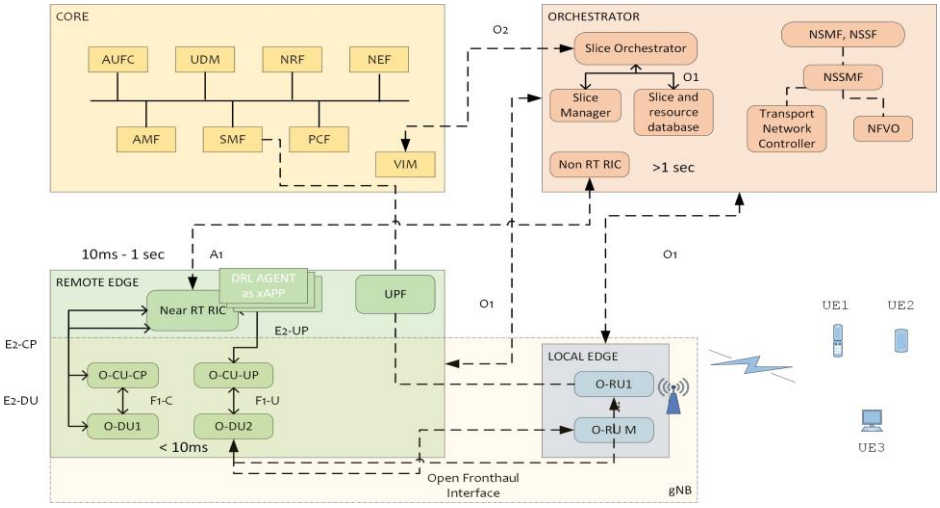


Figure 4-1 ORAN based Network Architecture

We consider a Radio Access Network (RAN) architecture as shown in Figure 4-1 based on the Open-Radio Access Network (ORAN) standard, as defined by the ORAN specifications [39]. The RAN comprises a number of users served by various network slices, each tailored to meet specific QoS requirements. The end users are associated with the ORAN radio unit (ORU), which acts as a transceiver radio unit with antennas and a low physical layer (PHY). This unit is further connected to the ORAN distributed unit (ODU) and the ORAN centralized unit (OCU), which operate at higher protocol layers. In this RAN architecture, each user is associated with and served by an ORU under the specific Network Slice Subnet Instance (NSSI) as defined by the ORAN standard [34]. Each NSSI corresponds to requested services and has specific KPIs. The connectivity extends to ODU, OCU, the Service Management and Orchestration (SMO) system, and the core network, providing end-to-end connectivity. The RAN Intelligent Controller (RIC) plays a crucial role in making resource allocation and management decisions intelligently at the network's edge. This increased openness allows for better decisions with more information on network parameters.

The ORU is located at the local edge, while the ODU, OCU, and near-real-time (RT) RIC are part of the remote edge implementation at the edge servers, thus coming under multi-access edge computing. The ORU communicates with the user equipment (UE) over wireless channels, while vendor- specific open fronthaul interfaces connect the ORU to the remote edge for both uplink and downlink transmissions. For the practical implementation of the proposed algorithm, the channel state information (CSI) can be extracted from the database of the near-RT RIC at the edge server as reported by the E2 node via E2-CP. Therefore, we assume that perfect CSI is available for resource allocation and management at the edge server. Additionally, each ODU and OCU at the remote edge server connects to the SMO via O1 and A1 interfaces. The RAN communicates with core network entities via a fronthaul link, such as Ethernet, PLC, or optical fronthaul. Notably, the fronthaul link has an

upper bound in terms of maximum bits transmitted per second and is subject to limited core network resources assigned to each slice or network subnet at the edge or cloud, including computational capabilities.

The RAN NSSI resource management is executed at two different control loop levels, namely non-real-time (non-RT) and near-real-time (near-RT), in line with ORAN specifications [39]. For each ORU, SMO provides a default RAN resource configuration via the O1 interface as part of non- RT resource management. It includes radio resources, such as bandwidth, and computational resources reserved for different types of slices. Consequently, each RAN NSSI has a pre- allocated portion of bandwidth to serve the associated users. The performance of each network slice is analyzed based on the requested QoS configurations to achieve Service Level Agreements (SLAs). Intelligent agents are used in the proposed dynamic RAN resource management policies to make sure that radio resources are given to different types of slices in the best way and meet the QoS requirements set by the

4.4 System Model

As shown in the Figure 4-2, consider the ORAN network with a general set of users K = {1, · · · , K}. The users can request services under any of the two slice categories eMBB and URLLC. Each user has QoS requirements to be fulfilled based on the type of slice requested. These QoS requirements are expressed in terms of the minimum data rate to be achieved per user Rmin and the maximum allowed delay per user dmax where k ∈ K. We assume that all users from the same slice have the same minimum data rate and maximum delay requirements. Both the slices follow different traffic generation models due to the variation in services under these slices. The packet arrival rates for eMBB and URLLC slices follow periodic deterministic traffic model with specific packet arrival intervals. The size of packets and number of packets for each user vary based on the slice category and traffic distribution models, respectively. We assume it is the same for all users under the same slice category indicated for each user k as Sk and Lk, where k . The slice type as well as service requirements are mentioned for each UE in the UE database available at the remote edge. The set of ORUs denoted b y M serves these users.

$$a_{k,m} = \begin{cases} 1, & \text{If } k \text{ is associated with } m \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Where, $m \in \mathcal{M}, k \in \mathcal{K}$

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1M} \\ \vdots & \ddots & \vdots \\ a_{K1} & \cdots & a_{KM} \end{bmatrix} \quad (2)$$

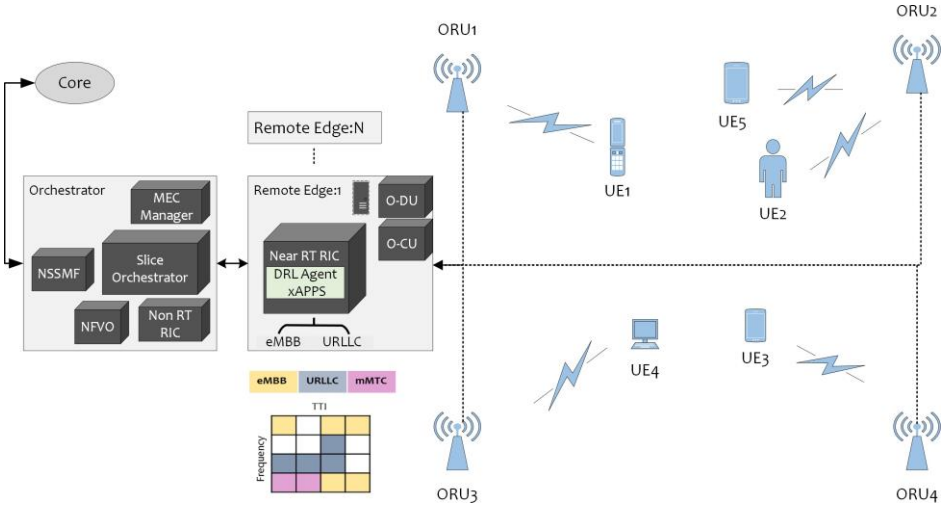


Figure 4-2 System Model

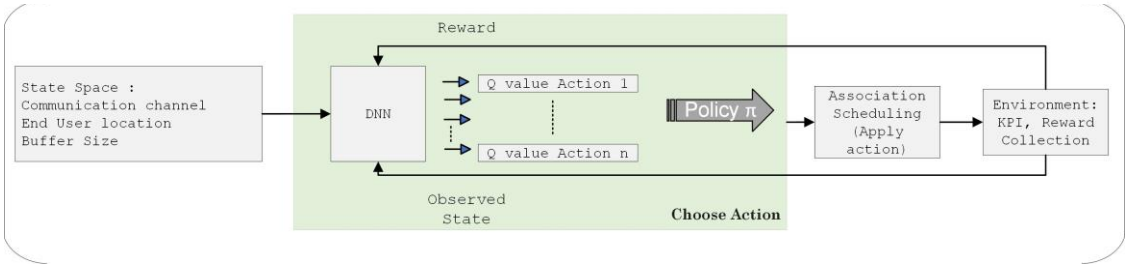


Figure 4-3 DRL Agent at Near-RT RIC (Remote Edge)

$$\mathcal{M}_s = \{m \in \mathcal{M} \mid a_{k,m} = 1\}, \; k \in \mathcal{K}_s \quad s \in \{E, U\} \tag{3}$$

To make the best use of resources within a slice, there is a separate intelligent agent for each RAN slice at the near-RT RIC. These agents learn how to better manage and assign radio resources within the ORU's assigned bandwidth. Its objective is to associate users with ORU and schedule resources in an optimal way to achieve the required QoS performance. We consider downlink (DL) frequency division duplexing (FDD) transmission. Let the binary variable indicate the association between ORU and users in DL and can be expressed as matrix A. Each user will be associated with one of the ORUs for transmission at a given time for DL. As discussed earlier, there are two types of slices: URLLC and eMBB. These slices can be instantiated in any ORU. Each slice has a separate association matrix as with corresponding ORUs and users. It is also assumed that a user can only be connected to one ORU at a time, and a user may be assigned one or multiple physical resource blocks (PRBs) based on the traffic at each user. The bandwidth allocated to each ORU is divided among the slices instantiated in the ORU. Hence, ORU has a specific number of resources PRBm available, and the corresponding slice has a total PRBm,s number of PRBs available where s E, U . We assume additive white gaussian noise for all users of independent circular symmetric natured complex random variables with a zero mean and σ2 variance. All users in the network follow random mobility model with speed V. As per 5G new radio (NR), we only consider numerology μ = 0.

$$\gamma_{k,m} = \frac{P_m |h_{k,m}|^2}{I_{k,m} + \sigma^2} \tag{4}$$

$$I_{k,m} = \sum_{i \in \mathcal{M}, i \neq m} P_i |h_{k,i}|^2 \tag{5}$$

Here, all the parameters indicated are with respect to a single TTI t; hence, for simplicity, the notation t is omitted from all variables unless the time duration is other than 1 TTI or a reference to a future or past TTI value is required. In DL, we use hk,m to denote the channel coefficient from ORU m to user k. The frequency-selective flat fading channel passes through Rayleigh fading in the wireless medium. Let us assume that Pm is the total transmit power of ORU. The proposed formulation aims to allocate and manage radio resources within each slice in a dynamic and optimal way such that all users satisfy the QoS requirements of requested services. The majority of existing proprietary and state-of- the-art based solutions address the importance of including intelligence at the inter-slice level. But it is equally crucial to include such intelligence in the intra-slice resource allocation to adapt to the diverse KPIs of future envisioned applications. The proposed work tries to fill these gaps while achieving optimal system performance. The SINR values γk,m are calculated as per equation (4) and they include effects such as path loss and shadowing based on the distance as well as the wireless channel quality between the user and ORU. Furthermore, it considers other interfering components in the network that affect the signal strength of the actual signal. The estimation of assigned PRBs is included in one of the

proposed algorithms, as prior knowledge of this parameter increases the KPI performance observed through several tests. The ability of ORU to serve traffic in a network is evaluated with a performance metric for ORU, τm defined as the ratio of buffer size in a given TTI t at ORU m to actual transmitted bits in the same TTI for the respective ORU. This metric is a type of queueing delay estimator that indicates how fast an ORU can serve the user. The metric τm is calculated as below:

$$\tau_m = \frac{C_m}{J_m} \tag{6}$$

$$C_m = \sum_{k \in \mathcal{K}_m} B_k$$
$$J_m = \sum_{k \in \mathcal{K}_m} TB_k \tag{7}$$

where Km is a set of users associated with m-th ORU, Bk is number of bits in buffer for each associated user and T Bk is number of bits transmitted for each associated user under ORU m at TTI t. Further, from the experienced value of τm,t in each TTI for each m-th ORU, we calculated global value. Initially, the global value is set to zero and updated with each TTI. Each experienced τm,t within each TTI is utilized to update the global τ gm,t as shown below.

$$\tau^g_{m,t} = \frac{\tau_{m,t} + \tau^g_{m,t-1}}{2} \tag{8}$$

These parameters represent crucial aspects to be considered in the decision-making process of associating users with ORUs. The proposed formulation computes the indicated parameters to evaluate their importance and learn optimal decisions for the given state of the network environment.

