Strategies for UPF Placement in 5G and Beyond Networks

By

IRIAN LEYVA-PUPO

Ph.D. Advisor

CRISTINA CERVELLÓ-PASTOR

Department of Network Engineering
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Thesis submitted to Universitat Politècnica de Catalunya in accordance with the requirements of the degree of DOCTOR OF PHILOSOPHY IN NETWORK ENGINEERING.

BARCELONA, NOVEMBER 2022

Abstract

With societal and technological developments, new services and use cases have emerged, such as autonomous vehicles and smart manufacturing. These services have stringent and diverse requirements for bandwidth, user density, latency, and reliability, which traditional evolved packet core (EPC) networks are unable to support. Therefore, a new network architecture capable of meeting these requirements while ensuring system efficiency, cost savings, and revenue growth was designed: the fifth generation (5G) of mobile networks.

The 5G system is capable of overcoming current network limitations and providing a wider range of business opportunities to almost all spheres of society. These networks are driven by software and programmability principles and heavily rely on technologies such as software-defined networking (SDN), network function virtualization (NFV), and multi-access edge computing (MEC). SDN and NFV provide flexibility, scalability, programmability, and agility, while NFV enhances QoS and resource usage, it allows VNFs to be deployed and provisioned on demand in the most suitable locations. MEC brings computing, storage, and networking capabilities to the network edge, reducing network response time and backhaul traffic as network functions and applications are deployed close to users.

However, the placement of network functions, especially 5G user plane functions (UPFs), at the network edge is challenging due to numerous factors and attendant trade-offs; 5G requirements, edge nodes constrained resources, dynamic network conditions, and rising expenditures all add complexity to the problem.

This doctoral thesis focuses on the design of strategies (i.e., exact and heuristic-based methods) to optimize the placement and reconfiguration of UPFs in 5G and beyond networks. These solutions seek to ensure QoS satisfaction while reducing the expenditures associated with deploying and operating 5G services. To this end, we study the UPF placement problem (UPP) using three lines of research: static placement, dynamic placement, and reconfiguration scheduling. For each, we consider packet data unit (PDU) service requests composed of single or multiple UPF instances.

For static UPF placement, we envision several solutions that aim to minimize expenditures related to the deployment and operation of UPFs while fulfilling service requirements. First, we address the case in which all UPF functionalities are centralized in a single instance. Then, we extend the problem to include more complex service topologies (i.e., single- and multiple-branch service function chains [SFCs]), which we refer to as the UPF placement and chaining (UPC) problem.

For the centralized UPF functionalities, we conceive two integer linear programming (ILP) models and a heuristic algorithm. These solutions contemplate several aspects of the system, such as node available capacities and service requirements for latency, reliability, and mobility. Then, we propose an ILP and two approximated solutions (i.e., a heuristic and a simulated annealing (SA)-based metaheuristic) to address the UPC problem. These solutions consider

additional aspects of the UPP, such as UPF-specific requirements, virtual network function (VNF) order in the service chains, and routing paths. The heuristic-based strategies combine various mechanisms to enhance their performance. Specifically, the heuristic algorithm reduces SFC rejections and provisioning costs by considering service demands, available resources, and the effects of VNF mapping decisions on the VNFs forming the service chain. The envisioned metaheuristic approach incorporates several mechanisms, such as restart-stop and variable Markov chain length, that reduce its solution time and improve the solution's quality.

Different approaches are adopted to address the dynamic UPF placement and session-mapping reconfiguration problem. We conceive two exact solutions and a heuristic algorithm to determine the best reconfiguration setup. Their primary goal is to minimize expenditures associated with the service operation and reconfiguration procedure and guarantee satisfaction with the service requirements. Therefore, these approaches consider multiple cost components (e.g., server activation, VNF deployment, and migration) and system specificities (e.g., node available capacity). The proposed heuristic aims to enhance the problem's solution efficiency in online scenarios by incorporating two strategies: partial unmapping of SFCRs and an improvement phase.

Furthermore, three scheduling mechanisms are provided to determine the best time to readjust the UPF placement and session-mapping configuration to cope with latency violations produced by user mobility. More specifically, we design two optimal stopping theory-based schedulers and a machine learning-based framework to anticipate poor QoS events and proactively trigger reconfiguration procedures. These scheduling solutions make reconfiguration decisions based on historical system data (e.g., system QoS), current QoS values, and a pre-established tolerance threshold.

Extensive simulation results validate the applicability and efficiency of the proposed solutions. Overall, the heuristic-based approaches provide near-optimal results with significantly lower execution times than the mathematical models. Moreover, the conceived scheduling methods display outstanding performance, reducing the number of reconfiguration events and maintaining the QoS of the system under desirable levels for nearly the entire simulation.

Keywords: 5G, ILP, machine learning, MEC, NFV, optimal stopping theory, placement, reconfiguration, scheduling, SFC, user plane function, VNF

ACKNOWLEDGEMENTS

Many are the incredible people who have made this endeavor possible. The following is an acknowledgment of them.

I would like to express my deepest appreciation to my Ph.D. supervisor, Cristina Cervelló Pastor, for being there for me unconditionally, carefully monitoring my progress, and providing invaluable directions. Her knowledge, unwavering support, dedication, and wealth of experience have inspired me throughout my research. Without her, this thesis would not have been possible.

I am also thankful to all of the professors and colleagues from the Network Engineering department (ENTEL) at UPC. In particular, to professor Sebastià Sallent Ribes, for his feedback, advice, and expertise. Special thanks to all my officemates for their friendship, ideas, support, and collaborations over these years.

I would like to extend my gratitude to Dr. Christos Anagnostopoulos and Dr. Dimitrios P. Pezaros for hosting me as a visiting researcher at the Network Systems Research Laboratory (NETLAB), University of Glasgow (UoG). For the inspiring discussions, guidance, and encouragement. Thanks should also go to my fellow researchers at UoG for their companionship and for making me feel at home.

I am grateful to Dr. Juan Felipe Botero and Dr. Chrysa Papagianni for the time and effort dedicated to assessing this study. Their suggestions and comments helped to improve the quality of this thesis.

I am forever indebted to my family and friends for their love and support. My deepest gratitude is to my mom and dad for their endless love and for encouraging me to be a better person and to pursue my dreams. They are my north and reason in life; without them, I would not be the person I am. I am grateful to my little sisters, my best friends, for always being there for me. I would also like to thank my aunt Ileana, and my extended family, especially Odalys and Edilia, for the words of trust, love, and wisdom.

I am especially thankful to my partner in life, Ale, for his unconditional love, friendship, patience, and support. I could not have undertaken this journey without you. No words can express my gratitude, admiration, and love for you.

TABLE OF CONTENTS

List of Tables

List of Figures

Figure **Page**

List of Abbreviations

(R)AN (Radio) Access Network

3GPP Third-Generation Partnership Project

5GS 5G System

AI Artificial Intelligence

ANN Access Network Node

AP Aggregation Point

BBU Baseband Unit

BLSTM Bidirectional Long Short-Term Memory

CAPEX Capital Expenditures

CMUP Cost- and Mobility-aware UPF Placement

CMUP-BS Cost and Mobility-aware UPF Placement with Backup Sharing

CUPR Cost-aware UPF Placement Reconfiguration

CUPS Control and User Plane Separation

DC Data Center

DN Data Network

DPC-UPCR Dynamic Priority and Cautions UPCR

DPS Dynamic Placement Scheduling

E2E End-to-End

eMBB Enhanced Mobile Broadband

EN Edge Node

EPC Evolved Packet Core

ILP Integer Linear Programming

IoT Internet of Thing

ISR Intelligent Scheduling of the Reconfiguration

LSTM Long Short-Term Memory

LTE Long-Term Evolution

MANO Management and Orchestration

MEC Multi-access Edge Computing

MILP Mixed Integer Linear Programming

ML Machine Learning

MLP Multilayer Perceptron

mMTC Massive Machine-Type Communication

NFV Network Function Virtualization

NFVI NFV Infrastructure

NOUP Near-Optimal UPF Placement

NS Network Slicing

OPEX Operational Expenditure

OSP Optimal Stopping Problem

OSR Optimal Scheduling of the Reconfiguration

OST Optimal Stopping Theory

PC-UPC Priority and Cautious UPF Placement and Chaining

PDU Packet Data Unit

PGW Packet Data Network Gateway

PoP Point of Presence

PPS Periodic Placement Scheduling

QoE Quality of Experience

QoS Quality of Service

RAT Radio Access Technologies

RCCPP Resilient Capacitated Controller Placement Problem

RCP Resilient Controller Placement

RCPP Resilient Controller Placement Problem

RRH Radio Resource Head

RTT Round-Trip Time

SA Simulated Annealing

SA-UPC Simulated Annealing-based UPF Placement and Chaining

SBA Service-Based Architecture

SDN Software-Defined Networking

SFC Service Function Chain

SFCR Service Function Chain Request

SGW Serving Gateway

SLA Service Level Agreement

SMF Session Management Function

SSC Session and Service Continuity

SSR Skeptical Scheduling of the Reconfiguration

SVC Support Vector Classification

SVM Support Vector Machine

SVR Support Vector Regression

UE User Equipment

UPC UPF Placement and Chaining

UPCR UPF Placement and Chaining Reconfiguration

UPF User Plane Function

UPP UPF Placement Problem

UPR UPF Placement and Reconfiguration

URLLC Ultra-Reliable Low-Latency Communications

VNF Virtual Network Function

VNFP Virtual Network Function Placement

VNFPC Virtual Network Function Placement and Chaining

CHAPTER 1

INTRODUCTION

The fifth generation (5G) of mobile networks marked the beginning of a new era characterized by a more intelligent and efficient world in which everyone and everything is connected. Bringing new opportunities to almost all spheres of society, including education, industry, transport, and healthcare, 5G technology is a game-changer. In the industry sector alone, more than 1000 applications are projected to benefit from 5G capabilities [1]. Furthermore, 5G is expected to offer tremendous growth in connectivity density, with millions of devices per km^2; mobile traffic capacity with downlink and uplink data rates of up to 20 Gbps and 10 Gbps, respectively; and end-to-end (E2E) data plane delay as low as 1 ms.

The 5G technology provides diverse use cases and services, and three main service types have been defined: enhanced mobile broadband (eMBB), ultra-reliable low-latency communications (URLLC), and massive machine-type communication (mMTC). Services under the eMBB category (e.g., augmented and virtual reality) are characterized mainly by an enhanced data rate and lower latency compared to long-term evolution (LTE) services. URLLC includes use cases with stringent latency and reliability requirements, such as emergency services and autonomous vehicles. Finally, mMTC services are characterized by a massive number of connected devices, for example, smart factories and agriculture applications.

However, many use cases (e.g., smart factory) cannot be defined by a single service type since they may combine requirements related to multiple categories, such as low latency and high device density. Additionally, service demands may vary in density, user mobility, energy efficiency, reliability, and so on. The 5G system (5GS) must support all these service types and their diverse requirements efficiently and cost-effectively. In general, 5G networks must fulfill eight technical requirements (see Fig. 1.1), which set them apart from their precedents.

Designing a single network capable of supporting all the above-mentioned service require-

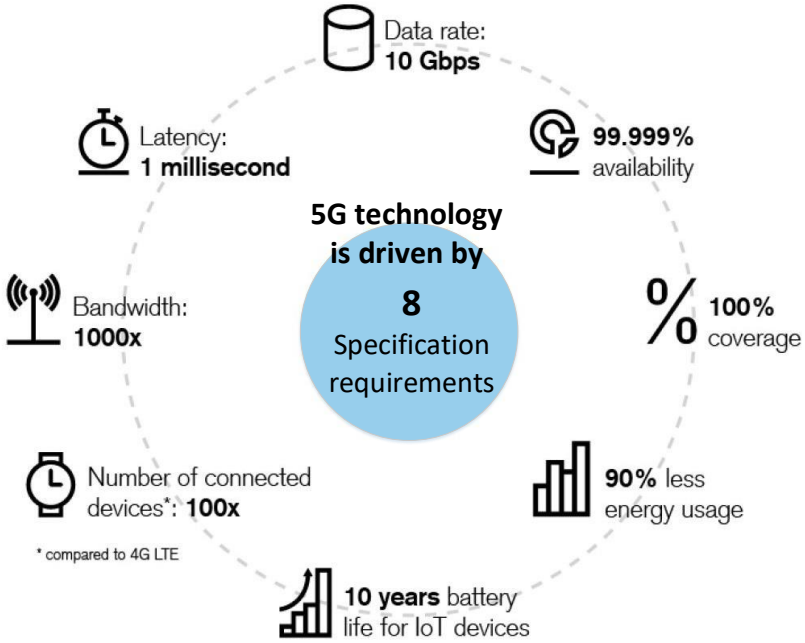


FIGURE 1.1. 5G technical requirements [2, 3].

ments while providing seamless access, high quality of service (QoS) and quality of experience (QoE), expenditure reductions, and revenue growth is a complex and challenging task. Doing so requires innovative and radical changes regarding networks' design and operation, users' communication (e.g., human-to-machine and machine-to-machine communication), and services' creation, commercialization, provisioning, and management. An E2E transformation from the user terminal to the core network provides the required flexibility, efficiency, scalability, robustness, and feasibility to provide current and oncoming services and use cases while optimizing resource usage and expenditures.

In this regard, technologies such as software-defined networking (SDN), network function virtualization (NFV), and multi-access edge computing (MEC) have become fundamental pillars [4, 5] for deploying 5G networks. SDN provides control and forwarding plane separation, thereby enabling intelligent, programmable, scalable, and flexible architectures. NFV decouples network functions from proprietary hardware by defining them as software instances called virtual network functions (VNFs). It allows for flexible and dynamic deployment of VNFs in the most suitable locations, and it enables more efficient usage of infrastructure resources since VNFs can be instantiated and scaled on demand. Thus, NFV is expected to lower 5G capital expenditures (CAPEX) and operational expenditures (OPEX) and improve business agility by introducing new revenue-generating services more quickly and easily than before.

Finally, MEC brings computing, storage, and networking capabilities closer to users at the network edge. MEC will likely help satisfy demands for 5G requirements regarding expected

throughput, latency, scalability, availability, and automation [4]. Additionally, MEC promises enhanced security and privacy by preventing data from arriving at centralized servers and thus avoiding a centralized point of trust [6].

A key concept for the 5GS design is control and user plane separation (CUPS) in both access and core networks. CUPS enables flexible deployment of network functions at centralized or distributed locations. Moreover, it enables efficient usage of resources and better life-cycle management since network functions in both planes can be scaled and updated independently. Thus, CUPS is a straightforward solution capable of providing the necessary flexibility and capabilities to fulfill stringent 5G service demands (e.g., high connection density, high bandwidth, and low latency) while reducing expenditures and simplifying network management and configuration.

With the adoption of the CUPS strategy and the maturation of edge computing technology, applications and network functions such as user plane functions (UPFs) can be deployed more closely to users at the network edge. Doing so satisfies service latency and bandwidth requirements since it reduces the data path length considerably compared to cloud deployments. Moreover, NFV provides more efficient utilization of MEC resources. It also enhances QoS since VNFs can be scaled and instantiated on demand in the most suitable locations.

In the 5GS, UPFs are the evolution of traditional serving gateway (SGW) and packet data network gateway (PGW) in evolved packet core (EPC) networks under the CUPS concept. UPFs are the primary network function in the 5G user plane architecture defined by the Third-Generation Partnership Project (3GPP) [7, 8]. UPFs' primary function is to process data plane packets between the access network and data network (DN). They may perform different functionalities, acting as anchor points for the mobility of intra-/inter-radio access technologies (RAT) or as external packet data unit (PDU) session points of interconnection to the DN. Additionally, they are responsible for packet inspection, routing and forwarding, traffic steering, lawful interception, and QoS handling. However, these functionalities do not need to be supported by a single UPF instance but rather can be implemented and tailored as required, consequently providing higher specialization, flexibility, scalability, and faster processing times in the user plane.

To meet diverse 5G requirements, UPFs can be flexibly deployed in diverse service scenarios and forms (e.g., dedicated, cloudified, and lightweight UPF implementation) in which they perform different roles. Typical deployment scenarios are central, regional, and edge data centers (DCs) [9], each of which has particularities, advantages, and disadvantages. For instance, centralized deployments allow for fully converged access and service continuity since the UPF service areas and capacities are greater. However, these greater capacities come at the expense of higher network response time and backhaul traffic. Therefore, this deployment type is suitable for non-latency-sensitive services, such as web browsing.

Regional UPF deployment, commonly placed in the city DC [9], helps reduce backhaul congestion and response time by performing local data service offloading. Therefore, this deployment is appropriate for video applications. Finally, edge UPFs are suitable for services with low latency

or security requirements that must be processed at the network edge. However, this approach implies a significant increase in the UPF number.

Of these scenarios, UPF deployment at the network edge is the most challenging due to numerous aspects and involved trade-offs. Edge-based systems are characterized by limited resources and a high number of computational nodes, in the order of thousands [10]. This situation increases the complexity of the UPF placement problem (UPP) since many potential candidate servers must be analyzed to determine the UPFs' most suitable locations. Moreover, the constrained capabilities of edge nodes (ENs) limit the available resources of UPFs to values considerably lower than those of centralized deployments. Consequently, a significant number of ENs and UPFs must be deployed and managed.

Services with low latency and high bandwidth demands require UPFs to be placed more closely to users to reduce the service data path and, consequently, the network response time and link congestion. Additionally, ultra-reliable services require the assignment of redundant network functions and links to ensure high availability. Furthermore, high-density scenarios require an increase in UPF processing capacities and the number of deployed instances. In MEC ecosystems, frequent PDU session relocations are also more likely due to UPF's smaller service areas and the presence of highly mobile users. All these translate into a rise in both CAPEX and OPEX.

However, this situation could be avoided by reducing the number of UPFs and consolidating their deployment into a small subset of ENs. Nevertheless, doing so may compromise QoS and QoE, violate service requirements, and increase routing costs. Therefore, a fair trade-off between all the involved objectives (i.e., cost reduction and service requirement fulfillment) must be found.

UPFs' different roles must also be considered, as these roles add to the complexity of the UPP problem. These roles may have diverse requirements and need to be chained in a specific order by accounting for their inter-dependency under constrained latency demands and resource availability at edge locations.

Furthermore, in dynamic environments where users' traffic demands and access points change over time, readjustment of the UPF placement and PDU session mapping is necessary for optimizing resource utilization and avoiding QoS and QoE degradation. However, the activation time of these reconfiguration events must be optimally selected since they may introduce temporal service interruption, additional delay and signaling exchange, and increased expenditures.

The literature addressing the placement of 5G network functions, particularly UPF placement [11–14], is scarce. Most studies [12, 13] have focused on minimizing the user plane latency, dismissing the optimization of UPF associated costs. Moreover, no study has investigated the effects of UPF reconfiguration events, such as reconfiguration costs and session relocations. Therefore, studying strategies to plan and reconfigure the 5G UPF placement and PDU session mapping effectively is of utmost importance.

Extensive research has examined topics and problems closely related to UPF placement

and reconfiguration problems, including the placement of EPC gateways [15–19] and generic virtual network function placement (VNFP) problems [20–25]. However, the solutions proposed by this research have several limitations that restrict their applicability to solve the 5G UPP. For instance, they do not consider a 5G architecture based on the 3GPP standard or 5G strict requirements. Moreover, limited literature has integrated aspects such as latency, resource utilization, reliability, relocation, and cost optimization in the proposed solutions.

5G is the fastest-growing generation of cellular networks to be deployed [26]. In 2019, 5G became a commercial reality with around 10 million subscriptions. Since then, 5G adoption has proliferated, reaching 161 million connections in 2020 and 550 million at the end of 2021 [27]. The number of connections is estimated to reach 1 billion by the end of 2022 and 2 billion in 2025, representing 25% of all mobile connections [28]. According to Ericsson's mobility report, 5G will become the dominant mobile access technology by subscriptions in 2027, with around 48% of all mobile subscriptions [29].

Nevertheless, despite its fast development, many design aspects and research topics still need to be addressed, such as efficient network management under increasingly complex networks [30]. Therefore, this doctoral thesis focuses on designing solutions to solve the 5G UPP in MEC ecosystems efficiently. These solutions aim to reduce expenditures while fulfilling 5G service requirements and ensuring proper levels of QoS and QoE.

1.1 Research Problem and Objectives

As previously stated, this doctoral thesis tackles the UPF placement problem in MEC environments. Therefore, we identified three main research challenges that must be addressed to solve this problem. The following challenges have guided and motivated our investigation.

- How to plan an initial UPF deployment capable of minimizing overall provisioning expenditures while satisfying 5G stringent service requirements?
- How to appropriately readjust a UPF placement setup and their assigned PDU sessions to guarantee a cost-effective usage of resources while enhancing the system QoS?
- What is the best time to reevaluate the UPF placement and session mapping configuration so that the impact of reconfiguration events is reduced while QoS is kept under acceptable levels?

We define a set of primary and secondary objectives to address these research problems. The main goals of this doctoral thesis can be summarized as follows:

1. Design, develop, and assess exact and approximate solutions to help service providers and network operators plan and operate the deployment of 5G UPFs cost-effectively in static and dynamic MEC environments while ensuring 5G services meet satisfaction demands.

2. Design, implement, and evaluate scheduling mechanisms to determine proactively the best time to readjust UPF placement to enhance the overall system QoS while reducing the negative impact of reconfiguration events.

Furthermore, a set of secondary objectives is specified to help attain the main research goals, as follows.

1. Conduct an extensive and systematic literature review on topics and technologies closely linked to our research problem (e.g., 5G, NFV, and MEC).

2. Define a reference architecture for 5G networks capable of providing flexibility, scalability, and programmability to fulfill 5G service requirements while cost-effectively addressing the UPP.

3. Identify the existing solutions related to the UPP (EPC gateways, 5G network functions, and VNFP problems) to identify their solution approach, scope, and limitations.

4. Formulate exact mathematical programming models and heuristic algorithms in line with 5G system requirements to solve the UPP in both static and dynamic scenarios.

5. Evaluate the effects of user mobility, through simulations, on the UPF placement and PDU session-mapping configuration regarding the system QoS.

6. Develop mechanisms to optimize resource utilization dynamically and enhance the system QoS in terms of network response times.

7. Assess the effectiveness and benefits of the proposed methods by comparing their performance with existing solutions.

1.2 Contributions

The main contributions of this thesis, which are based on several articles published in recognized journals and conferences (see Appendix A), can be summarized as follows:

1. A comprehensive review of the literature addressing the VNFP problem in different scenarios (static and dynamic), network architectures (generic, EPC, and 5G), and service function chain (SFC) topologies (single- and multi-branch).

2. A rigorous analysis of existing solutions for placing user plane network functions in EPC and 5G network topologies by addressing their solution approaches, requirements, and limitations.

3. A formulation of the 5G UPP in an MEC ecosystem addressing different variants of the problem (e.g., PDU session data paths formed by single or multiple UPFs) in static and dynamic scenarios. The main target is determining the best UPF placement and service demand mapping configuration to reduce network operators' and service providers' expenditures while ensuring 5G stringent service requirements.

4. Three exact solutions based on integer linear programming (ILP) to address the UPP during the planning phase. These solutions optimize multiple cost components by considering technical and non-technical aspects of the system, such as resource limitations in the underlying physical infrastructure, user mobility, and 5G service requirements (session processing capacity, latency, and reliability demands).

5. Development of two heuristic-based algorithms to reduce the time complexity of mathematical models when solving the UPP in large-scale static scenarios.

6. A proposed simulated annealing (SA) metaheuristic that enhances the quality of heuristic algorithms in static scenarios. This solution incorporates several strategies that significantly improve its efficiency and effectiveness compared with classical SA approaches.

7. Two multi-objective ILP models conceived to reconfigure the UPF placement and PDU session mapping in dynamic scenarios. These models' primary goal is minimizing capital and operational costs related to reconfiguration events while ensuring service requirements. In addition, a heuristic algorithm called the dynamic priority and cautions UPCR (DPC-UPCR) is devised to remap SFC requests (SFCRs) and readjust UPF placement in online scenarios efficiently.

8. Three scheduling mechanisms based on optimal stopping theory (OST) and machine learning (ML) principles that determine when to readjust the UPF placement and session-mapping configuration. These mechanisms determine the best reconfiguration time based on historical data, instantaneous values of sessions with latency violations, an upper QoS threshold, and expected values (e.g., cost, reward, or QoS). Their primary objective is to anticipate the occurrence of QoS degradation events to activate a UPF reconfiguration proactively.

1.3 Thesis Outline

A detailed description of each chapter of the work presented in this doctoral thesis is provided below.

Chapter 2 presents a comprehensive background regarding different aspects of 5G networks, such as key technology enablers and architectural approaches for the core network design. This chapter also provides an extensive review of relevant research studies addressing the VNFP

problem. It analyzes their solutions as well as their main contributions and drawbacks. This analysis allows us to identify the most popular methods and mechanisms in the literature for solving placement and reconfiguration problems, as well as the leading open issues concerning the UPP.

Chapter 3 introduces the 5G UPP in an MEC environment under limited resources and stringent service demands. This chapter provides several solutions (i.e., two ILP models and a heuristic algorithm) to solve the problem. These solutions aim to reduce service providers' and network operators' expenditures when planning the deployment of new services while ensuring 5G service demands for latency and reliability.

Chapter 4 proposes a multi-objective ILP model for the dynamic reconfiguration of the UPF placement and session mapping. This model aims to minimize operational and deployment costs and QoS degradation events while executing placement reconfiguration events. Therefore, it considers multiple cost components associated with the UPF placement (e.g., deployment, operation, and migration costs) and system QoS (network response time and reassigned sessions). Furthermore, the chapter presents a scheduling mechanism that determines the optimal reconfiguration time according to instantaneous values of latency violations and established QoS thresholds.

Chapter 5 addresses the UPF placement and chaining (UPC) problem. In contrast to previous chapters, Chapter 5 contemplates the possibility of splitting UPF functionalities into different instances and chaining them together as required to serve PDU session requests. Specifically, this chapter proposes one exact and two approximated solutions to address the problem. These solutions' primary objective is optimizing expenditures associated with service provisioning and QoS regarding network response time. Several aspects of the system, such as service demands, network capacity limitations, and UPF requirements, are contemplated for this aim, and performance simulations showcase the effectiveness of the conceived solutions.

Chapter 6 investigates the problem of dynamic UPF placement and chaining reconfiguration (UPCR). It describes an ILP model and a heuristic-based solution. The devised solutions aim to determine the best UPC setup during reevaluation events to reduce expenditures while satisfying service requirements. The solutions comprise multiple cost components (e.g., server activation, VNF deployment, and session reassignment) and system specificities. This chapter also presents a scheduling technique to determine when to readjust the UPC configuration to cope with latency violations produced by user mobility.

Chapter 7 introduces an ML-based framework to anticipate QoS degradation events. Based on system information, this solution applies ML algorithms to predict the QoS value or status at a given time horizon. Proactive UPCRs are determined by leveraging the predictor output and an established QoS threshold. Extensive simulation results validate this solution's applicability.

Finally, *Chapter 8* summarizes the main conclusions of this thesis and highlights possible future research directions.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

This chapter reviews the state-of-the-art literature and background information related to concepts and technologies covered by this doctoral thesis. First, Section 2.1 outlines the leading technology enablers necessary for developing 5G and beyond networks. Then, Section 2.2 introduces different architectural designs proposed in the literature for implementing the core of 5G networks. Next, Section 2.3 focuses on existing solutions for addressing the virtual network function placement (VNFP) problem, analyzing the solutions' main contributions and limitations. Section 2.4 provides an overview of the most popular methods for solving placement and reconfiguration problems. Finally, Section 2.5 highlights the main limitations and open research challenges regarding the UPF placement problem (UPP).

2.1 5G Key Enablers

The 5G network architecture must be flexible, adaptable, elastic, scalable, sustainable, programmable, and cost-effective to fulfill current and future use case and service requirements. To achieve these requirements, technologies including software-defined networking (SDN), network function virtualization (NFV), and multi-access edge computing (MEC) have been defined as 5G key pillars.

2.1.1 Software-Defined Networking

SDN [31] is a networking paradigm developed and standardized by the Open Networking Foundation (ONF). SDN is characterized by decoupling the control and data planes. In essence, the network intelligence is logically centralized at one or a set of control entities commonly referred to as SDN controllers. Moreover, the data forwarding plane is simplified and abstracted

for applications and network services. Therefore, the data plane is less complex, allowing fast-forwarding mechanisms and better data plane performance [5].

SDN architecture provides flexibility, dynamism, adaptability, and efficient management of network resources. It accelerates service provisioning and facilitates the dynamic and automatic programmability of network nodes in data forwarding.

2.1.2 Network Function Virtualization

NFV [5, 32] decouples network functions from proprietary hardware by defining them as software instances, called VNFs, that run over the NFV infrastructure (NFVI). The European Telecommunications Standards Institute Industry Specification Group for NFV (ETSI ISG NFV) is the leading working group in developing requirements and guidelines for NFV.

NFV enables flexible and dynamic deployment of VNFs in the most suitable locations. It also allows for more efficient use of infrastructure resources since VNF capabilities can be scaled according to traffic demand variations. Thus, NFV is not only expected to lower CAPEX and OPEX but also to improve business agility by introducing new revenue-generating services and functions more quickly and easily than before.

2.1.3 Multi-access Edge Computing

MEC [5, 33, 34] provides an IT service environment and cloud-computing capabilities at the edge of the network within the (radio) access network ((R)AN) and in close proximity to subscribers [33]. The ETSI ISG on MEC produces standards and norms regarding MEC use cases, reference architecture, requirements, and so on.

Some distinctive features of MEC are low latency, high bandwidth, proximity, location awareness, and real-time insights in radio network information [33]. These characteristics make MEC appealing for network operators and service providers. MEC also enables the flexible deployment of network functions and applications closer to the users, thereby reducing network congestion by eliminating the need to route data through the core network and improving QoE by offering lower response time.

MEC is recognized as a key 5G technology that complements NFV, enables innovative services, optimizes network performance, and creates new business opportunities [33]. MEC will likely help satisfy 5G requirements for expected throughput, latency, scalability, availability, and automation [4]. Additionally, it provides enhanced security and privacy by preventing data from arriving at centralized servers, thereby avoiding a centralized point of trust [6].

2.1.4 SDN, NFV, and MEC Combination

From previous subsections, we can appreciate that SDN, NFV, and MEC provide significant benefits and capabilities to pave the way toward 5G and beyond networks. Moreover, their advantages

multiply when combining them since they are tightly interrelated and highly complementary.

SDN helps NFV by providing isolation, abstraction, network resource sharing, VNF interconnections, and service organization. In an NFV ecosystem where the VNFs are created, adjusted, and removed on demand, VNF interconnection is challenging. This is particularly the case in MEC environments, where the VNFs are distributed and flexibly deployed under constantly changing traffic conditions and stringent latency requirements. In this context, SDN has been considered a complementary technology for improving the flexibility and simplicity of delivering the network service [5]. Precisely, SDN provides flexible and efficient network connectivity among VNFs. In turn, NFV allows the virtualization of SDN elements, such as controllers and forwarding entities, thereby allowing the dynamic adjustment of their assigned resources as well as their deployment in optimal locations.

The combination of MEC and NFV ensures low latency, as well as reliability for end-users. Implementing VNFs at the network edge shortens the service's E2E data path, reduces bandwidth consumption, and avoids traffic congestion at the network core. NFV also improves MEC resource utilization by scaling it up or down according to network demands. Moreover, the integration of NFV and MEC enhances the service scalability since the data can either remain at the network edge or be offloaded to the cloud when computing demands peak.

Furthermore, SDN can enhance the flexibility of the MEC-based infrastructure by enabling a global view of the underlying network. For instance, SDN can enable traffic steering rules for service chaining, computational load balancing, unified control plane interfacing, and retrieving the network context or device information.

Overall, only a 5G network built upon SDN, NFV, and MEC principles will achieve the required performance, scalability, and agility. The combination of these principles enables dynamic and flexible deployment, on demand scaling of network functions, enhanced QoS, and reduced costs.

2.2 5G Network Architecture

This section discusses some of the principal approaches proposed in the literature for designing the 5G core network architecture. We have classified these design lines into two main groups: evolutionary and revolutionary.

2.2.1 Evolutionary Approaches

Different 5G core architectures have been proposed to address EPC network limitations. These architectures are mainly evolutionary as they are based on the current EPC system (see Fig. 2.1) with some modifications that will allow them to become SDN and NFV compatible. In this regard, the literature review highlights three major approaches: SDN-based, NFV-based, and SDN/NFV-based.

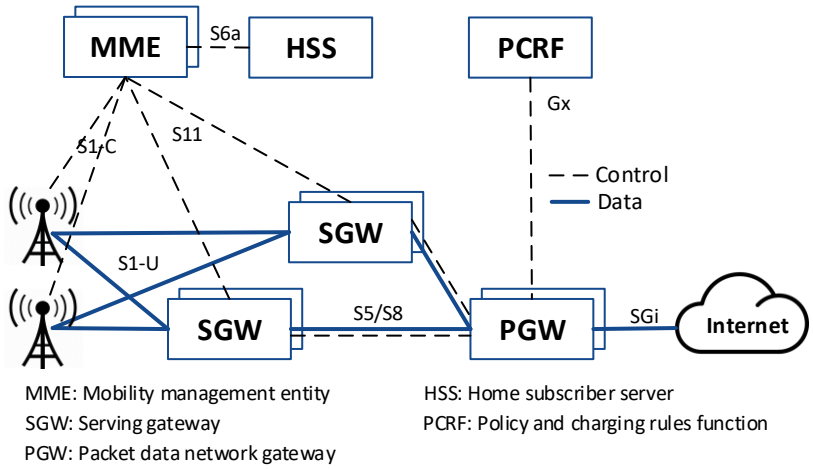
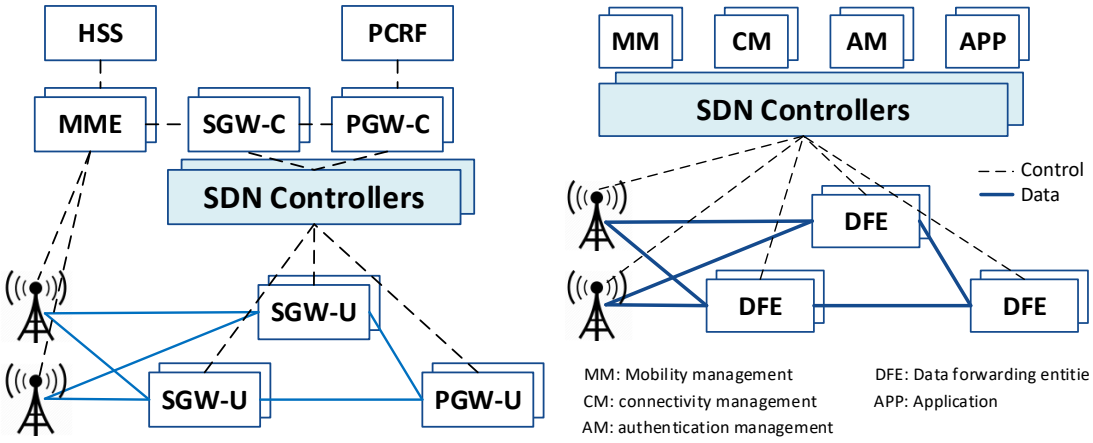


FIGURE 2.1. EPC system architecture [35].

2.2.1.1 SDN-based Cellular Network

This subsection covers 5G architecture proposals based mainly on SDN principles. The utilization of SDN in these proposals can be either partial [36] or complete [37], as shown in Fig.2.2. Partial SDN adoption implies the decoupling of user and control planes of some network functions of the EPC (e.g., SGWs and PGWs) through using SDN. In contrast, a complete SDN design requires major modifications since the traditional EPC architecture is purely based on SDN.



(**a**) Partial SDN-based 5G architecture (**b**) Complete SDN-based 5G architecture

FIGURE 2.2. Examples of SDN-based architectures for 5G networks [35].

In [36], a 5G architecture partially based on SDN is presented. This architecture is similar to the current EPC architecture, except for the SGW implementation. More specifically, the SGW is replaced by OpenFlow switches, and an OpenFlow controller called a resources allocator is

introduced. In contrast, Kozat et al. [37] propose a hierarchical SDN control plane architecture in which the traditional EPC entities no longer exist. Instead, a hierarchical control plane enables different grades of performance and service differentiation.

Each implementation option has advantages and disadvantages. On the one hand, complete SDN architecture provides more flexibility and programmability to the network, but compatibility with the existing EPC is reduced. On the other hand, partial SDN adoption has a lower impact on compatibility but does not fully exploit the benefits of a completely programmable network.

2.2.1.2 NFV-based Cellular Network

In the NFV-based EPC approach, also known as virtual EPC (vEPC), the EPC entities are migrated from dedicated hardware to commodity servers (see Fig. 2.3). Besides the NFV advantages mentioned in Subsection 2.1.2, this approach seems to be the most common and practical to implement as it does not require modifications to the core entities or their interfaces and protocols, which guarantees compatibility with the current EPC elements.

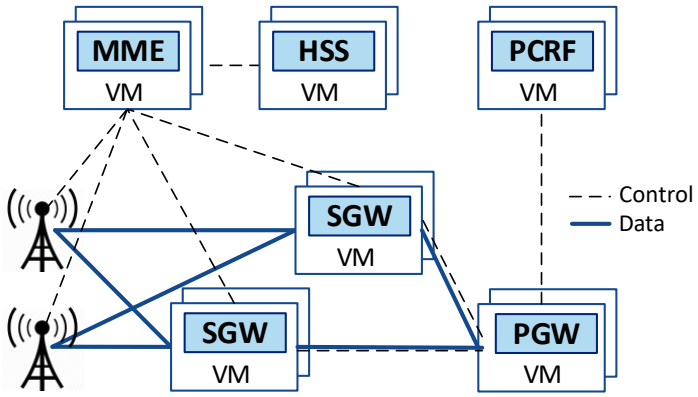


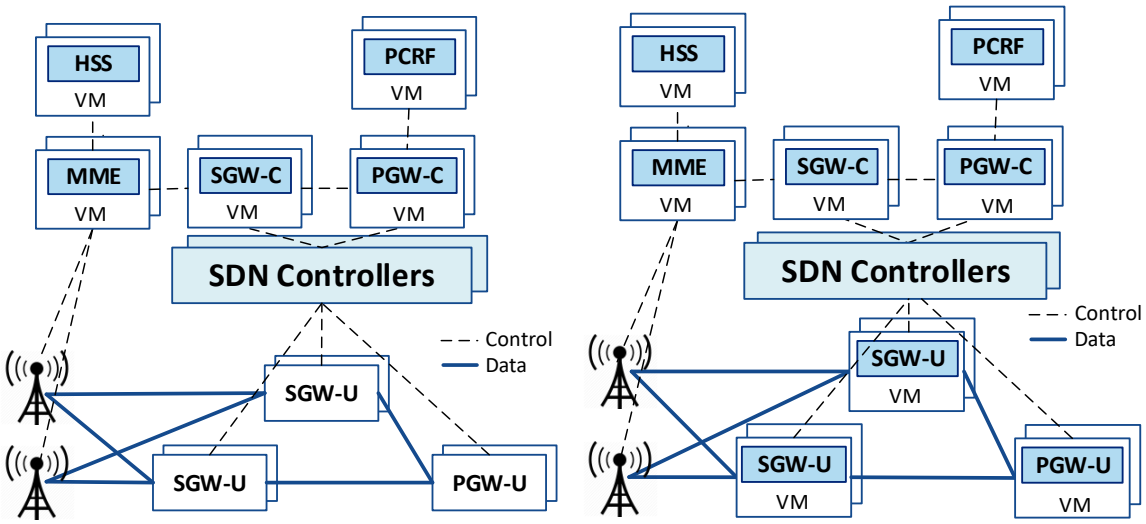
FIGURE 2.3. Example of NFV-based architecture for 5G networks [35].

In [38], an architecture for the 5G control plane based on the virtualization of the EPC entities is presented. This paper uses the vEPC entities to offload the mobile traffic from the legacy EPC on demand by dynamically creating a vEPC network architecture and allocating the necessary components. Three models for offloading traffic are proposed: entirely offloading (full vEPC), data plane-only offloading (vSGW and vPGW), and signaling-only offloading.

However, this architectural design still has some limitations. For instance, the processes for scaling and placing on demand are inefficient since the control and user planes of some network functions, such as SGWs and PGWs, are coupled.

2.2.1.3 SDN/NFV-based Cellular Network

SDN/NFV-based solutions combine SDN and NFV technologies in their architectural design. In this approach, the EPC gateways (SGW and PGW) are split into control and user plane functions following SDN principles. The gateway control planes and the remaining EPC entities are virtualized. The user plane functions can either be virtualized (fully virtualized) or remain on dedicated hardware (partially virtualized), as shown in Fig. 2.4.



(**a**) Partial SDN/NFV-based 5G architecture (**b**) Complete SDN/NFV-based 5G architecture

FIGURE 2.4. Examples of SDN/NFV-based architectures for 5G networks [35].

An example of this design approach is presented in [39], in which the user and control planes of the SGWs and PGWs are decoupled using an SDN controller as the intermediate layer. The SGW and PGW control functionalities are then merged as one entity. The implementation of this architecture can be fully or partially virtualized.

Unlike the NFV-based approach, the control and user planes are decoupled in this case. This facilitates independent scalability of network elements and more flexible placement. However, introducing a new SDN controller and its interfaces to communicate with the control and user planes results in higher response times in the network. Furthermore, the scalability of the SDN controllers is a significant issue, which could be overcome using multiple controllers or a hierarchical design of controllers [35].

Despite the plethora of architecture proposals and innovative designs, these architectures still present some significant limitations, which render them incapable of supporting 5G envisioned services. The leading cause of these limitations lies in adopting an EPC architecture with minor modifications. In this context, a more revolutionary and radical approach capable of overcoming all existing network limitations is mandatory.

2.2.2 Revolutionary Approaches: The 3GPP Standard

The 3GPP is the primary driver for developing 5G standards [4]. To meet the requirements of current and forthcoming services, which have diverse demands and characteristics, the 3GPP organization began developing a new 5G system architecture at the end of 2015.

The first standardized 5G architecture was defined by the 3GPP in the organization's technical specifications (TS) 23.501 [7] and 23.502 [8]. The architecture is an evolution of the 4G system and is based on the CUPS concept, service-based architecture (SBA), and network slicing (NS). It overcomes EPC-based architecture limitations because it has been designed to support SDN, NFV, and MEC technologies. 5G has been designed to be cloud-native in the sense that it should utilize NFV, SDN, and service-based interactions between control plane functions [40].

CUPS guarantees that the resources of each plane can be scaled independently, allowing for further improvements in the flexibility of the network architecture. Moreover, CUPS reduces the complexity of service configuration and the amount of signaling exchanged among network elements. Thus, CUPS allows network costs to be diminished and network performance to be improved noticeably.

SBA decouples the end-user service from the underlying network and platform infrastructure, enabling both functional and service agility. It enables operation in a cloud model, where different functions can be composed into an end-to-end service over standardized application programming interfaces (APIs). Furthermore, SBA simplifies how VNFs are added, modified, or removed from a network processing path (functional agility) and creates new service-specific service paths on demand (service agility) [40]. More specifically, it permits control plane functions to access any service that other logical nodes provide directly since all these network functions connect through a bus interface called a service-based interface (SBI) [41].

NS allows network operators to operate multiple logical network functions running on a standard, shared physical infrastructure. Network slices can be general slices that support a broad spectrum of services or fine-grained slices designed for specific services with certain requirements. Thus, they provide the required flexibility and cost efficiency that network and service providers need. This ability to support multiple customers and services is perhaps the most crucial commercial driver for 5G [40].

Some of the key principles that guide the design of the 5G architecture are as follows [4, 40]:

- Support of multi-vendor integration
- Flexibility for user plane and control plane deployment
- Independent scaling of control and user plane network functions.
- Support of stateless network functions
- Modular function design that allows for different network configurations according to the use case requirements (e.g., NS)

- Unified authentication independent of the access method

2.2.2.1 5G Core Network Functions: Architecture Overview

The 3GPP standard describes the interaction between 5G core network functions in service-based and reference point representation. In service-based representation, the control plane network functions connect through an SBI bus, as shown in Fig. 2.5. In reference point representation, a point-to-point connection exists between the network functions, as depicted in Fig. 2.6.

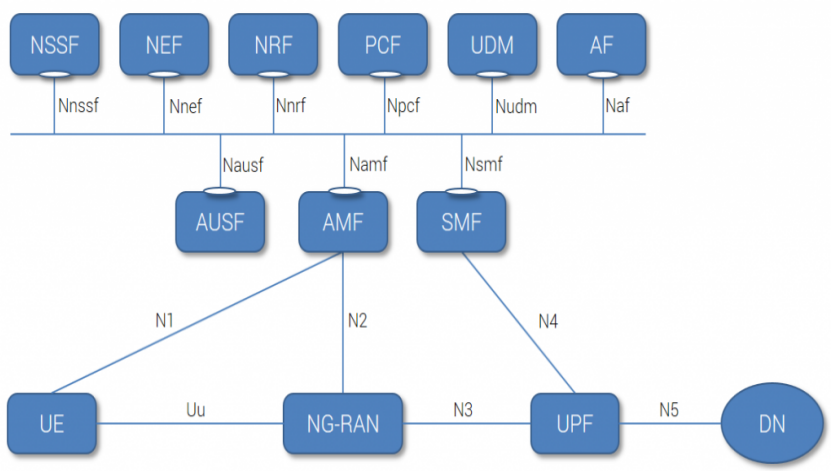


FIGURE 2.5. 5G network architecture in service-based representation [7].

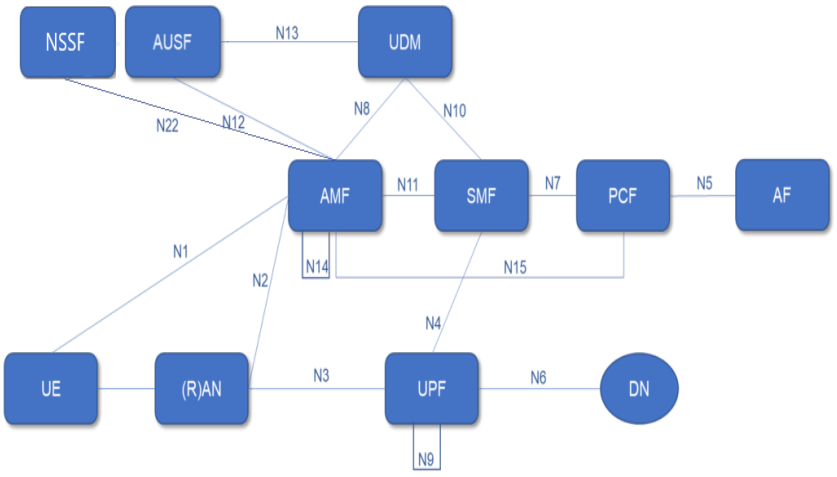


FIGURE 2.6. 5G network architecture in reference point representation [7].

The primary 5G core network functions and their capabilities can be summarized as follows:

- *Access and mobility management function (AMF):* Provides registration, mobility, connection, and reachability management. It supplies transport for session messages between the

user equipment (UE) and the session management function (SMF), access authentication, and authorization. It acts as a termination point for the (R)AN control plane interfaces (N2) and non-access stratum procedures (N1).

- *Session management function:* Handles the session establishment, modification, and release, including tunneling maintenance between UPFs and access network nodes. It is also responsible for DHCP functions and UE IP address allocation and management, and it supports charging interfaces and data collection. The SMF configures traffic steering capabilities at UPFs to route traffic to their proper destination and selects and controls the UPFs for PDU sessions.

- *User plane function:* Processes data plane packets between the access network and DN. It provides access control, packet routing and forwarding, QoS handling for the user plane, and lawful interception. Moreover, it acts as an anchor point for intra/inter-RAT mobility and an external PDU session point of interconnection to the DN.

- *Policy Control Function (PCF):* It is expected to perform similar functionalities to the Policy and Charging Rules Function (PCRF) in 4G networks. Namely, it supports a unified policy framework to govern network behavior, provides policy rules to control plane functions, and accesses subscription information for policy decisions.

- *Network exposure function (NEF):* Acts as an API gateway or proxy, providing security when external applications and functions access the 5G core nodes. The NEF allows network function exposure of capabilities and events to other network functions, translates internal-external information, and receives and stores information.

- *Network repository function (NRF):* Provides registration and discovery functionality so that network functions can discover each other and communicate via APIs. It maintains profiles of network function instances and their supported services within the network.

- *Unified data management (UDM):* Stores subscribers' data and profiles. It is similar to the Home Subscriber Subsystem (HSS) in 4G networks but is used for fixed and mobile access. It generates authentication credentials and access authorization based on subscription data and keeps records of the network functions serving the users.

- *Authentication server function (AUSF):* Acts as an authentication server by storing data for UE authentication.

- *Network slice selection function (NSSF):* Selects the network slice instances to serve the UE as well as the AMF to be used by the user.

- *Application function (AF):* Interacts with the 3GPP core network to provide application services to the subscriber.

The 5G user plane's main network function is the UPF, while all the rest core network functions (e.g., AMF, AUSF, SMF, and PCF) constitute the control plane of the 5G system. Apart from these network functions, other network elements form the 5G architecture:

- *(Radio) access network:* It is compounded by a set of access network nodes (ANN) that connect users to the 5G core (AMF and UPF).
- *Data network:* It offers services (e.g., operator services, internet access, and third-party services) to users.
- *User equipment:* Defines any device connected to the network.

The 5G system promises ubiquitous access to a wide range of services and applications. The 5G access network comprises fixed and heterogeneous wireless resources (Wi-Fi, 4G RAN, and 5G RAN). The 5G RAN, also known as next-generation RAN (NG-RAN), comprises the gNB, which is composed of the radio unit (RU), the distributed unit (DU), and the centralized unit (CU). Hereinafter, we refer to the RU as the radio resource head (RRH) and the union of the DU and CU as the baseband unit (BBU). Different terms are used in the literature to refer to these elements; see [42].

There are two main possibilities for deploying gNBs: distributed RAN (D-RAN) and centralized RAN (C-RAN). In a flat or distributed gNB deployment, the RRH and the BBU are installed at the cell site, similar to a traditional RAN architecture. Meanwhile, in a C-RAN deployment, the BBU is removed from the cell site and physically centralized, while the RRH remains distributed at the antenna site. The connection between RRHs and BBUs is called fronthaul, and the connection between BBUs and the core network elements is referred to as backhaul.

2.2.2.2 5G User Plane Functions

The 5G user plane consists of a single network function type, the UPF, which processes PDU sessions between the access network and external networks, such as the internet, application-specific DNs, and local area data networks (LADNs). Namely, UPFs act as gateways, combining functionalities from traditional SGW and PGW in the EPC architecture.

UPFs may perform different functions, such as an external PDU session point of interconnection to DN and an anchor point for intra/inter-RAT mobility. They are also responsible for packet inspection, routing and forwarding, traffic steering and usage reporting, lawful interception, and QoS handling for the user plane (e.g., data-rate enforcement). In addition, UPFs provide downlink packet buffering, downlink data notification triggering, and uplink traffic verification. They rely on the SMFs in the control plane to perform these functionalities; these SMFs are responsible for selecting and managing the UPFs associated with a PDU session.

The PDU session data path usually requires a single UPF instance, which may perform all above-mentioned functionalities. However, the 5G standard allows UPF to be split between

multiple roles to divide the load and enable functionality specialization. Thus, UPF functionalities can be selected and tailored as necessary, either during or after session establishment and transparently to the user. Regarding UPFs' functionalities, four main roles can be defined:

- *PDU session anchor (PSA) or anchor UPF (aUPF)*[1]: Terminates PDU sessions at the DN end and is responsible for IP anchoring.
- *Intermediate UPF (IUPF)*: Forwards traffic between the access network and the PSA. This type of UPF can be used to guarantee the continuity of the service, either when the user moves in a vast range network or due to transport network limitations.
- *Uplink classifier (UL-CL)*: This functionality allows local traffic to be steered toward local and central services. It determines how the user traffic should be routed based on matching traffic filters provided by the SMF. Additionally, it classifies flows based on the source and destination IP addresses. This functionality can be inserted in the PDU session data path to create new data paths for the same session [7].
- *Branching point (BP)*: A PDU session may be associated with multiple IPv6 prefixes (multi-homing PDU sessions), which provide access to the DN through multiple PSAs that branch out from a common UPF that supports the BP function [43]. The UPFs that support the BP functionality forward UL traffic toward different PSAs based on the source IPv6 prefixes, and they also merge the downlink traffic from different PSAs toward the UE.

UL-CLs and BPs are inserted into the session data path between the access network and PSAs. However, unlike the IUPFs, these functionalities allow a single PDU session to be served by multiple PSAs connected to the same DN to support SSC mode 3, selective traffic routing, and user plane re-selection. The latter is especially useful for edge computing scenarios and LADN, where services are hosted closer to the users. UPF roles are not mutually exclusive. Specifically, UPFs acting as UL-CLs or BPs may also support the PSA functionality. Moreover, a single UPF entity can hold different roles for one or various PDU sessions [44].

Additionally, according to UPFs' location in the data path, they can be divided into two categories: terminated and intermediate UPFs (tUPFs and iUPFs). The first category is compounded by PSAs, whereas the second one embraces all the UPFs (i.e., IUPF, UL-CL, and BP) that connect with the access network and other UPFs or solely with other UPFs. Notably, UPFs that support BP and UL-CL functionalities are classified as iUPFs, but not all iUPFs support UL-CL or BP. In this work, the term miUPF is used to distinguish intermediate UPFs that connect to multiple PSAs (i.e., UL-CL and BP).

The 5G 3GPP specifications place no limit on the number of UPFs simultaneously serving a PDU session. These UPFs can be chained together in different topologies as required. This, along with the role specialization option in the user plane, allows for UPF deployment in multiple

[1]In this work, the terms PSA and aUPF are used interchangeably.

configurations and locations, which are fundamental for reducing the network response time. For instance, some UPFs may be placed in the core network, while others may be located closer to the access network.

The UPF has four distinct reference points (i.e., N3, N4, N6, and N9), as shown in Fig. 2.7. The N3 interface transport GPRS Tunneling Protocol user plane (GTP-U) packets between the RAN (gNodeB, gNB) and the initial UPF in the uplink direction. The SMF controls (setup, modify, and delete) the UPFs to manage PDU sessions through the N4 interface, which is a control plane interface that employs the packet forwarding control protocol (PFCP). The UPFs connect to the DN through the N6 interface. Finally, the N9 interface is used for the communication between UPFs (e.g., PSA and iUPF) when intermediate UPFs are inserted into the session data path.

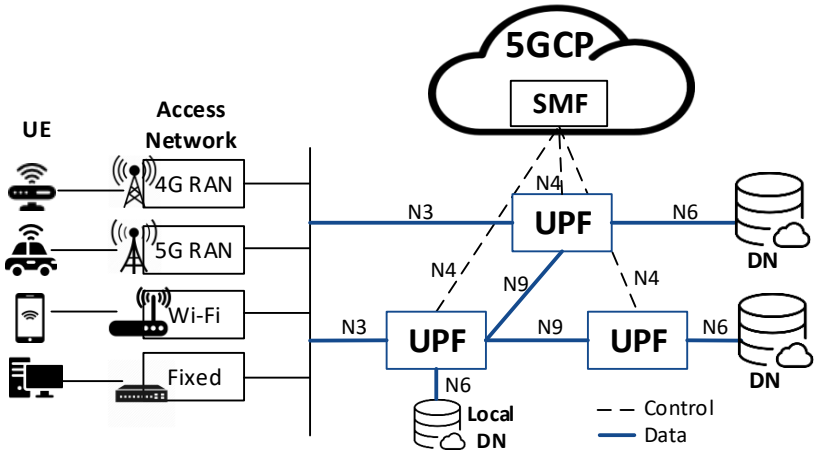


Figure 2.7: 5G user plane overview with reference point representation.

2.3 VNF Placement Problem

In the following subsections, we analyze several studies related to the UPP by describing their design goals and solution approaches. We cover various research studies dealing with the placement of both generic VNF instances with single and multiple constituent VNFs as well as EPC and 5G-specific network functions.

2.3.1 Generic VNF Placement

The VNFP problem refers to the placement of VNF instances over an NFV-based network infrastructure. Solutions to the VNFP problem can be grouped into different categories according to selected criteria [45]. For example, they can be classified as generic or specific solutions based on the VNF type. The first category focuses on defining placement strategies without particularizing their solution to a specific network function type. In contrast, the second category addresses its solutions to specific network function types by considering distinctive requirements

or metrics. Both categories can be solved by adopting static or dynamic approaches concerning flow handling. Additionally, the VNFP has been addressed in different deployment environments, such as MEC [46, 47], central cloud [22], or a mixture of both technologies [20, 21, 48].

This subsection presents relevant studies addressing the placement of generic VNF instances for both single and multiple VNF types, while Subsections 2.3.2 and 2.3.3 investigate the adopted state-of-the-art approaches for placing some specific network elements (e.g., EPC gateways and 5G UPFs). For more insights about the VNFP problem, the reader can refer to the following surveys: [45, 49–51]

2.3.1.1 Single VNF Placement

A significant number of research papers address the placement problem of single VNF instances using either static [20, 21, 47, 48] or dynamic approaches [22, 46, 52, 53].

Jemaa et al. [20] present a multi-objective mixed integer linear programming (MILP) model to address the VNF placement and provisioning problem in a two-tier carrier cloud infrastructure. They aim to reduce cloud overload and resource utilization and avoid violation of SLA requirements by considering latency and host capacity. Therefore, they apply optimization techniques and queuing QoS models. Behravesh et al. [21] investigate the joint user association and VNFP problem to reduce service-provisioning costs (i.e., VNF deployment and routing). They formulate the problem as an MILP and consider multiple restrictions, such as available link and host resources, service delay, and path continuity. However, they do not account for service reliability requirements.

Yala et al. [48] approach the VNFP problem with the design goal of jointly optimizing latency and availability. Since they face conflicting objectives, they adopt a weighted sum method to transform the problem into a single objective. Moreover, they design a genetic algorithm (GA)-based metaheuristic to solve the problem in polynomial time. Both solutions consider availability requirements, cost budget, and node capacity constraints. Neither latency nor backup instance restrictions are considered.

In [47], the latency-aware and survivable VNF mapping problem is formulated as a multi-objective MILP problem. The authors of this study seek to deploy VNF instances reliably while optimizing provisioning costs (i.e., node activation, VNF deployment, and routing costs). They consider several restrictions, such as backup VNF instances, anti-affinity, capacity, and latency demands. Additionally, they show that the problem is NP-hard and propose a simulated annealing (SA) metaheuristic to obtain near-optimal solutions in polynomial time.

Ghaznavi et al. [52] propose an optimization model and a heuristic for the elastic placement of VNFs in response to traffic variations (workload arrival and departure). Their objective function comprises multiple cost components (VNF installation, transportation, reassignment, and migration) that are mainly subject to link capacity constraints. The main design goal in [22] is to optimize the VNF relocations and the QoE in 5G DCs jointly. The problem is formulated as

an MILP and transformed into a single-objective optimization function by applying weight factors. The authors demonstrate that the problem is NP-hard for large-scale scenarios and propose an ant colony optimization (ACO) metaheuristic to increase the problem solution efficiency. These solutions, which focus on re-evaluating the VNFP due to user mobility, run on each DC to manage the VNF requests of its underlying eNBs. However, both studies are reactive in nature.

Cziva et al. [46] propose an ILP model to minimize user latency subject to capacity limitations and delay requirements when placing VNFs in MEC environments. Additionally, they present a scheduler mechanism to re-evaluate the VNFP dynamically according to a maximum threshold of the cumulative sum of latency violations and an expected migration cost. However, their conceived placement solution does not consider VNF reconfiguration costs when readjusting the placement.

In [53], Kawashima et al. present a solution for the dynamic VNFP based on a model predictive control. Their objective is to predict the traffic demands to begin migration decision-making in advance. The proposed optimization model minimizes the weighted sum of the number of active hosts and the expected number of migrated VNFs. Although multiple restrictions are considered, including VNF mapping, path continuity, and available resources in nodes and links, the service latency requirement is not considered in their solution.

2.3.1.2 Single-branch SFC Placement

The placement of VNFs on top of the NFVI, as well as the mapping of virtual links among them to compose a network service (i.e., service function chain), is a popular research topic, and one of the most important and sophisticated problems within NFV management and orchestration (MANO) operations [54]. Despite its novelty, the virtual network function placement and chaining (VNFPC) problem, also referred to as the SFC placement or mapping problem, has been widely addressed in the literature.

Mouaci et al. [23] attempt to solve the VNF placement and routing problem. To this end, they introduce a path-based MILP model to optimize node opening and VNF deployment costs. Their model considers VNF order, anti-affinity, routing, and latency restrictions but ignores the link capacity limitation. In [55], Alleg et al. formalize the VNFPC problem as an ILP and propose a degree-based heuristic solution. They seek to minimize the E2E delay and resource allocation cost. Their model accounts for node and link capacity, VNF mapping, E2E delay, and flow conservation constraints.

Song et al. [56] investigate the resource-efficient VNFP problem to minimize computing and communication (bandwidth) resources. An ILP and a heuristic are provided based on the hidden Markov model. In [57], the main objective is optimizing the utilization of VNF instances and links in a cloud data center environment. To this aim, a mathematical model and a heuristic approach are developed, and an online scenario in which service demands arrive at different time instants is assumed. Both [56] and [57] consider several aspects related to the VNFPC problem,

such as resource capacity, VNF order dependency, and flow conservation. However, they overlook the E2E service latency requirement.

The main design goal in [24, 25] is optimizing routing and VNF deployment costs. Allybokus et al. [24] formulate the problem as an MILP that encompasses several constraints, such as service delay, VNF order, anti-affinity, and resource limitations. Moreover, they develop a heuristic algorithm that combines the linear relaxation of the problem with a greedy approach. Similarly, Li et al. [25] formalize the problem as a multi-objective ILP model and propose a greedy-based heuristic to solve the problem in large-scale networks. They examine the SFC placement problem in a hierarchical MEC ecosystem to minimize the weighted costs associated with substrate resources subject to capacity and propagation delay restrictions. In [58], Liu et al. formulate the SFC traffic steering problem as an ILP aimed at minimizing node running, VNF deployment, and communication costs. They present an algorithm for improving the solution efficiency of the problem by considering it as a multi-stage decision problem in which the decision for each stage relies on the stage's current state.

The previous studies approach the VNFPC problem using static scenarios. More specifically, they consider static SFC demands and network topologies. However, in dynamic environments in which user demands and locations change over time, this solution approach may lead to sub-optimal deployments with higher operational costs and poor QoS. Thus, a different approach is adopted in [59–72] by which the VNFPC configuration is dynamically readjusted to cope with service demands.

Liu et al. [59] investigate the joint optimization problem of new and in-service SFCR provisioning. Their main objective is maximizing operator revenue by optimizing the profit of acceptance requests and reducing deployment costs. They formulate the problem as an ILP model and present a column generator-based solution to reduce time complexity. However, they do not consider service latency requirements, and the proposed solution presents scalability limitations. Askari et al. [60] design an algorithm for dynamic SFC placement in metro-area networks. Their main objective is reducing resource consumption regarding the number of active VNF nodes while ensuring service latency requirements and low blocking probability. However, neither [59] nor [60] contemplates placement and chaining reconfiguration costs in their solutions.

In [61], the VNF migration cost is optimized subject to node capacity and SFC latency restrictions. The problem is formulated as an ILP model, and a heuristic algorithm is developed to solve it in polynomial time. Additionally, in [62], the authors provide a multi-objective ILP model to minimize SFC reconfiguration costs for value-added services in content delivery networks. Several cost components are considered (i.e., routing, migration, VNF hosting, and instantiation), along with resource capacity, service latency, and mapping constraints.

Li et al. [63] address the problem of dynamic VNF mapping and scheduling to maximize revenue by increasing the service acceptance ratio. They formulate the problem as an MILP subject to service latency and available infrastructure capacity. They design a two-stage online

algorithm that greedily maps the SFC constituent VNFs based on their waiting time. A delay-aware rescheduling scheme is also triggered if the latency requirement is not satisfied; this scheme remaps existing VNFs.

The authors of [64] and [65] investigate the impact of user mobility in MEC and fog environments, respectively, with the primary objective of enhancing VNF migration performance due to user mobility. Chen and Liao [64] formulate the problem as an ILP aimed at optimizing user satisfaction by minimizing the effects of handovers in the SFC migration delay and service downtime. To this aim, they consider resource limitations in MEC servers and link bandwidth, service propagation delay requirement, and VNF migration time. They show the problem as NP-hard and propose a heuristic solution called Follow-Me Chain. Likewise, Zhao et al. [65] examine how to remap and migrate SFC constituent VNFs efficiently so that the reconfiguration cost, the blocking ratio, the migration time, and the downtime of the SFC requests are minimized. They thus develop a mathematical model, a heuristic solution, and two SFC migration strategies. However, these two studies adopt a user-based reactive approach since they assume the migration decision upon user handover.

Pei et al. [66] seek to minimize the overall SFC embedding cost, which comprises bandwidth, memory, CPU, E2E delay, and VNF placement, through the dynamic VNFP. They formulate the problem as a binary integer programming model and provide two algorithms, called SFC eMbedding APproach (SFC-MAP) and VNF Dynamic Release Algorithm (VNF-DRA). The SFC-MAP algorithm maps SFC requests in the infrastructure while VNF-DRA periodically checks the VNF utilization rate to release those instances with low utilization.

Liu et al. [67] address the SFC dynamic reconfiguration problem to balance the service provider revenue and reconfiguration costs. They formulate the problem as an ILP model and propose a heuristic solution that combines Tabu search (TS) and Fuzzy C-means. The reconfiguration trigger condition is based on the substrate network utilization thresholds (i.e., lower and upper bounds). Likewise, in [68], the migration of VNFs is triggered based on network resource overload or node failures. Their study formulates the VNF instance migration and SFC reconfiguration problem as an ILP model and derives a heuristic algorithm for solving the problem in polynomial time. Their main objective is minimizing SFC E2E delay along with network imbalance. Since they deal with a multi-objective optimization problem (MOOP), they use a weighted sum method to reflect the relative importance of each term in the objective function.

The main limitation of the above-mentioned literature is its reactive nature since it decides the readjustment of the SFC mapping upon a given event (e.g., threshold violation). Thus, these approaches may further degrade the QoS and introduce higher delays.

The objective in [69] is minimizing VNF deployment and routing costs of Internet of things (IoT) mobile devices at the edge. Therefore, mobility prediction methods are applied to determine the probability of IoT devices visiting a set of destinations. The problem is formulated as an ILP model, and a heuristic is proposed to reduce the model computation time. The proposed

solutions are executed periodically as a batch-like processing service. Likewise, Wang et al. [70] present a user-managed framework for the online orchestration of SFCs at the edge in which every user is responsible for managing and allocating resources to its service based on learned information. The authors aim to minimize the E2E delay by leveraging contextual information (i.e., user demand and mobility). They formulate the problem as mixed integer programming (MIP) subject to affinity, link capacity, and path-related constraints and present a bandit-based algorithm. However, these two studies do not consider the cost associated with the placement reconfiguration procedure.

Gu et al. [71] propose an online learning algorithm to predict the SFC flow rate to make horizontal scaling decisions. They focus on minimizing NFV provider OPEX by proactively scaling and provisioning VNF instances. They formulate the problem as a multi-objective MILP model and design an algorithm for its online solution. These methods optimize the overall operational expenditure, a compound of multiple cost components (i.e., VNF running, deployment and migration, traffic forwarding, and backup facility costs). Moreover, they consider restrictions related to the physical infrastructure (i.e., node and link capacity) and path continuity. However, they neglect the service latency requirement.

In [72], a cluster-based proactive solution is introduced to reduce the complexity of SFC mapping and reconfiguration. This solution comprises an ILP model and an optimized k-medoids clustering approach. The mathematical model considers multiple optimization objectives: latency, service-level objective (SLO) violation cost, hardware resource utilization, and VNF readjustment cost. They contemplate restrictions regarding host and link resources and energy and operational cost budgets. Each cluster in the solution checks placement constraints and provides specific placement and readjustment configurations.

2.3.1.3 Multiple-branch SFC Placement

The studies discussed in the previous subsection consider single-branch SFCRs in which the constituent VNFs are sequentially connected in a line graph. However, the literature addressing multiple-branch SFC mapping is scarce.

Luizelli et al. [73, 74] approach the VNFPC as an optimization problem combined with heuristic solutions. Their main objective is reducing the number of deployed VNF instances subject to the resource limitations of nodes, VNFs and links, VNF mapping, E2E delay, and flow conservation rules. To overcome the scalability limitations of the models, a binary search heuristic is proposed in [73], while in [74], an algorithm based on a variable neighborhood search (VNS) metaheuristic is introduced. These studies consider VNF sharing and the existence of different SFC topologies.

The goal in [75] is minimizing the VNF deployment and link costs subject to resources, delay, and traffic-routing constraints. An MILP model and a heuristic algorithm are provided. These solutions consider traffic splitting into multiple branches and multiple ingress/egress nodes.

Similarly, Leivadeas et al. [76] adopt a path-splitting approach to address SFC mapping in an MEC-cloud infrastructure to minimize the overall deployment cost and E2E communication delays. They present two solutions: an MIP model and a TS algorithm. The MIP formulation accounts for aspects related to resource capacity, flow conservation, traffic splitting, and VNF mapping. However, the definition of constraints for ensuring the correct order of VNFs in the chain and service latency demands are omitted. Jalalitabar et al. [77] address the branching SFC embedding and routing problem, defining a set of policies to be considered, such as VNF dependency, branching, and anti-affinity. Additionally, they propose an algorithm for optimizing node and link utilization jointly.

In [78], Alhussein et al. address multicast service orchestration for single and multiple services. They aim to optimize two conflicting objectives (i.e., VNF deployment and link provisioning costs) that are subject to node capacity and flow conservation constraints. Additionally, they show the NP-hardness of the problem and conceive low-complexity heuristics to find efficient solutions. However, they do not consider E2E latency requirements, and they solve the problem with the assumption that VNFs cannot be shared among multiple SFCRs. Ren et al. [79] propose using the service function tree to embed SFC for multicast tasks. They propose an ILP model and a heuristic to optimize deployment and routing costs. They solve the problem subject to node capacity, flow conservation, and multicast flow constraints; however, no restrictions regarding either link capacity or E2E delay are included.

The main design goal in [80] is maximizing the number of admitted SFCRs in a multicast network by considering several restrictions, such as node and link-limited resources, delay requirement, and a maximum number of instances. In the study, the problem is formulated as an MILP, and a column generation decomposition and two heuristic solutions are provided.

In [81], Awad et al. address the problem of dynamic resource adaptation in which VNF resources can be readjusted by performing either migration or scaling procedures (vertical and horizontal). The problem is formulated as an ILP model that minimizes energy consumption, resource utilization, and SLO violation subject to node capacity limitations and service latency requirements. Additionally, several multi-objective metaheuristics are designed to reduce the problem solution time. Although these solutions contemplate linear and non-linear SFC topologies, restrictions regarding VNF inter-dependency, path continuity, VNF anti-affinity, and link bandwidth capacity are neglected. Moreover, a scheduler mechanism to decide when VNF resources must be readjusted is missing.

Miotto et al. [82] design a framework for adaptive VNFPC called NFV-PEAR. This framework has an optimization modulo that reconfigures the VNFPC according to demand fluctuations. The VNFPC problem is formulated as a multi-objective ILP model aimed at reducing resource consumption, VNF and flow mapping changes, and SFC reassignments.

2.3.2 EPC Core Network Function Placement

The placement of EPC network functions has been widely addressed in the literature. Some studies have focused on the placement of multiple EPC functions [83–87] through SFC approaches. Others have concentrated on single network functions, with the SGW and PGW being the most common, and approach the problem through VNFP solutions.

Authors of [83–85] optimize the placement of the most common network functions of the EPC architecture (i.e., MME, SGW, PGW, and HSS) modeled as service chains. Baumgartner et al. [83, 84] aim to minimize the utilization cost of links and nodes while considering VNF requirements (e.g., processing and storage) and available infrastructure resources. Additionally, in [84], the authors extend their conceived solution by including aspects such as VNF processing time and E2E propagation delay. In [85], an MILP model is proposed to minimize the combination of several cost components (VNF deployment and infrastructure utilization) subject to resource capacity limitations, VNFP constraints, flow conservation, and latency requirements. Moreover, SFCRs are divided into user and control plane sub-chains.

Dietrich et al. [86] also approach the EPC VNFP in terms of bandwidth and resource utilization. Their primary focus is optimizing the load distribution among server nodes and links. To this aim, an MILP formulation, its relaxed variant, and a greedy heuristic are proposed. Capacity constraints and delay budgets between EPC components are some of the considerations for network function placement. Similarly, Papagianni et al. [88] propose an MILP model to minimize VNF deployment and bandwidth-associated costs through VNF sharing. They consider several placement restrictions, such as link and node capacities, VNF placement, and flow conservation.

To minimize bandwidth consumption, Gupta et al. [87] address the EPC VNF placement. They propose an ILP for VNF placement and traffic routing along the SFC constituent VNFs. To this end, they consider VNF interactions in the control and data planes, as well as application latency and NFVI resources (i.e., CPU). They not only show that distributed vEPC replicas reduce bandwidth but also that not all the vEPC functions need to be distributed.

Basta et al. [19] propose the possibility of using a 5G hybrid architecture in which the PGW and SGW functions could be deployed either by VNFs or SDN controllers. Their main aim is determining the optimal sizing and planning for DCs, VNFs (PGW and SGW), and SDN controllers by jointly analyzing SFCRs from both data and control planes. The authors present three models to minimize the network load and costs of DC resources. They account for several aspects of the system, such as SFC latency requirements, core network topology, and the DC number.

Unlike previous studies that model the EPC VNF placement through SFCs, Bagaa et al. [89] develop an algorithm based on the coalition formation game to find the optimal number and placement of the vEPC core instances over a federated cloud. Each cloud is considered a game participant in this solution that seeks to reduce the deployment cost while ensuring QoS.

Despite addressing the placement of EPC network functions, in which users are highly mobile,

none of the above-mentioned studies addresses this issue in their solutions. Authors in [90] amend this limitation by approaching the EPC network function placement in a mobility-aware manner. They formalize the problem as an ILP model and conceive a sub-graph-based approach to solve it in large topologies. Their main target is minimizing the average latency during handovers while meeting constraints for latency budget and processing resources.

Moreover, none of the aforementioned studies considers the necessity to readjust the placement configuration due to dynamic traffic variations. In this regard, Moratta and Kassler [91] proposed the use of robust optimization due to the difficulty of predicting VNF demands. They seek to minimize energy consumption by optimizing the VNF-assigned resources and flow routing jointly. They validate the effectiveness of the solution in a virtualized EPC scenario.

Other state-of-the-art approaches, such as [92, 93], address workload variation in the EPC core elements (i.e., MME, SGW, and PGW) through scaling mechanisms. The authors in [92] model the vEPC core as an open network of G/G/m queues and propose a dimensioning algorithm to auto-scale the VNFs dynamically. Additionally, Arteaga et al. [93] introduce a threshold-based mechanism for horizontally and vertically auto-scaling vEPC elements based on CPU usage and the number of registrations.

2.3.2.1 SGW and PGW Placement

The placement of SGW and PGW has been addressed in a wide variety of research studies [15–19] with different points of view.

In [15, 18, 94], the SGW placement problem is approached to minimize SGW relocations. Taleb and Ksentini [15] asserted the importance of avoiding mobile gateway relocations through the optimal placement of SGWs to reduce costs and enhance the overall QoE. To this end, they model the SGW placement as a service area planning problem and design a greedy algorithm. They aim to reduce the SGW relocation cost subject to capacity restrictions.

In [18, 94], the authors divide the SGW functionalities into user and control plane components denoted as SGW-U and SGW-C, respectively. Both articles focus on the placement of the SGW-C to reduce relocations and balance SGW-C load. In [18], two ILP models are conceived, and the Nash Bargaining game is adopted to determine a fair trade-off between both optimization objectives. However, latency requirements are overlooked, and the SGW-C placement is addressed by assuming that the SGW-Us have already been deployed. Similar optimization objectives are pursued by Basu et al. [94], albeit under latency restrictions. The latency parameter is measured regarding the transmission time between a service area and its assigned SGW-C and the SGW-C processing time.

The authors in [95] investigate the effects of SGW placement strategies on the backhaul bandwidth consumption. Specifically, they study three solution approaches for the SGW placement (i.e., centralized, distributed, and optimal deployment). They conclude that a distributed SGW placement in which each base station is co-located with an SGW performs similarly to an

optimized SGW placement. Moreover, this deployment approach significantly reduces the cost of backhaul traffic. However, no restrictions regarding the SGW capacity or service latency are considered.

The authors of [16] address placement of virtual PGW to reduce costs while ensuring QoE. The PGW load and imbalance are optimized by considering PGW capacity, application/service type, and geographical location when selecting a PGW to serve user requests. They model the PGW placement problem as a non-linear optimization problem and provide three heuristics to solve it. However, aspects such as relocation and service latency requirements are not considered.

Studies such as [17, 96] address the placement problem of SGW and PGW. Taleb et al. [17] present different solutions for optimizing user plane response time and SGW relocations by considering user mobility patterns along with delay and relocation constraints. However, since these design goals represent conflicting objectives, a solution based on game theory techniques is provided, similar to [18]. Nonetheless, the authors do not consider VNF resource requirements. In [96], Kiess and Khan analyze the gateway transmission cost for centralized and distributed architectures in a nationwide network. However, they do not address latency requirements.

The dynamic placement of both SGW and PGW is addressed in [97, 98]. These studies propose solutions based on constraint programming for the dynamic adaptation of SGW and PGW to user and service demands. Similar to [17], their main goal is to determine the optimal location for these network functions so that SGW relocations, the length of the path between the users and their assigned GWs, and the number of VNF instances are minimized.

Yousaf et al. [98] introduce the concept of softEPC for the virtualization of EPC network functions over a physical transport network topology. To investigate the virtualization benefits, they develop a load-aware greedy algorithm that dynamically places SGW and PGW instances according to service traffic demands to minimize network capacity (i.e., bandwidth and processing capacity). Other studies, such as [99], design load-balancing solutions for the dynamic gateway selection to avoid QoS degradation.

Based upon the previous studies, the main lines of research for solving the gateway placement problem in EPC networks are the optimization of relocations, network response time, load distribution, and deployment cost in regards to the number of gateway instances or traffic routing. Despite the wide variety of studies in this field, a solution integrating multiple aspects related to the problem, such as user mobility, resource capabilities (i.e., links, servers, and gateways), and service latency and reliability requirements, is missing. Moreover, none of the studies account for reliability when proposing solutions. Furthermore, solutions approaching the dynamic placement reconfiguration of these network functions are also lacking.

2.3.3 5G Core Network Function Placement

The literature reviewing the placement of 5G core network functions in a 3GPP-based architecture is scarce due to its novelty.

Bagaa et al. [100] address the core function placement in EPC/5G architectures with the main target of minimizing VNF deployment cost. To this aim, they propose an MILP model to derive the optimal number of VNF instances required to serve specific traffic demands. Additionally, they design an algorithm based on a coalition formation game to determine the best location of the EPC/5G core elements. Similarly, a Bayesian coalition formation game is proposed in [101] to determine 5G core VNF placement with unknown information about players (i.e., cloud providers). This study seeks to determine the optimal number of VNF instances and their location over a federated cloud at the minimum cost. Therefore, tracking areas are assigned to VNF instances based upon their available capacity. However, the latency requirement is neglected. The performance of the proposed approach is evaluated by creating a 5G core, which consists of AMF, SMF, and UPF. Neither [100] nor [101] accounts for VNF interrelation.

The authors of [102] devise a management architecture for 5G services by leveraging the combination of NFV, SDN, and MEC technologies. Their framework provides distributed and online deployment of VNFs. The VNF allocation problem is formulated as multi-objective mixed non-linear integer programming (MNIP), and an algorithm is proposed to solve the problem. Their solutions aim to minimize the overall cost associated with backhaul traffic, energy consumption in edge and cloud nodes, and revenue loss due to backhaul delays.

Do and Kim in [103, 104] address the placement of state management functions (StateMFs) –unstructured data storage function, user data repository, and NRF– over a geo-distributed cloud infrastructure. In [103], the design goal is to minimize the signaling transfer (state transfer) cost between the StateMFs and their traffic load. They considered the frequency of handovers between the StateMF service areas and the number of PDU sessions per user in the problem formulation. However, technical requirements like 5G latency or network function maximum capacity are not addressed.

Similarly, in [104], the primary goal is to minimize the state transfer between StateMFs and the total packet latency in the entire network. Although packet latency is analyzed as a design parameter, the chosen approach is not the most suitable. The overall latency can demonstrate a brief status of the network, but it cannot show the worst case. Both studies formulate the problem as a multi-objective optimization model and propose three solutions for the StateMF placement based on ε-constraint and adaptive weighted sum.

In [105], the joint placement of RAN and core network functions (e.g., AMF, UDM, SMF, and UPF) in an edge-cloud environment is considered through the introduction of weights. Specifically, preference-weighted costs are assigned to network functions according to their type, service/slice, and layer number. This paper's main objective is minimizing costs subject to node capacity and latency constraints. The problem is formulated as a non-linear and non-integer programming model, and a GA solution is provided to reduce the solution's complexity.

The authors in [14, 106] address the joint user association and SFC placement problem in a 5G network. They formulate the problem as an ILP model and design a heuristic algorithm

to tackle scalability issues associated with exact solutions. In [106], they address different optimization approaches (E2E latency, provisioning cost, VNF migration, and handovers) subject to several constraints, such as latency, capacity, and path continuity. Additionally, in [14], different VNF scaling strategies (i.e., vertical, horizontal, and hybrid) and their effects on the service provisioning cost are investigated. In this study, 5G core functions are divided into three categories (i.e., stateful, control, and user plane functions). Furthermore, SFCRs are formed by elements for both the user and control planes. However, the authors did not consider certain aspects of SFC placement, such as VNF order or E2E latency requirements.

2.3.3.1 5G UPF Placement

The number of studies related to the 5G UPF found in the literature review is very limited. On the topic, we encounter three main research directions. The first investigates the implementation of UPF functionalities through technologies such as SDN, NFV, and P4, while the second and third research groups focus mainly on designing strategies for placing and selecting UPFs.

Studies such as [107–109] examine the implementation of UPFs through different technologies with the primary goal of improving UPF performance (throughput and response time) and scalability. For example, Costa-Requena et al. [107] propose the deployment of UPF as a separate programmable module integrated with SDN switches to fulfill 5G latency and bandwidth requirements. They investigate the performance of 5G UPFs as an SDN-based solution in a realistic testbed. Specifically, they analyze two deployment scenarios (i.e., cloud and edge). Their results show that an edge deployment is possible without affecting UPF performance and helps reduce network congestion.

In [108], the authors virtualize the UPF using Docker containers and Intel DPDK to provide high-performance packet processing. Fattore et al. [109] propose a light UPF solution to adapt UPF deployment to constrained edge devices (i.e., drones). In this solution, UPF functionality and base stations are co-located with the drones and implemented through open-source software (vector packet processing [110]) on a lightweight, single-board computer. This solution approach reduces user plane delay and backhaul traffic. Other studies, such as [111–113], implement the UPF as a P4 program running on programmable switches.

Fewer studies address the UPF placement problem [11–14]. Li et al. [11] examine the problem of joint placement for edge servers and UPFs. Their main objective is minimizing service latency subject to cost and resource limitations (i.e., UPF and EN capacity). They formulate the problem as an ILP model and show its NP-hardness. Then, they prune the solution space and design a heuristic algorithm to simplify the problem and improve the solution's efficiency. They assign service demands per traffic generators (access nodes), not at PDU session or user levels.