4.4.1 Intelligent QoS aware Resource Allocation (IQRA)

The IQRA focuses on selecting the optimal association decision based on a combination of the above-discussed parameters calculated in current TTI. It sits at near-RT RIC as an xAPP. Initially, a database is generated that stores all possible combinations of associations between the available users and ORUs of a specific slice. With this database as a reference, the algorithm can calculate and estimate the required parameters that contribute to association and scheduling decisions within the available slice resources. As discussed in the earlier section we indicate association with matrix A defined in equation (2). For a simplified approach in IQRA, the association is expressed in a vector form deducted from the same matrix A. Each vector element is the value of the associated ORU m, where the index of the vector element indicates the UE k. It is defined as given below. This definition reduces the number of possible combinations for associating UEs with ORUs. For given K and M there are I such combinations available. Each i-th combination represents a unique A matrix A as per equation (2). For each of these combinations, SINR γk,m,i is calculated as per equation (4). Based on the range of calculated γk,m,i value in dBm, the modulation and coding scheme (MCS) such as QPSK, 16 QAM, 64 QAM, etc. is selected for each user from the look-up table (LUT). The MCS scheme determines the modulation order Ok,i for each user in every i-th combination and the number of bits to be transmitted per resource element.

$$\vec{A} = \{m_1, m_2 ....., m_k\} \tag{9}$$

As per 5G NR configurations, the number of symbols per slot, number of slots per subframe and total number of subcarriers can be determined. Hence, we can calculate bits per PRB for each user. From all these parameters, the required number of PRBs at user k can be estimated as follows using available number of bits in buffer at user k, Bk and the number of bits that can be transmitted at user k in each PRB,

$$PRB_{k,i}^{\text{required}} = \lceil \frac{B_k}{B_{\text{PRB}_{k,i}}} \rceil \quad (10)$$

Further, scheduling is performed based on the required number of PRBs for each user and the available bandwidth for each ORU. The proposed user association selection scheme is independent of the selected scheduler. In Section VIII, Proportional Fair (PF) scheduler is simulated. After scheduling, the estimate of assigned resources for each user is available.

$$\tau_{m,i}^{g} = \frac{\frac{C_{m,i}}{J_{m,i}} + \tau_{m,t-1}^{g}}{2} \quad (11)$$

The estimation of assigned PRB and tau matrix for each i-th association contributes to the association decision AD as given below:

$$AD_{s,i} = w_{s1} \sum_{k,m} a_{k,m,i} \cdot PRB_{k,i}^{\text{assigned}} - w_{s2} \sum_{m} a_{k,m,i} \cdot \tau_{m,i}^{g} \quad (12)$$

The values for ws1, ws2 are learned and updated as described in Section 6 by intelligent agent for each slice. The optimal association decision is selected from all the available i combinations in the set with the following equation:

$$AD_{s,opt} = \arg\max_{i} (AD_{s,i}) \quad (13)$$

subjected to:

$$\begin{aligned} &C1: d_k \leq d_k^{max}, \ k \in \mathcal{K} \\ &C2: R_k \geq R_k^{min}, \ k \in \mathcal{K} \\ &C3: \sum_{m \in \mathcal{M}} \sum_{k \in \mathcal{K}} R_{k,m} \leq N_{fronthaul} \\ &C4: a_{k,m} \in \{0,1\}, \ \forall_{k,m} \in \mathcal{K}, \mathcal{M} \\ &C5: \sum_{k} PRB_k^{\text{assigned}} \leq PRB_s, \ k \in \mathcal{K}_s, \ s \in \{E,U\} \end{aligned} \quad (14)$$

4.4.2 Low Complexity Intelligent QoS aware Resource Allocation (LIQRA)

This algorithm is proposed for the same objective in intra- slice RAN RRM with reduced computational complexity, which is crucial for some applications. The algorithm is constructed similarly, but evaluates association decisions in a different way. It takes association decisions individually for each user k and estimates the matrix A. The algorithm aims to make an intelligent association decision based on a combination of signal-to-noise ratio (SNR) and the global ORU metric. Once the association decision is taken and matrix A is estimated, SINR is calculated based on equation (4) and hence, MCS is selected. The values for ws1, ws2 are selected by an intelligent agent for each slice. Further, the required PRBs and other parameters for each user k are calculated as indicated in the earlier algorithm to schedule radio resources for end users.

4.4.3 Key Performance Indicator

Once the resources are allocated, the transmission takes place, and experienced key performance indicators such as achieved throughput, delay, and bit error rate (BER) are collected. These KPIs are stored, and the mean performance of the last TTIs for each user is forwarded to DRL. Additional KPIs, such as successful packet transmissions and packet drop rate, for each UE are also evaluated over the defined time interval. The throughput of user k is calculated by multiplying the assigned number of PRBs with bits per PRB selected for transmission in that TTI. The total delay experienced per user is a

combination of delays experienced by each packet in queue, transmission, and processing. The BER for each user k associated with m-th ORU in every t TTI for frequency selective flat fading channel is calculated as given in [40] The packet loss rate for every T TTIs is calculated as the ratio of packets discarded in given time interval to the total number of packets transmitted for each user.

4.5 Proposed DRL based Intelligent Agents

In this study, distinct intelligent agents for the slice types, eMBB and URLLC, are proposed. The UA decision is parameterized as discussed in above section. The DRL technique namely DQN is used by the intelligent agent to discover the significance of either the assigned PRB estimate or the SNR metric γ and the ORU metric τ for suggested strategies. Based on the state of the network environment at any given time, the deep neural network (DNN) determines the weights for each of these parameters and evaluates the optimum course of action. The DNN trains and learns the weight values for various network environment states and scenarios. IQRA and LIQRA are then given with these values for ws1 and ws2 separately for each slice. The formulated solutions in the earlier section take into account the weights and evaluate a decision for ORU selection to serve the corresponding end user. This results in better performance for the KPIs compared to baseline and state-of-the-art solutions. The proposed Markov's decision process (MDP) establishes the significance of the weights for UA parameters using DQN based intelligent agents. Each intelligent agent learns different values based on the impact of individual experienced user KPIs under each slice category.

4.5.1 Markov Decision Process

The Markov Decision Process (MDP) for the intelligent agents based on the above formulation is defined with a tuple of Ss, As, Rs , Γs corresponding to state, action, reward, and discount factor, where subscript s indicates slice type for intelligent agents. The state space Ss of the environment includes the current log normalized channel matrix HM×K, number of packets to be transmitted in the buffer per user Pk, and the distance between each user and ORU in the current TTI t. The state space definition remains the same for the different slices, as it represents the current state of the environment based on which the actions will be chosen by the intelligent agent. Different intelligent agents learn the importance of the UA parameters, which are specific and valid to the varying traffic flow for the services and user-specific to each slice. The number of actions is limited, and the agent learns the importance of each parameter to make an optimal decision for each slice separately. Several deep learning techniques and MDP designs come across issues while achieving convergence with a huge action space. The intelligent agents will learn the optimal way to associate user-ORU and eventually schedule the resources based on observation at a given point in time in an environment of wireless communication. The reward function is designed in such a way that the agent receives the reward for the user with a positive value equal to a fraction of exceeding QoS thresholds. Whereas, if any user fails to reach the QoS threshold, it is awarded a negative value equal to the amount of KPI that failed to reach the threshold. This reward is further normalized by the threshold values themselves. The total reward is the accumulated reward of each individual user divided by the total number of users. Both IQRA and LIQRA can use this definition of reward function. We can define slices dedicated to specific service and set the priorities of KPIs in reward function according to the requirements of that service. This ensures that algorithms achieve optimal system performance by keeping the service requirements in check and the quality of experience of each user maintained. Once the agent learns optimal weights for different states, the slices obtain desired and better individual user and system performance. The conducted research has analyzed various designs of reward functions to get better convergence and KPIs. We use deep Q learning methods from DRL algorithms.

4.6 Simulation Setup

The designed simulator is 3GPP-standard-compliant for wireless communication models. The network elements are defined and implemented in line with the ORAN architecture. The simulation set-up is based on ITU-R M recommendations for testing and verification. It is implemented in a Python environment. It uses OpenAI-gym, Keras, and Tensorflow modules to implement DQN-based agents. The intelligent DQN agents for both slices are tested and tuned by running multiple combinations of hyperparameters. The final selected values are mentioned in Table I. For each network slice, the

number of users requested to be served and their QoS requirements are available in the database at near-real-time RIC. We have tested the proposed algorithms for a number of settings of QoS values based on different packet arrival rate (PAR). The simulation set up include packet arrival based on distribution models as well as constant bit rate (CBR). The number of packets available to transmit for each user is determined per TTI based on the mentioned models with different times of arrival in reference to the current time stamp.Similarly, for all UEs, the location update is calculated for eachTTI based on the mobility model with a fixed 30 kmph speed for all users.

As we have discussed in the system model, we assume 5G NR numerology 0 for all users, which corresponds to subcarrier spacing of 15 KHz and 12 subcarriers per PRB, hence the bandwidth of a single PRB $W_{P\ RB}$ is 180 KHz. Here, $W_{m,s}$ bandwidth is allocated to each ORU under each slice category where it is serving both eMBB and URLLC slice users, and then the number of PRBs available at each ORU is given with $W_{m,s}/W_{P\ RB}$ as shown in table II. The symbol duration t_{symb}, number of symbols per slot, and otherparameters are calculated based on a selected combination of numerology and subcarrier spacing. The pathloss and shadow-ing models are defined as a combination of dense urban andhotspot models given in table A1-5 of ITU-R M.2412-0 [41] and
other communication model parameters are listed below. The ITU-R M specification [41] specifies the guidelines for the evaluation of radio interface technologies, specifically for simulation and testing purposes. The ORUs are placed around 50 meters away from each other, with their heights 3-3.5 meters and maximum power levels are 200 mW (linear power of ORU for 24.25–27.5 GHz) according to A1-16 and A1-23 of [41] compliant with ORAN standardization and the ITU framework to support network slicing [42].

4.7 Results and analysis

The results presented in this section are based on the simulation parameters for the environment and DQN agents that were previously described. The eMBB reward function, R_E uses $\alpha_E = 0.7$. Whereas the URLLC reward function, $R_U$uses $\alpha_U = 0.4$. Different values of α_s have been tested for both slices. The algorithms are simulated for 3000 iterations of DRL and 30 seconds of simulation time. The figure shows the reward values of eMBB and URLLC slices forboth the IQRA and LIQRA algorithms. Figure 4-4 and 4-5 show the value for reward for each run, where LIQRA and IQRA refer to reward values plotted with a sliding windowof 100 iterations, and mean refers to the mean reward value. The reward function depends highly on KPI values such as throughput and delay. The instantaneous KPI values are calculated within each TTI, which is 1 ms in duration, and subsequently, individual reward values vary frequently. There-fore, the mean and sliding values of reward have been plotted to analyze the performance of both algorithms. We can see that convergence is achieved within the first 200–300 iterationsof DQN. As seen in Figure 4-4 and 4-5 IQRA performs better for eMBB, and LIQRA has better reward values with URLLC slice. LIQRA has SNR and τ as UA parameters,whereas IQRA has an estimate of PRB assignment instead of SNR. This is weighted by w_{s1} and w_{s2} for both algorithms.In IQRA, the number of PRBs towards end users is given as one of the parameters to take an UA decision. This associates users with ORUs who can assign more PRBs, achieving higherthroughput. The eMBB slice has more stringent throughput thresholds; hence, the more PRBs assigned to end users, the better throughput is achieved, resulting in better reward values. Whereas, for URLLC, latency requirements are more strict, with the first parameters being only SNR helps the algorithm assign users to ORU with better signal quality, giving more importance to the τ metric.

Figure 4-6 to 4-9 shows the comparison for system performance be- tween the baseline approach, state-of-the-art (SOTA), referredto as DRLUA now onward, and the proposed algorithms for both slices. The baseline approach uses a simple maxSNR method for user-ORU association, whereas DRLUA uses a number of parallel DNNs approach along with DQN as proposed in [36]. All of them are implemented on top of proportional fair schedulers. Subfigure 4a and 4b show the Em-pirical Cumulative Distribution Function (ECCDF) for systemthroughput over DRL iterations for baseline, IQRA, LIQRA, and DRLUA. Both the proposed algorithms outperform the baseline and SOTA approaches. The resource allocation toend users is performed separately for each slice category.

Subfigure 4c and 4d show the ECCDF for latency for all four approaches in the simulation.

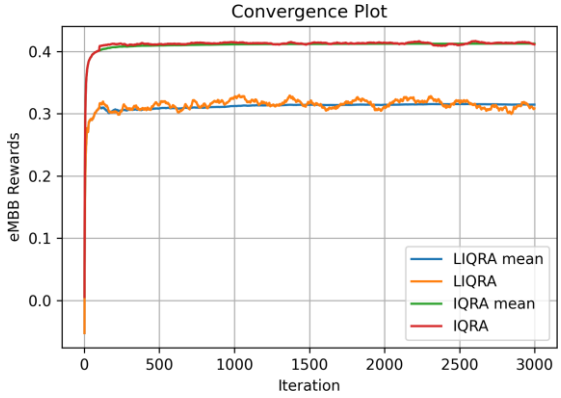


Figure 4-5 reward performance for DRL Intelligent agent

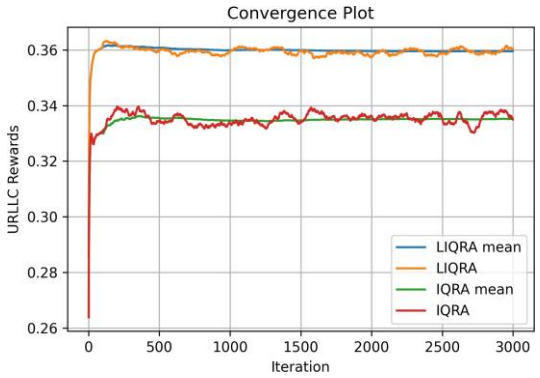


Figure 4-4 URLLC reward performance for DRL intelligent agent

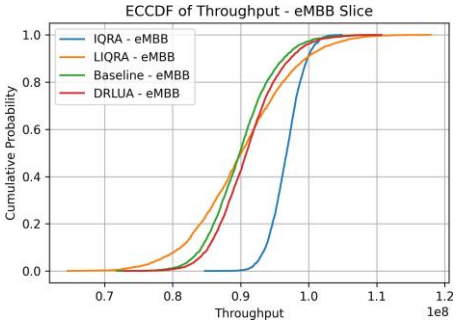


Figure 4-6 ECCDF of Throughput for eMBB Slice

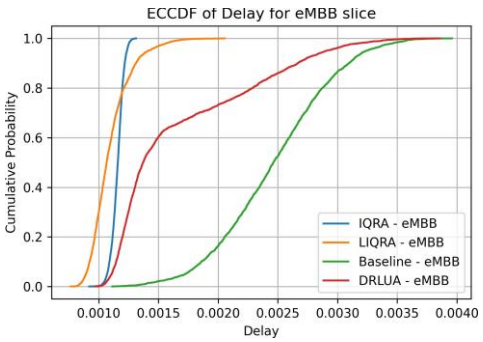


Figure 4-8 ECCDF of Latency for eMBB Slice

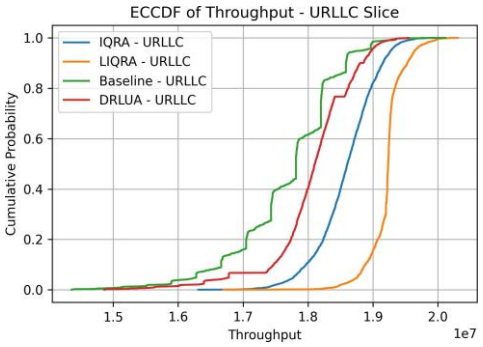


Figure 4-7 ECCDF of Throughput for URLLC Slice

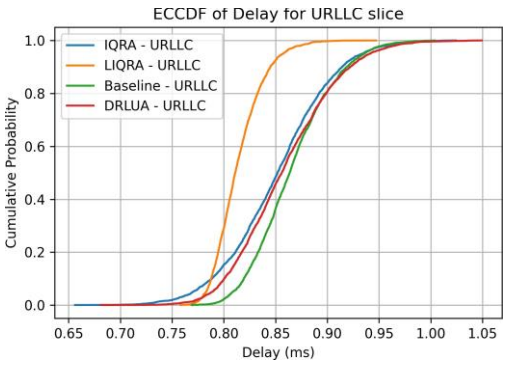
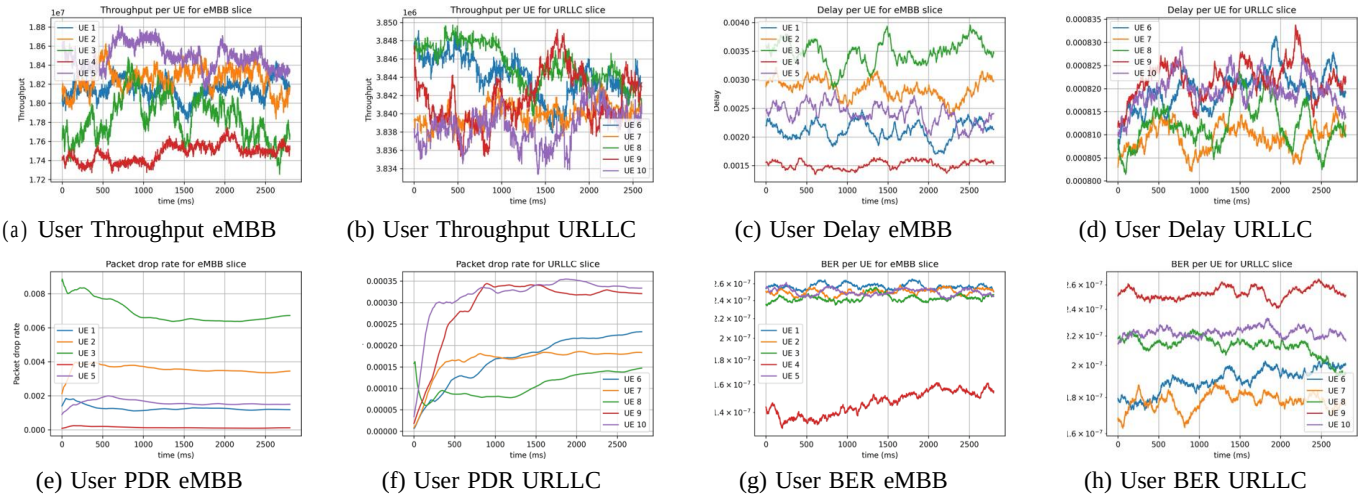


Figure 4-9 ECCDF of Latency for URLLC Slice

The algorithm makes decisions based on weighted valuesof UA parameters learned from intelligent agents. This allows the proposed methods to customize the importance of decision-making parameters specific to the current state of the wireless environment. For the eMBB slice, IQRA works better among the 4 approaches, as it considers estimates of PRBs and theτ metric as explained earlier in reward convergence results.In simulation runs, we observe that IQRA selects higher values for w_{s1} compared to w_{s2} in the majority of cases for eMBB slice. Hence, users are associated with ORUs by giving more importance to the quantity of available resources for transmission considering the allowed delay threshold for eMBB is larger, up to 10 ms. Whereas for URLLC, basedon the network state, the importance lies with τ as the delay thresholds are more strict up to 2 ms. The algorithms allocate more resources to the users based on the QoS thresholds,availability of resources, and state of the network, including the current traffic load. It prioritizes and balances the decisionsfor UA.



(a) User Throughput eMBB (b) User Throughput URLLC (c) User Delay eMBB (d) User Delay URLLC

(e) User PDR eMBB (f) User PDR URLLC (g) User BER eMBB (h) User BER URLLC

Fig. 5: User KPI for eMBB and URLLC slice with IQRA2 Algorithm

Figure 5 shows KPI performances, namely, throughput, delay, packet drop rate, and bit error rate for each user under both eMBB and URLLC slices, respectively, using the LIQRAalgorithm. Subfigure 5a and 5b show experienced throughput for individual users under eMBB and URLLC slices. All users achieve throughput greater than the set QoS thresholds, which are 16 Mbps and 3.8 Mbps for E and U, respectively. Where subfigure 5c shows the delay values for users in eMBB slice. Here, all users experience a delay of less than 4 ms on averagefor all successfully transmitted packets. Subfigure 5e shows that all users in the eMBB slice have less than 0.5% packet drop rate. Only 3rd users experience an 0.9% packet droprate, which is reduced to 0.7% as DRL agents make better association decision. Here, all users experience PDR less than 1% which is acceptable performance for eMBB services. Further, Subfigure 5d shows all users under URLLC slice experience an average delay less than 1 ms. The QoS thresholdfor URLLC, d^{max} is 2 ms; hence, users are associated with the respective ORU by giving more weight to τ_m which ensures the available buffer traffic at each UE can be servedwithin the delay budget. Subfigure 5f shows that majority usersexperience PDR less than $1e^{-4}$. Subfigure 5g and 5h give insights about BER performance for users, which is less than $1e^{-6}$ acceptable for all services considered under both slices.