The authors in [12] introduce a framework to place UPFs at edge locations dynamically to minimize the number of hops between user base stations and their assigned anchor UPFs (i.e., latency optimization). They evaluate the latency and execution time of different UPF allocation

algorithms (e.g., random, greedy-based, and k-means). These algorithms are simple solutions that periodically determine UPF placement according to a specific number of UPF instances. Moreover, the authors did not consider UPF capacity constraints, service latency requirements, or more complex aspects associated with placement reconfiguration events, such as numbers of migrated VNFs and reassigned sessions.

Subramanya et al. [13] address the problem of joint user association and SFC placement in a hierarchical MEC environment. They formulate the problem as an ILP model whose optimization objective is minimizing E2E service delay. A heuristic algorithm is proposed to overcome the scalability limitations of the exact solution. This solution maps SFCRs in the network according to latency requirements. Moreover, they apply machine learning (ML) techniques to proactive auto-scaling UPFs. Specifically, two neural network-based models (i.e., a classifier and a regressor) are presented to predict the required number of UPFs according to traffic demands. Nevertheless, no placement constraints associated with UPF requirements or VNF-order are considered since the authors assume that SFCRs are composed of only one UPF.

Harutyunyan et al. [14] study the joint user association and SFC placement in a 5G network. Their primary focus is investigating the effects of VNF scaling decisions (i.e., horizontal, vertical, and hybrid scaling) on the service provisioning cost. To this aim, they propose an ILP model and a heuristic algorithm to minimize service provisioning costs while satisfying users' data rate requirements. The authors represent the user service requests as SFCs formed by one UPF instance.

Table 2.1 summarizes the studies addressing the UPP and their solution approach, optimization objective, and considerations. As seen in this table, the main optimization objective for solving this problem is the minimization of user plane latency. Of the analyzed articles, only [14] considers optimizing deployment costs, measured in terms of VNF scaling and usage of links and gNB nodes. Furthermore, the problem is typically formulated as a mathematical problem, and a heuristic-based solution is presented to overcome the scalability limitations of exact solutions due to the NP-hard nature of the problem [11]. Additionally, none of the articles addresses reconfiguration costs associated with UPF deployment in dynamic environments. Moreover, only studies [13, 14] consider UPFs to be part of the SFC data paths. Nevertheless, both works assume that all UPF functionalities are centralized in a single UPF instance.

Finally, other studies [114–117] propose different solution approaches for selecting the best UPF in the session data paths. For example, Ge et al. [114] introduce an SFC framework for determining the optimal set of UPFs that traffic flows should traverse. They define different flow categories and SFC templates to identify the set of UPFs forming a flow data path. Moreover, through testbed implementation, they demonstrate that UPF deployment methods directly impact SFC performance. Nevertheless, they do not discuss any solution concerning the UPP. In [115], the problem of dynamic selection of UPF through an evolutionary game theory model is addressed. In this study, users attempt to optimize their performance by selecting UPF instances

Table 2.1: Summary of UPF placement solutions.

Paper	Objective	Constraints	Solution approach		Flow handling		SFC
			Exact	Aprox.	Static	Dynamic	
[11]	Min latency	service latency cost EN capacity UPF capacity BS-UPF mapping	x	x	x		
[12]	Min latency	Number of UPFs		x		x	
[13]	Min latency	VNF-node mapping VNF capacity EN capacity gNB capacity path continuity link bandwidth	x	x			x
[14]	Min cost (VNF scaling, link and gNB)	gNB related EN capacity VNF capacity UE-VNF mapping empty VNF link bandwidth path continuity	x	x			x

with lower latency. However, the authors assume that UPFs have already been deployed and that UPF locations do not change over time.

In [116, 117], the authors introduce the concept of anticipatory user plane management to decrease the user plane reconfiguration time during handover. They rely on the predictions of individual user behavior (e.g., target access point) to proactively select and configure the PDU session data paths. Two solutions (i.e., anticipatory iUPF placement and proactive UPF configuration) are provided in [116]. However, these are management mechanisms for selecting iUPF, not for placement optimization. In [117], the authors present an overview of a heuristic approach for iUPF placement optimization. This solution selects the best location by evaluating a cost function that ranks feasible candidates according to 5G use cases (i.e., URLLC and eMBB).

2.4 Placement Optimization and Scheduling Strategies

This section provides an overview of the most prominent placement strategies and reconfiguration mechanisms adopted in the literature for solving the VNFP problem.

2.4.1 Placement Methods

Placement strategies addressing the VNFPC problem can be grouped into two main categories: mathematical modeling (linear and non-linear programming) and approximate algorithms (specific heuristic and metaheuristic). Mathematical programming techniques use mathematical

models to describe the characteristics of the optimal solution for an optimization problem. The problem's solution consists of the optimum variable values that maximize or minimize the objective function and satisfy a specified set of constraints. Linear programming (LP) is an optimization method in which the objective function and constraints are expressed linearly. LP solutions can be placed into three categories according to the domain of the decision variables: LP, ILP, and MILP. In non-linear programming, at least one of the constraints or the objective function represents a non-linear relationship.

From the literature review, we observed that linear programming, especially ILP [25, 46, 55, 61, 62] and MILP [21, 23, 24, 47, 63], is the most commonly used technique. However, only a small subset of studies applies non-linear programming methods [102, 105] to formulate the problem. These models aim to optimize single or multiple criteria, though most approach the problem in a multi-objective fashion. Multi-objective optimization problems typically have conflicting design goals, so optimizing one objective comes at the expense of others. There are various methods to solve MOOP [118], such as weighted sum [48, 52, 68], ϵ-constraint [103, 104], and game theory (e.g., Nash bargaining game) [18].

Although mathematical programming methods provide optimal solutions to the problem, they are characterized by scalability limitations. In other words, they cannot solve the problem for large-scale scenarios in a reasonable time due to the NP-hardness nature of the VNFPC problem. Conversely, metaheuristic and heuristic solutions do not necessarily achieve optimal results but have significantly lower execution times than exact methods. Therefore, a common approach in the literature is the combination of both methods. More specifically, exact solutions are usually used to solve the problem in small scenarios or as benchmarks for assessing the performance of heuristic-based algorithms. Heuristic or metaheuristic methods are the research community's preferred approaches to provide online solutions to the problem.

Approximate algorithms can be categorized into two classifications: specific heuristics and metaheuristics [119]. Heuristic methods are a procedure that determines good or near-optimal solutions to an optimization problem [120] by trial and error in a reasonable time. They are problem dependent since they are designed to solve a particular problem [119]. In contrast, metaheuristics [119, 121] are more general methods that can be applied to many problems. They typically perform better than simple heuristics due to the diversification and intensification of the search space, also referred to as exploration and exploitation. In essence, metaheuristics are strategies that guide the search process, intending to determine near-optimal solutions. They can generally solve optimization problems faster than exact approaches and provide more robust solutions than specific heuristics.

A wide range of metaheuristic algorithms exists, as well as classification categories, such as type of search strategy (local or global), the number of simultaneously improved solutions (single or population-based), and source of inspiration (e.g., human-, nature-, and physics-based). Many metaheuristics are inspired by natural processes, such as evolution (GA) and the process of

annealing a metal (SA), while others, such as Tabu search and VNS, are not. Moreover, they can be classified as simple or classical, improved, hybrid, or adaptive according to their implementation design.

The number of works proposing approximation algorithms to address the VNFPC problem is extensive. Most of these studies design specific algorithms [25, 58, 61, 65, 67, 68, 71, 80] that typically follow greedy-based approaches to solve the problem. A lower percentage adopts metaheuristic solutions. Among these metaheuristics, SA [47], GA [48, 105], ACO [22], TS [67, 76], and VNS [74] are the most frequently used.

2.4.2 Reconfiguration Scheduling Mechanisms

In *event-based mechanisms*, the execution of a reevaluation procedure is initiated as a consequence of an event (e.g., node failure). These mechanisms are reactive by nature and include more specific mechanisms, such as time- and threshold-based reconfigurations.

Time-based mechanisms determine the readjustment of the placement configuration based on time indicators such as periodicity, specific hours of the day, or dates. This strategy is simple and easy to implement since it only needs the configuration of the time condition. However, the execution of these mechanisms does not guarantee a better system performance (e.g., QoS or utilization) since it is unaware of the system status. Moreover, the number of reconfiguration events depends on the specified time condition. For instance, in [12], a time-slotted model is considered to determine the best UPF deployment at each time interval to minimize the user-perceived latency.

In *threshold-based strategies*, reconfiguration events are triggered according to metric values and predefined thresholds (upper or lower), which can be static or variable. Threshold-based solutions are mainly based on metrics related to traffic variations, as well as resource availability [52, 66–68, 82, 93]. Therefore, specific system metrics, such as VNF utilization, may need to be monitored over time.

The authors of [52] optimize the VNF placement in response to new service requests and workload variations (i.e., demand arrival and departure). In [66], the proposed strategy periodically checks VNF utilization rates and releases the ones with values lower than a specified threshold, which can be dynamically adjusted according to load variation. Miotto et al. [82] propose using a VNF CPU threshold to readjust the VNFPC. Similarly, a scaling mechanism based on CPU utilization and used registration capacity per second is proposed in [93]. This solution uses three region zones in the monitored metrics to determine the scaling strategy to be applied. In [67], the SFC readjustment is triggered when the real-time utilization ratio of the servers reaches the lower/upper threshold and lasts for a certain duration. This solution aims to reduce reconfiguration costs due to frequent reconfigurations by verifying that the threshold violation is sustained over time.

In [68], the trigger condition for VNF instance migration is determined based upon network

resource overload (i.e., computing capacity and link bandwidth) and physical node failure. Li et al. [63] introduce a rescheduling scheme to readjust VNF mapping and accommodate incoming services without degrading the QoS of previously mapped SFCs. The VNF mapping and scheduling are readjusted when newly arrived service requests cannot be mapped into the network without guaranteeing their delay requirement. Additionally, the authors of [64, 65] reconfigure the SFC mapping upon user handover to ensure user connectivity.

A recent trend in modern networks is not only to react to changes but also to foresee them to diminish their effects, such as QoS deterioration. The variety of predictive methods and their applicability are substantial. Some of the most popular techniques are stochastic methods (optimal stopping), probabilistic forecasting (Markovian models), and machine learning (neuronal networks). In this study, we refer to the methods under this category as *prediction-based mechanisms*.

The optimal stopping theory (OST) [122] is concerned with the problem of determining the best time to take a given action based on sequentially observed random variables to maximize (minimize) an expected reward (cost). Optimal stopping problems (OSPs) exist in many areas, such as statics, economics, and computing science. The secretary, parking, or house-selling problems are some of the most common OSPs.

The OST has been widely adopted to solve optimization problems [46, 123–125] due to its effectiveness and simplicity. In [46], Cziva et al. propose a dynamic placement scheduler to forecast when the VNF placement needs to be readjusted. They rely on OST principles to dynamically determine the optimal reconfiguration time. Their main objective is to guarantee the established QoS levels (i.e., the cumulative sum of sessions with latency violations), whereas frequent placement recalculations are avoided.

Anagnostopoulos and Kolomvatsos [123] propose a model based on OST to determine the correct time to take a mitigation action in ENs (e.g., upgrade the current services/resources or offload tasks). In this manner, the ENs can adapt their configuration to ensure the desired QoS. In [124], a dynamic service migration strategy is presented to optimize the energy consumption of the MEC platform. To this aim, OST is applied to obtain the optimal migration energy expectation and select the target migration node. Similarly, in [125], Wu et al. use OST to choose the best nodes for the cache placement to maximize energy saving.

Machine learning [126, 127] is another strategy that has attracted much attention for managing resource allocation proactively (e.g., estimating resource demands or determining when the VNFP should be readjusted). ML algorithms can capture hidden data patterns and work quickly in changing network conditions [50]. These algorithms can be classified into three main categories: supervised, unsupervised, and reinforcement learning. Supervised learning solutions learn the relationship between feature data and labeling data, while unsupervised algorithms must find the data structure on their own since no labeling data are provided. Reinforcement learning models learn to take actions through system feedback (reward) in dynamic environments.

The objective in [128] is to optimize MEC resource usage by making virtual infrastructure management decisions (scaling, migrating, or retaining) based on user mobility predictions. In [129], the authors present several classification models based on deep neuronal networks and long short-term memory (LSTM) to accurately forecast the required number of VNF instances to satisfy traffic demands and avoid QoS degradation. Similarly, [130] integrates ML algorithms (i.e., support vector regression [SVR], linear regression [LR], and multi-layer perceptron [MLP]) with VNF placement and scaling solutions to predict VNF-required resources to serve a given traffic load. Subramanya and Riggio [131] apply federated learning techniques to scale VNF instances proactively in an MEC environment according to QoS or cost optimization objectives. To this aim, they model the problem as a time series forecasting problem that predicts the required number of VNF instances to satisfy traffic demands at a given time horizon.

Bendriss et al. [132] design an ML framework for SLA management. They apply MLP and LSTM algorithms to predict violations in SLA (SLO breaches or no SLO breaches). The authors in [133] approach the problem of SLA and SLO violation prediction for a latency-sensitive VNF through a multi-label classification methodology. They divide SLA into a set of SLO categories and apply a deep neural network (MLP)-based multi-label classification to predict SLO breaches associated with an application state.

2.5 Open Issues

Extensive research can be found regarding topics and problems closely related to 5G UPF placement (e.g., SGW and PGW placement or generic VNFPC problems). However, existing solutions still possess several limitations that restrict their applicability for solving the UPP. Some of these limitations and open issues can be summarized as follows:

- Despite the wide range of studies tracking the EPC gateway placement (i.e., SGWs and PGWs), a solution integrating user mobility, latency, and reliability requirements, as well as available resource capacities, is missing.

- Most existing studies addressing the VNFPC for 5G networks assume generic network functions, and only a few consider the 5G architecture proposed by the 3GPP, including its characteristic network elements and their specificities.

- Some UPF characteristics and particularities, such as the possibility of deconstructing UPF functionalities in smaller micro-services and chaining them together, have not yet been investigated in the literature.

- Solutions approaching the VNFPC for non-linear SFC topologies (multi-branch SFCs) in static or dynamic environments are lacking.

- Little or no research has approached the dynamic placement reconfiguration of UPF or its analogous in EPC networks (i.e., SGW and PGW).
- Limited literature has addressed the dynamic VNFPC reconfiguration problem through configuration readjustment solutions (mathematical models, heuristics, or metaheuristics) and scheduling mechanisms to determine the most convenient time to rearrange its configuration.
- Most state-of-the-art solutions decide to rearrange the VNFPC configuration according to VNF resource utilization to select the best scaling mechanism (horizontal, vertical, or hybrid) capable of meeting user traffic demands. However, other criteria, such as the user-perceived delay, must be investigated since resource utilization may not always reflect degradation in the QoS. More specifically, VNF utilization may be under acceptable levels, but users may experience high network response time. This may be caused by an increase in propagation delay due to user mobility and the absence of available VNFs in the new point of attachment. Moreover, most existing solutions are reactive and make this decision based on events (threshold violation or predefined time intervals) or per-user basis in mobile scenarios.
- Most research studies providing VNF scaling solutions do not determine the best placement to deploy new instances when horizontal auto-scaling is applied.
- Despite the plethora of studies proposing reconfiguration solutions for the VNFPC, most focus only on optimizing aspects relevant to service providers, such as node activation, VNF deployment, traffic routing, and VNF migration. User aspects like latency and session reassignment are neglected.

CHAPTER 3

STATIC UPF PLACEMENT

This chapter is based on:

- **I. Leyva-Pupo**, C. Cervelló-Pastor, and A. Llorens-Carrodeguas, "Optimal Placement of User Plane Functions in 5G Networks," in *International Conference on Wired/Wireless Internet Communication (IFIP WWIC'17)*, Bologna, Italy, Jun. 2019, Lecture Notes in Computer Science, vol. 11618, Springer, Cham, 2019.
- **I. Leyva-Pupo**, A. Santoyo-González, and C. Cervelló-Pastor, "A Framework for the Joint Placement of Edge Service Infrastructure and User Plane Functions for 5G," *MDPI Sensors*, vol. 19, no. 18, pp. 3975, 2019.

To address 5G-and-beyond networks' stringent service requirements, such as ultra-low user plane latency and high bandwidth, the placement of applications and network functions at the network edge –mainly user plane functions– is necessary. This approach implies a significant increase in UPF numbers, which are expected to grow 20–30 times their original amount [134]. However, MEC nodes are characterized by limited resources compared to cloud infrastructure, which leads to the necessity of effective placement solutions to help network operators and service providers to make efficient use of their resources when planning the UPF deployment.

In this context, this chapter presents two mathematical models and a heuristic-based algorithm to address the UPP in MEC ecosystems. The envisioned solutions aim to guarantee 5G service demands (i.e., latency and reliability) while reducing expenditures. Additionally, they enable enhanced QoE and extra operational cost savings by optimizing the occurrence of UPF relocations when planning the UPF placement for services with mobility requirements.

The rest of this chapter is structured as follows. Section 3.1 provides insights into the UPP and introduces the network model and notation. Next, Sections 3.2 and 3.3 present two optimal solutions to the problem. Section 3.4 provides a heuristic approach and its complexity analysis. The performance of the proposed solutions is assessed and analyzed in Section 3.5. Finally, Section 3.6 summarizes the main conclusions of this chapter.

3.1 Problem Statement: Static UPF Placement

Increasing traffic demands and the stringent requirements of forthcoming services, such as latency and reliability, require further network transformations. Specifically, by placing network functions (e.g., UPFs) closer to the users at the network edge, network metrics like response time and bandwidth consumption can be significantly reduced. Additionally, service reliability could be enhanced by assigning several instances of the same type (e.g., UPFs) to access nodes, thus providing higher resilience against failures.

However, these deployment approaches imply an increase in the UPF number, which results in higher costs and UPF relocations, which are largely due to user mobility and handover procedures in mobile networks. Concretely, a UPF relocation occurs when a user with an active PDU session registers in a radio access node served by a UPF instance different from the one assigned to its source access node. Frequent and unnecessary relocations can severely impact the overall QoE by introducing additional delays and signaling overhead for bearer establishment during handover procedures [15]. Moreover, relocations may increase operational costs due to extra signaling traffic exchanged among UPFs to maintain session and service continuity. Therefore, efficient UPF placement strategies that satisfy 5G service requirements, such as latency and reliability, while reducing expenditures are important.

3.1.1 Network Model

The 5G network topology is represented as a graph G(N,E), where N denotes the set of network nodes and E the links among them. The network nodes are formed by UPF candidate location (N_c) and access nodes (N_r). The candidate locations may comprise edge servers and data center facilities, while N_r can include fixed and radio access technologies. The capacity of a UPF instance is indicated by C_u, while L_{rc} represents the lowest propagation delay between an ANN $r \in N_r$ and a candidate location $c \in N_c$. L_{req} and V_u denote the maximum permissible propagation latency between an ANN and its assigned UPFs and the minimum number of assigned UPFs required by an access node to satisfy reliability requirements, respectively. Tables 3.1 and 3.2 summarize the sets, parameters, and variables used in the proposed solutions to solve the UPP.

The minimum number of UPFs ($V_u = K_u + 1$) to which an access node must be assigned to

Table 3.1: Used notation for sets and parameters.

Notation	Description
N_r	Set of access nodes
N_c	Set of UPF candidate locations
d_r	Traffic demand of access node $r \in N_r$
α	Maximum percentage of capacity utilization allowed in the main UPFs
C_u	Capacity of a UPF instance
V_u	Minimum number of UPFs required by an access node
K_u	Minimum number of backup UPFs required by an access node ($K_u < V_u$)
L_{rc}	Propagation delay between access node $r \in N_r$ and candidate $c \in N_c$
L_{req}	Propagation delay requirement between access nodes and their assigned UPFs
h_{ij}	Frequency of handovers between access nodes r_i and r_j ($r_i, r_j \in N_r$)
F_c	Cost of deploying and operating a UPF at candidate node $c \in N_c$
F_h	Cost associated with UPF relocations
m	1 if mobility optimization is required

ensure the desired reliability levels ($R[r]$) can be determined upon expression (3.1).

$$R[r] = (1 - p_r) \cdot \left[1 - \prod_{\forall u \in V_u[r]} [1 - (1 - p_u)]\right] \tag{3.1}$$

where p_r represents the failure probability of access node $r \in N_r$ and p_u the UPF failure probability.

Table 3.2: Used notation for binary variables.

Notation	Description
x_c	1 if there is a main UPF placed at candidate node $c \in N_c$
y_c	1 if there is a backup UPF placed at candidate node $c \in N_c$
z_c	1 if backup UPF at node $c \in N_c$, shares its capacity
p_{rc}	1 if access node $r \in N_r$ has a main UPF in node $c \in N_c$
b_{rc}	1 if access node $r \in N_r$ has a backup UPF in node $c \in N_c$
a_{ijc}	1 if access node r_i or $r_j \in N_r$ is assigned to a main UPF in candidate node $c \in N_c$
k_{ijc}	1 if access node r_i or $r_j \in N_r$ is assigned to a backup UPF in node $c \in N_c$
$w_{rcc'}$	1 if access node $r \in N_r$ has assigned a main UPF in node $c \in N_c$ and a backup in node $c' \in N_c$

3.2 Model 1: Optimal Cost and Mobility-aware UPF Placement

The optimal cost- and mobility-aware UPF placement (CMUP) solution aims to minimize deployment and operational costs associated with UPF placement while considering 5G service requirements and limited capacity in the candidate locations (e.g., ENs). This can be achieved by reducing the number of UPF instances and placing them in the optimal locations as determined by activation and operational costs. Additionally, the CMUP model also optimizes operational costs related to UPF relocations by accounting for user mobility effects when forming UPF

service areas. Therefore, considering the notation in Tables 3.1 and 3.2, the CMUP model can be formulated as follows:

$$\text{Min} \sum_{c \in N_c} F_c \cdot (x_c + y_c) + m \cdot \sum_{c \in N_c} \sum_{r_i, r_j \in N_r} F_h \cdot h_{ij} \cdot a_{ijc} \tag{3.2}$$

The first term in the objective function (3.2) indicates the cost associated with the deployment and operation of primary and backup UPF instances (F_c) regarding a given candidate location, while the second is related to relocation costs (F_h) in the main UPFs. The relocation cost component is determined in terms of the handover frequency (h_{ij}) between radio access nodes served by different UPFs. Since not all services have mobility requirements, the proposed model specifies whether the UPF relocation cost is an optimization objective to be considered using the binary indicator m.

To generate feasible solutions to the problem, the following constraints must be satisfied.

A primary or backup UPF can be placed at each candidate location for a given service category, but both types cannot be placed simultaneously. This distinction in the location of primary and backup UPFs enhances the system's availability concerning node failures. Specifically, in the case of a failure in a node hosting a main UPF instance, the location of its backup UPF may remain unaffected, thereby avoiding service affectations. Moreover, this approach also allows for energy saving. Backup UPFs can be instantiated only when failures occur since they do not serve any access node during normal network operation (i.e., no-failure scenarios).

$$x_c + y_c \leq 1 \qquad \forall c \in N_c \tag{3.3}$$

The inequalities (3.4) and (3.5) prevent the assignment of ANNs to the candidate locations where no UPF instance is placed as either the main or backup UPF. In addition, expression (3.6) restricts the assignment of an access node to a specific main UPF when both the node and the UPF are collocated. Specifically, if a primary UPF instance is placed on a candidate node along with an ANN, the UPF must serve the access node. This helps reduce network traffic and response time for users associated with this access node. Otherwise, the ANN can be assigned to a main UPF instantiated in any other location, as indicated by the constraint (3.7).

$$p_{rc} \leq x_c \qquad \forall r \in N_r, \forall c \in N_c \tag{3.4}$$

$$b_{rc} \leq y_c \qquad \forall r \in N_r, \forall c \in N_c \tag{3.5}$$

$$Loc_r = Loc_c \Rightarrow p_{rc} \geq x_c \qquad \forall r \in N_r, \forall c \in N_c \tag{3.6}$$

$$Loc_r \neq Loc_c \Rightarrow p_{rc} \leq x_c \qquad \forall r \in N_r, \forall c \in N_c \tag{3.7}$$

To meet the required service reliability levels, the ANN demands must be served by a minimum number of primary and backup UPFs ($V_u = 1 + K_u$). This allows a service can resist up to a maximum of K_u simultaneous failures in the UPF instances serving an access node, thereby

mitigating the UPF failure effects, such as service interruption and QoS degradation.

$$\sum_{c \in N_c} p_{rc} \geq 1 \qquad \forall r \in N_r \tag{3.8}$$

$$\sum_{c \in N_c} b_{rc} \geq K_u \qquad \forall r \in N_r \tag{3.9}$$

Expression (3.10) ensures the fulfillment of service latency requirements (L_{serv}) by forcing the assignment of access nodes to the UPFs with either main or backup roles that comply with the specified budget of propagation delay (L_{req}) between access nodes and UPF candidate locations (L_{rc}). The value of the L_{req} parameter is determined by considering the round-trip time (RTT) processing delay (L_{proc}) of all the network elements present in the service data path (e.g., access node and UPF), as well as the service latency requirement.

$$2 \cdot L_{rc} \cdot (p_{rc} + b_{rc}) \leq L_{req} \qquad \forall r \in N_r, \forall c \in N_c \tag{3.10}$$

where $L_{req} = L_{serv} - 2 \cdot L_{proc}$.

Equations (3.11) and (3.12) guarantee that the total traffic demand assigned to main and backup UPFs, respectively, does not exceed the UPFs' maximum processing capacity. Furthermore, the α factor further restricts the maximum capacity utilization of the main UPFs to avoid slowing their performance.

$$\sum_{r \in N_r} d_r \cdot p_{rc} \leq \alpha \cdot C_u \qquad \forall c \in N_c \tag{3.11}$$

$$\sum_{r \in N_r} d_r \cdot b_{rc} \leq C_u \qquad \forall c \in N_c \tag{3.12}$$

Expression (3.13) aims to determine the occurrence of UPF relocations due to user handovers between access nodes served by different main UPFs. It describes the relationship between two access nodes with reference to their assignment to a main UPF location. This expression is an additional restriction to the problem to consider when optimizing user mobility effects on the UPF placement (i.e., $m = 1$).

$$a_{ijc} = p_{ic} \oplus p_{jc} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c \tag{3.13}$$

Notice that constraint (3.13) is non-linear. However, it may be expressed in a linear form as follows:

$$a_{ijc} \leq p_{ic} + p_{jc} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c \tag{3.14}$$

$$a_{ijc} \geq p_{ic} - p_{jc} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c \tag{3.15}$$

$$a_{ijc} \geq p_{jc} - p_{ic} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c \tag{3.16}$$

$$a_{ijc} \leq 2 - p_{ic} - p_{jc} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c \tag{3.17}$$

Finally, expression (3.18) indicates the binary nature of the variables x_c, y_c, p_{rc}, b_{rc} and a_{ijc}.

$$x_c, y_c, p_{rc}, b_{rc}, a_{ijc} \in \{0,1\} \qquad \forall r \in N_r, \forall r_i, r_j \in N_r, \forall c \in N_c \tag{3.18}$$

Thereby, the linear form of the CMUP model can be summarized as follows:

$$\text{Min} \sum_{c \in N_c} F_c \cdot (x_c + y_c) + m \cdot \sum_{c \in N_c} \sum_{r_i, r_j \in N_r} F_h \cdot h_{ij} \cdot a_{ijc}$$

s. t.:

$$x_c + y_c \leq 1 \qquad \forall c \in N_c$$

$$p_{rc} \leq x_c \qquad \forall r \in N_r, \forall c \in N_c$$

$$Loc_r = Loc_c \Rightarrow p_{rc} \geq x_c \qquad \forall r \in N_r, \forall c \in N_c$$

$$Loc_r \neq Loc_c \Rightarrow p_{rc} \leq x_c \qquad \forall r \in N_r, \forall c \in N_c$$

$$\sum_{c \in N_c} p_{rc} \geq 1 \qquad \forall r \in N_r$$

$$\sum_{c \in N_c} b_{rc} \geq K_u \qquad \forall r \in N_r$$

$$2 \cdot L_{rc} \cdot (p_{rc} + b_{rc}) \leq L_{req} \qquad \forall r \in N_r, \forall c \in N_c$$

$$\sum_{r \in N_r} d_r \cdot p_{rc} \leq \alpha \cdot C_u \qquad \forall c \in N_c$$

$$\sum_{r \in N_r} d_r \cdot b_{rc} \leq C_u \qquad \forall c \in N_c$$

$$b_{rc} \leq y_c \qquad \forall r \in N_r, \forall c \in N_c$$

$$a_{ijc} \leq p_{ic} + p_{jc} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c$$

$$a_{ijc} \geq p_{ic} - p_{jc} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c$$

$$a_{ijc} \geq p_{jc} - p_{ic} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c$$

$$a_{ijc} \leq 2 - p_{ic} - p_{jc} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c$$

$$x_c, y_c, p_{rc}, b_{rc}, a_{ijc} \in \{0,1\} \qquad \forall r, r_i, r_j \in N_r, \forall c \in N_c$$

3.3 Model 2: Optimal Cost and Mobility-aware UPF Placement with Backup Sharing

The optimal cost and mobility-aware UPF placement with backup sharing (CMUP-BS) model pursues similar optimization objectives to the CMUP; see (3.19). Its main difference from the solution approach presented in Section 3.2 is the extension of the cost component associated with UPF relocations to consider the effects of handovers on backup UPFs. Additionally, it introduces the concept of capacity sharing in the backup UPFs to reduce their required number.

$$\text{Min} \sum_{c \in N_c} F_c \cdot (x_c + y_c) + m \cdot \sum_{c \in N_c} \sum_{r_i, r_j \in N_r} F_h \cdot h_{ij} \cdot (a_{ijc} + k_{ijc}) \tag{3.19}$$

Constraints (3.3) to (3.11), as well as constraint (3.13), are included in the CMUP-BS model. Moreover, an additional set of restrictions mainly related to the possibility of sharing backup UPF capacities must be considered. This set of extra constraints can be defined as follows:

The occurrence of relocations in the backup UPFs is indicated by constraint (3.20). This expression can be linearized by repeating the same procedure used for (3.13).

$$k_{ijc} = b_{ic} \oplus b_{jc} \qquad \forall r_i, r_j \in N_r,\ \forall c \in N_c \tag{3.20}$$

Inequality (3.21) restricts the possibility of capacity sharing to backup UPFs. This capability is recommended only for UPFs operating in backup mode since the main UPF instances must serve all their assigned users during normal network conditions. The main goal of sharing the capacity of backup UPFs is downsizing their number of deployed instances. This downsizing may be possible by sharing the backup capacity among access nodes assigned to different main UPFs since the simultaneous failure of all UPF instances is unlikely. In this regard, expression (3.22) indicates the relationship between primary and backup UPF instances that may serve a given access node.

$$z_c \leq y_c \qquad \forall c \in N_c \tag{3.21}$$

$$w_{rcc'} = p_{rc} \wedge b_{rc'} \qquad \forall c, c' \in N_c, \forall r \in N_r \tag{3.22}$$

Since constraint (3.22) is non-linear, further transformation is required to express it in a linear form. Specifically, it can be replaced with the following linear expressions:

$$w_{rcc'} \leq p_{rc} \qquad \forall c, c' \in N_c, \forall r \in N_r \tag{3.23}$$

$$w_{rcc'} \leq b_{rc'} \qquad \forall c, c' \in N_c, \forall r \in N_r \tag{3.24}$$

$$w_{rcc'} \geq p_{rc} + b_{rc'} - 1 \qquad \forall c, c' \in N_c, \forall r \in N_r \tag{3.25}$$

Vaticinating the combination of UPF instances, either primary or backup, that will fail at a given time, along with the amount of backup capacity required to serve the affected access nodes, is nearly impossible. To overcome this limitation, we made the following assumption: *If a backup UPF shares its capacity, the total demand of its assigned access nodes belonging to the same main UPF cannot exceed the backup capacity divided by the maximum number of failures the system must resist.* Thus, any backup UPF sharing its capacity will be able to serve all the affected access nodes in the case of up to K_u failures in the UPFs. This approach considers the worst-case scenario in which all failed instances are primary UPFs. The overall traffic demand assigned to backup UPFs with capacity sharing will be greater than the backup UPF's maximum available capacity, which allows for reductions in the required number of instances. This consideration will not affect the UPFs' performance since their placement has been planned to serve only a portion of the traffic assigned to a main UPF.

Expressions (3.26) and (3.27) restrict the capacity utilization in backup UPFs with and without capacity sharing, respectively. Constraint (3.26) indicates that when a backup UPF

shares its capacity, the demands of its assigned access nodes mapped to the same main UPF should be less than the backup's capacity divided by the number of UPF failures that the system must resist. In contrast, the overall demands assigned to backup UPFs with no capacity sharing cannot exceed their maximum capacity.

$$z_c = 1 \Rightarrow \sum_{r \in N_r} d_r \cdot w_{rcc'} \leq C_u / K_u \qquad \forall c, c' \in N_c \tag{3.26}$$

$$z_c = 0 \Leftrightarrow \sum_{c \in N_c} \sum_{r \in N_r} d_r \cdot w_{rcc'} \leq C_u \qquad \forall c' \in N_c \tag{3.27}$$

Though constraints (3.26) and (3.27) are non-linear, they can be expressed in a linear form as follows:

$$\sum_{r \in N_r} d_r \cdot w_{rcc'} \leq C_u / K_u + M_1 \cdot (1 - z_c) \qquad \forall c, c' \in N_c \tag{3.28}$$

$$\sum_{c \in N_c} \sum_{r \in N_r} d_r \cdot w_{rcc'} \leq C_u + M_2 \cdot z_c \qquad \forall c' \in N_c \tag{3.29}$$

$$\sum_{c \in N_c} \sum_{r \in N_r} d_r \cdot w_{rcc'} \leq C_u + \varepsilon + M_3 \cdot (1 - z_c) \qquad \forall c' \in N_c \tag{3.30}$$

where M_1, M_2, and M_3 are sufficiently large constants and ε is a small positive tolerance ($\varepsilon > 0$).

Constraint (3.31) stipulates that x_c, y_c, p_{rc}, b_{rc}, a_{ijc} and k_{ijc} are binary variables.

$$x_c, y_c, p_{rc}, b_{rc}, a_{ijc}, k_{ijc} \in \{0, 1\} \qquad \forall r, r_i, r_j \in N_r, \forall c \in N_c \tag{3.31}$$

Finally, the linearized CMUP-BS model can be defined as follows:

$$\text{Min} \sum_{c \in N_c} F_c \cdot (x_c + y_c) + m \cdot \sum_{c \in N_c} \sum_{r_i, r_j \in N_r} F_h \cdot h_{ij} \cdot (a_{ijc} + k_{ijc})$$

s. t.:

$$x_c + y_c \leq 1 \qquad \forall c \in N_c$$

$$p_{rc} \leq x_c \qquad \forall r \in N_r, \forall c \in N_c$$

$$b_{rc} \leq y_c \qquad \forall r \in N_r, \forall c \in N_c$$

$$z_c \leq y_c \qquad \forall c \in N_c$$

$$w_{rcc'} \leq p_{rc} \qquad \forall c, c' \in N_c, \forall r \in N_r$$

$$w_{rcc'} \leq b_{rc'} \qquad \forall c, c' \in N_c, \forall r \in N_r$$

$$w_{rcc'} \geq p_{rc} + b_{rc'} - 1 \qquad \forall c, c' \in N_c, \forall r \in N_r$$

$$Loc_r = Loc_c \Rightarrow p_{rc} \geq x_c \qquad \forall r \in N_r, \forall c \in N_c$$

$$Loc_r \neq Loc_c \Rightarrow p_{rc} \leq x_c \qquad \forall r \in N_r, \forall c \in N_c$$

$$\sum_{c \in N_c} p_{rc} \geq 1 \qquad \forall r \in N_r$$

$$\sum_{c \in N_c} b_{rc} \geq K_u \qquad \forall r \in N_r$$

$$2 \cdot L_{rc} \cdot (p_{rc} + b_{rc}) \leq L_{req} \qquad \forall r \in N_r, \forall c \in N_c$$

$$\sum_{r \in N_r} d_r \cdot p_{rc} \leq \alpha \cdot C_u \qquad \forall c \in N_c$$

$$\sum_{r \in N_r} d_r \cdot w_{rcc'} \leq C_u / K_u + M_1 \cdot (1 - z_c) \qquad \forall c, c' \in N_c$$

$$\sum_{c \in N_c} \sum_{r \in N_r} d_r \cdot w_{rcc'} \leq C_u + M_2 \cdot z_c \qquad \forall c' \in N_c$$

$$\sum_{c \in N_c} \sum_{r \in N_r} d_r \cdot w_{rcc'} \leq C_u + \varepsilon + M_3 \cdot (1 - z_c) \qquad \forall c' \in N_c$$

$$a_{ijc} \leq p_{ic} + p_{jc} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c$$

$$a_{ijc} \geq p_{ic} - p_{jc} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c$$

$$a_{ijc} \geq p_{jc} - p_{ic} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c$$

$$a_{ijc} \leq 2 - p_{ic} - p_{jc} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c$$

$$k_{ijc} \leq b_{ic} + b_{jc} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c$$

$$k_{ijc} \geq b_{ic} - b_{jc} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c$$

$$k_{ijc} \geq b_{jc} - b_{ic} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c$$

$$k_{ijc} \leq 2 - b_{ic} - b_{jc} \qquad \forall r_i, r_j \in N_r, \forall c \in N_c$$

$$x_c, y_c, p_{rc}, b_{rc}, a_{ijc}, k_{ijc} \in \{0, 1\} \qquad \forall r \in N_r, \forall r_i, r_j \in N_r, \forall c \in N_c$$

3.4 Heuristic: Near-Optimal UPF Placement

The CMUP and CMUP-BS solutions for no mobility considerations can be seen as variants of the facility location problem (FLP) [135, 136] and the resilient controller placement problem (RCPP) [137–139], two well known NP-hard problems. Additionally, when minimizing the effects of user mobility on UPF relocations is an optimization objective, the proposed models can be expressed as a combination of the previous models (i.e., FLP and RCPP) and the location area planning problem [15, 140], which is also NP-hard. Thus, both CMUP and CMUP-BS models are NP-hard in either variant (with and without user mobility considerations).

Exact solutions to NP-hard problems, such as ILP models, may require excessive computation time and resources. Moreover, optimal solutions in large-scale scenarios in which the number of possible combinations is significantly high may be impossible. To cope with this limitation, we devise a low-complexity heuristic, referred to as near-optimal UPF placement (NOUP), capable of determining efficient solutions to the problem in polynomial time.

The proposed heuristic aims to determine the best candidate locations and service areas to attend to the service demands of a given set of access nodes. These demands represent services with similar placement requirements, such as latency, reliability, and user mobility.

Since the access nodes may require the assignment of multiple UPF instances to guarantee certain reliability levels, a sequential assignment is adopted. Concretely, UPF levels are placed according to their role (primary and backup), starting with the highest-priority level (main) and ending with the lowest-priority level (last backup level). This ensures that the best locations always belong to UPFs with higher roles in the service. The pseudo-code of the NOUP solution is depicted in Algorithm 1.

Algorithm 1: Near-Optimal UPF Placement (NOUP)

Input: $N_r, N_c, N_{r-c}, N_{c-r}, C_{u_{max}}, D_r, K_u, H_{ij}$, m
Output: $S_u, S_{sa}, S_{r_{reject}}$

$S_u, S_{sa}, S_{r_{reject}} \leftarrow \emptyset$ // Initialize output variables
$flag\_main \leftarrow$ True // Place a main UPF level
forall *l in range*$(K_u + 1)$ **do**
  $S_u^l, S_{sa}^l, S_{r_{reject}}^l \leftarrow \emptyset$
  $S_r^l \leftarrow$ Set of access nodes that require a UPF at level l
  $N_c^l \leftarrow$ Set of available candidate nodes at level l
  $N_{c-r}^l \leftarrow$ Set of available candidates per access node at level l
  while $S_r^l \neq \emptyset$ **do**
    $score_{best} \leftarrow 0$
    forall $c \in N_c^l$ **do**
      $S_{rc}^l \leftarrow$ Unassigned access nodes near c ($N_{r-c}^l[c]$)
      $S_{sa_c}^l, S_{r_{rejected}}^c \leftarrow$ **Procedure 1** // Form service area for candidate c ($S_{sa_c}^l$)
      if $S_{sa_c}^l \neq \emptyset$ **then**
        $score_c \leftarrow$ Evaluate candidate c
        if $score_c > score_{best}$ **then**
          $score_{best} \leftarrow score_c$ // Update best candidate score
          $c_{best} \leftarrow c$ // Update best candidate
          $S_{sa_{best}}^l \leftarrow S_{sa_c}^l$ // Update best service area
      else
        $N_c^l \leftarrow N_c^l - c$ // Remove candidate with empty SA
      $S_{r_{reject}}^l \leftarrow S_{r_{reject}}^l + S_{r_{rejected}}^c$
    if $score_{best} \neq 0$ **then**
      Update $S_u^l, S_{sa}^l, S_r^l, N_c^l, N_{c-r}^l, N_{r-c}^l$
    else
      $S_{r_{reject}}^l \leftarrow S_{r_{reject}}^l + S_r^l$
      break
  $S_u^l, S_{sa}^l, S_{r_{reject}}^l, N_c^l \leftarrow$ **Procedure 2** // Improve placement solution at level l
  Update $S_u, S_{sa}, S_{r_{reject}}$
  if $l = 0$ **then**
    $N_c \leftarrow N_c - N_c^l$ // Remove candidate with main UPFs
    $flag\_main \leftarrow$ False // Start placing backup UPF levels

As inputs, the NOUP strategy takes the network topology with the sets of UPF possible locations (N_c) and access nodes (N_r); the UPF maximum processing capacity ($C_{u_{max}}$) and service requirements, such as traffic processing demand per access node ($d_r \in D_r$), number of UPF levels

to ensure the desired reliability ($K_u + 1$); and the frequency of handover between access nodes (H_{ij}). The frequency of handover is only needed when considering the effects of user mobility on UPF placement ($m = 1$). Additionally, the sets of candidate locations per access node (N_{c-r}) and access nodes near each candidate (N_{r-c}) must be specified. These sets may be previously computed using service latency requirements. This algorithm outputs the best UPF locations (S_u) and service areas (S_{sa}) along with the set of rejected access nodes ($S_{r_{reject}}$) per UPF level.