4.8 Conclusion

From the results presented in the above section, we can conclude that the designed intelligent agents and proposedalgorithms improved the performance of intra-slice resource allocation compared to baseline and SOTA techniques. The intelligence agent learns different weights w_{s1} and w_{s2} for each slice for the respective parameters affecting association decisions. From the comparison, we can see the proposedIQRA algorithm is more suitable for eMBB, whereas LIQRA performs better for URLLC slices. IQRA provides 11.5% and 7.42% improvement in throughput for eMBB slices compared to baseline and SOTA, respectively. While LIQRA provides 19.94% and 16.54% improvement in latency compared to baseline and SOTA approaches, respectively. LIQRA improveslatency by achieving a minimum latency value of 45.5% less than compared to the baseline approach. It also improves throughput for eMBB slices by 6.7%. Whereas, IQRA improves throughput by achieving an increment in maximum throughput value up to 26.7% compared to the baseline approach. It also improves the minimum latency value to less than 8.2% compared to baseline.

5 Multi-Agent DQN with Sample-Efficient Updates for Large Inter-Slice Orchestration Problems

5.1 Introduction

Network slicing is a paradigm that promises to enable one of the key-characteristics envisioned for beyond 5G networks, the reliable support of a massive number of services with widely diverse Quality of Service (QoS) requirements [43]. It leverages network function virtualization and software-defined networking technologies to create virtual networks ("slices") on top of the physical network infrastructure, which can be tailored to the needs of a specific service. The two main goals of slicing are: (i) the fulfilment of the desired QoS metrics (defined by Service Level Agreements (SLAs)); (ii) the efficient utilization of the limited network resources. Since the resource demands of the hosted slices are dynamically changing (due to traffic variations), dynamic slice orchestration is necessary to accomplish the aforementioned goals [3].

A slice is a ``VNF chain" comprising Virtual Network Functions (VNFs) and Virtual Links (VLs). Different optimization problems for network slicing have been considered in the literature, with the main representatives being (i) the placement (embedding) of slices onto the physical network (VNFs and VLs must be mapped to physical nodes and links respectively) [44]; (ii) the allocation of a physical node's resources to the hosted slices (users) [45], [46].

Initial attempts tried to tackle slice placement as an "one-shot" optimization problem, using mainly heuristic algorithms (due to non-polynomial complexity) [47]. More recent works formulated it as a Reinforcement Learning (RL) problem to account for the (unknown) changing VNF demands and the reconfiguration cost [48], [37]. However a number of challenges still remain: (i) most works focus on single domain setups [47] or simple VNF chains [37], and/or consider simple performance metrics [48], [37] (no end-to-end slice-specific Key Performance Indicators); (ii) RL based solutions have to deal with astronomically high action spaces [48], [37], due to the combinatorial nature of placing multiple VNFs upon multiple physical nodes (considering multiple slices/domains exacerbates this problem); (iii) data-efficient algorithms are required, given the scarce(r) availability of cellular network related data [48].

In the previous deliverable (D3.2, Section 4) we addressed challenge (i) by introducing a generic, queuing network based model that captures complex VNF chain topologies and end-to-end performance metrics (supporting multi-domain, multi-slice, end-to-end setups), and challenge (ii) by introducing a multi-agent algorithm of independent DQN agents (iDQN) that can reduce the action space complexity by many orders of magnitude compared to standard DQN. Nevertheless, as convergence can still be slow (requiring a large amount of training data), in this deliverable we introduce two additional mechanisms to store to and select from the experience replay buffer (for more efficient parameter updates), which aim to improve the training speed of DQN agents (challenge (iii)). Finally, as this deliverable focuses on performance evaluation, we validate our multi-agent DRL scheme by extensive simulations, using real traffic data. We summarize below the main contributions of this work:

(C.1) We provide a model that attempts to capture the problem of dynamic slice embedding and reconfiguration supporting a multi-domain setup and diverse, end-to-end SLAs (Section 5.2.1).

(C.2) To deal with the prohibitive state and action complexity of tabular RL algorithms, we propose a novel scheme based on independent DQN agents (iDQN): the DQN (Deep-Q Network) component implements approximate Q-learning, based on simple, generic Deep Neural Networks (DNNs) for value function approximation, radically reducing state space complexity; the independent agents then tackle the equally important issue of exploding action complexity arising from the combinatorial nature of embedding multiple VNFs per slice, multiple slices, over multiple domains and computing nodes therein (Section 5.3.2).

(C.3) We introduce a prioritized experience replay [49] on top of standard DQN (either single-agent or multi-agent) as well as a mechanism that stores some additional information per experience to reduce the number of computations during parameter updates (Section 5.3.3).

(C.4) We validate the proposed multiagent DQN scheme with all the above speed up extensions (iDQN+) in a large scale scenario, using real traffic data, and confirm its superior performance compared to vanilla iDQN and static baselines (Section 5.4).

5.2 RL framework for inter-slice orchestration

5.2.1 Environment: A B5G system

The system model was introduced in the previous deliverable D3.2 (Section 4), where the reader can refer to for a more detailed description. Figure 5-1Figure *3-1* outlines its main components, the physical network and network slices, as well as the notation, in a toy example[1].

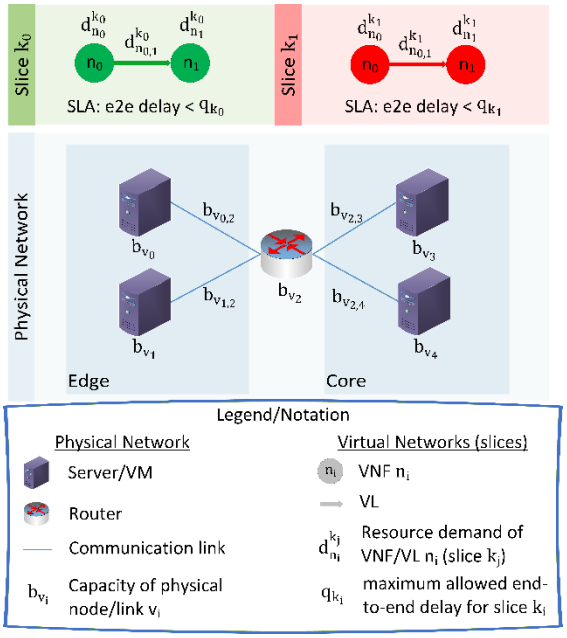
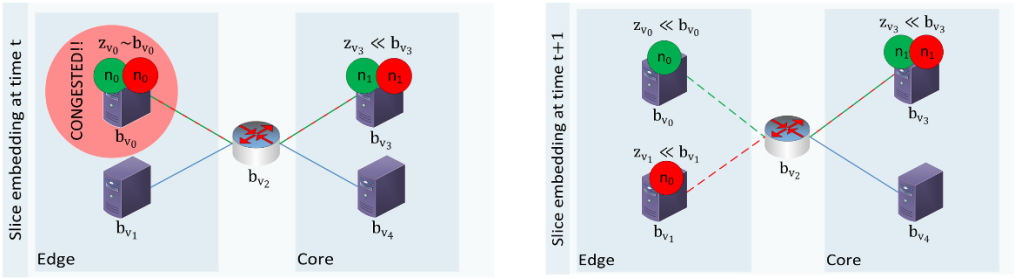


Figure 5-1. Toy example exhibiting the main components of our environment.

Slice orchestration. The assignment of each VNF/VL to a physical node/link[2] (at every timestep t) determines both the slice and the network performance. Slice performance deteriorates by SLA violations while network performance by unnecessary use of network resources. Hence, a VNF might "migrate" to another node at a next time unit with a view to improve system's performance. But migration of VNFs does not come for free; a reconfiguration cost should be taken into account (e.g. management overhead, delays leading to monetary penalties [46], [37]). A simple example is given in Figure 5-2. In Figure 5-2-(a), the initial embedding of slices (at time t) for the toy scenario of Figure 5-1 is depicted. The aggregate resource demand of the hosted VNFs at physical node v_0 ($z_{v_0} = d_{n_0}^{k_0} + d_{n_0}^{k_1}$) is close to b_{v_0}; the node is congested and causes high delays (and probably SLA violations) for both slices k_0 and k_1. At the next timestep $t+1$ (Figure 5-2-(b)), VNF n_0 of slice k_1 migrates to v_1 to avoid SLA violations, in exchange with i) a reconfiguration cost; ii) a node activation cost (server v_1 is turned "on" from the idle state).

[1] We stress that the above system model goes well beyond the depicted toy example, being able to capture VNF chains with probabilistic routing between VNFs, loops (i.e. traffic potentially going through the same VNF more than once), etc. See [81] for more details.

[2] A VL can be mapped to a path (its load is imposed to each traversed link).



a) System at timeslot t b) System at timeslot t+1

Figure 5-2. Slice embedding example.

Following, we define two of the most important slice-related quantities, the configuration and demand vectors, which we will be frequently using throughout this section:

Configuration $c \in \mathcal{C}$: mapping of all VNFs to physical nodes[3] at time t (e.g. in Figure 5-2 (b), the configuration is: $c = \left(c_{n_0}^{k_0}, c_{n_1}^{k_0}, c_{n_0}^{k_1}, c_{n_1}^{k_1}\right) = (v_0, v_3, v_0, v_3)$, where $c_{n_i}^{k_j}$ indicates the host node of VNF n_i, slice k_j).

Demand $d \in \mathcal{D}$: denotes the resource demands of all hosted slices at time t (e.g. in Figure 5-2 (a), the demand is: $d = \left(d_{n_0}^{k_0}, d_{n_1}^{k_0}, d_{n_{0,1}}^{k_0}, d_{n_0}^{k_1}, d_{n_1}^{k_1}, d_{n_{0,1}}^{k_1}\right)$).

Queuing Model. The impact on the performance of slices when multiple VNFs are assigned to the same physical node and their aggregate resource demand is close to or exceeds the node's capacity is captured using a queuing model. Each physical node/link is modelled as an M/G/1/PS queue [50] (such models tend to capture well the characteristics of proportionally fair schedulers, commonly used for resource scheduling [51]). Then, the average delay experienced by any VNF/VL hosted on node/link v_i is given by the function[4]:

$$f_{v_i}^{delay}(c, d) = \frac{1}{b_{v_i} - z_{v_i}(c, d)} \qquad (1)$$

$$\text{where } z_{v_i}(c, d) = \sum_{k_j \in \mathcal{K}} \sum_{n_l \in \mathcal{N}_{k_j} \cup \mathcal{L}_{k_j}} d_{n_l}^{k_j} \cdot x_{n_l, v_i}^{k_j} \qquad (2)$$

In (2), $\mathcal{K}$ is the set of slices, $\mathcal{N}_{k_j}$, $\mathcal{L}_{k_j}$ the sets of VNFs and VLs of slice k_j respectively, and $x_{n_l, v_i}^{k_j}$ a binary variable (1 if VNF/VL n_l of slice k_j is hosted by physical node/link v_i). For complex slices the end-to-end delay can be calculated by a Jackson network type of analysis. For a simple slice (VNF chain) the corresponding end-to-end delay $F_{k_i}^{delay}(c, d)$ is the sum of delays across all host nodes and links along its path (e.g. in Figure 5-2 the delay of slice k_0 is: $F_{k_0}^{delay}(c, d) = f_{v_0}^{delay}(c, d) + f_{v_{0,2}}^{delay}(c, d) + f_{v_{2,3}}^{delay}(c, d) + f_{v_3}^{delay}(c, d)$).