Algorithm 1 begins by creating the output variables where the placement configuration will be saved. Then, the Boolean $flag\_main$ is set to true to indicate that the procedure begins with the deployment of main UPFs (line: 2). Afterward, the algorithm begins an iterative process to determine the best candidates and service areas for each level of UPF (lines: 3–28). Inside this loop, three temporal variables corresponding to each output element in the current UPF level are initialized as empty sets. Next, access nodes requiring the assignment of UPFs at the specified level (S_r^l) are selected, along with the set of available candidates (N_c^l) for the deployment of the UPFs (lines: 5–6). After this preparatory stage, an assignment process determines the best combination of UPF locations and their service areas. This process is repeated while there are unassigned access nodes and available candidates (lines: 8–26). At each iteration, all available candidates are analyzed by formulating their potential service areas to choose the best candidate (lines: 10–20).

The UPF service areas are created by calling Procedure 1 (line: 12) upon the unassigned access nodes near each candidate location that requires a UPF at the specified level (S_r^l). The first step of this procedure is to initialize the output variables $S_{sa_c}^l$ and $S_{r_{rejected}}^c$ where the access nodes forming the candidate service area and the critical nodes rejected by the candidate during the procedure will be stored. Then, the unassigned ANNs are sorted according to their proximity to the candidate location. Next, the procedure verifies whether the candidate under analysis corresponds to a main UPF co-located with an unassigned access node (line: 3). This ensures the satisfaction of constraint (3.6) when deploying a level of main UPFs. If this condition is met, the co-located access node is assigned to the candidate service area, and the set of unassigned access nodes is updated along with the candidate's available capacity (lines: 3–5). The UPF available capacity for assigning the co-located node is not verified since we assume that UPFs have enough capacity to serve the access node's maximum demand. Next, the procedure searches for critical ANNs in the set of unassigned access nodes near c (line: 6). Here, the term *critical access nodes* is used to identify those access nodes that have only one available candidate for deploying their UPFs.

After determining the set of critical access nodes ($S_{r_{critical}}$), Procedure 1 initiates an iterative process to assign the access nodes to the candidate service area. This process is executed while there are unassigned access nodes in the neighborhood of the candidate ($S_{rc}^l \neq \emptyset$) and the candidate has enough capacity available to serve its less loaded neighbor ANN (lines: 7–30). Inside this loop, critical access nodes are prioritized to reduce the likelihood of leaving them

Procedure 1: UPF Service Area Creator

1 $S^l_{sa_c}, S^c_{r_{rejected}} \leftarrow \emptyset$
2 Sort unassigned access nodes (S^l_{rc}) by proximity to c
3 **if** $flag\_main$ ***and*** $loc(c) = loc(S^l_{rc}[0])$ **then** // Check if c and $S^l_{rc}[0]$ are co-located
4 $S^l_{sa_c} \leftarrow S^l_{sa_c} + S^l_{rc}[0]$ // Assign the co-located access node to c service area
5 Update S^l_{rc}, C_{u_c}
6 $S_{r_{critical}} \leftarrow$ Set of critical access nodes in S^l_{rc}
7 **while** $S^l_{rc} \neq \emptyset$ ***and*** $C_{u_c} \geq min(D_{rc})$ **do** // D_{rc}: Traffic demands of $r \in S^l_{rc}$
8 $S_{rc} \leftarrow S^l_{rc}$
9 **if** $S_{r_{critical}} \neq \emptyset$ **then**
10 **if** $\sum_{r \in S_{r_{critical}}} d_r \leq C_{u_c}$ **then** // Check if all the $r \in S^l_{rc}$ can be assigned to c
11 $S^l_{sa_c} \leftarrow S^l_{sa_c} + S_{r_{critical}}$ // Assign all the critical access nodes to c
12 Update $S^l_{rc}, S_{r_{critical}}, C_{u_c}$
13 **else** $S_{rc} \leftarrow S_{r_{critical}}$
// Select best access node (r_{best}) to be assigned to c service area
14 **if** $flag\_main$ ***and*** $m = 1$ **then**
15 $r_{best} \leftarrow$ Select $r \in S_{rc}$ with the highest handover frequency w.r.t. c current service area
16 **else**
17 $r_{best} \leftarrow$ Select the access node $r \in S_{rc}$ closest to c
18 **if** $d_{r_{best}} \leq C_{u_c}$ **then** // Check selected access node for assignment
19 $S_{r_{affected}} \leftarrow \emptyset$
20 **if** $flag\_main$ ***and*** $loc(r_{best}) \in loc(N^l_c)$ **then**
21 $S_{r_{affected}} \leftarrow$ Search for affected access nodes in S^l_r // Verify location constraint
22 **if** $S_{r_{affected}} = \emptyset$ **then** // If no access node is affected
23 $S^l_{sa_c} \leftarrow S^l_{sa_c} + r_{best}$ // Assign the r_{best} to c service area
24 $C_{u_c} \leftarrow C_{u_c} - d_{r_{best}}$ // Update c available capacity
25 **if** $S_{r_{critical}} \neq \emptyset$ **then**
26 $S_{r_{critical}} \leftarrow S_{r_{critical}} - r_{best}$
27 **else if** $S_{r_{critical}} \neq \emptyset$ **then**
28 $S_{r_{critical}} \leftarrow S_{r_{critical}} - r_{best}$
29 $S^c_{r_{rejected}} \leftarrow S^l_{r_{rejected}} + r_{best}$
30 $S^l_{rc} \leftarrow S^l_{rc} - r_{best}$ // Remove r_{best} from the set of unassigned access nodes
31 **return** $S^l_{sa_c}, S^c_{r_{rejected}}$

unattended (i.e., without a UPF) (lines: 9–13).

The first step when dealing with a set of critical access nodes is to determine whether the candidate has enough available capacity to cover the demands of the entire set (line: 10). If all the critical nodes can be served, they are added to the candidate service area and the sets of unassigned and critical ANNs (S^l_{rc} and $S_{r_{critical}}$) are updated by removing these nodes. Additionally, the amount of available capacity in the candidate is readjusted (lines:11–12). Otherwise, further analysis is required to determine which critical node is more convenient to map to the candidate service area. To do so, the unassigned access nodes in S_{rc} are replaced with the critical nodes (line: 13). The S_{rc} is an auxiliary set used to store the unassigned access nodes under analysis at each iteration stage. When there are no critical access nodes, the execution of

lines 10–13 is omitted, and the entire set of unassigned nodes is considered for the selection of the best node.

The best access node to be assigned (r_{best}) is determined in the remaining steps of the procedure (lines: 14–30). At each iteration, the best node is selected from the set S_{rc} according to the UPF role (main or backup) and mobility considerations, which are indicated by the parameters *flag_main* and m, respectively. In particular, when the service area under analysis corresponds to a main UPF with mobility requirements ($m = 1$), the best ANN is the unassigned node ($r \in S_{rc}$) that has the highest overall handover frequency concerning those ANN currently forming the candidate service area. Otherwise, the closest ANN to the candidate is elected as the best option (lines: 17).

Once r_{best} has been identified, the procedure checks whether the candidate has enough available capacity to serve the node service demand (line: 18). If this is the case, another verification is required before deciding whether the selected node can be assigned to the service area (lines: 20–21). This action is related to constraint (3.6); thus, it is only required when planning the service area of a main UPF. Specifically, the procedure verifies whether the assignment decision affects other unassigned access nodes ($r \in S_r^l$). Access nodes are affected when r_{best} is co-located with a candidate that is the only option for deploying their main UPF. Consequently, the selected best node could be mapped to the candidate, provided that no affected access node is detected (lines: 22–24).

When an assignment occurs, the candidate's service area and available capacity are readjusted. Additionally, the set of critical nodes ($S_{r_{critical}}$) is updated every time a critical access node is assigned. If a critical node cannot be served by the candidate's available capacity, it is classified as rejected and is transferred from the set of critical nodes to the output set $S^c_{r_{rejected}}$ (lines: 27–29). Last, the set S^l_{rc} is updated by removing the selected access node (line: 30) regardless of whether it was assigned.

After executing Procedure 1, Algorithm 1 proceeds to inspect the candidate service area (lines: 13–20). If the service area is not empty, the candidate is evaluated by considering utilization and maximum latency metrics. Moreover, relocation avoidance is analyzed for placement with mobility consideration. This evaluation process returns a candidate score ($score_c$) and compares it with the best score found thus far. If a more optimal location is detected, it is saved along with its service area configuration, and the best score is modified (lines: 15–18). In contrast, the candidate is removed from the available candidates when it cannot serve any unassigned access node (line: 20), and the set of rejected access nodes is automatically adjusted in the next step.

Once all the candidates have been evaluated, the algorithm determines whether a best candidate is available (lines: 22–26). When user mobility is considered, the best candidate has the highest number of assigned access nodes and avoids the highest number of UPF relocations. Then, the involved sets are updated (i.e., S^l_u, S^l_{sa}, S^l_r, N^l_c, N^l_{c-r}, and N^l_{r-c}) to reflect the selected placement configuration. Alternatively, the set of rejected access nodes at the current level is

extended to include the current set of unassigned access nodes, and the assignment process is interrupted due to the absence of feasible candidates (lines: 25–26).

At the end of each assignment process, an improvement process occurs (line: 27) through the execution of Procedure 2. The main goal of this procedure is enhancing the quality of the current placement configuration by finding candidates for the set of rejected access nodes and reducing the number of UPFs and UPF relocations for placement with mobility requirements. Following these design objectives, the procedure's pseudo-code is structured in two phases. First, it begins with the mapping of the rejected access nodes since guaranteeing service requirements is critical (lines: 1–14). Second, the phase related to cost reductions is executed as the mapping of the rejected ANNs may produce variations in the number of UPFs (lines: 15–36).

The mapping phase, which occurs in Procedure 2, attempts to reduce the number of rejected access nodes by reassigning already assigned access nodes with the hope of enabling enough capacity in their candidates to serve their demands. To accomplish this, the procedure begins by sorting the unassigned access nodes in ascending order according to their available candidates at the beginning of the assignment procedure. This increases the chance of finding a candidate for the most critical nodes. Then, the ANNs assigned to its potential candidates for each rejected access node ($r \in S_{r_{rejected}}$) that could release enough resources to serve it are selected (line: 5).

These ANNs are sorted by their number of candidates (descending) and traffic demands (ascending) to speed up the search process. For each selected node, the capacity available in its near candidates is analyzed to determine whether remapping is possible (lines: 2–13). If it is, the best candidate is selected to reassign the node under analysis, and the rejected node is mapped to the released candidate. The corresponding sets are then updated, and the search for a candidate for the unassigned node ends (lines: 9–14). Otherwise, the process continues until all the possibilities of reassigning a mapped access node are exhausted. If no valid candidates are found, further analysis of the situation by the network operators or service providers is required. For instance, the rejected nodes could be assigned to a UPF located in a further candidate by relaxing the latency requirement, or additional infrastructure could be deployed. However, these approaches imply either QoS degradation or additional expenditures.

The first action in the second phase of Procedure 2 is computing the overall available capacity ($C^l_{u_T}$) at the current UPF level (line: 15) to determine whether the number of UPFs can be reduced. In particular, some UPFs could be discarded when the overall available capacity is greater than the UPF maximum capacity (lines: 16–24). In this case, the UPFs are sorted in descending order based on their available capacity (line: 17) so that those with the lowest utilization have higher priority during the removal analysis. For each UPF, the procedure verifies whether all the UPF's assigned access nodes can be served by other UPFs (lines: 18–24). This allows the UPF to be removed, as none of its assigned access nodes will be unattended. When this condition is satisfied, the access nodes are reassigned, and the UPF under analysis is removed from the set of UPFs (lines: 22–24). The sets of available candidates and UPFs at the current level (N^l_c and S^l_u), as

Procedure 2: UPF Placement Improvement

```
   // Phase 1: Mapping rejected access nodes
 1 Sort r ∈ S^l_{r_rejected} by their initial number of available candidates in ascending order
 2 forall r ∈ S^l_{r_rejected} do
 3     N^r_c ← Set of candidates near r (N^l_{c−r})
 4     forall c ∈ N^r_c do
 5         S^c_r ← Set of access nodes (r_c) in S^l_{sa_c} with d_{r_c} ≥ d_r and more than one candidate
 6         Sort r_c ∈ S^c_r according to their number of candidates and demands
 7         forall r_c ∈ S^c_r do
 8             N^{r_c}_c ← Search for available candidates to remap r_c
 9             if N^{r_c}_c ≠ ∅ then
10                 c_best ← Select best candidate (the most loaded) to reassign r_c
11                 Remap r_c to c_r and map r to c
12                 Update S^l_{r_rejected}, S^l_sa, S^l_u,..
13                 break
14         if N^{r_c}_c ≠ ∅ then break
   // Phase 2.1: Reducing UPF costs
15 C^l_{u_T} ← Compute the overall UPF available capacity
16 if C^l_{u_T} ≥ C_u then
17     Sort u ∈ S^l_u according to their available capacity in descending order
18     forall u ∈ S^l_u do
19         forall r ∈ S_{sa_u} do
20             u_target ← Find a candidate to remap r
21             if u_target = ∅ then break
22         if u_target ≠ ∅ then
23             Reassign ∀r ∈ S_{sa_u} and remove u
24             Update N^l_c, S^l_u, S^l_sa, C_u
   // Phase 2.2: Reducing relocation costs
25 if flag_main and m=1 then
26     S_r ← Set of assigned access nodes at the current level
27     while True do
28         score_{r_best} ← ∞
29         forall r ∈ S_r do
30             u_best, score_r ← Find a candidate with lower relocations
31             if score_r < score_{r_best} then
32                 Update score_{r_best}, r_best, u_best
33         if score_{r_best} ≠ ∞ then
34             Reassign r_best to u_best
35             Update S^l_sa, C_u
36         else break
37 return S^l_u, S^l_sa, S^l_{r_reject}, N^l_c
```

well as UPF service areas (S^l_{sa}) and available capacity are then updated (line: 24).

The final part of the improvement procedure is executed when a reduction in the number of relocations among main UPFs is desired (lines: 25–36). To this aim, the mapped access nodes are analyzed to find better UPFs in terms of relocation avoidance. At each iteration of the main loop,

the best candidate for each ANN is determined, and the overall number of relocations that the remap will produce is computed (lines: 28–32). Once all access nodes have been assessed, the remap with the lowest relocations is applied if any (lines: 33–35). The previous step implies a readjustment of the UPF service areas and available capacities. This iterative process is repeated while improvements in the number of relocations are detected.

Procedure 2 could be extended to include more improvement actions, such as latency reduction and balanced load distribution among UPFs. It returns the final sets of UPF locations, UPF service areas, rejected access nodes, and available candidates (line: 27 in Algorithm 1). This information is then used by Algorithm 1 to update the output variables (line: 28). Finally, the indicator $flag\_main$ is modified to indicate that the remaining levels correspond to backup UPFs (line: 31). The main procedure of the algorithm is repeated until the placement of all UPF levels has been analyzed (lines: 3–28).

3.4.1 Complexity Analysis

This subsection addresses the time complexity of the NOUP algorithm. The overall complexity of Algorithm 1 is determined by the loop in lines 3–31, which will be executed L times, where L denotes the required number of UPF levels (i.e., $L = V_u = K_u + 1$). The complexity associated with the iterative processes inside this loop is given by the assignment procedure (lines: 8–26), arbitrarily referred to as W, the number of candidate locations (C) evaluated in its innermost loop; and the improvement procedure (P_2), which is executed at the end of each assignment process. Another significant time-consuming aspect of the candidate evaluation process is the creation of the service areas in Procedure 1, which is denoted as P_1. W cannot be found beforehand since it depends on several factors, such as the number of assigned nodes at each iteration, their traffic demands, and UPF maximum capacity. As a result, a first overview of the total time complexity of NOUP can be expressed as $\mathcal{O}(L \cdot (W \cdot C \cdot P_1 + P_2))$.

The time complexity of Procedure 1 is linked to the sorting process in line 2 and the iterative process concerning the assignment of the best ANN to the candidate service area (lines: 7–30). Assuming the worst-case scenario in which the entire set of access nodes ($S_r = A$) forms the candidate neighboring area ($S_{rc}^l = A$), the time complexity of the sorting process is $\mathcal{O}(A \cdot \log A)$, in which the while procedure is executed at most A times. Generally, $S_{rc}^l << S_r$ and this iterative procedure is typically interrupted due to capacity limitations in the candidate. Moreover, the size of the set of unassigned access nodes is reduced with the best candidate selection in the assignment process. Inside the while loop, different processes are triggered to determine the best access node according to UPF roles and mobility considerations; these determine the complexity of the procedure. The most demanding process is that which occurs for a main level of UPFs with mobility requirements; this process has a complexity of $\mathcal{O}(A \cdot S + A)$. In this expression, the first term corresponds to the process of determining the ANN with the highest handover frequency concerning the candidate service area (S). A represents the complexity of determining the set

of affected access nodes. Therefore, in the worst-case scenario, the computation complexity of Procedure 1 is $\mathcal{O}(A \cdot \log A + A \cdot (A \cdot S + A)) = \mathcal{O}(A^2 \cdot S)$.

Similarly, Procedure 2 has the worst time complexity during the placement of primary UPFs with mobility considerations. In this case, its total time complexity is $\mathcal{O}(R \cdot C \cdot S \cdot (\log S + C) + U \cdot (\log U + S) + T \cdot A \cdot U \cdot S)$. The first term derives from the mapping of rejected access nodes (phase 1), while the second is derived from reducing the number of deployed UPFs. The execution of these processes depends on the presence of unassigned access nodes ($R << A$) at the end of an assignment procedure and on the possibility of removing UPF instances (U), respectively. Additionally, the $T \cdot A \cdot U \cdot S$ term corresponds to the UPF relocation reductions where T indicates the time complexity of the iterative process (lines: 27–36). Thus, the maximum run time of Procedure 2 can be defined as $\mathcal{O}(R \cdot C \cdot S \cdot (\log S + C) + T \cdot A \cdot U \cdot S) = \mathcal{O}(S \cdot (R \cdot C \cdot (\log S + C) + T \cdot A \cdot U)$. Table 3.3 showcases the computational complexity of these procedures according to the UPF role and mobility considerations.

Table 3.3: Time complexity of the procedures.

Flag_main	**m**	**Procedures**	
		Procedure 1 (P_1)	**Procedure 2** (P_2)
False	-	$\mathcal{O}(A \cdot \log A + A) = \mathcal{O}(A \cdot \log A)$	$\mathcal{O}(R \cdot C \cdot S \cdot (\log S + C) + U \cdot (\log U + S)$
True	0	$\mathcal{O}(A \cdot \log A + A^2) = \mathcal{O}(A^2)$	$\mathcal{O}(R \cdot C \cdot S \cdot (\log S + C) + U \cdot (\log U + S)$
True	1	$\mathcal{O}(A \cdot (\log A + A \cdot S)) = \mathcal{O}(A^2 \cdot S)$	$\mathcal{O}(R \cdot C \cdot S \cdot (\log S + C) + U \cdot (\log U + S) + T \cdot A \cdot U \cdot S)$

Globally, the computational complexity of the proposed heuristic can be specified as $\mathcal{O}(W \cdot C \cdot (A \cdot \log A \cdot L + A^2 \cdot (K_u + S^m)) + L \cdot R \cdot C \cdot S \cdot (\log S + C) + m \cdot T \cdot A \cdot U \cdot S)$ where m is a binary indicator used to represent the computational time associated with mobility considerations. The first term of this expression is derived from the assignment process and the service area creation procedure, while the other two represent the maximum run time of the improvement procedure. This expression can be simplified by determining its dominant terms and considering $U \cdot S \approx A$. Thus, overall complexity of NOUP is in the order of $\mathcal{O}(W \cdot C \cdot A^2 \cdot (K_u + S^m) + L \cdot R \cdot C \cdot S \cdot (\log S + C) + m \cdot T \cdot A^2)$ = $\mathcal{O}(W \cdot C \cdot A^2 \cdot (K_u + S^m) + L \cdot R \cdot C \cdot S \cdot (\log S + C))$. Given that, generally, the size of the sets of unassigned access nodes (R), UPF service area (S), and candidate locations is considerably smaller than the overall set of access nodes ($R, S, C << A$), the worst time complexity of NOUP can be defined as $\mathcal{O}(W \cdot C \cdot A^2 \cdot (K_u + S^m))$.

3.5 Evaluation and Results

In this section, we describe the experiments conducted to evaluate the performance of the devised solutions. Then, we analyze the results.

3.5.1 Simulation Setup

To evaluate the performance of the proposed solutions, we generated a map grid of 10,000 km^2, with fixed and radio access nodes randomly deployed to emulate urban and rural regions (see Fig. 3.1). Two urban areas referred to as City_1 and City_2 were considered, while the rest of the territory represented rural zones. In the cities, the radio access nodes represented centralized BBUs with a maximum coverage radius of 3 km, whereas, in the rural areas, they were distributed with the RRHs and had coverage radii of 10 and 20 km. The frequency of handovers and traffic demands was measured at the BBU level for the urban scenarios. According to their underlying user density (region), the access nodes had different traffic demands ranging from 0–1 Tbps. These traffic demands were generated by considering several services with different requirements in terms of latency, reliability, bandwidth, and user mobility [4, 141]. Table 3.4 summarizes the requirements of the use cases.

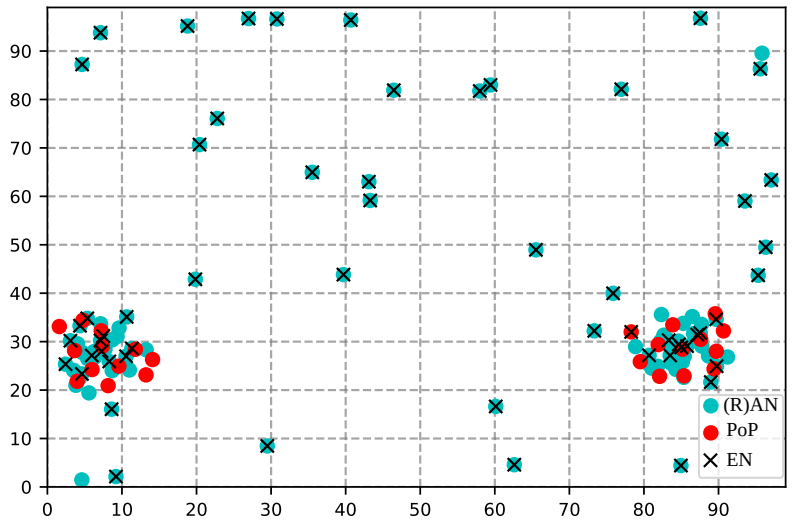


Figure 3.1: Map grid representing nodes distribution per type.

We classified the aforementioned services into two categories according to latency and reliability requirements: high-requirement services (i.e., automated factories, massive IoT, and cooperative sensing) and low-requirement services (i.e., traffic efficiency, 50 Mbps everywhere, and home and office). Thus, the categories presented services with different mobility requirements, allowing us to assess the performance of the proposed solutions under similar placement conditions for UPF placement with and without mobility considerations.

For the first group, 1 ms in the user plane delay and one backup UPF were assumed, while the latency requirement was fixed to 5 ms for the low-demand category, and no backup was needed. To ensure service delay below 1 ms in the user plane, the overall latency RTT in the user-ANN

Table 3.4: 5G service requirements.

Service	**Latency (ms)**	**Data rate (Mbps)**		**Density (users/km^2)**		**Reliability (%)**	**Mobility (m)**[1]
		Rural	**Urban**	**Rural**	**Urban**		
Automated factories	≤ 1	1		10^4	0	99.999	0
massive IoT	≤ 1	1		10^3	10^4	99.999	0/1
Cooperative sensing	≤ 1	5		10	10^2	99.999	1
Traffic efficiency	≤ 5	25		5	50	90	1
50 Mbps everywhere	≤ 10	50		50	400	90	1
Home and office	≤ 10	50	300	10^2	10^3	90	0

[1] Binary indicator for mobility requirements: $m = 1$ service with mobile users, 0 otherwise.

segment had to be less than 0.5 ms [142]. Considering that UPFs and DNs are co-located with a maximum overall processing time of 0.3 ms, a latency budget (L_{req}) of 0.2 ms RTT was left for the segment ANN-UPF. For the low-demand services, the L_{req} parameter was relaxed to 1 ms. These constraints for latency propagation were translated into Euclidean distances considering the propagation time of direct links between any ANN-EN pair and optical fiber as the underlying transport layer (5 $\mu s/km$) [19, 143]. Furthermore, the minimum number of UPFs to which a given access node must be assigned ($V_u = K_u + 1$) to ensure the desired reliability levels was determined upon (3.1) by considering $p_r = 10^{-6}$ and $p_u = 10^{-4}$. Parameter values related to the UPF placement are provided in Table 3.5.

Table 3.5: Simulation parameters for the UPP.

Notation	**Description**	**Value**
L_{req}	Propagation delay requirement (ms)	[0.2, 1]
C_c	Resource capacity of a server (Tbps)	2.5
C_u	Resource capacity of a UPFs (Tbps)	[0.5-2.5]
d_r	RTT overall delay in the (R)AN (ms)	0.5
d_u	Processing time of UPFs (ms)	0.1
d_{DN}	Processing time of DN (ms)	0.1
-	Propagation delay in optical links (μs/km)	5
p_r	Failure probability of access nodes	10^{-6}
p_u	Failure probability of UPFs	10^{-4}

For the UPF placement, a set of candidate locations formed by the point of presence (PoP) and ENs was considered. The number of PoPs for the cities was estimated based on real data[1] from the PoPs of internet service providers operating in Spain. Additionally, the number and location of the ENs required to cover the access node demands were determined by running a hybrid SA algorithm [144]. To this aim, the access nodes and their underlying demands were simplified and modeled as traffic generators [144]. These generators were considered with the PoPs as EN potential locations. Table 3.6 depicts some characteristics of the scenarios mentioned above, such

[1] https://www2.telegeography.com/en/globalcomms-database-service

as the numbers of candidate and access nodes per type as well as overall traffic demands per service category.

Table 3.6: Network nodes and traffic distribution per region.

Region	**Candidate nodes**		**Access nodes**		**Total demands (Tbps)**	
	EN	**PoP**	**Radio**	**Fixed**	**Category 1**	**Category 2**
City_1	13	12	10	22	2.67	17.93
City_2	12	12	11	21	2.34	14.62
Rural	33	0	16	20	6.34	15.66

We coded all solutions (e.g., exact and heuristics) proposed in this thesis using Python programming language 3.7. The mathematical models were implemented in the Python-based package Pyomo [145], and Gurobi [146] was selected as its underlying solver. All experiments were performed on a computer with 64 GB of RAM and a 3.30 GHz Intel Core-i9 processor.

3.5.2 Models' Performance

This subsection discusses the performance of the exact solutions envisioned for the UPP. We selected the urban City_1 scenario for high-requirement services since it had more candidate locations and traffic demands than City_2. Additionally, in this region, the effects of user mobility on UPF relocations could be better appreciated because of the higher difference in the numbers of radio and fixed access nodes (see Table 3.6) and smaller coverage radius than in the rural zones.

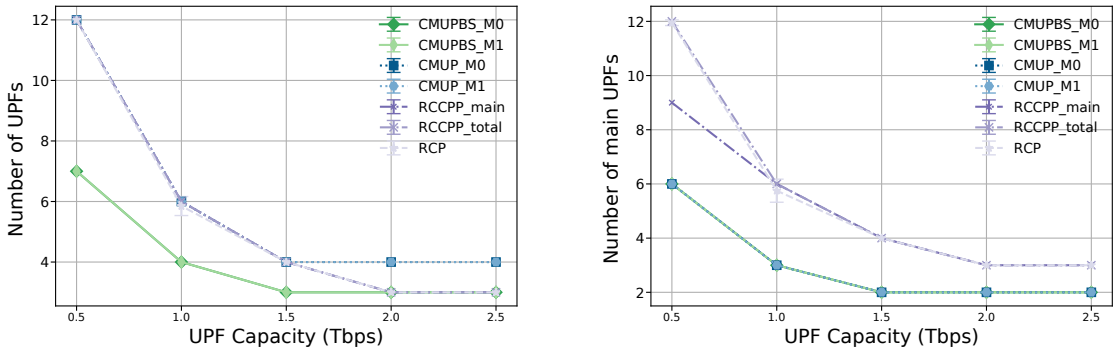
We compared the proposed models with two relevant studies on optimizing placement solutions with reliability requirements. Specifically, the resilient controller placement (RCP) [147] and the resilient capacitated controller placement problem (RCCPP) [137] were selected as baselines. Both methods pursue optimization objectives similar to that of our models since they aim to minimize the number of deployed instances (i.e., SDN controllers) subject to latency, resilience, and capacity constraints. To adapt these benchmarks to the UPP, we considered the SDN controllers and switches as UPFs and access nodes, respectively. For the RCCPP model, the constraint associated with inter-controller delay was relaxed.

These solutions were evaluated using numerous metrics, such as the number of deployed UPFs, UPF utilization, UPF relocations, and computational time. Some evaluation criteria, such as UPF relocations, maximum delay, and load distribution, were measured only in the main UPFs because the backups did not process traffic from their assigned access nodes under normal network conditions. As the RCCPP solution does not distinguish primary from backup UPFs, two variants were contemplated when analyzing RCCPP's performance for metrics related to the number of active instances. The first variant was RCCPP_total, which assumed that all UPFs were active, whereas the other was called RCCPP_main and considered that the main UPF of an ANN was its nearest one.

We show results with a 95% confidence interval for these experiments, as estimated by running the solutions 10 times on each evaluation scenario.

3.5.2.1 Number of UPFs

Figure 3.2 depicts the obtained results regarding the number of UPFs (total and active UPFs) for all solutions under study. The models based on the backup capacity sharing approach (CMUP-BS_M1 and CMUP-BS_M0) provided the best performance since they required the lowest number of UPFs (see Fig. 3.2(**a**)). The CMUP-BS achieved reductions ranging from 25–40% compared to the baselines for capacity values below 2 Tbps. This difference was more significant for small values of processing capacity where the total number of deployed UPFs was higher. In contrast, CMUP had the worst performance when the UPF capacity was high ($C_u \geq 2$ Tbps) with an additional UPF.



(**a**) Overall number of UPFs

(**b**) Number of active UPFs

Figure 3.2: Number of UPFs for different UPF capacity values.

All conceived solutions always required considerably fewer active UPFs than the benchmarks, as illustrated in Fig. 3.2(**b**). In particular, CMUP and CMUP-BS decreased the number of active UPFs in at least one to three instances compared to the established baselines. Thus, the distinction between primary and backup UPFs provided significantly more energy savings since backup UPFs could only be activated when failures occurred.

In general, the CMUP-BS method was more cost-effective than the other approaches since it reduced the overall number of deployed UPFs by sharing the backup capacity. Specifically, for these experiments, it met the reliability requirement of all access nodes by placing one backup UPF. Furthermore, both variants of the proposed models –with and without mobility considerations– always obtained similar results for all capacity values for the total and active number of UPFs.

3.5.2.2 UPF Utilization

Figure 3.3 represents UPF load distribution in the primary UPFs against various UPF processing capacities for all analyzed solutions. It also includes the utilization distribution of the overall number of UPFs deployed by the RCCPP model (RCCPP_total). As shown, the devised solutions (CMUP and CMUP-BS) outperformed the baselines for all evaluated capacity values. Our models provided a UPF average utilization between 50% and 90% with a characteristic imbalance below 25%. Conversely, the RCCPP_main and RCP benchmarks were characterized by lower utilization in their primary UPFs, with typical values around 50%, while the average utilization provided by RCCPP_total exceeded 90% of the UPF capacity most of the time. Furthermore, the RCCPP_main and RCP baselines provided an uneven load distribution, with imbalance values ranging from 20–95%. This behavior was due to the reference models, which distributed traffic demands among all deployed instances without UPF role differentiation.

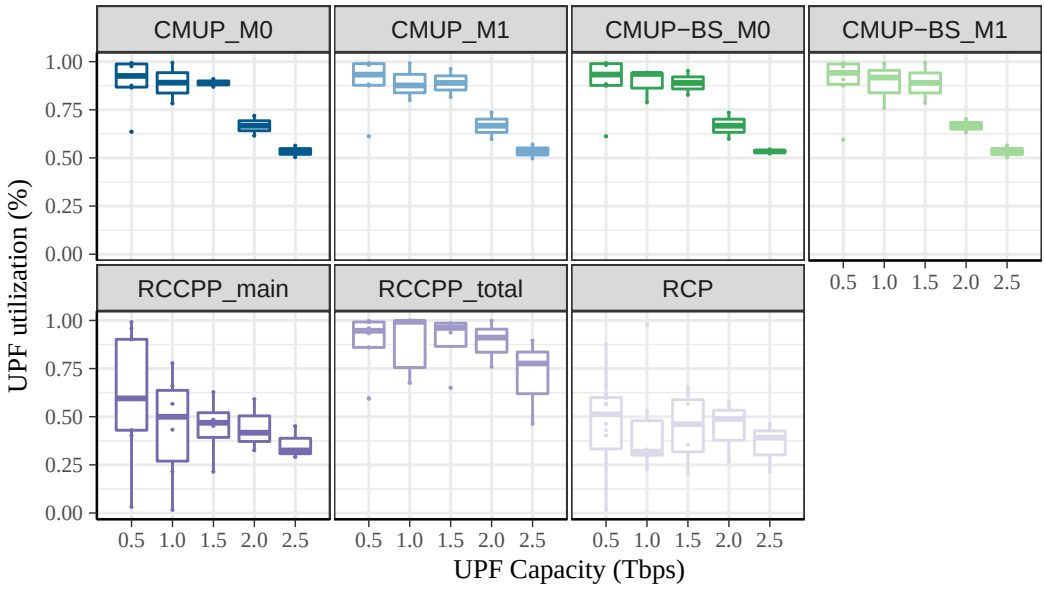


Figure 3.3: UPF utilization for different UPF capacity values.

These satisfactory outcomes in the devised models were partly due to the utilization of the α factor, which restricted the assigned capacity in the main UPFs (see expression (3.11)), thereby allowing for enhanced load distribution in the main UPFs. This factor was determined as a function of the total traffic demands and the expected number of UPFs for each capacity value. Overall, a decreasing trend in the primary UPF imbalance and average utilization with the capacity rise was observed for all solutions.

3.5.2.3 UPF Relocations

Figure 3.4 summarizes the average UPF relocation rates for all evaluated methods. As expected, the mobility-aware placement approaches always obtained the best results. A remarkable difference in relocations was evident in the proposed models, despite possessing the same number of active UPFs. More specifically, the CMUP_M1 and CMUP-BS_M1 solutions produced up to 50% and 60% fewer UPF relocations compared to CMUP_M0 and CMUP-BS_M0, respectively. These results elucidate significant performance improvements when considering user mobility requirements during the UPF placement planning. Subsequently, mobility-aware placement solutions can guarantee enhanced QoE without additional deployment costs regarding the number of deployed UPFs.

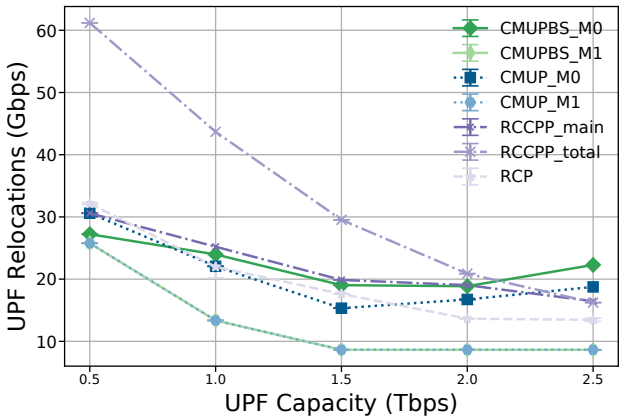
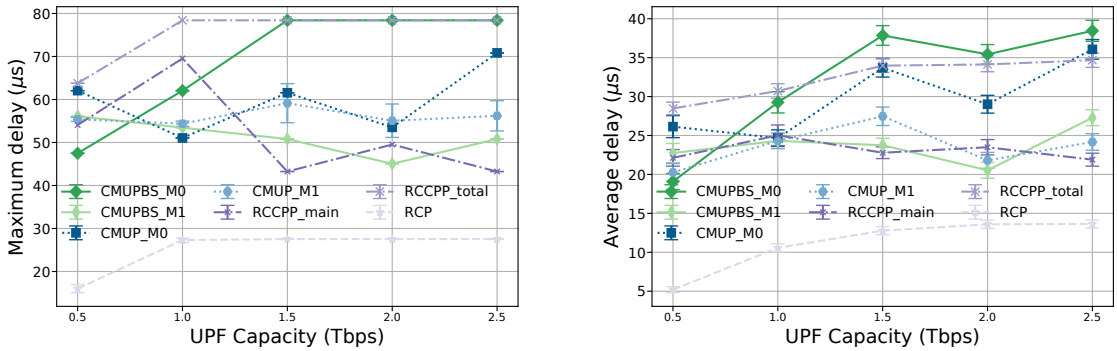


Figure 3.4: UPF relocation rate for different UPF capacity values.

3.5.2.4 Maximum and Average Latency

Figure 3.5 provides the results regarding the worst and average propagation delay between access nodes and their primary UPFs. The RCP model consistently obtained the best performance for these metrics, with less than 30 μs and 15 μs in its maximum and average delays. This was because the RCP solution was the only method that contemplated latency optimization in the objective function. In contrast, the RCCPP_total and the CMUP-BS_M0 approaches had the worst performance, with maximum values close to 80 μs. Nevertheless, this value was under the established threshold required to meet the propagation delay budget in the segment ANN-UPF ($L_{req} \leq 100$ μs in one way). Furthermore, the proposed models obtained similar outcomes to the RCCPP baseline in terms of mean and maximum delays, despite having considerably fewer active UPFs.



(**a**) Average delay

(**b**) Maximum delay

Figure 3.5: Average and maximum delays in the segment ANN-UPF for different UPF capacity values.

3.5.2.5 Execution Time

Figure 3.6 depicts the average execution times required by the envisioned models and the selected benchmarks. The RCCPP and CMUP_M0 models provided the best performance with average computational times of 1 s. In contrast, the mobility-aware approaches (CMUP_M1 and CMUP-BS_M1) demanded the highest execution times with differences of up to three orders of magnitude compared to the other models. This showed the main drawback of these approaches since such high running times limit their applicability for online resource allocation in real scenarios. Nevertheless, due to their remarkable performance in terms of relocation reductions, they could be used to benchmark heuristic approaches or for planning phases. Moreover, no significant difference between the solutions for no mobility considerations and the baselines was observed most of the time.

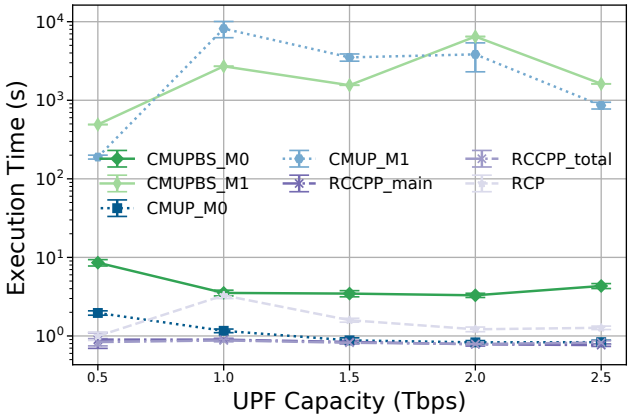


Figure 3.6: Execution times for different UPF capacity values.

No trend was noticed regarding the effects of UPF capacity on computational times. For instance, models such as RCCPP, RCP, CMUP-BS_M1, and CMUP_M1 required the lowest processing time for the smallest UPF capacity value, while the CMUP-BS_M0 and CMUP_M0 had their worst performance.

3.5.3 NOUP Performance

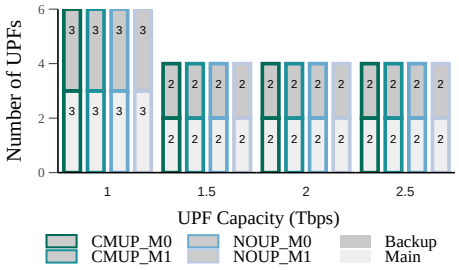
In this subsection, we evaluate the mobility and no-mobility considerations of the proposed heuristic (NOUP_M0 and NOUP_M1) for the three simulation scenarios presented in Table 3.6, as well as for both service categories. Since these categories have different service requirements, the assignments of their demands to the UPFs were determined separately. Specifically, the UPF placement problem was first solved for high-demand services, and after updating available resources in the underlying infrastructure, the second group of service placements was addressed. The performance of NOUP was compared with both variants of the CMUP model (i.e., CMUP_M0 and CMUP_M1) in terms of the number of UPFs, UPF utilization, UPF relocations, and computation time for several values of UPF capacity.

3.5.3.1 Number of UPFs

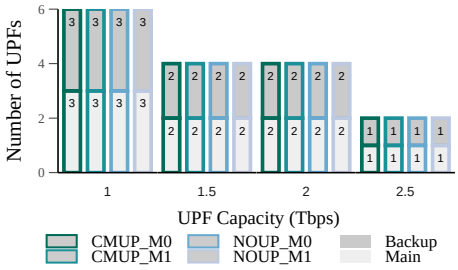
Figures 3.7 and 3.8 represent the total number of UPFs (main and backup) for each scenario and service category. As shown in these figures, NOUP provided near-optimal performance. Specifically, in urban scenarios, all solutions, both with and without mobility considerations, deployed the same number of UPFs for both service categories and all analyzed values of UPF capacity. The only exception was the low-demand category in the City_2 scenario for a UPF capacity of 1.5 Tbps. In this scenario, NOUP_M1 required one UPF more than the other solutions. We could believe that these satisfactory results are due to the shorter distance among ENs and the higher number of neighboring candidates per ANN that characterize these regions.