[3] To simplify our discussion, and w.l.o.g., we consider the mapping of VLs is predetermined. Routing variables could be easily included in our framework.

[4] When demand exceeds capacity, (\ref{eq:queuing_delay}) is extended to include a large penalty.

5.2.2 States, actions, and rewards

State space $\mathcal{S}$.

The state of the system at timestep t consists of (i) the assignment of all VNFs to physical nodes; (ii) the resource demands of all VNFs/VLs (both are necessary to calculate the instantaneous reward, to be elaborated shortly).

Definition 1 (State) $s = (c, d)$, $s \in \mathcal{S} = \mathcal{C} \times \mathcal{D}$

Remark: The above state space $\mathcal{S}$ grows exponentially fast with slice and VNF number. What is worse, continuous traffic demand (as will be the case for the dataset used in our simulations [52]) essentially renders vanilla RL methods (e.g., Q-learning) inapplicable, even for toy scenarios. To this end, approximate RL methods (e.g., using a Deep Neural Network (DNN) to encode the input) cannot be avoided for such problem. We are using the popular DQN architecture [53] to this end.

Action space $\mathcal{A}$.

The agent's action is the configuration to be applied in the next timestep (combinatorial). Note that we use an apostrophe to denote all quantities of $t + 1$.

Definition 2 (Action) $a = c'$, $a \in \mathcal{A} = \mathcal{C}$

Action complexity example: In a physical network with $V = 10$ nodes and $K = 10$ slices (one VNF per slice), the number of possible actions is $|\mathcal{A}| = V^K = 10^{10}!$ Standard DQN algorithms [53] are designed for small action spaces. To deal with this problem (on top of state space complexity), we use a multi-agent DQN architecture, that can reduce action state complexity by orders of magnitude.

Reward function $\mathcal{R}$.

We consider three individual costs that determine the total cost performance of the system. Given some observed state s, an agent action a, and the next state s', these costs are the following.

-Type 1 cost: SLA violation. A penalty is paid when the end-to-end delay $F_{k_i}^{delay}(s')$ perceived by a slice k_i is higher than q_{k_i} (defined by the SLA). This may take any suitable form (linear, quadratic, etc.). We give as an example the linear form:

$$g_1(s') = \sum_{k_i \in \mathcal{K}} \left(F_{k_i}^{delay}(s') - q_{k_i} \right) \cdot 1_{\{F_{k_i}^{delay}(s') > q_{k_i}\}} \qquad (3)$$

-Type 2 cost: Reconfiguration. Migrating VNFs from their host servers causes overhead and might incur delays, leading to SLA violations [46], [37].

$$g_2(s, a) = 1/2 \cdot \sum_{k_i \in \mathcal{K}} \sum_{n_j \in \mathcal{N}_{k_i}} \sum_{v_l \in \mathcal{V}} \left| \left(x_{n_j, v_l}^{k_i} \right)' - x_{n_j, v_l}^{k_i} \right|, \qquad (4)$$

where $\mathcal{V}$ is the set of physical nodes.

-Type 3 cost: Active nodes. It is the number of physical nodes that are "on" (hosting at least one VNF). The idle servers/VMs can be turned off (or set to sleep mode) and save energy/free up resources [54].

$$g_3(a) = \sum_{v_i \in \mathcal{V}} 1_{\{\sum_{k_j \in K} \sum_{n_l \in N_k} x_{n_l, v_i}^{k_j} \geq 1\}} \qquad (5)$$

The reward obtained is the negative weighted sum of the individual costs[5] (3) -(5):

$$r = \left(w_1 \cdot g_1(s') + w_2 \cdot g_2(s,a) + w_3 \cdot g_3(a)\right) \quad (6)$$

5.3 Approximate RL schemes

As is evident by the problem model, the inter-slice orchestration problem at hand is characterized by (i) unknown future resource demands; and (ii) delayed rewards (e.g. if the demand of a VNF is predicted to increase soon and stay high for a while, paying now a reconfiguration cost for its migration to a less busy server could lead to high future rewards). While this is the standard "playground" of RL, vanilla algorithms like Q-learning [55] are unable to handle the problem at hand, due to the prohibitive state and action spaces even in relatively small setups. We will first describe here the basic DQN and iDQN (multi-agent DQN) solutions we use as our starting point to deal with state and action complexity, respectively, then proceed with the proposed experience replay buffer heuristics.

5.3.1 DQN

To deal with the combinatorially large (potentially infinite) state space, an approach that has found significant success recently in many applications (e.g., games) is to learn a parameterized function $Q_\theta(s,a)$ (with the function commonly being a DNN), that approximates the original Q function (using much fewer learnable parameters than a complete state-action table). The advantage of a DNN is the automatic encoding of important features that would otherwise be problem dependent and tough to track (e.g. the discretization of continuous traffic demands). However, simply adding a DNN forfeits the convergence guarantees of tabular RL algorithms and leads to unstable learning in most practical scenarios. The recent Deep Q-Network (DQN) algorithm [53] is shown to often overcome these issues and will be our starting point.

A DQN agent is equipped with two DNNs, the so-called policy and target networks ($Q_\theta(s,\cdot)$ and $Q_{\theta'}(s,\cdot)$ respectively), which take as an input the state and output the Q values of all possible actions (configurations). Moreover, the visited transitions are stored in a replay buffer ($\mathcal{B}$). In Figure 5-3 we outline the main steps of the DQN algorithm.

Drawbacks. DQN does not scale well for very large action spaces. In our case, larger problem size means (a) (combinatorially) more outputs/actions for the DNN, (b) harder argmax operations, (c) slower exploration of the action space.

[5] Note that the goal of the RL agents is to maximize the accumulated rewards, and this is why we introduce a negative sign in Def.~\ref{def:reward} (in our problem we want to minimize the accumulated cost).

DQN algorithm

Step 1 (in agent): When at state s (Def. 1), take an ϵ-greedy action (Def. 2):

$$a \leftarrow \begin{cases} \text{random } a \in \mathcal{A}, & \text{with probability } \epsilon \\ \arg\max_{a \in \mathcal{A}} Q_\theta(s,a), & \text{with probability } 1-\epsilon \end{cases} \quad (7)$$

Step 2 (in env): Returns the next state s' and reward r (Def. 3)

Step 3 (in agent): store transition (s, a, s', r) in the replay buffer $\mathcal{B}$

Step 4 (in agent): copy the policy network parameters θ to the target network θ' (only every T timesteps)

Step 5 (in agent): pick M samples randomly from replay buffer and perform a gradient step on the minibatch:

$$\theta \leftarrow \theta - \eta \nabla_\theta (\mathbb{E}_{i \sim U(\mathcal{B})}[\delta_i^2]), \quad (8)$$

$$\text{where } \delta_i = Q_\theta(s_i, a_i) - (r_i + \gamma \max_{a_i' \in \mathcal{A}} Q_{\theta'}(s_i', a_i')) \quad (9)$$

In (8) η is the learning rate and δ_i the Temporal Difference (TD)-error of transition i.

Repeat steps 1 to 5 till termination criterion.

Figure 5-3. Main algorithmic steps of DQN.

5.3.2 iDQN

To deal with the above bottlenecks (a) and (b), we employ a multi-agent scheme to decompose the combinatorial action space into smaller subspaces. To this end, we consider one independent DQN agent (iDQN) per VNF, responsible only for placing the specific VNF on the physical network. The introduced modifications are outlined in Figure 5-4.

iDQN Modifications

Multi-agent scheme: one DQN agent j per VNF

Step 1: each agent j takes an ϵ-greedy action:

$$a^j \leftarrow \begin{cases} \text{random } a^j \in \mathcal{A}^j, & \text{with probability } \epsilon \\ \arg\max_{a^j \in \mathcal{A}^j} Q_\theta(s,a^j), & \text{with probability } 1-\epsilon \end{cases}$$

Then, the collective action is:

$$a = (a^0, a^1, ..., a^{j_{max}})$$

Step 3: each agent stores (s, a^j, s', r)

Steps 4-5: applied to all agents (with $\mathcal{A}^j$ instead of $\mathcal{A}$)

Figure 5-4. Modifications of iDQN algorithm with respect to DQN.

With iDQN we have managed to reduce memory requirements by avoiding the combinatorial output layer of the single-agent DQN scheme. Also, we have replaced the computationally expensive maximization operations of (7), (9), with much less expensive operations over $\mathcal{A}^j$. However, there is still much room for improvement regarding convergence speed and sample efficiency (drawbacks (b) and (c) of Section 5.3.1), which are crucial characteristics for a practical algorithm.

5.3.3 DQN+/iDQN+

The last step towards a more scalable solution is to improve convergence speed by (i) smarter picking of minibatches; (ii) DNN parameter updates with fewer computations (we introduce the "lazy" computation of the TD-target). Following, we demonstrate how DQN must be modified to incorporate these two speedup tricks (DQN+), which can be readily applied in the multiagent scheme (iDQN+) to further improve its convergence speed (especially in large scale scenarios).

Prioritized experience replay. We have observed that as the action space in our problem grows larger, actions with similar effect are over-represented in the replay buffer, while potentially more effective actions are under-represented (slowing down convergence). To this end, we employ a

prioritized experience replay [49], which prioritizes transitions with a larger TD-error to boost sample efficiency. Figure 5-5 outlines the modifications on top of the DQN algorithm.

DQN+ Modification 1:
Step 5 (in agent): M samples are picked from the buffer with a probability $P(i)$ for each of the N transitions:

$$P(i) = p_i^{\alpha^{\text{rep}}} / \sum_{j=1}^{N} p_j^{\alpha^{\text{rep}}}, \quad (10)$$

$$\text{where } p_i = |\delta_i| + 0.1 \quad (11)$$

and the gradient step is:

$$\theta \leftarrow \theta - \eta \nabla_\theta (\mathbb{E}_{i \sim P(\mathcal{B})}[(w_i^{rep} \delta_i)^2]), \quad (12)$$

$$\text{where } w_i^{rep} = (\frac{1}{N} \cdot \frac{1}{P(i)})^{\beta^{\text{rep}}} \quad (13)$$

Figure 5-5. Modification 1 of DQN+ algorithm with respect to DQN.

In (10), α^{rep} is a hyperparameter that determines the amount of prioritization ($\alpha^{\text{rep}} = 0$ leads to uniform sampling while $\alpha^{\text{rep}} = 1$ to full prioritization), while β^{rep} in (13) determines the amount of compensation applied by weighted importance sampling (to balance the bias introduced by prioritization).