Nevertheless, similar outcomes were acquired in the rural zones. Especially for high-demand services, for which the latency budget was more restrictive, both heuristic variants (NOUP_M1 and NOUP_M0) consistently deployed the same number of UPFs, and their difference with the optimal solution was at most one additional UPF for the lowest value of UPF capacity ($C_u = 1$ Tbps). For this use case, all solutions obtained more main UPFs than backups. This was due to two isolated access nodes that could not be assigned to backup UPFs without violating placement constraints (e.g., service latency and main-backup relationship).

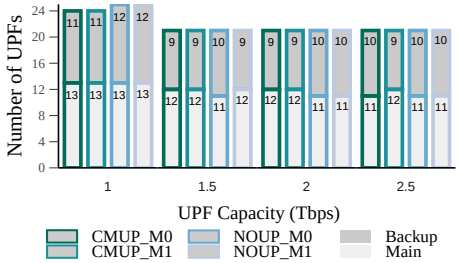
Overall, a reduction in the number of deployed UPFs with UPF capacity can be appreciated. This behavior was more noticeable for the low-demand category in rural areas since the latency requirement was less constrictive. For this case study, the total number of main UPFs was significantly smaller than high-demand services despite higher traffic demands. Furthermore, the inclusion of user mobility considerations in the UPF placement solutions design did not significantly alter the required number of UPFs.



(**a**) City_1 scenario.

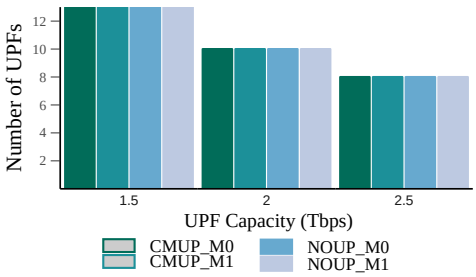


(**b**) City_2 scenario.

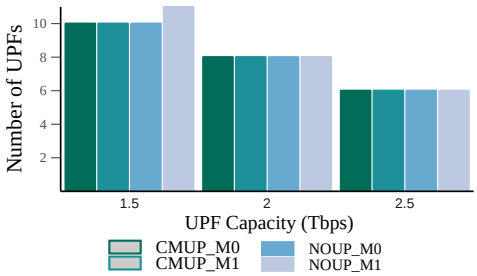


(**c**) Rural scenario.

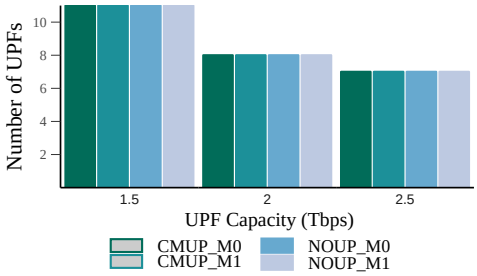
Figure 3.7: Number of UPFs versus UPF capacity for high-requirement services.



(**a**) City_1 scenario.



(**b**) City_2 scenario.

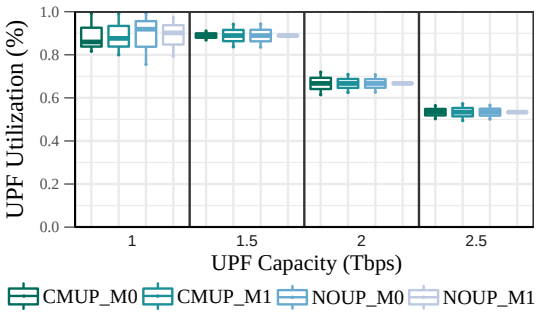


(**c**) Rural scenario.

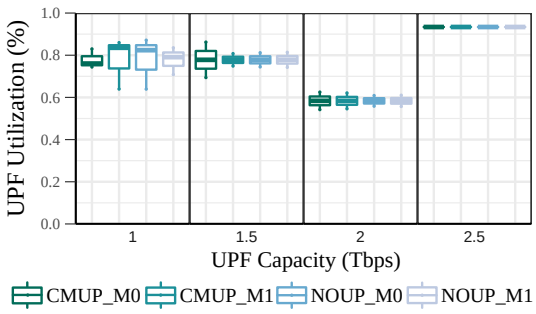
Figure 3.8: Number of UPFs versus capacity for low-requirement services.

3.5.3.2 UPF Utilization

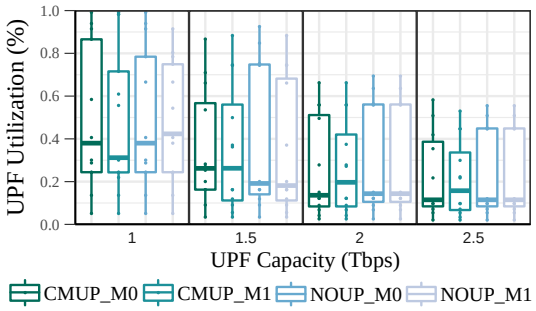
Figure 3.9 depicts the results regarding the main UPF load distribution for the high-demand service category. In urban scenarios, all solutions obtained similar utilization levels for all UPF capacity values. Specifically, the load distribution among UPFs was relatively even, with less than 25% in the maximum imbalance, and typical values around 10%. In contrast, the UPF load in rural areas was significantly uneven, with a characteristic imbalance of at least 50% and an average UPF utilization below 40%. This was due to isolated access nodes with low user density and service demands. Furthermore, as seen in the figure, the imbalance and UPF average utilization were reduced when UPF capacity increased. This behavior was evident in the City_1 and the rural scenario, where the number of UPFs remained constant most of the time.



(**a**) City_1 scenario.



(**b**) City_2 scenario.



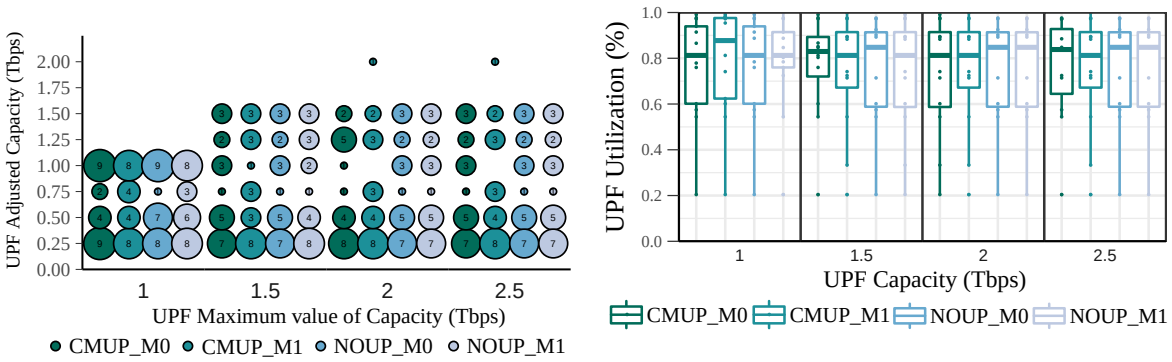
(**c**) Rural scenario.

Figure 3.9: UPF utilization versus UPF capacity for high-requirement services.

Rural scenarios, where service demands are scattered, require a different provisioning approach from urban scenarios when planning UPF placement for services with restrictive requirements (e.g., latency). For instance, deploying UPFs with smaller processing capacities that are scalable according to their underlying service demands may enhance the system's performance in terms of capacity utilization and imbalance. Therefore, we readjusted the UPFs capacity by considering a granularity factor of 0.25 Tbps.

Figure 3.10(**a**) summarizes the results regarding the required number of UPFs for every value of capacity considered. As shown in this figure, small processing values (i.e., 0.25 and 0.5 Tbps)

were the most frequent, while a UPF capacity higher than 1.5 Tbps was not often required. This was a general trend for all evaluated values of maximum capacity. The UPF utilization for the customized UPF capacities is shown in Fig. 3.10(**b**). After readjustment, significant improvements in the UPF utilization were achieved, with average values above 80% and most UPFs with loads above 60%. However, the imbalance was still significant, ranging from 45–80% because some isolated UPFs had few loads (around 20%). These results could have been improved by applying a smaller granularity factor, such as 0.1 Tbps.



(**a**) UPF adjusted capacities

(**b**) UPF utilization with adjusted capacity

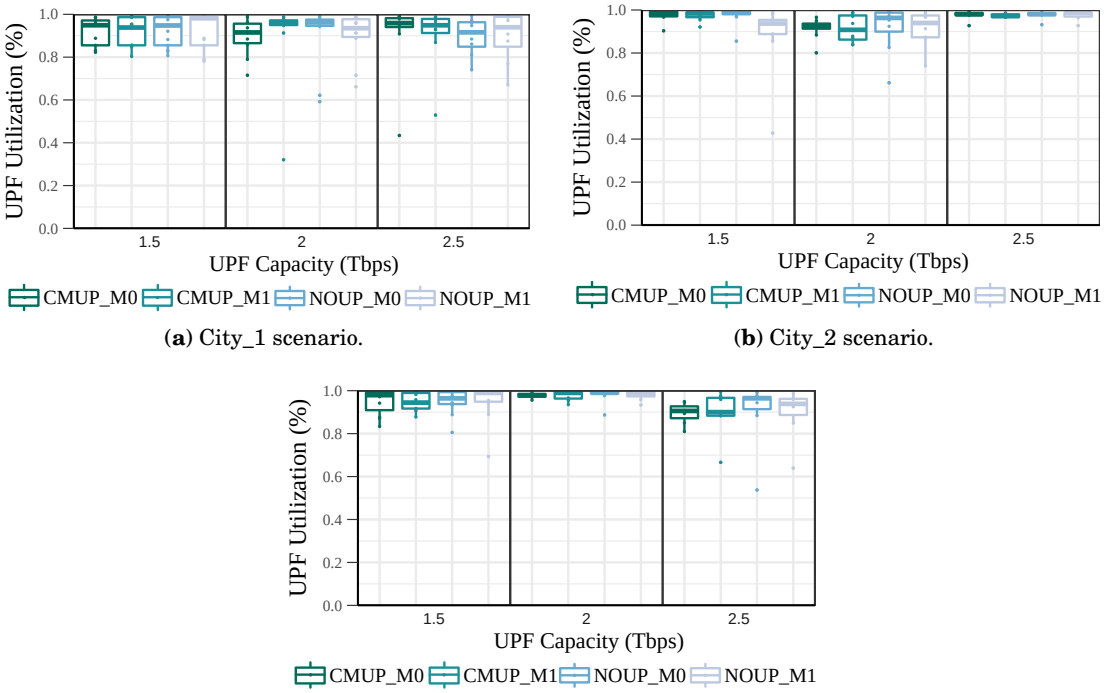
Figure 3.10: Adjusted capacity and load distribution versus UPF maximum capacity for services with high requirements in the rural scenario.

The load distribution for services belonging to the low-demand category is shown in Fig. 3.11. All proposed solutions obtained similar results in all scenarios, with an average utilization above 95% and a characteristic imbalance below 25%. These utilization values indicated that most of the UPFs were overloaded, although there were some exceptions (see outliers in Fig. 3.11) that had less than 70% utilization. This metric could have been improved by reducing the α factor in the UPF capacity. However, doing so would have increased the number of deployed UPFs.

Based upon these results, both solutions (i.e., NOUP and CMUP) for mobility and non-mobility requirements provided similar results in terms of load distribution, regardless of the service category or region under study.

3.5.3.3 UPF Relocations

The results regarding the relocation rate among the main UPFs for both service categories are shown in Figs. 3.12 and 3.13. As seen in the figures, the best results were always provided by placement solutions with mobility considerations (i.e., CMUP_M1 and NOUP_M1), while CMUP_M0 and NOUP_M0 produced the highest amount of relocations. This difference between user mobility and non-mobility-aware approaches was more noticeable for low-demand services, for which the relocation rate was significantly higher due to a greater number of services with mobility requirements and greater traffic demands. Specifically, for this use case, CMUP_M1



(**a**) City_1 scenario.

(**b**) City_2 scenario.

(**c**) Rural scenario.

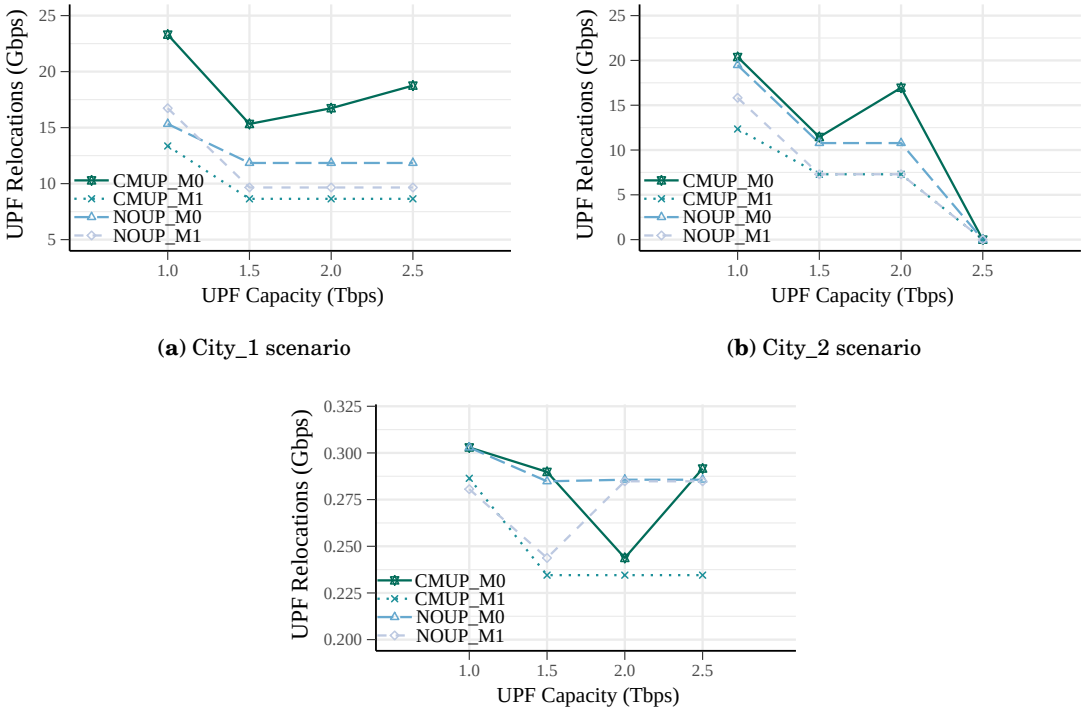
Figure 3.11: UPF utilization versus capacity for low-requirement services.

and NOUP_M1 reduced the relocation rates up to 72% and 57%, respectively, compared to their variants for non-mobility considerations.

For high-demand services in urban scenarios, the performance of NOUP_M1 was close to that of the optimal CMUP_M1. Conversely, a higher gap between these approaches was observed during the UPF placement for low-demand services. This was due to different placement conditions (available candidates) since a sequential placement approach was adopted during their evaluation, beginning with the most demanding services.

Notable reductions in the UPF relocation rate were attained when implementing user mobility-aware UPF placement solutions. Moreover, these metric values were usually decremented along with the number of deployed UPFs as the processing capacity increased. This is seen in Fig. 3.12(**b**); the occurrence of UPF relocations was completely avoided for $C_u = 2.5$ Tbps due to the existence of a unique UPF. Nonetheless, some exceptions to this behavior occurred, especially in the rural scenario for high-requirement services, for which the UPF relocation rates were incremented for NOUP_M1 and $C_u \geq 2$ Tbps.

Based upon these outcomes, a lower number of deployed UPFs does not necessarily imply fewer relocations. Instead, contemplating user mobility patterns when planning the UPF service area is critical.



(**a**) City_1 scenario

(**b**) City_2 scenario

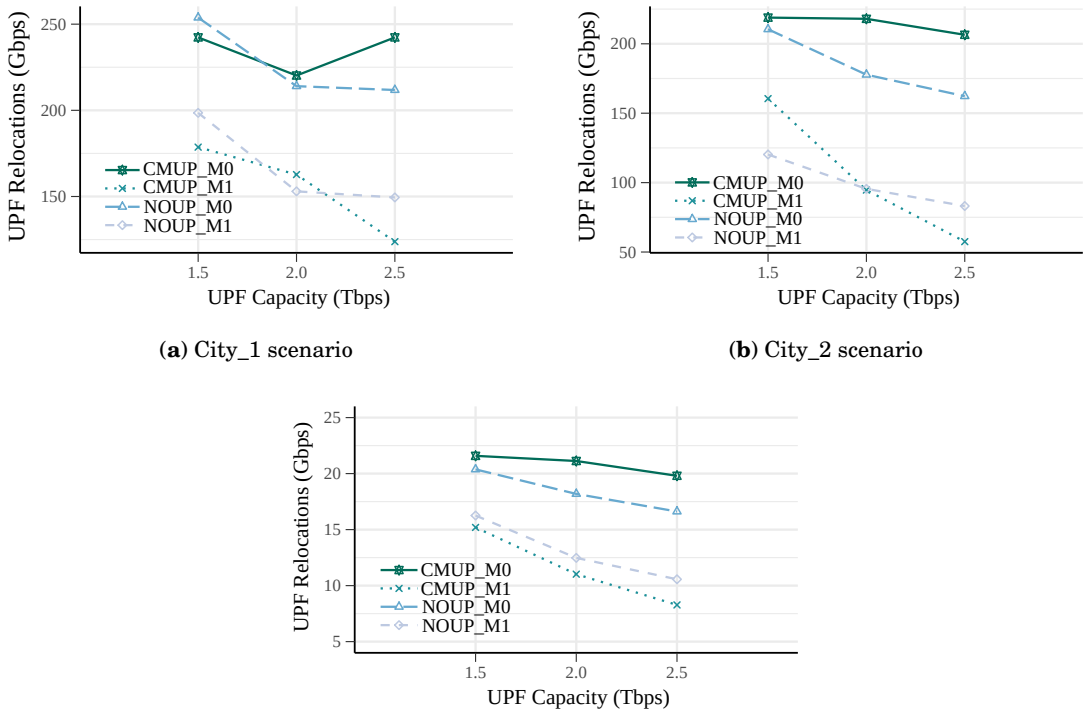
(**c**) Rural scenario

Figure 3.12: UPF relocation rate versus UPF capacity for high-requirement services.

3.5.3.4 Execution Time

Table 3.7 summarizes the execution times, in seconds, of the proposed heuristic and CMUP model for every scenario, service category, and UPF capacity considered in the simulations. As seen in this table, the NOUP solution consistently outperformed the ILP model for both variants (mobility and non-mobility considerations) with outstanding reductions in the computation times. Specifically, when compared to CMUP_M0, NOUP provided reductions ranging from 55% up to 95% and 35% to 45% for high-requirement services in urban and rural zones, respectively. Similar outcomes were achieved for the second group of services, with decreases in computing times between 70% and 85% for urban scenarios and 22% to 55% for the rural scenarios.

Moreover, the introduction of mobility requirements did not produce remarkable variations in the computation times of the heuristic approach. In contrast, these requirements severely impacted the CMUP_M1 exact solution as the number of combinations drastically increased when forming the UPF service areas. A notable difference in the execution times of both models was observed. Specifically, the solution time of CMUP_M1 was in the order of hundreds or even thousands of seconds, while CMUP_M0 barely required more than 1 s. These results disproved the CMUP_M1 model as a feasible solution for solving the mobility-aware UPF placement problem in practical scenarios and validated the necessity of heuristic-based approaches. Regarding the



(**a**) City_1 scenario

(**b**) City_2 scenario

(**c**) Rural scenario

Figure 3.13: UPF relocation rate versus capacity for low-requirement services.

Table 3.7: Execution times (s) for different values of UPF capacity.

Region	**Method**	**UPF capacity (Tbps)**						
		Group 1				**Group 2**		
		1.0	1.5	2.0	2.5	1.5	2.0	2.5
City_1	CMUP_M0	3.41	0.37	0.43	0.45	1.11	1.18	0.47
	CMUP_M1	10,428	8352	5370	940	244	190	121
	NOUP_M0	0.11	0.11	0.11	0.12	0.17	0.17	0.10
	NOUP_M1	0.16	0.14	0.15	0.13	0.21	0.16	0.13
City_2	CMUP_M0	3.16	0.43	0.45	0.38	0.56	0.52	0.48
	CMUP_M1	36,065	17192	4757	5.73	1420	176	30,058
	NOUP_M0	0.10	0.12	0.14	0.08	0.17	0.11	0.09
	NOUP_M1	0.12	0.14	0.14	0.14	0.16	0.14	0.12
Rural	CMUP_M0	0.61	0.59	0.52	0.57	0.58	0.51	0.32
	CMUP_M1	13.30	13.15	13.04	13.13	20,440	182,811	526
	NOUP_M0	0.37	0.36	0.33	0.29	0.33	0.25	0.09
	NOUP_M1	0.40	0.29	0.33	0.31	0.56	0.43	0.18

effects of UPF maximum capacity on the solution computation times, no significant difference was detected for the first group of services. For the second category, however, the execution time slightly decreased with capacity growth.

3.6 Conclusion

This chapter presented two ILP models and a heuristic-based solution to address the UPF placement problem in an MEC environment under limited resources and stringent service demands. These solutions aim to reduce the number of deployed UPF instances and their relocations while considering latency, reliability, and mobility service requirements. Achieving these aims will allow service providers and network operators to plan their infrastructure and available resources cost-effectively when deploying UPFs.

Extensive simulations and experiments were conducted to evaluate the solutions' performance. Overall, the proposed CMUP and CMUP-BS models evidenced a significant decrease in active UPFs compared with the baselines. This outcome was due to the distinction between UPF roles (i.e., main and backup). Moreover, CMUP-BS considerably reduced the overall number of deployed UPFs by sharing the capacity of the backup UPFs. Specifically, it required up to 40% fewer UPFs than the other models.

Additionally, the proposed mathematical models provided better load distribution concerning imbalance and average utilization metrics, partly due to the introduction of the alpha factor in the primary UPFs. Regarding the effects of user mobility on the UPF placement, the mobility-aware variants (i.e., CMUP_M1 and CMUP-BS_M1) outperformed their analogous, with remarkable reductions in the number of UPF relocations. Moreover, they demonstrated that the user QoE could be enhanced without incurring additional deployment costs regarding the number of deployed UPFs by accounting for user mobility patterns. However, this improvement was made at the expense of a dramatic increase in running times, therefore discarding the possibility of using mobility-aware models for online placement applications.

The NOUP algorithm obtained satisfactory results with near-optimal performance, especially in urban scenarios. The worst-case scenario required only one additional UPF compared to the optimal solution. Furthermore, NOUP's approach to mobility considerations provided outstanding reductions in the computation time compared to the mobility-aware models (i.e., CMUP_M1). Therefore, the obtained results showcased the ability of the proposed approaches to attain their objectives.

CHAPTER 4

DYNAMIC UPF PLACEMENT

This chapter is based on:

- **I. Leyva-Pupo**, C. Cervelló-Pastor, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic Scheduling and Optimal Reconfiguration of UPF Placement in 5G Networks," in *Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (ACM MSWIM'20)*, Alicante, Spain, Nov. 2020, Association for Computing Machinery, pp. 103-111, 2020.

MEC technology significantly reduces network response times and backhaul utilization since network functions and service applications are closer to the end users. However, this technology also presents new challenges for dynamic network function placement and resource management. ENs have limited resources (networking and computing) and thus host network functions with smaller processing capacities and service areas. The latter and the presence of highly mobile users increase the likelihood of QoS degradation events, such as UPF relocations and service latency violations. Thus, frequent and dynamic readjustment of the UPF placement and UPFs' assigned PDU sessions may be required to ensure satisfactory QoS levels. However, placement reevaluation events may produce additional delays in the sessions' data path and service interruptions due to UPF migration and session relocations.

Designing strategies to optimize the UPF placement and reconfiguration (UPR) dynamically is crucial for addressing this challenge. Therefore, this chapter presents a multi-objective ILP solution for determining the optimal UPF placement and PDU session assignment. This model aims to reduce operational and deployment expenditures (e.g., migration and routing costs) while ensuring service requirements and avoiding QoS degradation (i.e., session relocations). Additionally, a scheduling mechanism based on OST is presented to determine the optimal

reconfiguration time concerning QoS levels as measured in the instantaneous number of sessions with latency violations.

The remainder of this chapter is organized as follows. In Section 4.1, the UPR problem is presented along with a mathematical model proposed to solve the problem. Similarly, Section 4.2 introduces the conceived strategy for dynamically scheduling the UPR. In Section 4.3, we evaluate and discuss the performance of the proposed solutions. Finally, we highlight the main conclusions of this chapter in Section 4.4.

4.1 Optimal UPF Placement Reconfiguration

This section introduces the UPR problem. The problem is formulated as a multi-objective ILP model aimed at minimizing expenditures related to the UPF placement readjustment, guaranteeing service demands, and reducing service disruptions due to session relocations during reconfigurations.

4.1.1 Problem Statement

Given a placement configuration in which multiple UPF instances have been located in different EN locations and in which PDU session requests were mapped to these UPFs according to service demands and available resources, we must consider certain variations over time in their initial configuration due to user mobility and traffic demands. For instance, network response times may increase as users move, implying QoS degradation and higher routing costs for network operators. Under these circumstances, readjusting the current UPF placement and service request mapping configuration may be required to re-establish QoS satisfaction. However, this reconfiguration must be carefully set due to the multiple combinations and trade-offs.

Thus, we address the problem of finding optimal 5G UPF placement and user request mapping setup in a dynamic MEC system to enhance the system QoS. This new configuration must also use the infrastructure's available resources cost-effectively and avoid additional QoS degradation due to session relocations.

4.1.1.1 Network Model

We consider a 5G network represented by graph G(N,E,S), where N, E, and S denote the sets of nodes, links, and active PDU sessions, respectively. The set of network nodes comprises access nodes (N_r), aggregation points (APs) (N_a), and UPF candidate locations (N_c). Additionally, U represents the set of already deployed UPF instances.

A PDU session $s \in S$ is characterized by a latency requirement (L^s_{ser}), a minimum number of UPFs (Vu^s) which must be assigned to satisfy the service reliability levels, and resource demands (d^s), such as CPU and memory. Moreover, Cu_c indicates the maximum UPF processing capacity associated with a given candidate location, whereas T_u denotes the UPF maximum processing

time. For simplicity, we assume that all links have sufficient bandwidth capacity to process their assigned traffic. Tables 4.1 and 4.2 showcase the used notations and their descriptions.

Table 4.1: Used notation for sets and parameters.

Notation	Description
E	Set of network links
N	Set of network nodes
N_a	Set of aggregation points
N_r	Set of access nodes
N_c	Set of UPF potential locations (e.g., MEC servers)
U	Set of already deployed UPFs
S	Set of PDU session requests
L^s_{ser}	Service latency requirement of PDU session $s \in S$
L^s_{ra}	Propagation delay between ANN $r \in N_r$ of PDU session $s \in S$ and its AP $a \in N_a$
L^s_{ac}	Propagation delay between AP $a \in N_a$ of PDU session $s \in S$ and candidate $c \in N_c$
T^s_{proc}	Processing time in the data path of PDU session $s \in S$
T^s_{prop}	Propagation time in the data path of PDU session $s \in S$
Vu^s	Number of UPFs required for PDU session $s \in S$
d^s	Computing resources required by PDU session $s \in S$
Cu_c	UPF processing capacity (e.g., CPU and RAM) associated with candidate $c \in N_c$
Fd_c	Cost of installing a UPF in location $c \in N_c$
Fo_c	Cost of running a UPF in location $c \in N_c$
Ft_{ac}	Cost of routing associated to link ac, $a \in N_a$ and $c \in N_c$
$Fm_{c'c}$	Cost of migrating a UPF from location c to $c' \in N_c$
Fr^s	Cost of reassigning a PDU session $s \in S$
P^s_c	1 if PDU session $s \in S$ had a UPF in $c \in N_c$ before the reconfiguration
X_c	1 if there was a UPF deployed in node $c \in N_c$ before the reconfiguration
V^u_c	1 if UPF $u \in U$ was deployed in location $c \in N_c$

Table 4.2: Used notation for binary variables.

Notation	Description
x_c	1 if there is a UPF deployed at location $c \in N_c$
h_c	1 if it has been a change in location $c \in N_c$
n_c	1 if a new UPF has been deployed in location $c \in N_c$
δ_c	1 if there is a UPF deployed in location $c \in N_c$
v^u_c	1 if UPF $u \in U$ is deployed in location $c \in N_c$
$m^u_{c'c}$	1 if UPF $u \in U$ located in c' is migrated to $c \in N_c$
p^s_c	1 if PDU session $s \in S$ is assigned to a UPF in $c \in N_c$

4.1.2 Model: Optimal Cost-aware UPF Placement Reconfiguration

The cost-aware UPF placement reconfiguration (CUPR) model aims to reduce capital and operational expenditures associated with UPR events. To this aim, the following cost components have been considered:

- Deployment cost (C_{dep}): Encompasses costs associated with the deployment of new UPFs.

$$C_{dep} = \sum_{c \in N_c} Fd_c \cdot n_c \tag{4.1}$$

- Running cost (C_{run}): Includes costs related to UPF operation and is expressed in terms of the number of deployed UPFs.

$$C_{run} = \sum_{c \in N_c} Fo_c \cdot x_c \tag{4.2}$$

- Routing cost (C_{rou}): Deals with the cost of routing PDU sessions traffic from their APs to their assigned UPFs. The network response time can be improved by reducing the value of this component since it is expressed in terms of the propagation delays (L_{ac}).

$$C_{rou} = \sum_{c \in N_c} \sum_{a \in N_a} \sum_{s \in S} Ft_{ac} \cdot L^s_{ac} \cdot p^s_c \tag{4.3}$$

- Migration cost (C_{mig}): Expresses the cost of migrating UPF instances from one location to another. This cost is determined by considering already deployed UPFs (U) at the reconfiguration time.

$$C_{mig} = \sum_{c \in N_c} \sum_{c' \in N_c} \sum_{u \in U} Fm_{c'c} \cdot m^u_{c'c} \tag{4.4}$$

- Reassignment cost (C_{rea}): Embraces expenditures related to the relocation of service sessions during the placement reconfiguration. Specifically, this solution is expressed as a penalty (Fr^s) applied to service providers for interrupting PDU sessions due to reconfiguration events. Different penalties can be applied depending on user subscriptions and service types. The reassignment of a PDU session is indicated by a change in the location of its assigned UPFs, $[p^s_c - P^s_c]^+$ where $[f(x)]^+ = max\{f(x), 0\}$. Thus, the previous expression equals 1 when a session is assigned to a UPF placed at a different location from its location prior to the reconfiguration.

$$C_{rea} = \sum_{c \in N_c} \sum_{s \in S} Fr^s \cdot [p^s_c - P^s_c]^+ \tag{4.5}$$

The CUPR model seeks to minimize the above-mentioned cost components' effects when readjusting the UPF placement and mapping configuration. However, given its multi-objective nature and conflicting objectives, such as optimization of UPF operation and routing costs, a trade-off among these terms has to be defined when solving the UPR problem. Therefore, we adopt a weighted sum method to transform the problem into a mono-objective optimization problem. A weight factor (α_i) can be associated with each optimization term to specify its relative

importance in the overall objective function. Additionally, these terms should be normalized to avoid dominant effects. Thus, the CUPR model can be formulated as follows:

$$\begin{aligned}\text{Min:} \quad & \alpha_1 \cdot \sum_{c \in N_c} Fd_c \cdot n_c + \alpha_2 \cdot \sum_{c \in N_c} Fo_c \cdot x_c + \alpha_3 \cdot \sum_{c \in N_c} \sum_{a \in N_a} \sum_{s \in S} Ft_{ac} \cdot L^s_{ac} \cdot p^s_c + \\ & \alpha_4 \cdot \sum_{c \in N_c} \sum_{c' \in N_c} \sum_{u \in U} Fm_{c'c} \cdot m^u_{c'c} + \alpha_5 \cdot \sum_{c \in N_c} \sum_{s \in S} Fr^s \cdot [p^s_c - P^s_c]^+ \end{aligned} \tag{4.6}$$

subject to the below constraints.

Each PDU session request must be mapped to a minimum number of UPFs (Vu^s) to provide service while guaranteeing reliability. The value of the Vu^s parameter can be established by the network operator or determined using (3.1).

$$\sum_{c \in N_c} p^s_c \geq Vu^s \qquad \forall s \in S \tag{4.7}$$

Constraint (4.8) restricts the assignment of PDU sessions to those candidate locations where UPF instances are deployed. Namely, a PDU session $s \in S$ cannot be assigned to a candidate location $c \in N_c$ if no UPF is deployed. Additionally, inequality (4.9) prevents the deployment of empty UPFs by ensuring that all UPF instances have some sessions assigned.

$$p^s_c \leq x_c \qquad \forall s \in S, \forall c \in N_c \tag{4.8}$$

$$x_c \leq \sum_{s \in S} p^s_c \qquad \forall c \in N_c \tag{4.9}$$

Expression (4.10) stipulates that the service demands assigned to a UPF instance cannot exceed the physical resources available at its associated location.

$$\sum_{s \in S} d^s \cdot p^s_c \leq Cu_c \qquad \forall c \in N_c \tag{4.10}$$

As stated previously, the occurrence of UPF migrations is determined by considering the set of UPFs already instantiated at the reconfiguration time. In this regard, expression (4.11) indicates whether a UPF has been migrated from a source location $c \in N_c$ to a target candidate $c' \in N_c$. Moreover, constraint (4.12) expresses that a UPF instance can be migrated once during a placement readjustment event at most.

$$m^u_{c'c} = v^u_c \wedge V^u_{c'} \qquad \forall u \in U, \forall c', c \in N_c, c' \neq c \tag{4.11}$$

$$\sum_{c \in N_c} \sum_{c' \in N_c} m^u_{c'c} \leq 1 \qquad \forall u \in U \tag{4.12}$$

The set of already UPFs deployed before the reconfiguration event cannot increase in size due to a placement reevaluation. More specifically, the number of UPF instances forming this set ($u \in U$) either remains constant or decreases with the removal of some UPFs. However, new instances cannot be added to this set during reconfiguration. This distinction avoids mistakenly interpreting new deployments as existing ones during the model implementation.

$$\sum_{c \in N_c} \sum_{u \in U} v^u_c \leq |U| \tag{4.13}$$

The presence of a UPF at a given location can be the result of a new deployment or a UPF's deployment during previous placement events, such as an initial placement or a reconfiguration. The latter may be because the UPF did not change its location or migrated to this location during reconfiguration.

$$x_c = 1 \Rightarrow n_c \veebar \sum_{u \in U} v_c^u = 1 \qquad \forall c \in N_c \tag{4.14}$$

Constraint (4.14) is not linear. However, it can be expressed in a linear form with the help of the binary variable δ_c. This variable indicates the presence of an already deployed UPF $u \in U$ at a given location $c \in N_c$.

$$n_c + \delta_c \leq 2 - x_c \qquad \forall c \in N_c \tag{4.15}$$

$$n_c + \delta_c \geq x_c \qquad \forall c \in N_c \tag{4.16}$$

$$\sum_{u \in U} v_c^u \geq \delta_c \qquad \forall c \in N_c \tag{4.17}$$

$$\sum_{u \in U} v_c^u \leq |U| \cdot \delta_c \qquad \forall c \in N_c \tag{4.18}$$

Expressions (4.19) and (4.20) are related to the detection of changes with negative impact concerning the reconfiguration cost in the candidate locations. In particular, constraint (4.19) indicates that a negative change at a given location is produced by a new UPF deployment or the migration of an existing instance to the candidate. A change in a candidate is determined by comparing its current state with the one it had before the placement readjustment in terms of deployed UPFs, $h_c = [x_c - X_c]^+$. We omit changes regarding the removal of UPF instances. Additionally, inequality (4.20) restricts the type of change in the candidate nodes. Specifically, a variation in a candidate is caused by a new deployment or migration, but not for both reasons simultaneously.

$$h_c = n_c \veebar \sum_{c' \in N_c} \sum_{u \in U} m_{c'c}^u \qquad \forall c \in N_c \tag{4.19}$$

$$n_c + \sum_{c' \in N_c} \sum_{u \in U} m_{c'c}^u \leq 1 \qquad \forall c \in N_c \tag{4.20}$$

To keep the linearity of the model, constraint (4.19) can be replaced by the following expressions:

$$h_c \leq \sum_{c' \in N_c} \sum_{u \in U} m_{c'c}^u + n_c \qquad \forall c \in N_c \tag{4.21}$$

$$h_c \geq \sum_{c' \in N_c} \sum_{u \in U} m_{c'c}^u - n_c \qquad \forall c \in N_c \tag{4.22}$$

$$h_c \geq n_c - \sum_{c' \in N_c} \sum_{u \in U} m_{c'c}^u \qquad \forall c \in N_c \tag{4.23}$$

$$h_c \leq 2 - n_c - \sum_{c' \in N_c} \sum_{u \in U} m_{c'c}^u \qquad \forall c \in N_c \tag{4.24}$$

Inequality (4.25) guarantees service latency requirements by forcing the mapping of PDU sessions to those candidates capable of meeting the round trip user plane delay. The latency of a PDU session is computed regarding the processing time (T^s_{proc}) of the network elements that form its data path and the propagation delay (T^s_{prop}) between the elements.

$$T^s_{proc} + T^s_{prop} \leq L^s_{serv} \qquad \forall s \in S, \forall c \in N_c \tag{4.25}$$

where $T^s_{proc} = 2 \cdot (T^s_r + T^s_a + T_u \cdot p^s_c) + T_d$ and $T^s_{prop} = 2 \cdot (L^s_{ra} + L^s_{ac} \cdot p^s_c + L^s_{cd})$. The terms T^s_r, T^s_a, T_u, and T_d represent the processing time of access nodes, APs, UPFs, and DNs whereas L^s_{ra}, L^s_{ac} and L^s_{cd} indicate the propagation delays in the segments between them. In this study, application servers and UPFs are assumed to be co-located in the ENs. Thus the propagation delay in the segment UPF-DN is neglected ($L^s_{cd} = 0$).

The binary nature of the used variables is indicated below:

$$x_c, h_c, n_c, v^u_c, m^u_{c'c}, p^s_c \in \{0,1\} \qquad \forall s \in S, \forall u \in U, \forall c, c' \in N_c \tag{4.26}$$

Thereby, the conceived ILP model for the UPRP can be summarized as follows:

$$\text{Min } \alpha_1 \cdot C_{dep} + \alpha_2 \cdot C_{run} + \alpha_3 \cdot C_{rou} + \alpha_4 \cdot C_{mig} + \alpha_5 \cdot C_{rea}$$

s. t.:

$$\sum_{c \in N_c} p^s_c \geq V u^s \qquad \forall s \in S$$

$$p^s_c \leq x_c \qquad \forall s \in S, \forall c \in N_c$$

$$x_c \leq \sum_{s \in S} p^s_c \qquad \forall c \in N_c$$

$$\sum_{s \in S} d^s \cdot p^s_c \leq C u_c \qquad \forall c \in N_c$$

$$m^u_{c'c} = v^u_{c'} \wedge V^u_c \qquad \forall u \in U, \forall c, c' \in N_c, c' \neq c$$

$$\sum_{c \in N_c} \sum_{c' \in N_c} m^u_{c'c} \leq 1 \qquad \forall u \in U$$

$$\sum_{c \in N_c} \sum_{u \in U} v^u_c \leq |U|$$

$$n_c + \delta_c \leq 2 - x_c \qquad \forall c \in N_c$$

$$n_c + \delta_c \geq x_c \qquad \forall c \in N_c$$

$$\sum_{u \in U} v^u_c \geq \delta_c \qquad \forall c \in N_c$$

$$\sum_{u \in U} v^u_c \leq |U| \cdot \delta_c \qquad \forall c \in N_c$$

$$h_c \leq \sum_{c' \in N_c} \sum_{u \in U} m^u_{c'c} + n_c \qquad \forall c \in N_c$$

$$h_c \geq \sum_{c' \in N_c} \sum_{u \in U} m^u_{c'c} - n_c \qquad \forall c \in N_c$$

$$h_c \geq n_c - \sum_{c' \in N_c} \sum_{u \in U} m^u_{c'c} \qquad \forall c \in N_c$$

$$h_c \leq 2 - n_c - \sum_{c' \in N_c} \sum_{u \in U} m^u_{c'c} \qquad \forall c \in N_c$$

$$n_c + \sum_{c' \in N_c} \sum_{u \in U} m^u_{c'c} \leq 1 \qquad \forall c \in N_c$$

$$T^s_{proc} + T^s_{prop} \leq L^s_{serv} \qquad \forall s \in S, \forall c \in N_c$$

$$x_c, h_c, n_c, v^u_c, m^u_{c'c}, p^s_c \in \{0,1\} \qquad \forall s \in S, \forall u \in U, \forall c, c' \in N_c$$

4.2 Dynamic Scheduling for the UPR

This section presents a scheduling mechanism to determine the best time for the UPR. First, we formulate the problem as an optimal stopping problem (OSP) called skeptical scheduling of the reconfiguration (SSR). The optimal reconfiguration time is determined according to QoS metrics (i.e., number of sessions with latency violations) and OST principles. This section also provides some OST-related fundamentals and an optimal stopping rule for the SSR problem are provided.

4.2.1 Problem Statement: Skeptical Scheduling of the Reconfiguration

In dynamic mobile environments, placement conditions vary with user locations over time. This dynamism implies that an optimal placement configuration, determined under certain conditions, may no longer be optimal or feasible as time passes. For example, at a given instant, a placement event, either initial or reconfiguration, is executed, and all the PDU sessions are satisfactorily mapped to a given number of UPFs according to their service requirements (e.g., latency and processing demand). However, as users move, their access nodes change, and therefore, their perceived network response time also varies. The perceived delay may vary due to numerous factors, such as relocation of the assigned UPFs or a simple modification in the data path. These data path modifications may deteriorate the QoS when the distance between users and their assigned UPFs (propagation delay) increases and a nearby UPF is not available to provide the service. Specifically, a service latency violation occurs when the user plane response time of a PDU session surpasses its service latency requirement (i.e., $L^s > L^s_{serv}$).