Lazy computation of TD-target. The maximization operation in (9), required for every sample of the minibatch, becomes very expensive as the action space grows larger. To reduce the number of such computations we introduce a second modification (Figure 5-6).

DQN+ Modification 2:
Step 3 (in agent): calculate Q^{next}:

$$Q^{\text{next}} = \max_{a' \in \mathcal{A}} Q_{\theta'}(s', a') \quad (14)$$

and store $(s, a, s', r, Q^{\text{next}})$ in the replay buffer.
Step 5 (in agent): the TD-error for all M minibatch samples is now computed using the stored Q^{next} values:

$$\delta_i = Q_\theta(s_i, a_i) - (r_i + \gamma Q_i^{\text{next}}) \quad (15)$$

Figure 5-6. Modification 2 of DQN+ algorithm with respect to DQN.

This trick offers important real time gains, as DQN+ performs M times less computations per timestep, compared to DQN, for the calculation of the TD-target (in DQN+, (15) is calculated only for the visited transition while in DQN for each one of the M samples).

5.4 Simulation results

In this section, we have three main goals: (i) to establish that Q-learning based approximate algorithms, either single-agent or multi-agent, are able to obtain close to optimal solutions (i.e. ones found by tabular Q-learning or offline Policy Iteration), yet with much higher convergence speed as the problem size increases; (ii) to quantify the convergence speed gains offered by the speedup heuristics proposed in section 5.3.3 on top of of DQN and iDQN algorithms; and (iii) to validate our proposed enhanced multiagent solution (iDQN+) in a realistic large scale setup. For (i) we use small enough scenarios (to obtain the optimal policy) with synthetic data (for sensitivity analysis), while for the rest we use real data from the Milano dataset [52], in more realistic topologies with much larger state and action baseline complexity.

Policies. Below we list various algorithms that we will use in our validation, together with some key parameters for each. Note that not all algorithms will or can be used in every scenario (e.g., the first two take more than days to converge, except in the smaller Scenario 1).

• Policy Iteration (PI): Returns the optimal policy in an offline fashion and is applicable when the traffic dynamics are Markovian and known [55].

• Q-Learning (QL): This is a standard RL tabular method [55] that returns the optimal policy (for discrete traffic demands).
• DQN: This is the centralized approximation method (Section 5.3.1), using a DNN of 3 layers and 60 neurons per layer. We set its replay memory size to 5000 timeslots, the target update period to 500 timeslots and the minibatch to 32, since these parameters performed well in a variety of tested scenarios.
• iDQN: This is the proposed multi-agent approximation method (Section 5.3.2). Each agent (one agent per VNF) is a DNN of 3 layers and 60 neurons per layer. It uses the same parameter values as the DQN.
• DQN+/iDQN+: These variants are the above DQN/iDQN but with all the speedup heuristics we have proposed in Section 5.3.3.
• Group-all: A reference static policy, whose main goal is to merely minimize the active nodes cost (5); it places all VNFs in one server and does not react to changing demands (hence no reconfiguration cost either), but possibly suffering from frequent SLA violations.
• Split-all: A sister reference policy to Group-all which instead aims to minimize SLA violations (3) only, trying to "spread out" VNFs among all available servers as much as possible (no reconfiguration cost either).
• Random: It chooses randomly one of the possible configurations at each timestep.

Note that the discount factor was set to $\gamma = 0.9$ for all RL/MDP algorithms.

5.4.1 Scalability of tabular vs approximate RL schemes

In this first part, we focus on small scenarios, and synthetic Markov traffic (defined below), so that theoretically optimal algorithms (Policy Iteration and Q-learning) can be used as baselines (converge in reasonable time). Our aim is to carefully study how the increase of problem size (K: number of slices, V: number of servers) affects the convergence speed of the approximations, as well as their attained cost, compared to the optimal. For this reason, in terms of cost performance we use PI and QL as benchmarks (both find the optimal cost), and check how DQN and iDQN perform against them (i.e. how far from the optimal). And moreover, we observe the convergence speed of QL against DQN and iDQN.

Network Setup and Markov Traffic. We consider a single domain physical network and two scenarios (a small and a larger) with different problem sizes, i.e., V and K. Without loss of generality, we assume each slice consists of one VNF. The demand of each VNF $d_k(t) \in \mathcal{B} = \{0,1\}$ ("ON/OFF") and evolves, independently to the other VNFs, as a Markov process with transition probability matrix:

$$P_k = \begin{bmatrix} 0.98 & 0.02 \\ 0.02 & 0.98 \end{bmatrix} \qquad (16)$$

This captures a very simple scenario where each slice has bursty traffic periods followed by long silence periods, not necessarily coinciding, to better illustrate the optimality of the chosen actions, as well as the performance of static heuristics. Using (16) we generate a training and a testing dataset, with duration $2 \cdot 10^6$ and $8 \cdot 10^4$ timeslots respectively. The algorithms are trained and tested 10 times in the respective datasets with different initial random seeds.

Scenario 1. Here, we consider a PN with $V = 2$ servers and $K = 4$ slices to be configured (4096 state-action pairs).

-Convergence Speed (Figure 5-7): We depict the average cost value (over 10 runs, on the y-axis) as a function of time (counted in terms of iteration, on the x-axis), for QL, DQN, and iDQN. The two main points to observe are the following: (i) the approximate solutions, DQN and iDQN both achieve similar costs with the theoretically optimal QL (this has been confirmed for a variety of other small scenarios as well); (ii) yet, it is impressive that QL already takes quite a few more iterations to converge, even in such a small scenario (this is not so surprising since the size of the Q-value table is

already 256 by 16); (iii) since the action space in this scenario is quite small, there are no additional convergence advantages by IDQN (compared to DQN), as expected.

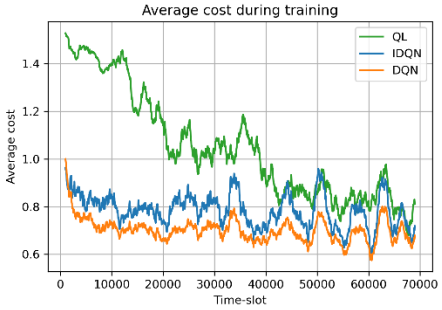


Figure 5-7. Convergence plot (Scenario 1: Markovian traffic – small problem size)

-Cost performance (Figure 5-8): We depict the cost per time-slot in the testing dataset via a box plot (for all the algorithms outlined in the beginning of the Section). The main observations are: (i) the approximations (DQN and IDQN) have no problem finding the optimal solution in such a scenario; (ii) even on this tiny scenario, we get a 20% improvement compared to the simple static heuristics, and a 60% compared to the random policy.

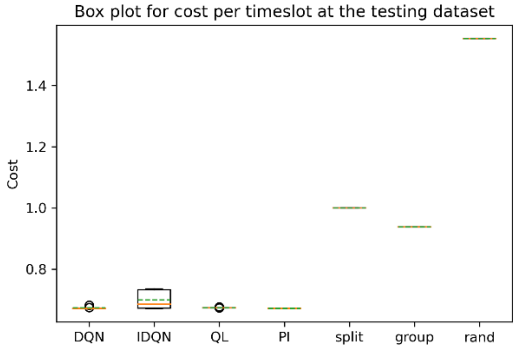


Figure 5-8. Box plot of cost in the testing dataset (Scenario 1: Markovian traffic – small problem size).

-Take-away message 1: DQN based approximate policies are able to find good quality solutions (certainly better than static reference policies), and quite faster than tabular Q-learning.

Scenario 2 (Figure 5-9**).** We increase the problem size (w.r.t. Scenario 1) and assume a physical network with $V = 3$ servers that hosts $K = 7$ slices (10^{10} state-action pairs). While this is still not a very large scenario, in practice, policy iteration and Q-learning already collapse, due to their memory and computation requirements; we therefore omit these from the respective plot. So, Figure 5-9 is similar to Figure 5-7, yet we only compare DQN and IDQN. An important observation here is that IDQN now presents considerable convergence time improvements (roughly 10x) compared to DQN; this is reasonable as the action space in this scenario is growing large (2187 actions per state). Nevertheless, the solution quality (average cost achieved) is comparable.

-Take-away message 2: for realistic size scenarios, even centralized DQN will collapse in terms of convergence time.

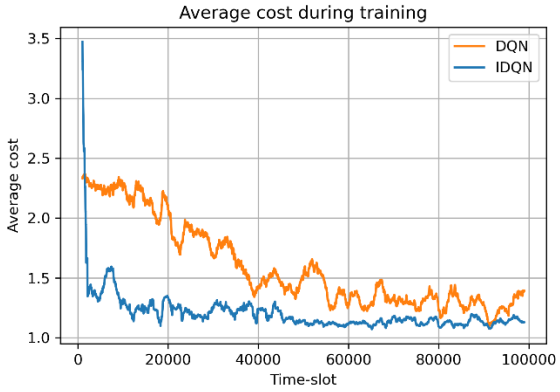


Figure 5-9. Convergence plot (Scenario 2: Markovian traffic – larger problem size).

5.4.2 Performance gains of DQN+/iDQN+

Having established that approximate schemes can increase convergence speed without significantly impacting the quality of the obtained policy, here we aim to validate that the speed up heuristics proposed in Section 5.3.3 can further boost convergence speed in a realistic setup. We will also introduce real traffic data to drive the demand, as well as a full-fledged multi-domain physical network, coupled with the end-to-end queuing delay SLAs to further induce realism into our scenarios.

VNF demands. We import them from the Milano dataset [52]. Due to their continuous values the state space is infinite for all RL algorithms of this section (thus only approximate algorithms can be directly employed). Milano timeseries consist of 8928 samples per base station (1 sample every 10 minutes), so we map to each VNF the normalized "internet" traffic demand of a different base station. W.l.o.g., we assume that VL demands are zero. We use the first 4464 Milano samples for training and the rest for testing, hence the duration of each training/testing episode is 4464 timesteps.

System Setup. The physical network consists of 2 domains, each of them comprising 2 nodes (servers) respectively. On top of it there are 4 slices (simple VNF chains) with 2 VNFs each (one VNF per domain). This results in 256 possible actions for single-agent DQN.

Training. Each algorithm is trained over 12 episodes, while this procedure is repeated over 20 individual runs with different random seeds (to get averaged results).

Impact of proposed speed up mechanisms on DQN/iDQN. In Figure 5-10, we compare the performance of vanilla DQN/iDQN algorithms with respect to their DQN+/iDQN+ counterparts. There are 4 main observations to take away, (i) iDQN is again confirmed to converge faster than DQN in this real traffic scenario (due to the additional approximation in action space); (ii) the speedup heuristics of DQN+/iDQN+ improve their convergence speed compared to their vanilla counterparts; (iii) the speed improvement for DQN is much larger than the speed improvement for iDQN, due to its larger action space (the gain for iDQN is expected to become more prominent in larger scenarios); (iv) iDQN+ is the fastest among the tested algorithms (but only slightly faster than DQN+).

-Take-away message 3: The proposed speedup heuristics on top of DQN/iDQN can offer significant convergence speed gains.

Figure 5-10. Convergence plot in real traffic small scale scenario.

5.4.3 Validation in large scale scenario

Having validated the gain offered by the proposed speed ups in a relatively small scenario, in this last part we examine the performance of iDQN+ in a larger setup. Since DQN cannot be used as a baseline here (due to the prohibitive action space size), we go back to the simple static heuristic policies (split-all, group-all), or even random actions, as minimum benchmarks to assess the obtained dynamic policies.

System Setup. The physical network consists of two technological domains, comprising 9 and 3 nodes (servers) respectively. On top of it there are 10 slices (simple VNF chains) with 2 VNFs each (one VNF per domain). This results to $2 \cdot 10^{14}$ possible actions for single-agent DQN.

Training and Testing. Each algorithm is trained over 22 training episodes, while this procedure is reperated for 10 individual runs with different random seeds. All the obtained policies are evaluated over 1 testing episode (we rollout the policy with no exploration and record the mean cost).

Cost performance. The results of the testing phase are given in the box plot of Figure 5-11. This plot compares the performance of iDQN and iDQN+, also including the static baseline policies split-all, group-all, and random. The main observations are: (i) iDQN+ performs 2x better than the split-all policy, which was the best of the static baselines; (ii) even the worst policies obtained by iDQN (with or without the speed up extensions) in all 10 runs perform much better than the static baselines; (iii) iDQN+ demonstrates 20% cost reduction compared to iDQN (converges faster as it achieves lower cost in the same amount of training steps); (iv) the performance gain of iDQN+ is more prominent in this larger scenario, compared to Section 5.4.2.

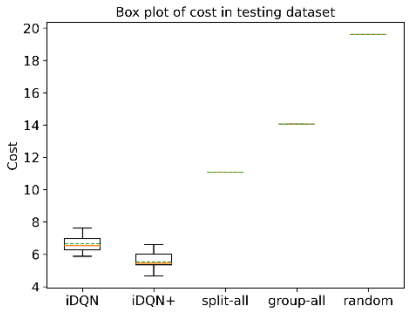


Figure 5-11. Box plot of cost in the testing dataset (large scale real traffic scenario).

-Take-away message 4: iDQN+ was validated to improve the cost performance of iDQN by 20% in a large scale scenario.

5.5 Conclusion

In this work we introduced a flexible RL framework for slice embedding (reconfiguration) supporting an inter-slice setting, multiple domains, and diverse end-to-end SLAs. We proposed a novel DRL scheme based on independent DQN agents that radically reduces both the state and action complexity, while we additionally introduced two heuristic mechanisms to speed up its convergence and thus improve scalability. Finally, we validated the performance of the proposed solution through extended simulations using a real traffic dataset.

6 Summary

In summary, this deliverable focuses on the evolution of wireless networks beyond 5G, emphasizing the importance of zero-touch AI-driven automation in telecommunications. It introduces an Explainable Federated Deep Learning (FDL) model for predicting dropped traffic in 6G networks, particularly in the Radio Access Network (RAN). The model incorporates closed-loop automation and Explainable AI (XAI) principles, addressing the complexity of network slicing. The study underscores the significance of the XAI approach for transparent and trustworthy decision-making in critical telecom services. The paper also presents a 6G RAN-edge network architecture and preliminary results, setting the foundation for implementing the proposed idea. The ultimate goal is to enhance trust and reliability in the 6G network deployment, making the approach valuable for telecom operators and service providers aiming to expand their services in the evolving landscape.

Further this deliverable presents comprehensive study on radio resource management for RAN edge domain at inter- and intra-slice level. It concludes several state-of-the-art work and proposes algorithms with novel approach to take resource allocation decision by parameterizing the crucial components related to scheduling and association. Several DRL techniques has been studied and a novel approach using DQN is presented for near-real time resource allocation which is implemented as xAPPs at near RT RIC of ORAN based architecture for beyond 5G and 6G networks. The algorithm presents quality of service (QoS)-aware intra-slice resource allocation that provides superior performance compared to baseline and state of the art strategies. The slice-dedicated intelligent agents learn how to handle resources at near-RT RIC level time granularities while optimizing various key performance indicators (KPIs). The study provides novel and intelligent resource allocation approach which are essential to support futuristic applications under 6G network.

Finally, in Chapter 5, this deliverable focuses on data-driven dynamic embedding of end-to-end slices in beyond 5G networks. It introduces a generic, queuing network-based model that captures the inter-slice orchestration setting, supporting complex VNF chain topologies and end-to-end performance metrics. To deal with the astronomically high action spaces emerging in such settings, it explores multi-agent DQN algorithms that can reduce action space complexity by orders of magnitude compared to standard DQN, providing a scalable solution. Moreover, as the training of DQN agents is not particularly data-efficient, which can hinder their practical application given the scarce(r) availability of cellular network related data, this deliverable investigates two mechanisms to store to and select from the experience replay buffer, aiming to speed up training The scalability and the convergence speed gains of the proposed scheme are validated through extensive simulations using real traffic data.

7 References

[1] X. You et al., «Towards 6G wireless communication networks: vision, enabling technologies, and new paradigm shifts,» Sci. China Inf. Sci. 64, 110301 (2021)..

[2] ETSI GS ZSM, «Zero-touch network and Service Management (ZSM); Reference Architecture,» ETSI GS ZSM 002, 2019.

[3] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, «How should I slice my network? A multi-service empirical evaluation of resource sharing efficiency,» de *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018.

[4] X. Foukas, G. Patounas, A. Elmokashfi and M. K. Marina, «Network Slicing in 5G: Survey and Challenges,» *IEEE Communications Magazine,* vol. 55, nº 5, pp. 94-100, 2017.

[5] V. Sciancalepore, F. Z. Yousaf and X. Costa-Perez, «z-TORCH: An Automated NFV Orchestration and Monitoring Solution,» *IEEE Transactions on Network and Service Management,* vol. 15, pp. 1292-1306, 2018.

[6] R. Wen, G. Feng, J. Tang, «On Robustness of Network Slicing for Next-Generation Mobile Networks,» *IEEE transaction on communication, vol. 67, no.1,,* pp. 430-444, Jan 2019.

[7] Wu, Wen and Zhou, Conghao and Li, Mushu and Wu, Huaqing and Zhou, Haibo and Zhang, Ning and Shen, Xuemin Sherman and Zhuang, Weihua, "AI-Native Network Slicing for 6G Networks," *IEEE Wireless Communications,* vol. 29, no. 1, pp. 96-106, 2022.

[8] B. Brik, and A. Ksentini, «On Predicting Service-oriented Network Slices Performances in 5G: A Federated Learning Approach,» de *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, 2020.

[9] S. Roy, H. Chergui, L. Sanabria-Russo and C. Verikoukis, «A Cloud Native SLA-Driven Stochastic Federated Learning Policy for 6G Zero-Touch Network Slicing,» de *ICC 2022 - IEEE International Conference on Communications*, Seoul, Korea, Republic of, 2022, 2022.

[10] F. Fossati, S. Moretti and S. Secci, «Multi-Resource Allocation for Network Slicing under Service Level Agreements,» de *10th International Conference on Networks of the Future (NoF),*, Rome, Italy, 2019, 2019.

[11] Li, Y., Huang, A., Xiao, Y., Ge, X., Sun, S., & Chao, H., «Federated Orchestration for Network Slicing of Bandwidth and Computational Resource.,» *ArXiv, abs/2002.02451,* 2020.

[12] C. Benzaid and T. Taleb, «AI-Driven Zero Touch Network and Service Management in 5G and Beyond: Challenges and Research Directions,» *IEEE Network,* vol. 34, pp. 186-194, 2020.

[13] S. Wang, M. Qureshi, L. Miralles-Pechuan, T. Huynh-The, T. R. Gadekallu, and M. Liyanage, «Explainable AI for B5G/6G: Technical Aspects, Use Cases, and Research Challenges,» *ArXiv,* 2021.

[14] C. Li, W. Guo, S. C. Sun, S. Al-Rubaye and A. Tsourdos, «Trustworthy Deep Learning in 6G-Enabled Mass Autonomy: From Concept to Quality-of-Trust Key Performance Indicators,» *IEEE Vehicular Technology Magazine,* vol. 15, pp. 112-121, 2020.

[15] Y. Wu, G. Lin and J. Ge, «Knowledge-Powered Explainable Artificial Intelligence for Network Automation toward 6G,» *{IEEE Network,* vol. 36, pp. 16-23, 2022.

[16] A. Terra, R. Inam, S. Baskaran, P. Batista, I. Burdick and E. Fersman, «Explainability Methods for Identifying Root-Cause of SLA Violation Prediction in 5G Network,» de *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020.

[17] W. Guo, «Explainable artificial intelligence for 6G: Improving trust between human and machine,» de *IEEE Communications Magazine, 58(6), 39-45.*, 2020.

[18] G. B. J. &. W. O. A. Rjoub, «Explainable AI-based federated deep reinforcement learning for Trusted Autonomous Driving.,» de *IEEE International Wireless Communications and Mobile Computing (IWCMC) (pp. 318-323).* , 2022.

[19] A. Cotter, H. Jiang, and K. Sridharan, «Two-Player Games for Efficient Non-Convex Constrained Optimization,» *coRR,* vol. abs/1804.06500, 2018.

[20] Sundararajan, M., Taly, A., & Yan, Q., «Axiomatic attribution for deep networks.,» de *In International conference on machine learning (pp. 3319-3328). PMLR.*, 2017.

[21] Shrikumar, A., Greenside, P., & Kundaje, A., «Learning important features through propagating activation differences.,» de *In International conference on machine learning (pp. 3145-3153). PMLR.*, 2017, July.

[22] Cotter, A., Gupta, M., Jiang, H., Srebro, N., Sridharan, K., Wang, S., Woodworth, B. and You, S., "Training well-generalizing classifiers for fairness metrics and other data-dependent constraints.," in *International Conference on Machine Learning*, 2019.

[23] G. J. Gordon, A. Greenwald, and C. Marks, «No-regret learning in convex games,» de *In Proceedings of the 25th international conference on Machine learning*, 2008,July.

[24] «https://github.com/marcoancona/DeepExplain».

[25] S. Zhang, An overview of network slicing.

[26] R. Wen et al., «On Robustness of Network Slicing for Next-Generation Mobile Networks,» *IEEE Transactions on Communications,* vol. 67, pp. 430-444, 2019.

[27] G. Wang et al, «Reconfiguration in Network Slicing - optimizing the profit and performance,» *IEEE transaction on network and service management,* pp. 591-605, 2019.

[28] Z. Xu, Y. wang, J. Tang, J. Wang and M. C. Gursoy, «A deep reinforcement learning based framework for power efficient resource allocation in clud RANs,» de *IEEE ICC*, 2017.

[29] Y. Yang, Y. Li, K. Li, S. Zhao, R. Chen, J. Wang, and S. Ci, «Decco: Deep learning enabled coverage and capacity optimization for massive MIMO sysrem,» *IEEE access.*

[30] J. Song, Y. Nam, H. Kwon, I. Sim, S. Maeng, and S. Jang, «Adaptive Generalized Proportional Fair Scheduling with Deep Reinforcement Learning,» Samsung Research, Samsung Electronics, Seoul, Republic of Korea.