Thus, we define $I^s_t \in [0,1]$ as a random variable indicating whether the QoS of a PDU session $s \in S$ is poor or not due to service latency violation at a given time t.

$$I^s_t = \begin{cases} 1 & \text{if the service latency requirement of session } s \in S \text{ is exceeded at instant } t \\ 0 & \text{otherwise} \end{cases}$$

Thereby, at time t, the total number of PDU sessions with latency violations (L_t) can be expressed as:

$$L_t = \sum_{s \in S} I^s_t \tag{4.27}$$

A UPF placement is no longer optimal when users have poor QoS ($L_t \neq 0$). In this case, reevaluation of the UPF placement configuration is required to re-establish the desired QoS level. However, reconfiguration events are not only resource and time-consuming but may occasion additional delays or even service interruption. Moreover, these events may increase expenditures due to the deployment of new UPFs, the migration of existing instances, and the reassignment of PDU sessions. Subsequently, unnecessary and frequent placement re-computations must be avoided as much as possible, triggered only when needed. Therefore, the problem of deciding the optimal readjustment time to reduce reconfiguration adverse effects must be determined while maintaining the overall QoS under proper values.

Therefore we assume that at each time t, the system can tolerate a maximum number of latency violations denoted as Θ ($\Theta > 0$) without needing to execute a reconfiguration event. Namely, the offered QoS is considered acceptable, and no placement readjustment is required as long as the number of sessions with latency violations does not surpass the Θ threshold. In contrast, violating this threshold deteriorates the system QoS, rendering a new placement reconfiguration mandatory. As mentioned previously, a readjustment event implies a reconfiguration cost, which is defined as a function of the expected number of affected sessions ($\mathbb{E}[S_r]$) due to the reconfiguration procedure; see (4.28). The service providers can establish the Θ parameter according to specific service-level agreements regarding application types and user profiles.

$$\mathbb{E}[S_r] = \sum_{c \in N_c} \sum_{s \in S} [p_c^s - P_c^s]^+ \cdot P(p_c^s \neq P_c^s) \tag{4.28}$$

The main objective of the SSR problem is determining in advance when the system is about to exceed the established QoS threshold so that UPF placement can be readjusted to avoid QoS deterioration, as well as frequent reconfiguration events. In other words, at each instant, the system tries to tolerate as many sessions with latency violations as possible as long as the QoS is kept under acceptable levels to delay or even avoid reconfiguration procedures. When the Θ parameter is exceeded, a placement reevaluation event is activated, thus incurring an expected reconfiguration cost. Accordingly, we represent the decision process for the placement reconfiguration as the following reward function:

$$Y_t(L_t) = \begin{cases} L_t & \text{if } L_t \leq \Theta \\ -\lambda \cdot \mathbb{E}[S_r] & \text{if } L_t > \Theta \end{cases} \tag{4.29}$$

where the weight factor λ indicates the importance of the reconfiguration cost (i.e., $\mathbb{E}[S_r]$) to the reward function.

Our aim with the SSR problem is determining the optimal time t^* when it is worthy of stopping observing the selected QoS metric denoted by the L_t parameter and reconfiguring the UPF placement. In other words, we need to determine the stopping rule that maximizes the expected reward function in (4.29). Therefore, we can formulate the SSR problem as follows:

Problem 1. *Given a sequence of events defined by the L_t parameter, a maximum QoS tolerance threshold Θ, and an expected reconfiguration cost $\mathbb{E}[S_r]$, the SSR problem seeks the optimal*

decision epoch t^* where the supremum of Y_t is attained:

$$\sup_{t \geq 0} \mathbb{E}[Y_t(L_t)] \tag{4.30}$$

4.2.2 Solution Fundamentals

The SSR problem belongs to the group of OSPs with infinite horizons where, at each time interval or decision epoch t, we must make one of the following decisions: (i) continue to the next time slot $(t+1)$ and do not reconfigure the placement or, (ii) stop and readjust the placement.

The theory of optimal stopping is concerned with the problem of choosing a time to take a given action based on sequentially observed random variables to maximize an expected payoff or minimize an expected cost [122]. OSPs are characterized by a sequence of observations X_1, X_2, $\ldots$, X_t whose joint distribution is assumed to be known, as well as a sequence of reward or cost functions Y_1, Y_2, $\ldots$, Y_t, where $Y_t = y_t(x_1, x_2, \ldots, x_t)$. Thus, an OSP can be defined as follows. A decision-maker or agent observes a sequence of random variables $(X_1, X_2, \ldots, X_t)$ and, at each time t, must decide whether to stop observing and receive a reward Y_t (or cost) or continue and observe the next variable (X_{t+1}) [122]. The objective is to stop observing at the best time t^* for which the expected payoff is maximized or the expected cost is minimized.

Due to its simplicity and efficiency, an approach widely used to solve OSPs is the one-stage-look ahead (1-SLA) rule, also referred to as the myopic rule. This rule indicates at each decision epoch t whether to stop or continue based on the expected value of the reward/loss function in the next time epoch $(t+1)$. Specifically, for OSPs aimed at maximizing an expected reward, the 1-SLA rule calls for stopping at the first time t for which the payoff Y_t for stopping is at least as good as the expected reward for continuing to the next stage and then stopping. Thus, the decision-making at each epoch depends only on the current observations and the expected reward in the next stage. Mathematically, the 1-SLA rule can be defined as follows:

Definition 1. *For stopping problems, aimed at maximizing an expected payoff Y_t, the 1-SLA rule is described by the stopping time*

$$t^* = inf\,\{t \geq 0 : Y_t \geq \mathbb{E}[Y_{t+1} | \mathbb{F}_t]\} \tag{4.31}$$

where $\mathbb{F}_t$ is the σ-fields generated by the observations X_1, X_2, $\ldots$, X_t. Specifically, it represents the knowledge of the random variable X_t up to time t.

Generally, the 1-SLA rule is not optimal [148]. However, it has been demonstrated to be optimal in monotone-stopping problems (see [122]).

Definition 2. *Let A_t denote the event $\{Y_t \geq \mathbb{E}(Y_{t+1} | \mathbb{F}_t)\}$. The stopping problem is monotone if the sets A_t are monotone non-decreasing, i.e., $A_0 \subset A_1 \subset A_2 \ldots$ almost surely (a.s.).*

A monotone stopping rule problem can be described as follows: If the 1-SLA rule calls for stopping at time t due to event A_t, then it will also call for stopping at all future stages (e.g., $t+1$, $t+2$, ...) regardless of the value of the future observations, since $A_t \subset A_{t+1} \subset A_{t+2} \ldots$

Theorem 1. *The 1-SLA rule is optimal in monotone-stopping rule problems.*

Proof. Refer to Chapter 5 of [122] for more details. ■

4.2.3 Optimal Skeptical Scheduling of the Reconfiguration

This subsection proposes an optimal stopping rule for the SRR problem based on the 1-SLA rule. This section also shows the optimality of this rule.

Theorem 2. *Given an upper bound Θ upon which the system QoS is considered to deteriorate and a sequence of latency violations $L_1, L_2, \ldots, L_t$ concerning the last optimal UPF placement ($L_0 = 0$), the optimal reconfiguration time (stopping time) t^* for the SSR problem stated in* (4.30) *is defined as:*

$$t^* = inf\{t \geq 0 : \sum_{l=0}^{\Theta} l \cdot P(L = l) - \lambda \cdot \mathbb{E}[S_r] \cdot (1 - \sum_{l=0}^{\Theta} P(L = l)) \leq L_t\} \tag{4.32}$$

where $P(L = l)$ and $\sum_{l=0}^{\Theta} P(L = l)$ denote the probability mass function (PMF) and cumulative distribution function (CDF), respectively, of the random variable L_t defined in (4.27).

Proof. Given that $L_t \leq \Theta$, the conditional expectation of Y_{t+1} (i.e., $\mathbb{E}[Y_{t+1}|L_t \leq \Theta]$) is given by

$$\begin{aligned}
\mathbb{E}[Y_{t+1}|L_t \leq \Theta] &= \mathbb{E}[L_{t+1}|L_t \leq \Theta, L_{t+1} \leq \Theta] \cdot P(L_{t+1} \leq \Theta) - \mathbb{E}[\lambda \cdot \mathbb{E}[S_r]|L_t \leq \Theta, L_{t+1} > \Theta] \cdot P(L_{t+1} > \Theta) \\
&= \mathbb{E}[L_{t+1}|L_{t+1} \leq \Theta] \cdot P(L_{t+1} \leq \Theta) - \mathbb{E}[\lambda \cdot \mathbb{E}[S_r]|L_{t+1} > \Theta] \cdot (1 - P(L_{t+1} \leq \Theta)) \\
&= \sum_{l=0}^{\Theta} l \cdot P(L = l) - \lambda \cdot \mathbb{E}[S_r] \cdot (1 - \sum_{l=0}^{\Theta} P(L = l))
\end{aligned}$$

Thus, by comparing the current reward, $Y_t(L_t) = L_t$, with the one expected at the next stage, we find that the UPF placement readjustment must be triggered at the first time instance t such that $\mathbb{E}[Y_{t+1}|L_t \leq \Theta] \leq L_t$. ■

According to Theorem 2, the UPF placement must be readjusted at the first time $t \geq 0$ where condition (4.32) is met since, at this moment, the supremum of the expected reward defined in (4.29) is attained. Based on Theorem 1, the 1-SLA stopping rule proposed in Theorem 2 is optimal for the SSR problem, as long as the problem is monotone.

Theorem 3. *For the SSR problem, the 1-SLA presented in* (4.32) *is optimal and maximizes the expected reward defined in* (4.29).

Proof. The SSR problem in (4.30) is monotone when the difference between the expected and current rewards ($\mathbb{E}[Y_{t+1}|L_t \leq \Theta] - Y_t(L_t)$) is non-increasing with L_t. The latter is satisfied if the $\mathbb{E}[Y_{t+1}|L_t \leq \Theta]$ is non-increasing and $Y_t(L_t)$ is non-decreasing. Given that the left side of expression (4.32) remains constant and its right side increases over L_t as long as the number of sessions with latency violations do not exceed the established QoS threshold ($L_t \leq \Theta$), the difference $\mathbb{E}[Y_{t+1}|L_t \leq \Theta] - Y_t(L_t)$ is not increasing. Therefore, the 1-SLA rule proposed in (4.32) is optimal for the SSR problem. ■

We assume that, at each instant, an agent is measuring the offered QoS and verifying condition (4.32) to determine when to readjust the UPF placement. Thus, a UPF placement reconfiguration will be triggered at the first time instant t where the current reward ($Y_t = L_t$) is at least as good as the expected reward at the next observation ($Y_t \geq \mathbb{E}[Y_{t+1}]$). In the case of overpassing the QoS threshold ($L_t > \Theta$), a placement reconfiguration is immediately triggered. After each reconfiguration event, the decision process is restarted with $t = 0$ and $L_0 = 0$.

4.3 Evaluation and Results

This section presents the simulation results obtained for the proposed solutions (i.e., UPR model and SSR scheduler). It begins by describing the simulation setup used during the conducted experiments. Afterward, we investigate the effects of the conceived solution for placement reconfiguration by considering various sets of weight factors. Finally, we evaluate the performance of the SSR mechanism and compare its behavior against three baselines.

4.3.1 Simulation Setup

The performance of the envisioned solutions for the dynamic UPR was evaluated in a medium-scale 5G scenario of 5x5 km^2. This setup represented an urban city in a MEC ecosystem in which edge nodes and aggregation points were co-located along with access nodes. The access nodes had different coverage radii according to their location in the city. In dense areas, the inter-site distance was 200 m, while they were spaced 500 m apart in less-populated regions. Additionally, 13 MEC servers with a coverage radius of 1 km and a processing capacity of 15 CPUs were available to host UPFs. We assumed that the topology of the AP was a full mesh where every AP had a direct link with the others.

For the service demands, we considered 1000 users, each one with an active PDU session. These sessions require just one UPF and 0.1 CPU to be served and have a service latency requirement of 1 ms. The users represented vehicles whose mobility was modeled in a downtown model using the mobility patterns generator CityMob[1]. In this model, the vehicle's traffic density was not uniformly distributed but was higher in downtown, where the users move more slowly

[1] http://www.grc.upv.es/Software/oldsw/citymob/citymob.rar

than in the outskirts of the city [149]. Table 4.3 summarizes the parameters required by the CityMob program and their specified values.

Table 4.3: Simulation parameters used in CityMob.

Notation	Description	Value
m	Mobility model	3[1]
n	Number of users	1000
t	Simulation time (s)	60000
s	Maximum speed of the users (m/s)	40
d	Distance between streets or block sizes (m)	100
w x d	Dimensions of the grid (km^2)	5x5
a	Number of accidents	0
x, y, X, Y	Downtown limits (km)	1, 1, 2, 2
p	Probability of starting a user located in the downtown	0.45

[1] The m parameter takes numeric values to indicate the mobility model (e.g., $m = 3$ for the downtown model).

The initial UPF placement and their assigned PDU sessions were determined using the UPR model by removing the terms that depend on previous time instances (i.e., migration and reassignment costs) from the objective function and setting the indicators P_c^s, X_c, and V_c^u at zero. Specifically, we considered the following weight factors $\alpha_1 = 0.3$, $\alpha_2 = 0.3$, and $\alpha_3 = 0.4$ for the deployment, running, and routing costs, respectively. Table 4.4 provides the values of the simulation parameters used for the UPR problem.

Table 4.4: Simulation parameters for the UPR.

Notation	Description	Value
N_c	Set of candidate locations (i.e., ENs)	13
N_r	Set of access nodes	121
N_a	Set of aggregation points	13
U	Set of already deployed UPFs	7
S	Set of PDU sessions	1000
T_r	RTT delay in the RAN (μs)	500
T_u	Processing time of UPFs (μs)	100
T_a	Processing time of AP (μs)	15
T_d	Processing time of DN (μs)	100
–	Propagation delay in optical links ($\mu s/km$)	5
–	Number of gNBs per MEC server	[8,10]
L_{ser}^s	Service latency requirement of a PDU session (ms)	1
d^s	Computing demand of a PDU session (CPU)	0.1
Cu_c	Capacity of a UPF (CPU)	15

4.3.2 UPR Solution Performance

This subsection evaluates the performance of the proposed ILP model for the UPF placement reconfiguration. It focuses on the results provided by the optimization objectives under study by

analyzing their associated costs and their effects on more general aspects of the system. Several metrics were considered, such as the number of deployed and migrated UPFs, maximum and average propagation delays in the segment ANN-UPF, average UPF imbalance, and reassigned sessions. Additionally, metrics related to the offered QoS (i.e., sessions with latency violations), the number of reevaluation events (RE), and average execution times were collected. We determined the optimal reconfiguration time based on the SSR mechanism by considering a maximum tolerance threshold of 3% of sessions with latency violations.

Table 4.5 showcases the obtained results for different sets of weight factors in the objective function and a simulation period of five hours. Given the high number of weight combinations, we did not include all the Pareto-optimal solutions but rather a representative set. Mainly, we focused on the analysis of the importance of the routing cost (C_{rou}) since the activation of a placement reconfiguration event by the envisioned SSR strategy depends directly on the optimization of this parameter to reduce the number of sessions with latency violations.

Table 4.5: Simulation results for the UPR model for different sets of weight factors.

ID	**Weight Components**					**Metrics**											
						Max Delay (μs)	**Mean Delay (μs)**	**No. UPF**	**No. Mig.**	**Imb (%)**	**No. Relocations**			$\sum_t L_t$	**No. RE**	**Execution Time (s)**	
	α_1	α_2	α_3	α_4	α_5	**Aver**	**Aver**	**Aver**	**Max**	**Aver**	**Max**	**Aver**	**Total**	**Total**		**Max**	**Aver**
a	0.2	0.2	0.1	0.3	0.2	18.42	6.43	7	0	0.29	67	48	3763	4878	79	26.79	18.32
b	0.2	0.2	0.2	0.2	0.2	18.32	6.32	7	0	0.28	71	50	4808	4942	96	36.92	20.96
c	0.2	0.2	0.3	0.1	0.2	17.55	5.66	7	0	0.23	124	94	6971	4595	74	46.88	29.27
d	0.2	0.2	0.4	0	0.2	17.58	4.97	7	0	0.30	213	160	9300	4410	58	56.27	31.86
e	0.2	0.2	0.4	0.2	0	12.91	3.06	7	0	0.32	745	716	19343	3683	27	53.72	38.55
f	0.2	0.1	0.5	0.1	0.1	14.23	3.09	7	1	0.31	675	545	11987	2327	22	45.40	19.88
g	0	0.3	0.5	0.2	0	9.75	2.10	7	3	0.28	880	782	3127	2301	4	52.92	33.12
h	0.1	0.1	0.6	0	0.2	12.75	3.54	7	1	0.25	422	422	422	164	1	8.07	8.07
i	0.1	0.2	0.6	0	0.1	10.03	2.15	7	3	0.28	772	634	5704	2296	9	21.55	17.96
j	0.1	0.1	0.7	0	0.1	10.62	2.15	7	3	0.28	772	642	4492	2254	7	9.42	7.56
k	0.1	0.1	0.7	0.1	0	7.91	2.07	8	0	0.73	805	805	805	88	1	10.27	10.27
l	0	0.1	0.8	0.1	0	9.01	1.76	8	2	0.63	842	842	842	88	1	5.23	5.23
m	0	0.1	0.9	0	0	9.01	1.44	9	2	0.66	850	850	850	88	1	5.29	5.29

The propagation delays in the segment ANN-UPF (L_{ac}) decreased with the importance of the routing cost (α_3). Specifically, the maximum and average delays after reconfiguring the placement were decremented around two and three times for experiments with $\alpha_3 \geq 0.7$ compared to the delay obtained by experiments with lower importance in the routing cost (i.e., row IDs **a-i**). By more closely examining the tests for which the optimization of the routing term was equally important (i.e., **d–e**, **f–g**, **h–i**, and **j–k**), we found that these outcomes were not only conditioned by the use of higher values in the α_3 weight but also by the importance of other terms in the objective function. These pairs of experiments evidenced more significant reductions in the maximum and average delays when omitting or decrementing the weight factor linked to the reassignment cost component.

Moreover, we also noticed that a higher weight factor value did not necessarily imply a better performance of its associated cost, as we have to contemplate all terms under optimization and their attendant trade-offs. An example of these trade-offs can be observed by comparing experiments with row IDs **f** and **e** or **h** and **g**. In this case, experiments **e** and **g** had better performance in terms of delay than samples **f** and **h**, respectively, despite having a smaller

α_3 factor. The cause of this outcome was the omission of the reassignment cost component in experiments **e** and **g**.

Conversely, we observed an increased number of deployed and migrated UPFs and reassigned sessions since more transformations were produced to reduce the routing cost further. This behavior was more remarkable for values of $\alpha_3 \geq 0.5$, where the placement reevaluation required the migration or/and deployment of additional UPFs. For those experiments where new UPFs were placed, the average UPF imbalance increased by more than 30%. A considerable increase was also observed in the maximum and average numbers of reassigned sessions during reconfiguration events. In particular, for $\alpha_3 \geq 0.5$, most sessions were relocated with typical values ranging between 60% and 80%, while for $\alpha_3 \leq 0.3$, less than 13% of the sessions were reassigned. These results were also reflected in the total number of reassigned sessions, which increased with the routing component's importance in the objective function. This trend was more noticeable in experiments in which the UPF placement remained unchanged regarding the number of deployed instances and selected locations (i.e., experiments **a–e**) since frequent reconfiguration events were necessary due to the high number of sessions with latency violations. In contrast, fewer reconfigurations were triggered for the remaining experiments, translating into lower session reassignments.

Overall, the number of placement reevaluations decreased with the increasing importance of the routing cost, as seen in Table 4.5. This was because more users were reassigned to nearer UPFs to reduce the impact of the routing component in the objective function. However, this behavior was not steady but instead presented some exceptions (i.e., row IDs **b**, **i**, and **j**) in which a higher value in α_3 with reference to previous experiments resulted in more reconfigurations. In some cases, this increment was caused by a variation in the importance of the reassignment cost (α_5), as evidenced when comparing experiment **j** with **k**. Thus, when this term (i.e., C_{rea}) was considered ($\alpha_5 \geq 0.1$) in the optimization problem, fewer sessions were reassigned, increasing the probability of latency violations and, therefore, re-computation events.

However, by comparing experiments **b** or **i** with others with similar weights, such as **a** and **h**, respectively, we can see that the reassignment cost does not cause this behavior. The latter may be due to different reconfiguration conditions, such as session mapping, user locations, and affected sessions. Additionally, all these use cases produced different placement configurations, even when their weight factors were similar. In the end, these slight differences accumulate, and their effects are reflected in global metrics, such as the overall number of reconfigurations and sessions with poor QoS.

Concerning the execution times of the proposed ILP model, the model always determined the optimal UPF placement and session mapping reconfiguration in less than a minute. Concretely, it provided maximum values ranging from 5–57 s and average computing times of approximately 30 s.

These experiments revealed a complex and robust relationship among the optimization

objectives of the UPR problem. Hence, optimizing one or more parameters significantly impacted the others since we dealt with conflicting objectives. Specifically, we can classify these objectives into three main categories. The first is related to the number of UPFs formed by the deployment and running costs. The second is linked with the relocation of PDU sessions and comprises the migration and reassignment components, while the third includes the routing cost. In general, no single best solution addresses multi-objective optimization problems but rather a set of multiple optimal solutions that form the Pareto-Fronts. Therefore, selecting one solution over another depends on optimization goals.

4.3.3 Dynamic Scheduling for the UPR

In this subsection, we assess the performance of the conceived SSR mechanism to determine the optimal UPR reconfiguration time. We analyzed its performance in terms of the number of reconfiguration events, number of reassigned sessions during the reconfiguration events, number of sessions with latency violations at the reconfiguration moment, and QoS status at each sampling time. Moreover, to evaluate the benefits of the SSR solution, we compared it with the following benchmarks:

- *Periodic placement scheduling (PPS):* The UPF placement is readjusted periodically at fixed time intervals. In particular, two variants with low and high reconfiguration periods (i.e., every 5 and 60 minutes) were selected; these were referred to as PPS_P5 and PPP_P60, respectively.

- *Dynamic placement scheduling (DPS):* This strategy embraces the scheduling solution presented in the related work [46]. This approach determines the optimal reconfiguration time according to the system's maximum allowed number of cumulative latency violations. A maximum tolerance threshold of 1000 latency violations was considered.

We conducted our experiments using a set of 1000 users, each with one active PDU session. The effectiveness of the SSR mechanism and the baselines was evaluated by considering two sets of weights in the objective function of the UPR model. The first set, denoted as weight_set_1, considered that all cost components in (4.6) are equally important (i.e., $\alpha_i = 0.2$). In contrast, the second set of weights (weight_set_2) gave more importance to the routing cost and ignored the optimization of the terms related to deployment and reassignment expenditures ($\alpha_1 = \alpha_5 = 0$, $\alpha_2 = 0.3$, $\alpha_3 = 0.5$, and $\alpha_4 = 0.2$).

For these simulations, we ran the system for 10 hours and collected QoS metrics every minute for a total of 600 samples. The expected number of relocated sessions during a reconfiguration event was estimated based on the obtained results for the PPS with a reconfiguration period every 5 minutes (PPS_P5). Furthermore, a maximum QoS tolerance threshold (Θ) of 3% of active sessions with latency violations was established for the SSR scheduler. The instantaneous number of sessions with latency violations of the system was modeled as a Poisson distribution with a

mean rate of 21 sessions ($\mu = 21$). We fitted this distribution based on the observed number of latency violations at each time instance for a UPF placement without reconfiguration during the entire simulation time (see Fig. 4.1). In addition, we assumed each user's mobility was independent of the mobility of other users; that is, the movement of one user does not affect other users.

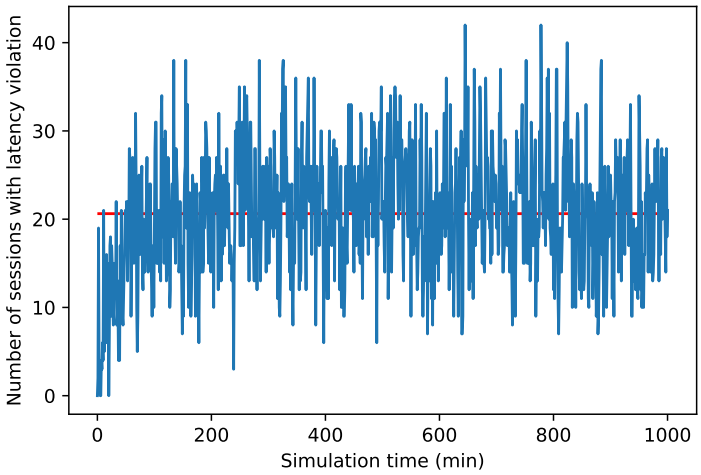


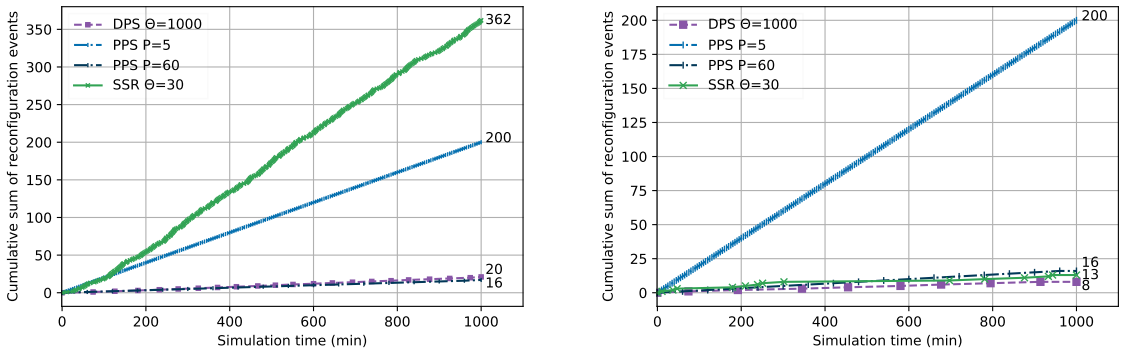
Figure 4.1: Number of sessions with latency violations over time for a static UPF placement.

4.3.3.1 Reconfiguration Costs

In the following sections, we analyze the performance of the SSR and baseline solutions according to several metrics associated with the reconfiguration costs.

Number of reconfiguration events: Figure 4.2 shows the cumulative sum of UPF placement readjustment events for both sets of weight factors. When all optimization objectives were equally important (weight_set_1), the SSR strategy provided the worst results, triggering almost twice as many reconfiguration events as the PPS_P5 approach, with a frequency of about 2–3 minutes between reconfigurations. The leading cause of this behavior lay in the SSR mechanism, which unlike the others, directly depended on the instantaneous values of the selected QoS metric (i.e., L_t). In particular, this set of weights produced minor transformations in the placement configuration, mostly reassigning only the sessions with latency violations. The latter resulted in a UPF placement with poor QoS given the frequent and persistent number of users with latency violations.

In contrast, significant improvements in the number of readjustment events were achieved by the conceived mechanism when considering the second set of weight factors (see Fig. 4.2(**b**)) since this set allowed for more transformations in the placement configuration. This was possible because the set favored the routing cost and ignored the session reassignment term. In this case, the SSR method not only outperformed both periodic schedulers with an average reconfiguration time of 70 minutes but obtained results similar to the DSP baseline. Moreover, the number



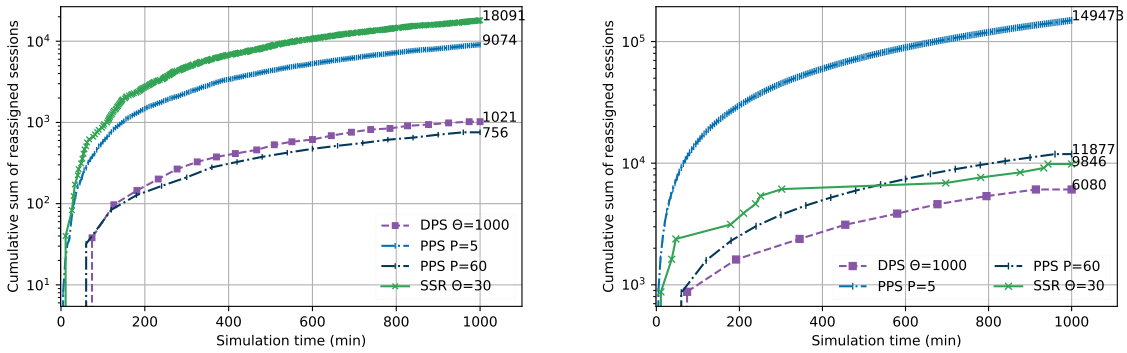
(**a**) weight_set_1 (**b**) weight_set_2

Figure 4.2: Cumulative sum of UPF placement reconfiguration events.

of placement reevaluation events triggered by the SSR and DPS mechanisms decreased when the second set of weights was selected, as seen in the sub-figures. Specifically, for this set, the SSR and DPS solutions required around 95% and 60% fewer reconfigurations, respectively. By contrast, this metric remained constant for the periodic approaches since they were unaware of the placement conditions.

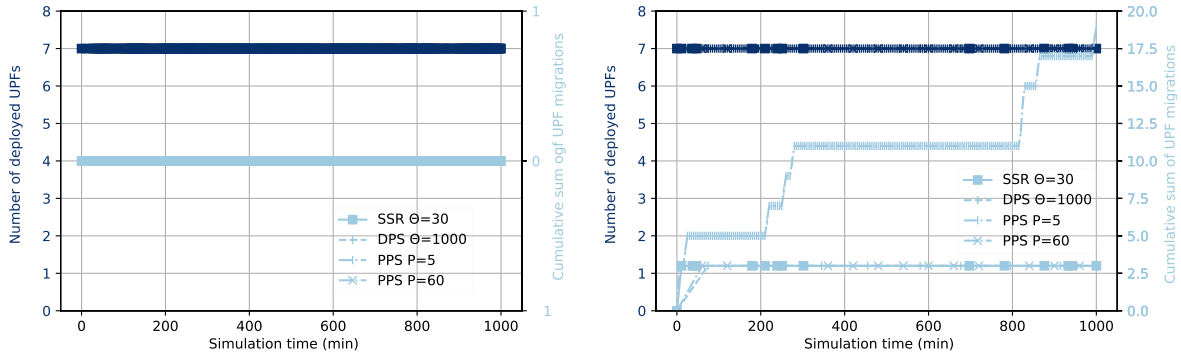
Figures 4.3 and 4.4 show the effects of the selected weight factors on the number of relocated PDU sessions, as well as the number of deployed and migrated UPFs during the placement reconfiguration. As seen in the figures, the cumulative sum of reassigned sessions and migrated UPFs rose dramatically when the second set of weights was used. This behavior was more remarkable in the periodic schedulers, especially PPS_P5, since they produced the same number of reevaluation events in both scenarios. The **number of reassigned sessions** for the periodic mechanisms increased by more than 15 times compared with the first weights' obtained results. Moreover, as expected, the scheduler solutions with more reconfiguration events always provided the worst results regarding the number of relocated sessions. Specifically, for the weight_set_1, the SSR and PPS_P5 schedulers produced many more reassignments than the other two approaches, while for the second set of weights, SSR obtained similar results to PPS_P60 and DPS.

Figures 4.4(**a**) and 4.4(**b**) depict the **number of deployed and migrated UPFs**, in dark and light blue, respectively, for both sets of weights. No variation in the overall number of deployed UPFs produced for either of the analyzed sets was observable during the entire simulation time. This behavior was because the deployment cost always had equal or higher importance than the migration and reassignment components. Alternatively, modifications in the UPF locations were produced for all schedulers when using weight_set_2. Specifically, SSR, DPS, and PPS_P60 only modified UPF placement during the first reconfiguration event with three migrations, whereas PPS_P5 activated 15 migration events with one UPF migration most of the time. Overall, the PPS_P5 baseline had the worst performance, with 19 UPF migrations.



(**a**) weight_set_1 (**b**) weight_set_2

Figure 4.3: Cumulative sum of session relocations.



(**a**) weight_set_1 (**b**) weight_set_2

Figure 4.4: Number of deployed and migrated UPFs.

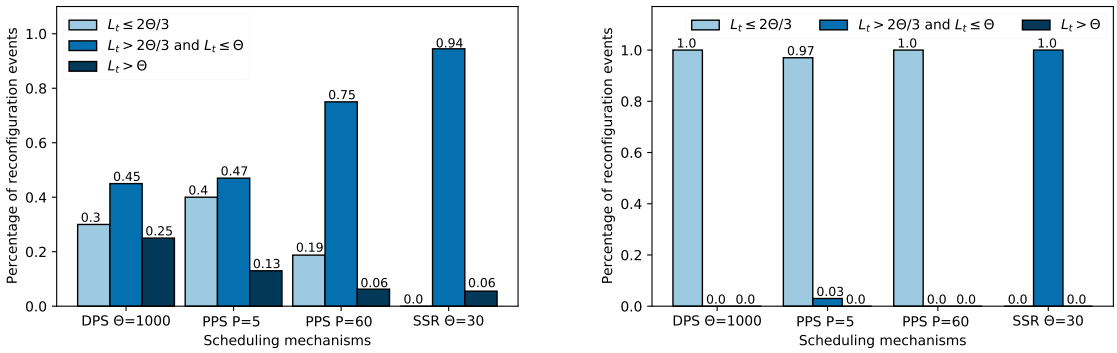
4.3.3.2 QoS Metric

To assess the suitability of the scheduler approaches, we analyzed the behavior of the L_t parameter and the overall system QoS.

We began by analyzing the **system QoS status at the reconfiguration moment** to verify under which conditions these events were triggered. We classified the QoS into three levels concerning the relationship between the selected QoS metric and the tolerance threshold to create a more intuitive representation. The system QoS was considered good for a low number of latency violations ($L_t \leq \frac{2}{3}\Theta$), acceptable for moderate latency violations ($L_t > \frac{2}{3}\Theta$ and $L_t \leq \Theta$), and poor for L_t values above the established threshold ($L_t > \Theta$).

Figure 4.5 showcases the obtained results for both weight sets. As shown, the proposed mechanism provided the best performance since it activated the reconfiguration events when the number of sessions with latency violations was considerably high (i.e., $L_t > \frac{2}{3}\Theta$). Moreover, it did

not decide to activate a placement readjustment when the value of this metric was low $L_t \leq \frac{2}{3}\Theta$) and thus avoided unnecessary reconfigurations. Conversely, many placement reevaluations triggered by the baselines were determined when the latency violations were scarce or above the established QoS upper threshold. Specifically, for the first set, the percentage of UPR events triggered under good QoS ranged from 19% to 40% while between 6% and 25% of the events were activated when the system QoS had deteriorated. The benchmarks performance was even worse for weight_set_2. For this set, they executed the UPR reconfiguration under good QoS levels ($L_t \leq \frac{2}{3}\Theta$) most of the time, as shown in Fig. 4.5(**b**). In this scenario, the SSR solution had an outstanding performance as it always reconfigured the placement when the QoS threshold was close to being exceeded.



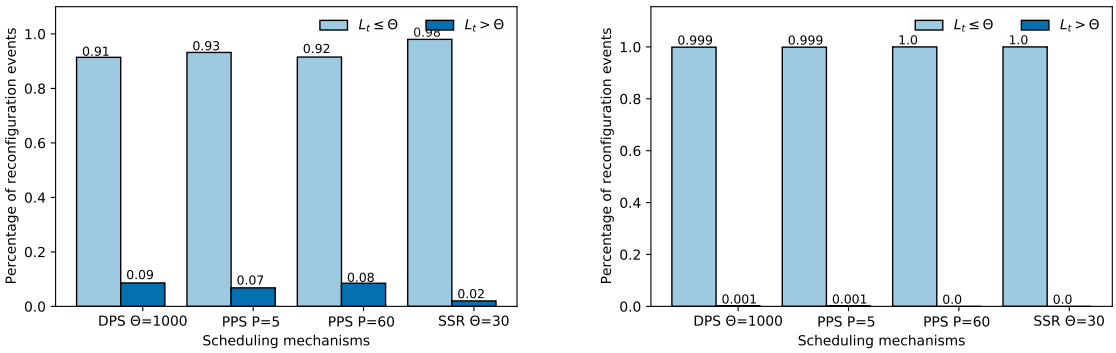
(**a**) weight_set_1

(**b**) weight_set_2

Figure 4.5: Percentage of reconfigurations grouped according to L_t values with reference to the Θ threshold at the reconfiguration time.

Figure 4.6 also represents the **QoS status** regarding the established Θ threshold during the entire simulation time. For the first set of weights (see Fig. 4.6(**a**)), none of the schedulers could avoid the occurrence of QoS degradation events due to the high number of sessions with poor QoS. Nevertheless, the proposed mechanism offered a better QoS than the rest, with only 2% of the samples with bad QoS. This outcome was expected since the SSR solution was designed to diminish events with bad QoS. Most of the time, the tolerance threshold was exceeded almost immediately after a reconfiguration due to the high number of sessions with latency violations. These were mainly due to high user mobility and limited placement transformations (e.g., reassigned sessions and UPF migrations) performed during the reconfigurations.

In contrast, when the second set of weights was chosen, more stable UPF placement and session mapping were produced at the cost of higher transformations and, therefore, expenditures. In this case, the number of reconfiguration events was decreased for the QoS-aware scheduling approaches (i.e., SSR and DPS). However, a significant improvement in the overall QoS was achieved since the Θ threshold was hardly overpassed. Additionally, the SSR strategy reduced the number of events with bad QoS to zero, thus providing similar or better results to solutions



(**a**) weight_set_1

(**b**) weight_set_2

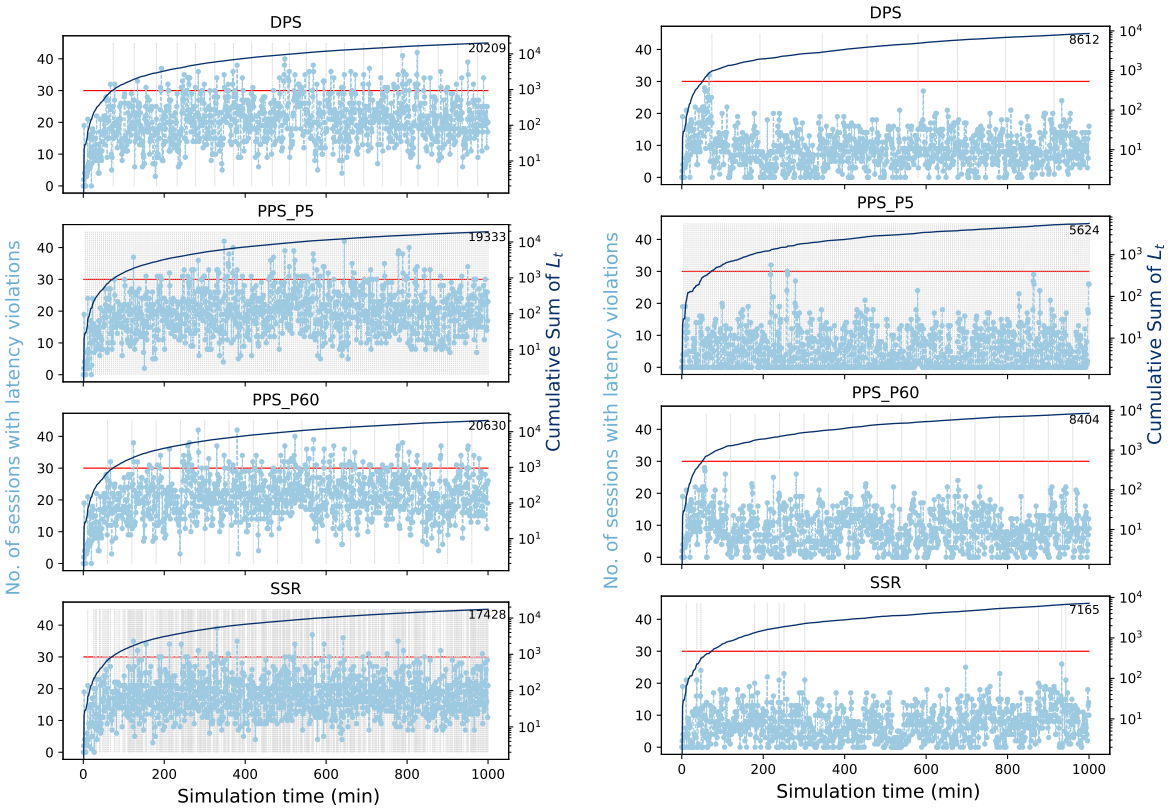
Figure 4.6: Values of L_t with reference to the Θ threshold during the simulation time.

with a greater number of reconfiguration events (i.e., PPS_P5 and PPS_P60).

Figure 4.7 presents the instantaneous and cumulative values of sessions with latency violations. Moreover, it also indicates the reconfiguration events activated for each scheduler with dashed grey lines. As shown, the SSR solution had a lowest number of events overpassing the QoS threshold ($L_t > \Theta$) for both sets of weights. Furthermore, in the cases where this threshold was violated, it immediately executed a placement readjustment, which was not the case for the other approaches, which did not account for the instantaneous values of the offered QoS. Most of the time, the baselines reconfigured the placement when the number of sessions with latency violations was low. This behavior was more remarkable for the second set of weights than the first, as shown in Fig. 4.7(**b**).