[31] F. Wei, G. Feng, Y. Sun, Y. Wang,Y. C. Liang and D.I. Kim, «Dynamic Network Slice Reconfiguration by exploiting deep reinfocement learning,» de *IEEE conference on communications*, Ireland, 2020.

[32] N.C. Lugong et. al, «Application of deep reinforcement learning in communication and networking: a survey,» *IEEE coimmunication survey and tutorial,* pp. 2664-2732, 2020.

[33] ORAN WG1, «Slicing-Architecture- v07.00: Technical Specification",» 2022.

[34] ORAN TR, «ORAN WG1 Use Case Analysis Report v08.00,» 2022.

[35] S Zhang, «An Overview of Network Slicing for 5G,» *IEEE wireless communication,* pp. 111-117, June 2019.

[36] Z. Li, M. Chen, K. Wang, C. Pan, N. Huang and Y. Hu, «Parallel Deep Reinforcement Learning Based Online User Association Optimization in Heterogeneous Networks,» de *International Conference on Communications Workshops (ICC Workshops)*, Dublin, Ireland, 2020.

[37] F. Wei, G. Feng, Y. Sun, Y. Wang, S. Qin and Y. -C. Liang, «Network Slice Reconfiguration by Exploiting Deep Reinforcement Learning With Large Action Space,» *IEEE Transactions on Network and Service Management,* vol. 17, nº 4, pp. 2197-2211, 2020.

[38] P. Tavakoli, A. Pardo, and K. Kormushev, «Action Branching Architec- tures for Deep Reinforcement Learning,,» de *AAAI Conference on Artificial Intelligence*, 2018.

[39] ORAN Alliance, «ORAN Architecture Specification Version 1.0.0,» 2019.

[40] M. K. Simon and M. S. Alouini, Digital Communication over Fading Channels: A Unified Approach to Performance Analysis, John Wiley & Sons, 2005.

[41] ITU-R M.2412-0 , «Guidelines for evaluation of radio interface technologies for IMT-2020,» 2017.

[42] I.-T. Y. Recommendation, «Framework for the support of network slicing in the IMT-2020 network,» 2020.

[43] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini and H. Flinck, «Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions,» *IEEE Communications Surveys & Tutorials,* vol. 20, nº 3, pp. 2429-2453, 2018.

[44] S. Vassilaras et al., «The Algorithmic Aspects of Network Slicing,» *IEEE Communications Magazine,* pp. 112-119, 2017.

[45] D. Bega, M. Gramaglia, M. Fiore, A. Banchs and X. Costa-Perez, «DeepCog: Cognitive Network Management in Sliced 5G Networks with Deep Learning,» de *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019.

[46] D. Bega, M. Gramaglia, M. Fiore, A. Banchs and X. Costa-Perez, «AZTEC: Anticipatory Capacity Allocation for Zero-Touch Network Slicing,» 2020.

[47] F. Schardong, I. Nunes, A. Schaeffer-Filho, «Nfv resource allocation: a systematic review and taxonomy of vnf forwarding graph embedding,» *Computer Networks,* 2021.

[48] P. T. A. Quang, Y. Hadjadj-Aoul and A. Outtagarts, «A Deep Reinforcement Learning Approach for VNF Forwarding Graph Embedding,» *IEEE Transactions on Network and Service Management,* vol. 16, nº 4, pp. 1318-1331, 2019.

[49] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, «Prioritized experience replay,» de *ICLR*, 2016.

[50] M. Harchol-Balter, Performance Modeling and Design of Computer Systems: Queueing Theory in Action, USA: Cambridge University Press, 2013.

[51] T. Bonald, and A. Prouti`ere, «Wireless downlink data channels: User performance and cell dimensioning,» de *MobiCom*, 2003.

[52] Telecom Italia, «Milano Grid,» 2015.

[53] V. Mnih, K. Kavukcuoglu, D. Silver, et al., «Human-level control through deep reinforcement learning,» *Nature,* vol. 518, pp. 529-533, 2015.

[54] M. Shojafar, N. Cordeschi and E. Baccarelli, «Energy-efficient adaptive resource management for real-time vehicular cloud services,» *IEEE Transactions on Cloud Computing,* 2019.

[55] D. Bertsekas, Reinforcement Learning and Optimal Control, Athena Scientific, 2019.

[56] B. Han et al., «Admission and Congestion Control for 5G Network Slicing,» de *IEEE Conference on Standards for Communications and Networking (CSCN)*, Paris, 2018.

[57] S. Moazzeni et al, «A Novel Autonomous Profiling Method for the Next-Generation NFV Orchestrators,» de *IEEE Transactions on Network and Service Management*, 2021.

[58] 3GPP, «Study on Management and Orchestration of Network Slicing,» 3GPP, TR 28.801, 2018.

[59] L. Le, D. Sinh, B. P. Lin, and L. Tung, «Applying Big Data, Machine Learning, and SDN/NFV to 5G Traffic Clustering, Forecasting, and Management,» de *4th IEEE Conference on Network Softwarization and Workshops, NetSoft 2018*, Montreal, QC, Canada, 2018.

[60] I. Yahia, J. Bendriss, A. Samba, and P. Dooze, «CogNitive 5G networks: Comprehensive operator use cases with machine learning for management operations,» de *20th Conference on Innovations in Clouds, Internet and Networks, ICIN 2017*, Paris, France,, 2017.

[61] W. Saad, M. Bennis, and M. Chen, «A vision of 6G wireless systems: Applications, trends, technologies, and open research,» *IEEE Network,* vol. 34, pp. 134-142, 2020.

[62] J. Mei, X. Wang and K. Zheng, «An intelligent self-sustained RAN slicing framework for diverse service provisioning in 5G-beyond and 6G networks,» *Intelligent and Converged Networks,* vol. 1, nº 3, pp. 281-294, Dec 2020.

[63] L. S. Russo, L. Righi, D. Pubill, «LTE as a Service: leveraging NFV for realising dynamic 5G network slicing,» de *IEEE Global Communications Conference (GLOBECOM),DOI: 10.1109/GLOBECOM38437.2019.9014330,*, 2019.

[64] H. Bakker, M. Doll et al., «"RAN architecture components – final report",» 5G NORMA deliverable D4.2, June 2017.

[65] H. Chergui and C. Verikoukis, «"Offline SLA-Constrained Deep Learning for 5G Networks Reliable and Dynamic End-to-End Slicing",» *IEEE Journal on Selected Areas in Communications,* pp. DOI: 10.1109/JSAC.2019.2959186,, 2019..

[66] H. Chergui and C. Verikoukis, «"Big Data for 5G Intelligent Network Slicing Management",» de *IEEE Network, DOI:10.1109/MNET.011.1900437, July 2020.*.

[67] M. Maule, P. -V. Mekikis, K. Ramantas, J. Vardakas and C. Verikoukis, "Real-time Dynamic Network Slicing for the 5G Radio Access Network", IEEE Global Communications Conference (GLOBECOM), DOI:10.1109/GLOBECOM38437.2019.9013965, February 2020..

[68] M. Z. Chowdhury, M. Shahjalal, S. Ahmed and Y. M. Jang, «6G Wireless Communication Systems: Applications, Requirements, Technologies, Challenges, and Research Directions,» *IEEE Open Journal of the Communications Society,* vol. 1, pp. 957-975, 2020.

[69] G. Camps-Valls, M. Reichstein, X. Zhu and D. Tuia, «ADVANCING DEEP LEARNING FOR EARTH SCIENCES: FROM HYBRID MODELING TO INTERPRETABILITY,» de *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, 2020.

[70] X. Shen et al., «AI-Assisted Network-Slicing Based Next-Generation Wireless Networks,» *IEEE Open Journal of Vehicular Technology,* vol. 1, pp. 45-66, 2020.

[71] B. Monchai, A. P. da Silva, X. Vasilakos, R. Nejabati, and D. Simeonidou, «Auto-3P: An autonomous VNF performance prediction & placement framework based on machine learning,» *Computer Networks 181,* vol. 107433, 2020.

[72] T. Subramanya and R. Riggio, «Centralized and federated learning for predictive VNF autoscaling in multi-domain 5G networks and beyond.,» *IEEE Transactions on Network and Service Management,* vol. 18, pp. 63-78, 2021.

[73] M. Bunyakitanon, X. Vasilakos, R. Nejabati and D. Simeonidou, «End-to-End Performance-Based Autonomous VNF Placement With Adopted Reinforcement Learning,» *IEEE Transactions on Cognitive Communications and Networking,* pp. 534-547.

[74] D. Naboulsi, M. Fiore, S. Ribot and R. Stanica, «Large-Scale Mobile Traffic Analysis: A Survey,» *IEEE Communications Surveys & Tutorials,* vol. 18, pp. 124-161, 2016.

[75] R. R. Hoffman, S. T. Mueller, G. Klein and J. Litman, «Metrics for Explainable AI: Challenges and Prospects,» *ArXiv,* vol. abs/1812.04608, 2018.

[76] 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 Communications Magazine,* June 2020.

[77] C.J.C.H Watkins, and P. Dayan, «Q-learning,» *Machoine Learning,* vol. 8, p. 279–292, 1992.

[78] R. Su et al., «Resource allocation for network slicing in 5g telecommunication networks: A survey of principles and models,» *IEEE Network,* vol. 33, nº 6, pp. 172-179, 2019.

[79] N. Zhang, S. Zhang, P. Yang, O. Alhussein, W. Zhuang and X. S. Shen, «Software defined space-air-ground integrated vehicular networks: Challenges and solutions,» *IEEE Communications Magazine,* vol. 55, 2017.

[80] N. Rajatheva et al., «White paper on broadband connectivity in 6G,» *arXiv preprint arXiv:2004.14247,* 2020.

[81] P. Doanis, T. Giannakas and T. Spyropoulos, «Scalable end-to-end slice embedding and reconfiguration based on independent DQN agents,» de *IEEE GLOBECOM*, Rio de Janeiro, 2022.

[82] H. Xiang, M. Peng, Y. Sun, S. Yan, «"Mode Selection and Resource Allocation in Sliced Fog Radio Access Networks: A Reinforcement Learning Approach",» *IEEE Transactions on Vehicular Technology,* pp. vol. 69, no. 4, pp. ,4271, April 2020.

[83] C. Zhang, M. Dong, K. Ota, « "Vehicular Multi-slice Optimization in 5G: Dynamic Preference Policy using Reinforcement Learning",» de *Global Communications Conference,*, 2020.

[84] M. Yan, G. Feng, J. Zhou, Y. Sun, and Y.-C. Liang , «Intelligent resource scheduling for 5G radio access network slicing,» *IEEE Transactions on Vehicular Technology,* pp. vol. 68, no. 8, pp. 7691–7703, 2019.

[85] B. Khodapanah, A. Awada, I. Viering, A. Noll Barreto, M. Simsek, and G. Fettweis , «Framework for slice-aware radio resource management utilizing artificial neural networks,» *IEEE Access,* pp. vol. 8, pp. 174972- 174987, 2020.

[86] R. Schmidt, C.-Y. Chang, and N. Nikaein, «Slice scheduling with QoS-guarantee towards 5G,» de *IEEE Global Communications Conference (GLOBECOM)*, 2019.