Regarding the cumulative sum of latency violations, this metric was highly related to the number of readjustment events. As shown in Fig. 4.7(**a**), the overall number of sessions with bad QoS decreased as the reconfigurations increased. In this case, the SSR approach provided the best results, with reductions of at least 10% compared to the baselines. Similarly, for the second set of weights, the scheduler with more reconfigurations (i.e., PPS_P5) provided the lowest amount of sessions with latency violations. Nonetheless, SSR outperformed the other two benchmarks (DPS and PPS_P60) since it could keep the QoS parameter under acceptable values all the time.

Through the experiments and results, we validated the effectiveness and assertiveness of the conceived scheduling solution for the UPR problem. We showed that the SSR mechanism could reduce the number of events with bad QoS by accounting for instantaneous values of the selected metric (L_t). Specifically, SSR decreased the number of events with bad QoS by at least 5% compared to established baselines for UPF placement configurations with frequent and elevated sessions with latency violations. Furthermore, we found that the proposed SSR mechanism ensured the established QoS levels and avoided unnecessary placement recalculations. Its benefits were more remarkable when combined with UPR models that allowed further transformations



(**a**) weight_set_1 (**b**) weight_set_2

Figure 4.7: Number of sessions with latency violations and their cumulative sum over time.

during the placement reconfiguration. In contrast, periodic reconfiguration strategies produced a lower or higher number of reevaluations according to the selected period. However, they often resulted in unnecessary reconfigurations or poor QoS.

4.4 Conclusion

This chapter presented the dynamic UPR problem. To address this problem, we proposed a multi-objective ILP model (CUPR) to determine the optimal placement and mapping configuration along with a scheduling mechanism (SSR) based on OST to decide the best re-computation time.

The CUPR solution seeks the optimal UPF placement (number of UPFs and candidate locations) and PDU session mapping that allows more efficient use of the infrastructure resources while guaranteeing service latency requirements and minimizing service interruption. To this aim, its objective function consideres several cost components associated with UPF deployment and operation (e.g., running and migration costs), as well as the system QoS expressed in terms of network response time (routing cost) and session reassignments. Based on experimental simulations, we demonstrated that a unique best solution does not exist for solving the UPR

problem since doing so depends on the selected cost components and their respective importance in the objective function. For instance, better QoS regarding lower network response times and fewer readjustment events can be achieved by increasing the importance of the routing cost. However, this came at the expense of more transformations during the placement reconfiguration, such as UPF deployments and migrations or session reassignments.

In addition, the conceived SSR strategy was shown to be a practical and straightforward approach for proactively determining the optimal reconfiguration time. Concretely, it revealed that, by accounting for instantaneous values of the selected QoS metric (i.e., sessions with latency violations), the desired levels of QoS can be guaranteed most of the time, and the number of placement reevaluation events can be significantly reduced. Therefore, substantial improvements were obtained by selecting a reconfiguration model that favored the routing cost over others.

CHAPTER 5

STATIC UPF PLACEMENT AND CHAINING

This chapter is based on:

- **I. Leyva-Pupo**, and C. Cervelló-Pastor, "Efficient solutions to the placement and chaining problem of User Plane Functions in 5G networks," *Journal of Network and Computer Applications*, vol. 197, p. 103269, 2022.

We addressed the UPP in previous chapters by considering that a single UPF instance forms PDU session data paths. This assumption is suitable for general planning and operation purposes. For example, in some deployment scenarios in which the number of candidate locations or available resources is scarce, a single UPF instance can support various functionalities. Nevertheless, 5G standards allow PDU sessions to be served simultaneously by multiple UPFs, chained together and dividing load and functionalities. Based upon this fact, the work in this chapter targets to extend previous solutions by introducing chaining requirements to the UPP. In short, this chapter presents the UPF placement and chaining (UPC) problem.

The chapter provides an exact solution and two heuristic-based approaches to tackle the UPC problem. These solutions aim to minimize network operator expenditures and enhance the QoS measured in terms of network response time. The ILP model is bound by stringent service latency requirements and physical network capabilities, such as server processing capacity and link bandwidth. Moreover, it encompasses several aspects, such as UPF-specific requirements, VNF order in the SFC, and routing path. To overcome scalability issues associated with exact solutions, we design a heuristic and a metaheuristic algorithm called priority and cautious UPF placement and chaining (PC-UPC) and simulated annealing-based UPF placement and chaining (SA-UPC), respectively. These approaches introduce several mechanisms to boost their performance. Specifically, PC-UPC avoids rejections of SFCRs by prioritizing mapping the most

demanding services and considering the effects of VNF mapping decisions on the subsequent VNFs forming the chain. The SA-UPC approach incorporates several strategies, such as variable Markov chain length (VMCL) and restart-stop, which significantly improve the computation time and quality of the output solution.

The chapter is structured as follows. Section 5.1 introduces the UPC problem and the system model and notation. In Section 5.2, we present an optimal solution to the UPC problem. Next, Section 5.3 describes the proposed heuristic and metaheuristic solutions and analyzes their complexity. The obtained simulation results are discussed in Section 5.4, and finally, Section 5.5 concludes the chapter.

5.1 Problem Statement: Static UPF Placement and Chaining

In 5G networks, unlike previous mobile network generations, the UPF can be divided into different functional roles called micro-services, which can be chained together and steered as necessary. This functionality specialization allows PDU sessions to be served simultaneously by multiple UPFs. However, this complicates the UPF placement problem due to the presence of different UPF roles with diverse and specific requirements that must be chained in a specific order by accounting for their inter-dependency.

In this chapter, we define the UPC problem as follows: *Given a virtualized infrastructure and a set of PDU session requests, the optimal number and location of UPF instances, as well as their routing paths must be determined to optimize the overall provisioning cost and QoS. Moreover, session requirements such as VNF order and latency must be met while considering physical and virtual network infrastructure limitations, such as available resources and topology.*

Thus, we deal with the problem of determining the optimal placement and chaining of UPF instances at the network edge so that 5G service requirements are satisfied while minimizing expenditures.

5.1.1 System Model and Considerations

We represent the 5G network topology as a directed graph G(N, E), where N and E indicate the set of nodes and links, respectively. The set of network nodes is formed by VNF candidate locations (N_c), aggregation points (N_a), and access nodes (N_r). Each candidate node $c \in N_c$ has an associated processing capacity (C_c), expressed in the number of CPUs.

A physical link between two nodes $u, v \in N_c$ in the direction u to v is denoted as $(u, v) \in E$. Each link is characterized by a propagation delay ($d_{u,v}$) and a bandwidth capacity ($\beta_{u,v}$). The link latency indicates the time required for a data package to go from a source to a target node. It is computed in terms of the propagation delay and the processing time of the transmission nodes. Additionally, the set of all paths (P) in the network, as well as the subset of paths between a specific pair of nodes ($P_{n,m} \subset P$), are included. Every path $p \in P$ is identified by its endpoints (n,

m) and an ID (h) to distinguish different paths between the same pair of nodes. Moreover, the mapping of a physical link $(u,v) \in E$ to a specific path $p \in P$ is described by the parameter $H^p_{u,v}$, where $H^p_{u,v} = 1$ if the link (u, v) belongs to path p, 0 otherwise.

Table 5.1 provides the notations related to the physical and virtual networks. Different VNF types (T) can be deployed in the network, where $t \in T$ denotes a specific type. In particular, we represent the UPF types as follows; t=1: aUPF, t=2: miUPF (i.e., UL-CL and BP), and t=3: IUPF. Moreover, each VNF type characterizes by a processing capacity (C_t), a processing delay (d_t), and a maximum number of instances that can be deployed (I_t).

Table 5.1: Used notation for physical and virtual networks.

Notation	Set	Param.	Description
N	x		Set of all network nodes
N_r	x		Set of access nodes, $N_r \subset N$
N_c	x		Set of VNF candidate locations (e.g., MEC servers), $N_c \subset N$
N_a	x		Set of aggregation points, $N_a \subset N$
E	x		Set of physical links
P	x		Set of paths between all network nodes
$P_{n,m}$	x		Set of paths between nodes n and m, $(n,m \in N)$, $P_{n,m} \subset P$
T	x		Set of VNF types
C_c		x	Processing capacity at candidate location $c \in N_c$
C_t		x	Processing capacity of VNF of type $t \in T$
$\beta_{u,v}$		x	Bandwidth capacity of link (u,v)
$d_{u,v}$		x	Latency associated to link (u,v)
d_p		x	Latency associated to path $p \in P$
d_t		x	Processing delay of VNF of type $t \in T$
I_t		x	Maximum number of instances of type $t \in T$
$H^p_{u,v}$		x	1 if path $p \in P$ is mapped to physical link $(u,v) \in E$
V^t_n		x	1 if node $n \in N$ supports VNFs of type $t \in T$
$T^{f,t}_s$		x	1 if VNF $f \in F_s$ in SFCR $s \in S$ is of type $t \in T$
$O^{f,g,b}_s$		x	1 if VNF f goes just before VNF g $(f,g \in F_s)$ in branch $b \in B_s$ of SFCR $s \in S$
$Q^{f,b}_s$		x	1 if VNF $f \in F_s$ is present in branch $b \in B_s$ of SFCR $s \in S$

The set of active PDU sessions is represented by S. We model PDU session requests as SFCRs and represent their properties by a 10-tuple <u_s, r_s, F_s, C_s, β_s, L_s, B_s, $T^{f,t}_s$, $O^{f,g,b}_s$, $Q^{f,b}_s$>. The user ID, access node, and VNF associated with a given SFCR are denoted as u_s, r_s, and F_s, respectively. For simplicity, we define the source access node of a service request as a VNF of type $t = 0$ and extend the set of VNFs forming the SFCR to include this node ($F^+_s = F_s \cup r_s$). Every PDU session is characterized by a processing capacity demand (C_s), a required bandwidth (β_s), and a maximum E2E user plane delay (L_s). Additionally, parameters B_s, $T^{f,t}_s$, $O^{f,g,b}_s$, and $Q^{f,b}_s$ specify the number of branches in an SFCR, the VNF type, VNF order, and VNF presence in each branch, respectively. Our SFCR model does not include the destination nodes since we assume that they are DNs co-located with the aUPFs, and therefore, their location before the UPF placement is unknown. Table 5.2 summarizes the notation used for modeling SFCRs.

In this study, we assume that PDU sessions can have different characteristics (e.g., number and type of constituent UPFs and SFC topology), which may vary according to several aspects,

Table 5.2: Sets and parameters related to SFCRs.

Notation	Description
S	Set of PDU sessions (SFCRs)
u_s	ID of the user requesting PDU session $s \in S$
r_s	Access node ($r_s \in N_r$) of SFCR $s \in S$
F_s	Set of VNFs forming SFCR $s \in S$
$\|F_s^t\|$	Number of VNFs of type $t \in T$ in SFCR $s \in S$
C_s	Computing resources required by SFCR $s \in S$
β_s	Bandwidth capacity required by SFCR $s \in S$
L_s	E2E latency requirement of SFCR $s \in S$
B_s	Number of aUPFs (branches) in SFCR $s \in S$

such as service type and network conditions. In particular, we consider three basic topologies when modeling the SFCRs as depicted in Fig. 5.1. These topologies account for the main UPF functionalities, although they can be extended to include other VNF types or UPF roles and combined to create more complex structures. Additionally, according to 3GPP specifications [7, 8], the following aspects must be considered when orchestrating PDU sessions in 5G networks: (1) at least one UPF is required to act as a PSA; (2) all UPFs acting as PSAs must terminate the data path with the DN; (3) more than one iUPF (e.g., IUPF, UL-CL, and BP) may be inserted in the path, but only one connects with the ANN via the N3 interface, except for session continuity during UL-CL/BP relocation [7]; (4) if a UL-CL or BP functionality is inserted in the data path of a PDU session, then multiple PSAs are assigned to this session; and (5) the use of UL-CL and BP is independent of the SSC mode and is mostly linked to QoS metrics or network policy rules.

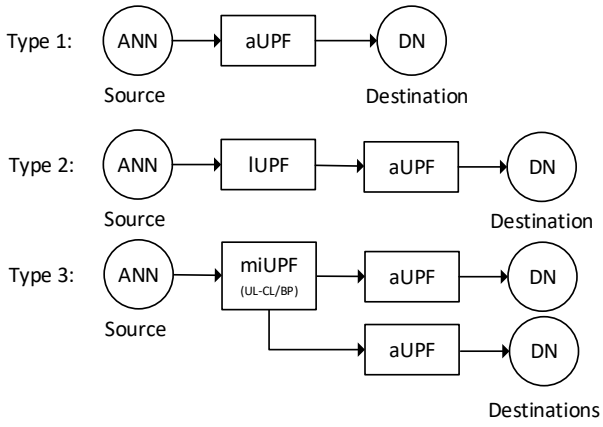


Figure 5.1: Uplink representation of PDU sessions with different SFC topologies.

To simplify the UPC problem, the following assumptions were made:

- The use of the UL-CL or BP functionality is determined by the PDU session type (e.g., IPv4 or IPv6), which is out of this thesis's scope. Hence, both types of UPFs are treated interchangeably as miUPF.
- In the 3GPP technical specifications, there is no restriction concerning the maximum

number of UPFs assigned to a given PDU session. Nevertheless, we limit the number of iUPFs serving a PDU session to one. This assumption avoids extra signaling between SMFs and UPFs and reduces resource utilization in the user plane.

- Every UPF instance has a unique and identical role for all its assigned PDU sessions.
- The propagation delay is constant and proportional to the link lengths.

The decision variables used for the UPC problem solution are described in Table 5.3.

Table 5.3: Used notation for the decision variables.

Notation	Description
w_n	1 if candidate node $n \in N_c$ is open
$x_{i,t,n}$	1 if instance $i \in I_t$ of VNF type $t \in T$ is deployed on node $n \in N$
$z^{f,s}_{i,t,n}$	1 if VNF $f \in F_s$ of SFCR $s \in S$ is mapped to instance $i \in I_t$ of VNF type $t \in T$ placed in node $n \in N$
$a^{f,b,s}_{i,t,n}$	1 if VNF $f \in F_s$ in branch $b \in B_s$ of SFCR $s \in S$ is mapped to instance $i \in I_t$ of VNF type $t \in T$ placed in node $n \in N$
$y_p^{f,g,b,s}$	1 if path $p \in P$ routes traffic between VNFs f and g ($f, g \in F_s$) of branch $b \in B_s$ in SFCR $s \in S$

5.2 Model: Optimal UPF Placement and Chaining

The UPC model aims to optimize provisioning costs and service response time during UPF placement and chaining. Therefore, it contemplates three cost components concerning the activation of candidate nodes, the deployment of VNF instances (i.e., UPF), and the routing of SFCRs. The objective functions of the UPC model can be defined as follows:

$$\text{Min} \sum_{n \in N_c} w_n \tag{5.1}$$

$$\text{Min} \sum_{i \in I_t} \sum_{\substack{t \in T, \\ t \neq 0}} \sum_{n \in N_c} x_{i,t,n} \tag{5.2}$$

$$\text{Min} \sum_{f,g \in F_s^+} \sum_{b \in B_s} \sum_{s \in S} \sum_{p \in P} d_p \cdot y_p^{f,g,b,s} \tag{5.3}$$

Equation 5.1 minimizes expenditures associated with opening candidate locations due to the deployment of VNFs. Likewise, the objective function (5.2) seeks to optimize the VNF deployment cost by reducing the number of instantiated VNFs. Expression (5.3) aims to decrease the traffic routing cost between VNFs that form the SFCs. We have expressed the routing cost regarding the propagation delay between the segments comprising the SFC data path. In this manner, more than one objective (i.e., routing cost and network response time) is optimized simultaneously.

Given the multi-objective nature of the UPR model, we adopt a weighted sum approach with normalized terms to transform it into a single-objective function:

$$\text{Min } \alpha \cdot \sum_{n \in N_c} w_n + \beta \cdot \sum_{i \in I_t} \sum_{\substack{t \in T, \\ t \neq 0}} \sum_{n \in N_c} x_{i,t,n} + \gamma \cdot \sum_{f,g \in F_s^+} \sum_{b \in B_s} \sum_{s \in S} \sum_{p \in P} d_p \cdot y_p^{f,g,b,s} \tag{5.4}$$

where the weight factors α, β, and γ express the relative importance of each term in the objective function, such that $\alpha + \beta + \gamma = 1$

To facilitate reading and improve the legibility of the model, we group its associated constraints into four main categories: VNF, SFC, QoS, and capacity.

VNF constraints: Expressions (5.5)–(5.8) restrict the VNF placement in the candidate locations. In particular, constraint (5.5) limits the maximum number of instances of a given type (I_t) that can be deployed. The parameter I_t can be determined by the infrastructure's available resources or the number of licenses. Expression (5.6) guarantees that each VNF is placed, at most, in one location. Moreover, inequality (5.7) forces the deployment of VNFs on the candidates that are open and have the capabilities to support the particular requirements of the VNF type. Additionally, expression (5.8) forces a candidate to be closed if it has not mapped any VNF instance.

$$\sum_{i \in I_t} \sum_{n \in N_c} x_{i,t,n} \leq I_t \qquad \forall t \in T; t \neq 0 \tag{5.5}$$

$$\sum_{n \in N_c} x_{i,t,n} \leq 1 \qquad \forall i \in I_t, \forall t \in T; t \neq 0 \tag{5.6}$$

$$x_{i,t,n} \leq w_n \cdot V_n^t \qquad \forall i \in I_t, \forall t \in T, \forall n \in N \tag{5.7}$$

$$w_n \leq \sum_{i \in I_t} \sum_{\forall t \in T} x_{i,t,n} \qquad \forall n \in N \tag{5.8}$$

SFC constraints: Constraints (5.9)–(5.19) are associated with the mapping of SFCRs. Inequality (5.9) ensures the assignment of each VNF service comprising an SFCR (i.e., $f \in F_s^+$) to one VNF instance. In addition, constraint (5.10) forces the mapping of the VNF service requests to the candidates hosting VNF instances of the requested type. Moreover, restriction (5.11) avoids the deployment of empty VNFs by ensuring that the launched instances have assigned service requests of at least one PDU session. Expression (5.12) states that each SFCR $s \in S$ must be assigned to at least a minimum number of VNFs of a given type ($|F_s^t|$).

$$\sum_{i \in I_t} \sum_{t \in T} \sum_{n \in N} z_{i,t,n}^{f,s} = 1 \qquad \forall f \in F_s^+, \forall s \in S \tag{5.9}$$

$$z_{i,t,n}^{f,s} \leq x_{i,t,n} \cdot T_s^{f,t} \qquad \forall f \in F_s^+, \forall s \in S, \forall i \in I_t, \forall t \in T, \forall n \in N \tag{5.10}$$

$$x_{i,t,n} \leq \sum_{s \in S} \sum_{f \in F_s} z_{i,t,n}^{f,s} \qquad \forall i \in I_t, \forall t \in T, \forall n \in N_c \tag{5.11}$$

$$\sum_{f \in F_s} \sum_{i \in I_t} \sum_{n \in N_c} z_{i,t,n}^{f,s} \geq |F_s^t| \qquad \forall s \in S, \forall t \in T; t \neq 0 \tag{5.12}$$

To reduce the effects of VNF failures on the SFCs, we define an anti-affinity rule for VNF instances of the same type that serve the same PDU session, see (5.13). Specifically, this constraint stipulates that VNFs of the same kind that serves a PDU session must be deployed in different locations. Therefore, if a failure occurs in a VNF's underlying physical infrastructure, the SFC can remain operative as long as it has a mapped VNF with the same role operating in another

candidate location. Similarly, constraint (5.14) forces the placement of aUPFs and IUPFs (i.e., iUPFs connecting to a single PSA) to different locations. Since IUPFs are inserted in a PDU session data path to guarantee service continuity due to PSA limitations such as coverage area or transport network.

$$\sum_{f \in F_s} \sum_{i \in I_t} z_{i,t,n}^{f,s} \leq 1 \qquad \forall s \in S, \forall t \in T; t \neq 0, \forall n \in N_c \tag{5.13}$$

$$\sum_{f \in F_s} \sum_{i \in I_t} z_{1,i,n}^{s,f} + \sum_{f \in F_s} \sum_{i \in I_t} z_{3,i,n}^{s,f} \leq 1 \qquad \forall s \in S, \forall n \in N_c \tag{5.14}$$

Restriction (5.15) ensures that the mapping of a VNF service $f \in F_s^+$, comprising an SFCR, to a given VNF instance is conditioned by a service request of this VNF type from at least one of its branches. Likewise, inequality (5.16) expresses that a VNF instance can serve a branch of an SFCR as long as the VNF has been mapped onto the network. Additionally, a VNF service request associated with a given branch can only be assigned to a VNF instance if the SFC topology requires this VNF type; see (5.17).

$$z_{i,t,n}^{f,s} \leq \sum_{b \in B_s} a_{i,t,n}^{f,b,s} \qquad \forall f \in F_s^+, \forall s \in S, \forall i \in I_t, \forall t \in T, \forall n \in N \tag{5.15}$$

$$a_{i,t,n}^{f,b,s} \leq z_{i,t,n}^{f,s} \qquad \forall f \in F_s^+, \forall b \in B_s, \forall s \in S, \forall i \in I_t, \forall t \in T, \forall n \in N \tag{5.16}$$

$$\sum_{i \in I_t} \sum_{n \in N} a_{i,t,n}^{f,b,s} \leq Q_s^{f,b} \cdot T_s^{f,t} \qquad \forall f \in F_s^+, \forall b \in B_s, \forall s \in S, \forall t \in T \tag{5.17}$$

Inequalities (5.18) and (5.19) guarantee the mapping of SFC data paths. Concretely, constraint (5.18) enforces the existence of a path between two consecutive VNFs in the required direction ($f \rightarrow g$), thereby ensuring the required routing order among VNFs that form the branches of an SFCR. Likewise, constraint (5.19) restricts the assignment of the VNFs forming an SFCR branch to those nodes through which its traffic passes. It also avoids loops in the traffic flow between two consecutive VNFs by preventing them from using more than one data path for communication.

$$\sum_{p \in P} y_p^{f,g,b,s} \geq O_s^{f,g,b} \qquad \forall f, g \in F_s^+, \forall b \in B_s, \forall s \in S \tag{5.18}$$

$$\sum_{p \in P_{n,m}} y_p^{f,g,b,s} \leq \sum_{i \in I_t} \sum_{t \in T} a_{i,t,n}^{f,b,s} \cdot \sum_{i \in I_t} \sum_{t \in T} a_{t,i,m}^{s,b,g} \qquad \forall f, g \in F_s^+, \forall b \in B_s, \forall s \in S, \forall n, m \in N \tag{5.19}$$

Constraint (5.19) is non-linear since it implies the product of two binary variables. To better illustrate the latter, we add the binary variable $\lambda_m^{f,b,s}$ which takes value 1 when $\sum_{t \in T} \sum_{i \in I_t} a_{i,t,n}^{f,b,s} = 1$ and 0 otherwise. To linearize this constraint, we introduce the binary variable $\delta_{n,m}^{f,g,b,s}$ such that:

$$\delta_{n,m}^{f,g,b,s} \leq \lambda_m^{f,b,s} \qquad \forall f, g \in F_s^+, \forall b \in B_s, \forall s \in S, \forall n, m \in N \tag{5.20}$$

$$\delta_{n,m}^{f,g,b,s} \leq \lambda_m^{g,b,s} \qquad \forall f, g \in F_s^+, \forall b \in B_s, \forall s \in S, \forall n, m \in N \tag{5.21}$$

$$\delta_{n,m}^{f,g,b,s} \geq \lambda_m^{f,b,s} + \lambda_m^{g,b,s} - 1 \qquad \forall f, g \in F_s^+, \forall b \in B_s, \forall s \in S, \forall n, m \in N \tag{5.22}$$

$$\sum_{p \in P_{n,m}} y_p^{f,g,b,s} \leq \delta_{n,m}^{f,g,b,s} \qquad \forall f,g \in F_s^+, \forall b \in B_s, \forall s \in S, \forall n,m \in N \tag{5.23}$$

QoS constraint: To guarantee proper levels of QoS, the round trip data plane delay of a PDU session cannot overpass its service latency requirement. The former is ensured by constraint (5.24), which forces the mapping of VNFs that form an SFCR along with their traffic routing to the candidates and data paths capable of meeting the specified threshold. This expression contemplates the VNF processing times and the propagation delay between VNFs for each branch forming an SFCR.

$$2 \cdot \left(\sum_{f \in F_s^+} \sum_{i \in I_t} \sum_{t \in T} \sum_{n \in N} d_t \cdot a_{i,t,n}^{f,b,s} + \sum_{f,g \in F_s^+} \sum_{p \in P} d_p \cdot y_p^{f,g,b,s} \right) + d_{DN} \leq L_s \qquad \forall b \in B_s, \forall s \in S \tag{5.24}$$

Capacity constraints: Expressions (5.25)–(5.27) represent the resource limitation in the candidate nodes, VNF instances, and physical links. Specifically, constraint (5.25) stipulates that the processing capacity demanded by the VNF instances embedded in a given candidate (MEC server) cannot exceed its physical resource capabilities. Likewise, constraint (5.26) restricts the amount of traffic processed by a VNF instance to avoid overload, given its limited processing capacity. Moreover, the available bandwidth on physical links must be sufficient to handle the traffic demands of their assigned data flows, see (5.27).

$$\sum_{i \in I_t} \sum_{\substack{t \in T, \\ t \neq 0}} C_t \cdot x_{i,t,n} \leq C_n \qquad \forall n \in N_c \tag{5.25}$$

$$\sum_{f \in F_s} \sum_{s \in S} C_s \cdot z_{i,t,n}^{f,s} \leq C_t \qquad \forall i \in I_t, \forall t \in T; t \neq 0, \forall n \in N_c \tag{5.26}$$

$$\sum_{f,g \in F_s^+} \sum_{b \in B_s} \sum_{s \in S} \sum_{p \in P} \beta_s \cdot y_p^{f,g,b,s} \cdot H_{u,v}^p \leq \beta_{u,v} \qquad \forall (u,v) \in E \tag{5.27}$$

Finally, the binary nature of the attendant variables is indicated as follows:

$$w_n, x_{i,t,n}, z_{i,t,n}^{f,s}, a_{i,t,n}^{f,b,s}, y_p^{f,g,b,s} \in \{0,1\} \quad \forall f,g \in F_s^+, \forall b \in B_s, \forall s \in S, \forall i \in I_t, \forall t \in T, \forall p \in P, \forall n \in N \tag{5.28}$$

The linear form of the UPR model is provided below:

$$\text{Min} \qquad \alpha \cdot \sum_{n \in N_c} w_n + \beta \cdot \sum_{i \in I_t} \sum_{\substack{t \in T, \\ t \neq 0}} \sum_{n \in N_c} x_{i,t,n} + \gamma \cdot \sum_{f,g \in F_s^+} \sum_{b \in B_s} \sum_{s \in S} \sum_{p \in P} d_p \cdot y_p^{f,g,b,s}$$

s.t.:

$$\sum_{i \in I_t} \sum_{n \in N_c} x_{i,t,n} \leq I_t \qquad \forall t \in T; t \neq 0$$

$$\sum_{n \in N_c} x_{i,t,n} \leq 1 \qquad \forall i \in I_t, \forall t \in T; t \neq 0$$

$$x_{i,t,n} \leq w_n \cdot V_n^t \qquad \forall i \in I_t, \forall t \in T, \forall n \in N$$

$$w_n \leq \sum_{i \in I_t} \sum_{\forall t \in T} x_{i,t,n} \qquad \forall n \in N$$

$$\sum_{i \in I_t} \sum_{t \in T} \sum_{n \in N} z_{i,t,n}^{f,s} = 1 \qquad \forall f \in F_s^+, \forall s \in S$$

$$z_{i,t,n}^{f,s} \leq x_{i,t,n} \cdot T_s^{f,t} \qquad \forall f \in F_s^+, \forall s \in S, \forall i \in I_t, \forall t \in T, \forall n \in N$$

$$x_{i,t,n} \leq \sum_{s \in S} \sum_{f \in F_s} z_{i,t,n}^{f,s} \qquad \forall i \in I_t, \forall t \in T, \forall n \in N_c$$

$$\sum_{f \in F_s} \sum_{i \in I_t} \sum_{n \in N_c} z_{i,t,n}^{f,s} \geq |F_s^t| \qquad \forall s \in S, \forall t \in T; t \neq 0$$

$$z_{i,t,n}^{f,s} \leq \sum_{b \in B_s} a_{i,t,n}^{f,b,s} \qquad \forall f \in F_s^+, \forall s \in S, \forall i \in I_t, \forall t \in T, \forall n \in N$$

$$a_{i,t,n}^{f,b,s} \leq z_{i,t,n}^{f,s} \qquad \forall f \in F_s^+, \forall b \in B_s, \forall s \in S, \forall i \in I_t, \forall t \in T, \forall n \in N$$

$$\sum_{i \in I_t} \sum_{n \in N} a_{i,t,n}^{f,b,s} \leq Q_s^{f,b} \cdot T_s^{f,t} \qquad \forall f \in F_s^+, \forall b \in B_s, \forall s \in S, \forall t \in T$$

$$\sum_{p \in P} y_p^{f,g,b,s} \geq O_s^{f,g,b} \qquad \forall f, g \in F_s^+, \forall b \in B_s, \forall s \in S$$

$$\delta_{n,m}^{f,g,b,s} \leq \lambda_m^{f,b,s} \qquad \forall f, g \in F_s^+, \forall b \in B_s, \forall s \in S, \forall n, m \in N$$

$$\delta_{n,m}^{f,g,b,s} \leq \lambda_m^{g,b,s} \qquad \forall f, g \in F_s^+, \forall b \in B_s, \forall s \in S, \forall n, m \in N$$

$$\delta_{n,m}^{f,g,b,s} \geq \lambda_m^{f,b,s} + \lambda_m^{g,b,s} - 1 \qquad \forall f, g \in F_s^+, \forall b \in B_s, \forall s \in S, \forall n, m \in N$$

$$\sum_{p \in P_{n,m}} y_p^{f,g,b,s} \leq \delta_{n,m}^{f,g,b,s} \qquad \forall f, g \in F_s^+, \forall b \in B_s, \forall s \in S, \forall n, m \in N$$

$$\sum_{f \in F_s} \sum_{i \in I_t} z_{i,t,n}^{f,s} \leq 1 \qquad \forall s \in S, \forall t \in T; t \neq 0, \forall n \in N_c$$

$$\sum_{f \in F_s} \sum_{i \in I_t} z_{1,i,n}^{s,f} + \sum_{f \in F_s} \sum_{i \in I_t} z_{3,i,n}^{s,f} \leq 1 \qquad \forall s \in S, \forall n \in N_c$$

$$2 \cdot \left(\sum_{f \in F_s^+} \sum_{i \in I_t} \sum_{t \in T} \sum_{n \in N} d_t \cdot a_{i,t,n}^{f,b,s} + \sum_{f,g \in F_s^+} \sum_{p \in P} d_p \cdot y_p^{f,g,b,s} \right) + d_{DN} \leq L_s \qquad \forall b \in B_s, \forall s \in S$$

$$\sum_{i \in I_t} \sum_{\substack{t \in T, \\ t \neq 0}} C_t \cdot x_{i,t,n} \leq C_n \qquad \forall n \in N_c$$

$$\sum_{f \in F_s} \sum_{s \in S} C_s \cdot z_{i,t,n}^{f,s} \leq C_t \qquad \forall i \in I_t, \forall t \in T; t \neq 0, \forall n \in N_c$$

$$\sum_{f,g \in F_s^+} \sum_{b \in B_s} \sum_{s \in S} \sum_{p \in P} \beta_s \cdot y_p^{f,g,b,s} \cdot H_{u,v}^p \leq \beta_{u,v} \qquad \forall (u,v) \in E$$

$$w_n, x_{i,t,n}, z_{i,t,n}^{f,s}, a_{i,t,n}^{f,b,s}, y_p^{f,g,b,s} \in \{0,1\} \qquad \forall f, g \in F_s^+, \forall b \in B_s, \forall s \in S, \forall i \in I_t, \forall t \in T, \forall p \in P, \forall n \in N$$

5.3 Approximate Solutions

The UPC problem is a specific variant of the VNFPC problem, which has been shown to be NP-Hard [25, 74, 78, 150]. Consequently, the UPC problem is also NP-Hard. Thus, developing heuristic algorithms is required to obtain efficient solutions for large-scale scenarios in polynomial

time. In this regard, we propose two solutions (i.e., a heuristic and an SA metaheuristic). The following subsections explain the conceived solutions in detail.

5.3.1 Heuristic: Priority and Cautious UPC

The PC-UPC algorithm maps SFCRs on the underlying infrastructure by applying priority classification of PDU session demands and cautiously locating their constituent VNFs. Algorithm 2 provides a general overview of the proposed PC-UPC heuristic. As input, it takes a set of SFCRs and the substrate network topology (e.g., nodes, physical links, and paths) along with their respective requirements and capabilities.

Algorithm 2: Priority & Cautious UPC (PC-UPC)

```
   Input: N, E, P, S
   Output: Set of UPFs (N_u), Sets of mapped and unmapped SFCRs (S_m and S_u)
 1 Initialize output variables and parameters
 2 forall s ∈ S do
 3     Determine available candidates for s and SFCRs near each candidate
 4     Classify s as critical or not regarding their number of available candidates
 5 Sort SFCRs (S_sort) according to the established criteria
 6 while S_sort ≠ ∅ do
 7     s ← S_sort[0]
 8     Procedure 3: SFCR-mapping Procedure
 9     if mapping_success then
10         S_m ← S_m ∪ s
11         Update network and infrastructure resources
12         if number of available servers changed then
13             Update available candidates near each SFCR
14             Sort SFCRs according to the established criteria
15     else
16         S_u ← S_u ∪ s
17     S_sort ← S_sort − s
```

Algorithm 2 begins by determining the candidate locations that could host the VNFs that form every SFCR as well as the SFCRs that each candidate could serve according to session latency requirements (line: 3). The first determination reveals whether an SFCR is in a critical stage, whereas the latter helps decide which server is more convenient to deploy a UPF in terms of popularity. Specifically, we classify an SFCR as critical if it has a number of candidates equal to or lower than the minimum needed to map its constituent VNFs (line: 4). Next, the algorithm sorts the SFCRs according to the established criteria (line: 5). Specifically, we order the PDU sessions in terms of criticism level, service latency requirements (ascending), length of the SFCS (descending), available candidates for their mapping (ascending), and closeness between their point of attachment to the network. This approach increases the possibility of successfully mapping the most critical and demanding PDU sessions. Once the SFCR order has been established, the algorithm begins the mapping process by selecting the most demanding SFCR and executing the SFCR-mapping procedure (lines: 7–8).

Procedure 3 in line 8 is responsible for determining the best candidates and routing data paths for the VNFs and path segments that form the SFCR under analysis. A VNF mapping decision is made by considering the mapping effects on the current stage and the next VNFs that remain unmapped. Specifically, this procedure analyzes how the placement decision of a given VNF affects other VNFs that directly depend on it (i.e., the next VNFs in every branch). This enables more effective VNF mapping and reduces SFCR rejection and reassignment procedures by avoiding dead ends during the SFC mapping. To keep the SFCR-mapping procedure simple and reduce its execution time, we only consider the influence of the VNF mapping decision in the current and subsequent stages. Nevertheless, it could be extended to include more stages.

The SFCR-mapping procedure begins by initializing the output variables. In particular, it marks the SFC mapping indicator as successful and creates an empty set to store the mapping information concerning selected VNF candidates and routing paths (line: 1). Afterward, it proceeds to map the SFCR constituent VNFs (F_s) by iterating through each VNF, thus forming the chain (line: 2). These VNFs are mapped by their order of appearance in the SFCR branches, beginning from the source access node to the destination UPFs (PSAs). Given that a VNF may have been previously mapped by the look-ahead process, the procedure must verify whether the chosen VNF has not been mapped before analyzing its mapping options (line: 3).

In the case of an unmapped VNF, the procedure sets the best mapping cost to a significant value (i.e., infinite) and determines the VNF source nodes, the next VNFs in the chain (if any), and possible candidates. This step is executed for each branch comprising the chain since each VNF can have different source nodes, destination VNFs, and candidate servers (lines: 5–8). The possible locations for the chosen VNF are determined by considering its source node and available propagation delay budget in each branch. From these candidates, the ones that are common to all the branches are selected (N_c^f), and their feasibility and mapping costs are analyzed (lines: 9–10).

The feasibility of a candidate is assessed based on the UPC constraints presented in Section 5.2. A candidate is feasible when all the attendant constraints are satisfied. For every feasible candidate, the mapping cost of the VNF under analysis (f) is computed according to the objective function (5.4). Since this cost is estimated based on the selected VNF mapping, considering the remaining fraction of unassigned PDU sessions when creating a new instance is required. This favors the instantiation of VNFs in the candidate nodes that can serve a greater number of unmapped SFCRs, which we refer to as the most popular candidate locations. Furthermore, the shortest virtual path with enough bandwidth to serve the traffic flow demand is also obtained for each feasible candidate.

Once the feasibility of the candidates has been evaluated, the procedure selects the subset of feasible candidates, denoted as $N_{c'}^f$, and sorts them in ascending order according to the estimated VNF mapping costs (lines: 11–12). For each candidate in this subset with a lower estimated cost than the current best cost $Cost_c < Cost_{best}$, the mapping of VNF f is simulated by reserving the required resources (lines: 13–37). The previous step reduces the solution computation time

Procedure 3: SFCR-mapping

```
1  mapping_success ← True, F_mapped ← ∅
2  forall f ∈ F_s do
3      if f ∉ F_mapped then
4          Cost_best ← ∞
5          forall b ∈ B_s do
6              if f ∈ b then
7                  Determine source and next VNFs
8                  Determine candidates for f
9          Select candidates common to all the branches (N_c^f)
10         Evaluate feasibility and mapping cost for c∈N_c^f
11         Select feasible candidates (N_c'^f)
12         Sort c ∈ N_c'^f by cost in ascending order (N_c'sort^f)
13         forall c ∈ N_c'sort^f do
14             if Cost_c < Cost_best then
15                 Make a copy of current placement configuration and SFCR mapping
16                 Simulate deployment of f in c
17                 if f ≠ last_vnf then
18                     Sort next_vnfs by their requirements
19                     forall nf ∈ next_vnfs do
20                         Determine candidates for nf (N_c^nf)
21                         Evaluate feasibility and mapping cost for all nc ∈ N_c^nf
22                         Select feasible candidates (N_c'^nf)
23                         if N_c'^nf ≠ ∅ then
24                             Select candidate with the best cost (nc_best)
25                             Simulate deployment of nf in nc_best
26                             Cost_c ← Cost_c + Cost_nc_best
27                         else
28                             Cost_c ← ∞
29                             break
30                     if Cost_c < Cost_best then
31                         n_best ← c
32                         Cost_best ← Cost_c
33                         if f = penultimate_vnf then
34                             Save placement configuration and SFCR mapping
35                 else
36                     Update placement configuration and SFCR mapping
37                     return
38         if Cost_best = ∞ then
39             mapping_success ← False
40             return
41         Update F_mapped, placement configuration and SFCR mapping
42 return
```

since poor candidates are instantaneously discarded. Furthermore, when f is not the last VNF in the chain data path, the look-ahead process analyzes the implications of its assignment to the current candidate for the next VNFs (lines: 17–34).

The look-ahead procedure begins by sorting the VNFs that directly depend on the VNF under analysis (*next_vnfs*). It sorts these VNFs according to their propagation latency budget, although other criteria, such as criticism level, may be used. For each next VNF (*nf*), candidates are

determined and evaluated according to their feasibility and mapping cost. If feasible candidates exist for the mapping of *nf*, the one with the lowest cost is selected, and the mapping of *nf* is simulated in the selected candidate by reserving the required resources (lines: 23–26). This step requires updating the mapping cost ($Cost_c$) by considering the expected cost of the *nf* mapping. If no feasible candidate is available for the assignment of a next VNF, the candidate $c \in N^f_{fc_{sort}}$ under analysis is discarded by setting its cost to infinite (lines: 28–29). The latter avoids selecting candidates that may lead to unmapped VNFs in the chain at future steps.

After analyzing the future implications of a given candidate selection, the procedure compares the candidate's mapping cost with the best cost found thus far to determine if it is a better candidate (lines: 30–34). If a better candidate is found, the current best cost and candidate are updated. In addition, the best-simulated mapping configuration is also saved when VNF f is the penultimate virtual node in the chain (lines: 33–34). This avoids repeating the mapping process for the next VNFs since they have already been analyzed as part of the look-ahead process.

Once all candidates have been evaluated, the one with the lowest mapping cost is selected, and the placement configuration is updated with the best SFCR mapping configuration. If no feasible candidate exists, the mapping process is marked as failed and interrupted. Otherwise, the mapping procedure continues until all the VNFs that form the SFCR have been analyzed. No look-ahead is performed for mapping the final VNF. Thus, the first candidate in the sorted set of feasible candidates is considered the best. This step requires the update of the placement and SFCR mapping configuration and ends the SFCR-mapping procedure (lines: 35–37).

Algorithm 2 classifies an SFCR as mapped or rejected according to the outcome of the SFCR-mapping procedure (i.e., the mapping_success indicator). When an SFCR is successfully mapped, the network and infrastructure resources are updated according to the performed mapping decisions. Additionally, the occurrence of any change in the set of available candidates is verified (lines: 12–14). In the latter case, the subset of candidates for each SFCR is updated by removing the unavailable locations. This last step requires reordering the set of unmapped SFCRs since their possible locations and criticism levels may have changed. In contrast, when the SFCR-mapping procedure fails, the selected SFCR is marked as unmapped (lines: 15–16). Finally, the currently picked SFC is removed from the set of PDU session requests (S_{sort}) regardless of its mapping outcome (line: 17). The SFCR-mapping procedure is executed as long as there are SFCRs pending analysis.

5.3.2 Metaheuristic: Simulated Annealing-based UPC

This section presents the SA-based metaheuristic for the UPC (SA-UPC) problem. This approach incorporates several strategies, such as restart-stop and variable Markov chain length, to enhance its performance. Subsection 5.3.2.1 provides a detailed description of the solution's main components, while Subsection 5.3.2.2 describes its procedure using a flowchart scheme.

5.3.2.1 SA-UPC Components

SA metaheuristics are characterized by several parameters that must be defined and adjusted according to the problem under study. In this section, we present the main components of the proposed solution. We focus on aspects such as neighborhood function, MCL, restart strategy, and termination condition.

The *initial solution* is the starting point for the solution space exploration. The proposed SA can accept any initial solution (e.g., random or heuristic-specific) as long as it is feasible. Specifically, the solution must satisfy the UPC constraints specified in Section 5.2. However, a good initial solution is recommended since it can provide better results than random approaches [151].

The *acceptance criterion* determines whether a candidate solution is accepted according to the acceptance probability (p_a). Our proposed solution adopts the Metropolis condition [152], which is the criterion utilized in the traditional formulation of the SA metaheuristic [153].

The *cooling or annealing schedule* specifies how the temperature decreases during the search process [154]. A set of parameters, such as initial temperature, decrement function, MCL, and final temperature, defines the cooling schedule. In this study, we utilize a geometric cooling schedule, which computes the next temperature value according to the decrement formula $T_{k+1} = \alpha \cdot T_k$. The α factor in the previous function has typical values ranging from 0.8 to 0.99. We select this scheduling technique due to its popularity and simplicity. However, other approaches, such as adaptive, arithmetic, or monotonic scheduling, can be used.

Given the wide variety of research studies [154–156] addressing the performance of the acceptance criterion and cooling scheduling parameters, we focus on the analysis of other aspects of the SA algorithm. Specifically, we investigate the neighbor solution (NS), MCL, and restart solution.

Most state-of-the-art SA-based solutions generate *neighbor solutions* based on the same neighborhood strategy during their search. Moreover, these strategies are typically characterized by introducing small changes to the current solution. We believe that combining different strategies can significantly improve the efficiency of the SA algorithm. Therefore, we propose two approaches (i.e., NS_ST and NS_3T) that combine intensification with diversification techniques when exploring the solution space. The benefits of this approach are twofold. First, it allows a better solution space exploration, and second, it prevents the SA algorithm from being trapped in local optima.

Our conceived neighborhood functions use three strategies to introduce changes to the current solution. The first mechanism exploits the current solution neighborhood by introducing small changes. In particular, it randomly chooses an SFCR from the set of PDU sessions and maps it onto the underlying infrastructure. This SFCR mapping can either be random or can follow a specific optimization criterion (i.e., minimize the objective function (5.4), in Section 5.2). With this strategy, the generated neighbor solutions will mainly differ from the current solution in mapping

one SFCR. However, this approach may also involve changes to other system parameters, such as the number of open servers and VNFs. Based on this variant, a subset of neighboring solutions of a pre-established size is generated.

The other two strategies aim to explore the solution space more efficiently and escape from local optima (if it is trapped in any) by further modifying the current solution. The second and third approaches remap all SFCRs mapped to a randomly selected VNF instance and candidate server, respectively. In these variants, mapping selected SFCRs is determined by following only the cost reduction criteria. This mapping approach attempts to avoid deterioration of the current solution's quality through overlay due to the presence of a more significant number of unmapped PDU sessions.

In all previously mentioned strategies, mapping the selected PDU sessions is performed by verifying placement and chaining constraints; see Section 5.2. Moreover, mapping SFCRs is done by considering only those candidates that satisfy service latency requirements (near candidates). In this way, we increase the likelihood of generating feasible solutions. Accordingly, we propose the following two methods to create neighbor solutions:

- NS_3T: The set of neighboring solutions is generated by simultaneously considering the three strategies mentioned above. Specifically, this function generates two neighbor solutions by introducing changes of types two and three, along with a subset of type one solutions.

- NS_ST: Unlike NS_3T, this method selectively applies a change based on the quality of the generated solutions and current temperature values. First, it simultaneously considers the three previous strategies when generating new solutions. This approach is applied until a better solution is detected. Once a better solution is found, it generates only neighboring solutions of the first type to explore its neighborhood better. The decision to apply a diversification strategy is made at the end of a temperature length with probability p_d. Using diversification depends on the fraction of remaining temperatures (T_r), and the best solution ratio updates at each temperature (Q). Specifically, the diversification strategy is selected when p_d is greater than Q and lower than T_r. Diversification probability is more probable at the beginning of the algorithm since both T_r and Q decrease with temperature.

Once all neighbor solutions have been generated according to the selected method, their associated cost is determined. Then, the one with the best cost is selected as the neighbor solution.

The *Markov chain length*, also known as temperature length (L_{mkv}), defines the number of iterations performed for a given temperature. A considerable length of Markov chains allows an exhaustive exploration of the solution. However, this comes at the expense of higher execution times. To avoid long Markov chains, cooling schedule methods with small decrements in the temperature may be adopted [157].

Using a fixed temperature length during simulation time is the most common practice. However, this may not be the most suitable strategy for large-scale optimization problems [158], thereby different strategies based on variable MCL have been adopted. For instance, the authors of [159, 160] vary the chain length inversely to the temperature update. In [161], the higher the temperature, the longer the MCL, whereas in [158], a higher MCL in intermediate temperatures is proposed.

As in the previous studies, we embrace a VMCL approach. However, instead of varying the chain length with the temperature, we adjust it according to the quality of the generated solutions. Specifically, our strategy assumes two possible values for the MCL and switches between them. It chooses a lower value ($L_{mkv_{min}}$) when no better solution is found at a current temperature; otherwise, it sets the length of the chain to a maximum value ($L_{mkv_{max}}$). Another technique is gradually increasing or decreasing this parameter using adjusting factors or an ordered list of possible values. However, we prefer our strategy due to its simplicity and ability to accelerate the algorithm execution time. The MCL is adjusted at the end of a temperature length when required. This strategy favors the exploitation of effective solutions and decreases the algorithm computation time.

Restart strategies are a widespread diversification mechanism used to avoid being trapped at strong local optima and increase the probability of finding a global optimum. This strategy is characterized by two main aspects: the restart conditions and the restart point.

The restart condition adopted by the conceived SA is straightforward and widely used in the literature. Our algorithm restarts if the current best solution has not improved for a fixed number of consecutive temperatures, which is referred to as the restart period (P_r). In addition, the restart point requires the specification of the restart temperature and solution. Many approaches can be followed for the temperature reset, such as re-establishing the initial temperature, multiplying the current temperature by a factor greater than one, or using the system's temperature value when the current best solution was found. Nevertheless, we do not adopt any of these re-heating methods; rather, we keep the current temperature value unchanged. We select this approach to reduce the probability of accepting worse solutions and the time needed for solution computation.

For the solution reset, a popular approach is selecting the current best solutions [162, 163]. However, a drawback to this approach is that it may always restart to the same point of the solution space if the best solution is not updated during the restart period. Thus, it may end up in the local optimum from which the algorithm is trying to escape. In contrast, some studies generate random restart solutions [164] or combine different strategies [165].

Our proposed strategy for selecting the restart solution is more in line with the latter since we adopt different methods to generate the solution. A solution is chosen randomly from a set of possible solutions (e.g., the initial, current, and best solutions), and a new restart point is constructed upon the selected solution. We randomly select a number of open servers much lower than the total and remap their assigned SFCRs based on the criteria of the best cost optimization.

We do this to construct a new solution that differs from the current solution but not completely to avoid selecting poor reset solutions. After defining a new restart point (solution), we must ensure that it is located in a different region of the solution space to escape from a current local optimum if trapped in one.

Furthermore, to promote diversification during the process, we determined whether the selected restart point has not been used in previous restarts. We record all the neighboring solutions generated during the current restart period and inspect whether the new solution is significantly different (an outlier) from the ones forming the set. The z-score formula is used to compare how far the restart solution cost is from the mean cost of the neighboring solutions. In this manner, we determine whether we are in the presence of an outlier, thereby ensuring the exploration of a different region and escaping from the local optimum.

Additionally, to promote diversification during restarts, we define a Tabu list in which the costs of the selected restart solutions are stored. The length of this list is conditioned by the number of permissible restarts (i.e., stopping condition). Overall, a new restart solution will be accepted as long as it has not been previously used as a restart point and it significantly differs from the solutions recently visited during the restart period. The acceptance criterion of the restart point is based on the solution cost instead of the placement configuration since this parameter is much simpler to analyze.

The *stopping condition* indicates when the SA algorithm should terminate exploring the solution space. Our proposed solution has two stopping conditions. The first is the one used by classical SA; the algorithm stops when a specific temperature value (T_f) is reached. The second condition terminates the execution when a fixed number of restarts (R_f) has been performed. Thus, our SA will stop executing when one of the aforementioned criteria is met.

5.3.2.2 SA-UPC Procedure

The flowchart of the proposed SA-UPC algorithm is represented in Fig. 5.2. This algorithm begins by generating an initial solution and assessing its cost according to (5.4). Afterward, it proceeds to initialize the SA parameters, such as initial and final temperatures (T_i and T_f), Markov chain length, and restart period (P_r), and it specifies the neighborhood solution strategy (i.e., NS_ST or NS_3T). Additionally, the current and best solutions (S_c and S_b), as well as their associated costs (F_c and F_b), are defined based on the initial solution parameters (i.e., placement and mapping setup and cost). Given that the probability of improving the current best solution and accepting worse solutions is initially higher, the L_{mkv} parameter is set to its minimum value.

The algorithm performs L_{mkv} explorations of the solution space for each temperature value. Specifically, it generates a set of neighbor solutions according to the specified method at each iteration of its inner loop. The UPC constraints are verified for each solution, and only feasible solutions are accepted. If an unfeasible solution is detected, a new one must be generated. Subsequently, the solutions' costs are determined, and the best solution (the one with the lowest

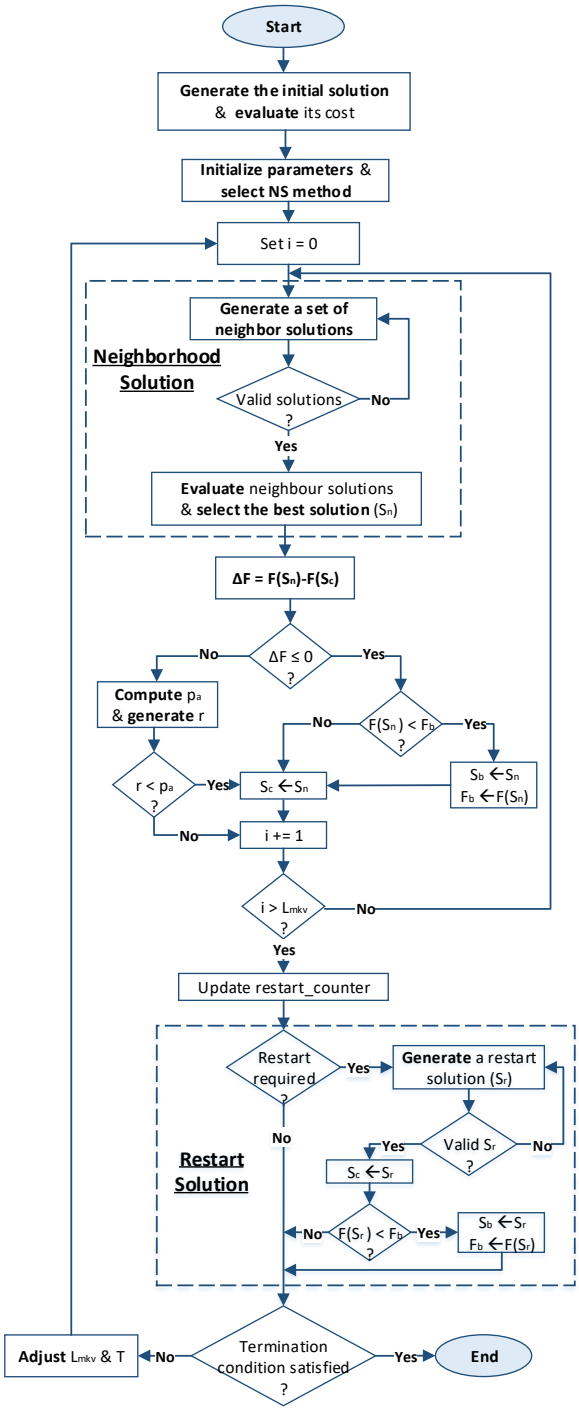


Figure 5.2: Flowchart of the proposed SA-UPC.

cost) is selected as the neighbor solution.

Next, the algorithm compares the costs of the neighbor and current solutions ($\Delta F = F(S_n) - F(S_c)$). If the neighbor solution is better than the current one, the latter is updated, and the algorithm verifies whether a better solution has been detected. If so, the placement configuration

of the best solution and its associated cost are replaced by the neighbor solution. When using the NS_ST method, the neighbor function is instructed only to generate solutions of type one during successive iterations until otherwise specified, and the MCL is instructed to switch to its maximum value at the next temperature iteration. These two strategies further exploit the solution space in the following iterations. Worse neighbor solutions are accepted as current solutions with probability p_a following the Metropolis criterion.

The algorithm updates the restart counter at the end of a temperature length. It increases this counter if no best solution improvement occurs during the process or sets it to zero otherwise. The algorithm restarts when no better solution is found during a restart period (P_r). As a result of this process, the current solution is modified, the MCL is increased, and the NS_ST method, if selected, adopts a diversification phase.

Finally, the termination criteria are verified. If a stopping condition is satisfied, the algorithm terminates and returns the best solution found thus far. Otherwise, it updates the current temperature value, the MCL, and the neighborhood strategy if required by the NS_ST method. Furthermore, the MCL parameter is set to its minimum value when the best solution is not updated during the current temperature value.

5.3.3 Complexity Analysis

This subsection discusses the complexity analysis of PC-UPC and SA-UPC algorithms.

In algorithm 2 (lines: 2–4), we first need to find the available candidates for each SFCR as well as the SFCRs near each candidate, which requires $S \cdot N_c$ steps. Next, the set of SFCRs is ordered according to the established criteria. Thus, the time complexity of this part is $S \cdot (N_c + K \cdot \log S)$, where K denotes the number of sorting criteria under consideration. Then, an iterative process for mapping the PDU sessions takes place (lines: 6–17). This process will be executed S times according to the number of PDU sessions. Inside this loop, Procedure 3 is responsible for determining the best SFCR mapping configuration. Moreover, when a variation in the number of available candidates is detected, the update of the subset of SFCR available candidates and SFCR mapping order is required.

Regarding the complexity of the SFCR-mapping procedure, for the mapping of a VNF service, we first need to determine its source and destination VNFs as well as available candidates for each branch forming the SFC, which runs in $B_s \cdot (N_c + F_s)$. Then, the subset of common candidates for all the branches is determined in the subsequent line. In the worst case, this requires $B_s \cdot N_c$, assuming that the selected VNF is presented in all the branches. Afterward, the feasibility of each candidate is assessed, and the feasible candidates are sorted according to their estimated cost. The complexity of this step is expressed as $N_c \cdot M + N_c \cdot \log N_c$, where M denotes the run time of the feasibility evaluation process. Please note that this expression considers all candidates' feasible locations for deploying the selected VNF, which rarely happens in practice.

For each feasible candidate, the implications of its selection are investigated by simulating

the VNF deployment and verifying the mapping costs and options for the next VNFs in the chain. This iterative process can be expressed as $N_c \cdot (B_s \cdot \log B_s + B_s \cdot N_c(1+M))$. Assuming that all the constituent VNFs are analyzed, $F_s \cdot (B_s \cdot (2 \cdot N_c + F_s) + N_c \cdot (M + \log N_c + B_s \cdot (\log B_s + N_c \cdot (1+M))$ steps are required for the mapping of one SFCR. Therefore, Procedure 3 complexity is at the level of $\mathcal{O}(F_s \cdot N_c^2 \cdot M \cdot B_s)$. Notice that not all the constituent VNFs and candidates are usually evaluated due to the 1-SLA and the candidate sorting procedures.

The total complexity of the conceived PC-UPC algorithm is $S \cdot (N_c + K \cdot \log S) + S \cdot (F_s \cdot N_c^2 \cdot M \cdot B_s + S \cdot N_c + K \cdot S \cdot \log S)$. From this expression, we can see that the dominant term is the one associated with the mapping of SFCRs and their constituent VNFs ($S \cdot F_s \cdot N_c^2 \cdot M \cdot B_s$). Generally, the number of constituent VNFs (F_s) and branches (B_s) in an SFCR is significantly smaller than the size of the service request and candidate node sets. Thereby, the overall complexity of the PC-UPF solution can be expressed as $\mathcal{O}(S \cdot N_c^2 \cdot M)$.

The computational time of the SA-UPC solution is proportional to the number of moves performed by the algorithm. These are conditioned by the temperature values and the MCL (L). Specifically, several neighbor solutions are generated for each temperature $t \in T$ based on the neighboring method and the MCL. Assuming the worst-case scenario, in which the NS_3T method and the maximum length of the Markov chain are always selected ($L = L_{mkv_{max}}$), we can define the SA-UPC running time as $T \cdot L \cdot S \cdot F_s \cdot N_c^2 \cdot M \cdot B_s$. In the previous expression, T represents the number of temperature evaluations determined by the cooling factor and initial and final temperatures for the geometric scheduling. Additionally, S denotes the total number of sessions remapped by the three types of changes, and $F_s \cdot N_c^2 \cdot M \cdot B_s$ is the maximum running time of the SFCR-mapping procedure. Generally speaking, $B_s \leq F_s << S$, thus the computational complexity of the SA-UPC algorithm can be expressed as $\mathcal{O}(T \cdot L \cdot S \cdot N_c^2 \cdot M)$.

5.4 Evaluation and Results

5.4.1 Simulation Setup

For the simulations, we considered a network topology representing a 5G medium-scale scenario of 5x5 km^2 (see Fig. 5.3). In this scenario, the gNBs were connected through aggregation points, and the gNB inter-site distances were 500 m and 200 m for gNBs located in urban and dense urban areas [166], respectively. The VNF candidate location comprised 13 MEC servers, which were co-located with the APs and had a service area with a radius of 1 km. We represented the bidirectional links as two individual links, one in each direction. Regarding link capacity, we assumed that the links between gNBs and APs had enough resources to support their associated users and traffic. In contrast, the bandwidth between ENs was set to 10 Gbps [64].

For the service demand, we contemplated three types of SFCRs formed by one to three UPFs. Their service requirements, such as latency, processing demand, and bandwidth, were randomly generated according to the parameters specified in Table 5.4. The number of active PDU sessions

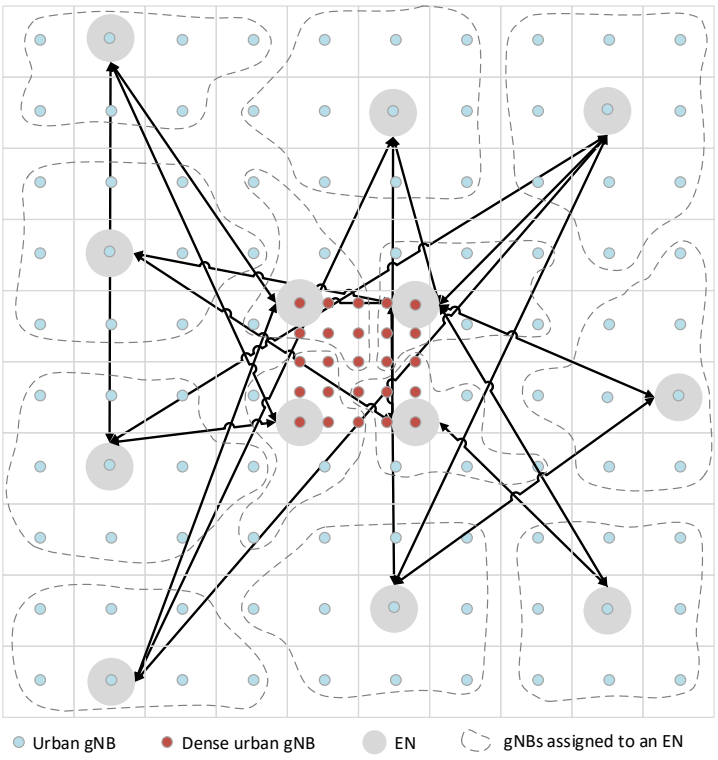


Figure 5.3: 5G access network topology.

varied in the interval of [10–400]. Table 5.4 presents the simulation parameters for the UPC model and proposed heuristics.

In the following sections, we analyze the simulation results. We ran every solution 15 times for each experiment, which enabled us to present results with confidence intervals of 95%.

5.4.2 PC-UPC Performance Evaluation

To evaluate the effectiveness of the proposed PC-UPC solution, we compared its performance with two benchmarks in terms of the total cost, the number of reassigned VNFs during the mapping procedure, and execution time. Two greedy-based baselines, referred to as Greedy and Sorted Greedy Heuristics (GH and SGH, respectively), were considered. Both solutions assign SFCRs to the VNF instances and servers that minimize the objective function (5.4). Moreover, SGH sorts SFCRs by following the criteria of our proposed algorithm (PC-UPC). Given that these baselines generate solutions with rejected SFCRs, we developed a VNF reassignment procedure. This procedure returns one step back in the mapping process when a VNF service request cannot be mapped due to the absence of feasible candidates. Concretely, it attempts to remap the previous VNF in the chain to a different candidate (the following best location, if any). This procedure repeats until the current VNF is mapped successfully or no feasible candidate remains for the previous VNF. In this manner, we avoid SFCR rejections in the baselines and guarantee similar

Table 5.4: Simulation parameters for the UPC.

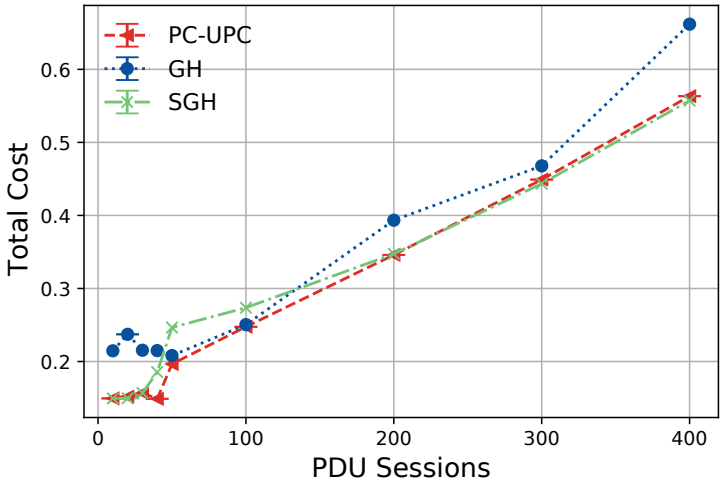
Parameter	**Value**
Number of gNBs	121
Number of ENs	13
Number of links	174
Number of shortest paths	1742
Scenario dimensions	5 x 5 km^2
Number of PDU sessions	[10–400]
Bandwidth requirement	[1 Mbps, 10 Mbps, 20 Mbps]
Processing demand	[0.1 CPU, 0.2 CPU]
Latency requirement	[0.9 ms, 1 ms]
RTT delay in the RAN (T_r)	500 μs
Processing time of UPFs (T_u)	50 μs
Processing time of DN (T_d)	100 μs
Processing time of AP (T_{ap})	5 μs
Propagation delay in optical links	5 μs
Links capacity	10 Gbps
MEC server capacity	16 vCPU
UPF processing capacity	2 vCPU
UPF types	1: aUPF, 2:miUPF, 3: IUPF
Number of VNF instances per type	[1: 72, 2: 16, 3: 16]
Weight factors	$\alpha = 0.4$, $\beta = 0.4$, $\gamma = 0.2$

comparison conditions with our conceived PC-UPC algorithm.

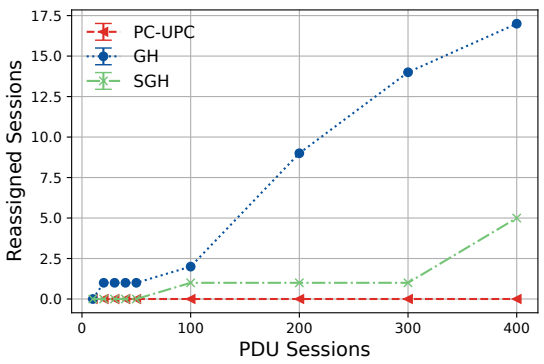
Figure 5.4 summarizes the gathered results. as shown in the figure, our proposed heuristic outperformed both benchmarks for all analyzed metrics. It had not only the best cost behavior but also the lowest execution time and reassignments. Specifically, the PC-UPC heuristic mapped all SFCRs without requiring the reassignment of any constituent VNFs. No the case was for the baselines, in which the number of remapped VNF instances increased with the number of active PDU sessions. Thus, the GH provided the worst results, with up to 13 more reassigned sessions than the SGH baseline. Moreover, the reference solutions would have rejected the reassigned sessions if it had not been for the reassignment procedure.

Higher cost reductions and fewer VNF reassignments were achieved during SFCR mapping when accounting for SFC requirements and current network conditions. This can be better appreciated by comparing the priority-based solutions (i.e., PC-UPC and SGH) with the GH. Consequently, these results demonstrated the importance and benefits of the look-ahead and priority sorting processes.

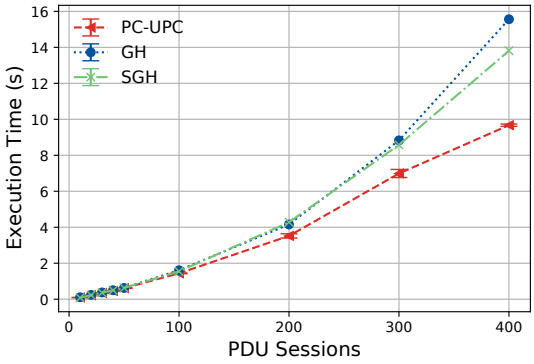
Regarding the execution time, no significant difference was found among the solutions for the number of PDU sessions equal to or lower than 200. In contrast, for PDU session values greater than 200, the conceived PC-UPC solution required between two and six seconds less than the benchmarks. This wider gap was due to the greater number of reassigned sessions during the mapping procedures. Moreover, the comparison of the greedy-based heuristics showed that the priority sorting process did not significantly influence the overall computing time.



(**a**) Normalized cost.



(**b**) Reassigned sessions.



(**c**) Execution time.

Figure 5.4: Performance of the heuristic-based solutions versus different numbers of PDU sessions.

The obtained results generally revealed an increase in the examined metrics with the number of PDU sessions for all solutions.

5.4.3 SA-UPC Performance Evaluation

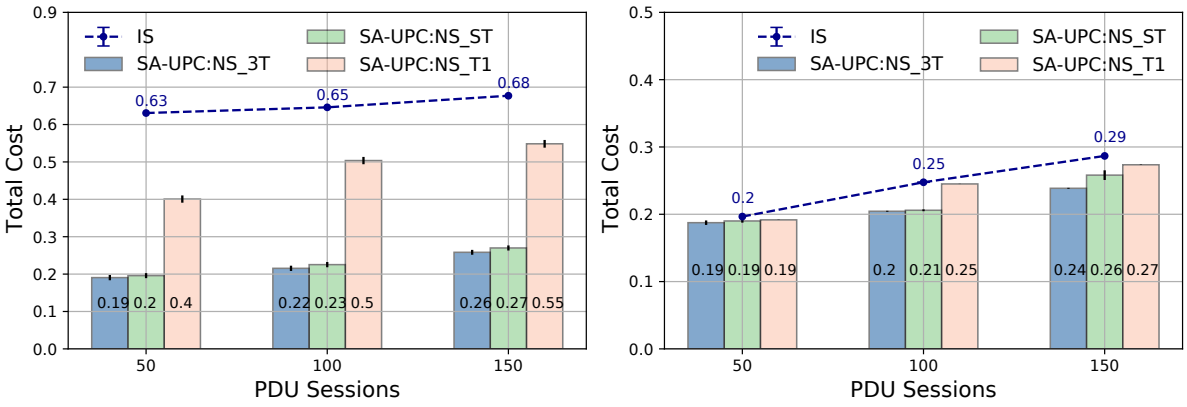
This subsection evaluates the performance of the presented SA-based solution. Since the proposed solution had several modifications compared to the classical SA (CSA), we ran various experiments by gradually introducing each change. This allowed us to investigate the advantages and implications of the changes in depth. First, we assessed the proposed neighborhood methods for a fixed MCL (FMCL) of 20 iterations without a restart-stop mechanism. Then, we studied the impact of the proposed restart-stop technique by comparing it with a classical restart approach that uses the current best solution as the restart point. Finally, we analyzed the performance of the variable MCL strategy versus a fixed approach. For these experiments, we considered initial and final temperatures of 100 and 0.01, respectively, and geometric scheduling with a cooling

factor (α) of 0.9.

5.4.3.1 Initial Solution and Neighborhood Method

This subsection provides the simulation results in terms of total cost and execution times for the proposed NS approaches (i.e., NS_3T and NS_ST). We compared the approaches' performance with a traditional method that introduces slight modifications to the current solution. We refer to this reference method as NS_T1 since it generates neighbor solutions by applying only type-one modifications. For type-one changes, we used a neighborhood set with a size of five samples. Moreover, we investigated their behavior when different approaches were applied to determine the initial solution (IS). Specifically, we considered two initial solution types: the first was generated by randomly assigning the SFCRs to candidate locations that complied with UPC constraints, while the second solution was the PC-UPC heuristic presented in Section 5.3.

Figure 5.5 summarizes the total cost obtained by the considered neighbor methods for different initial solutions and numbers of PDU sessions. This figure reveals that NS_3T and NS_ST always had the best performance, achieving substantial cost reductions compared to NS_T1. Their capacity to decrease the initial solution costs was more significant for the random initial solution, which was characterized by poor quality (see Fig. 5.5(**a**)). For this case scenario, the proposed neighborhood methods improved the initial solution by at least 60%, whereas NS_T1 only provided a cost reduction of up to 37% in the best-case scenario (i.e., $S = 50$).



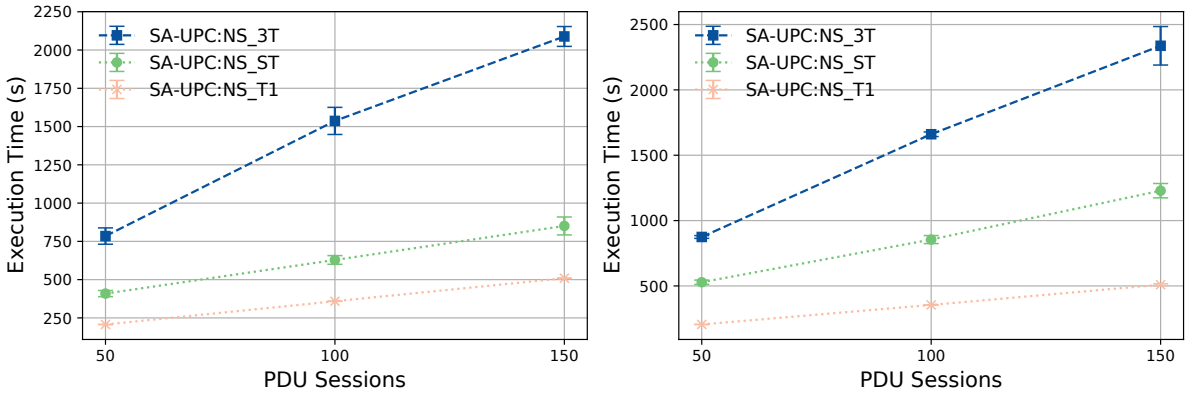
(**a**) Random initial solution

(**b**) PC-UPC as initial solution

Figure 5.5: SA-UPC placement cost for different neighborhood methods.

As seen in the figure, the initial solution quality highly conditioned the final cost obtained by the baseline method since it could not attain significant improvements. This can be better seen in Fig. 5.5(**b**), where the costs of the initial and final solutions for NS_T1 do not differ significantly. In contrast, the conceived strategies provided similar results regardless of the initial solution quality, with a maximum difference of 0.02 in the overall normalized cost.

Figure 5.6 displays the SA running times for each method, as well as the numbers of PDU sessions. The NS_T1 method had the lowest execution times, with values around two and four times smaller than NS_ST and NS_3T, respectively. This outcome was anticipated since our proposed methods performed more transformations in the generated neighbor solutions. Regarding the effects of the IS quality on this metric, the execution time of NS_T1 was similar for both initial solutions since it was unaware of the quality of the generated solutions. This was not the case for NS_ST and NS_3T strategies, which experienced a significant increment when the PC-UPC heuristic was selected as the initial state of the SA. The cause of this behavior lies in the IS quality. Specifically, good-quality initial solutions have more sessions mapped to the same VNFs and servers since fewer VNFs and servers are required. Thus, more PDU sessions are reassigned when applying type-two and -three changes. Moreover, NS_ST performs more diversification processes to improve the solution, given the fewer updates of the best solution.



(a) Random initial solution

(b) PC-UPC as initial solution

Figure 5.6: SA-UPC execution time for different neighborhood methods.

The simulation time of NS_ST remained much lower at all times than that of NS_3T, with a difference of at least 1.6 times for both types of initial solutions. Therefore, NS_TS provided similar costs to NS_3T at a lower execution time by adapting the neighborhood search to the generated solution quality and temperature values.

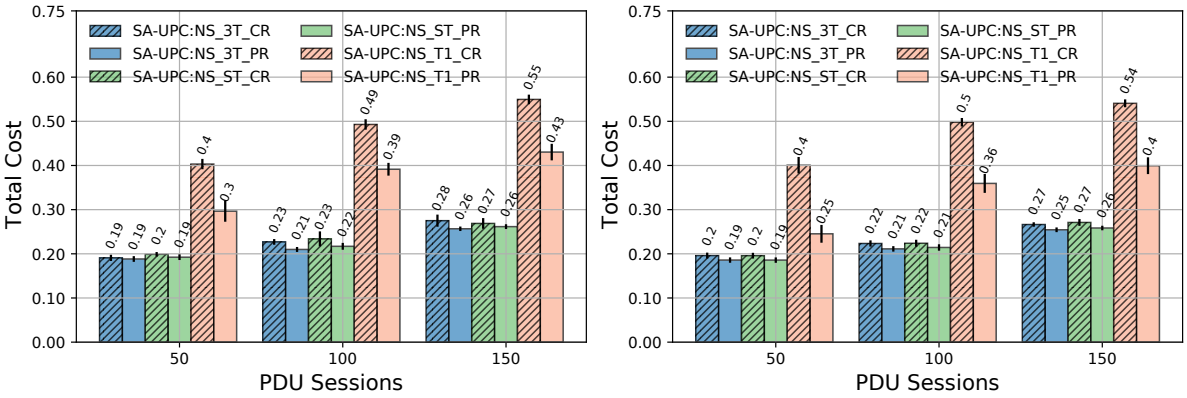
These results demonstrate that NS methods that combine exploration with exploitation of the solution space provide superior results to traditional methods. However, this outcome comes at the expense of higher execution times.

5.4.3.2 Restart–stop Criteria

In this subsection, we compare the performance of the proposed restart (PR) mechanism with a classical restart (CR) approach in which the current best solution is used as the restart point. We also analyze the impacts of using different stop periods on the final solution quality and SA execution time. We conducted these experiments for an FMCL of 20 iterations per temperature value and a random initial solution. We preferred a random initial solution because

the cost reductions in the output solution were more noticeable. Additionally, our proposed restart approach generated the restart solution based on the current best solution by reassigning the PDU sessions assigned to a maximum of two servers that were randomly selected.

Figure 5.7 provides the SA output costs for a restart period of three temperatures ($P_r = 3$) and different stop conditions (i.e., three and five restarts, $P_s = 3$ and $P_s = 5$). This figure shows that the proposed restart approach, represented by bars without stripes, provided lower-cost solutions than the baseline (striped bars) regardless of the selected NS method. Moreover, slight improvements in the proposed NS methods were observed for both the classical and proposed restart approaches when more restart attempts were considered in the stopping condition. In contrast, the algorithm based on the NS_T1 method obtained more significant reductions in its output cost for the stopping condition with more restart attempts (see Fig. 5.7(**b**)). This was because the baseline NS depends on additional diversification strategies, such as restarts, to explore the solution space further since it only introduces minor modifications to the generated neighbor solutions. Therefore, unlike the PR approach, the CR could not further improve the output solution quality for NS_T1, despite the increment in the number of restarts shown in Fig. 5.7(**a**). This outcome demonstrated that a classical restart may not help the SA algorithm escape from local optima and may thus be inefficient.



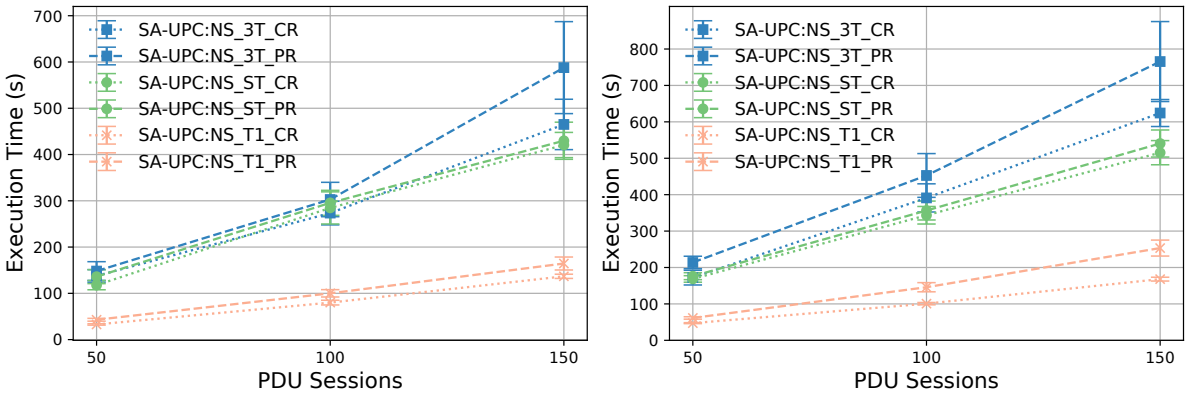
(**a**) $P_r = 3$ and $P_s = 3$.

(**b**) $P_r = 3$ and $P_s = 5$.

Figure 5.7: Obtained cost for different restart-stop periods.

In Fig. 5.8, the execution time of both restart-stop strategies is presented. The proposed restart mechanism required higher execution times than the baseline. However, this difference was insignificant except for the NS_3T method with 150 PDU sessions; most of the time, the proposed restart strategy improved the current best solution, which delayed the occurrence of restarts, and subsequently, that of meeting the stopping condition.

By comparing these results with those obtained for no restart-stop condition discussed in the previous subsection, we observed that the initial solution quality was significantly improved with the adoption of the proposed restart-stop strategy, as was the algorithm running time. Specifically, for NS_ST, NS_T1, and NS_3T methods, the restart-stop with $P_s = 3$ and $P_r = 3$ decreased the SA



(**a**) $P_r = 3$ and $P_s = 3$.

(**b**) $P_r = 3$ and $P_s = 5$.

Figure 5.8: SA algorithm execution time for different restart-stop periods.

execution time by at least 50%, 65%, and 70%, respectively. Correspondingly, for $P_s = 3$ and $P_r = 5$, the SA algorithm execution time was reduced by more than 40%, 50%, and 65%. Additionally, the utilization of the restart-stop mechanism significantly decreased the gap between NS_3T and NS_ST running times. These results demonstrate the effectiveness of the proposed restart-stop approach in increasing the probability of finding better solutions and reducing the SA running time. Nevertheless, the stop period must be carefully set since the algorithm execution time tended to increase with this parameter.

5.4.3.3 VMCL Strategy

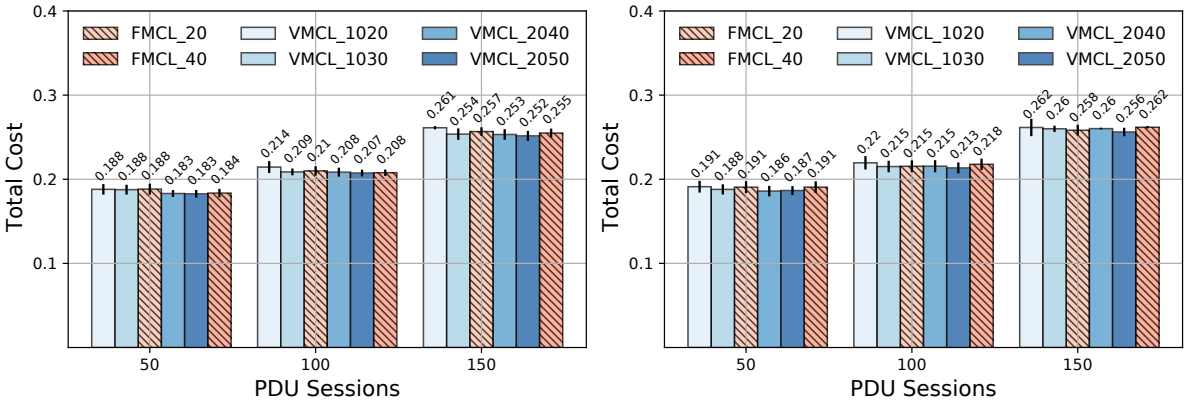
In this subsection, we evaluate the impacts of the envisioned VMCL strategy under different conditions. Concretely, we investigated the strategy's behavior for the proposed NS methods with and without the restart-stop condition and compared it to an FMCL approach. Different values of the MCL were assessed for fixed and variable approaches. To facilitate analysis of the results, we grouped them into two categories based on the MCL (see Table 5.5). As was described in the previous subsection, we considered a random solution as the algorithm's initial state.

Table 5.5: Values of MCL considered in the experiments.

Group	Approach	MCL	Notation
I	Variable	10 & 20	VMCL_1020
	Variable	10 & 30	VMCL_1030
	Fixed	20	FMCL_20
II	Variable	20 & 40	VMCL_2040
	Variable	20 & 50	VMCL_2050
	Fixed	40	FMCL_40

Figure 5.9 shows the impact of different MCL approaches on the UPC output cost provided by the SA solution. These results revealed no significant improvements in the quality of the output

solutions with the increased number of inner iterations. Comparison of FMCL_20 and FMCL_40 revealed that the output cost reductions obtained by FMCL_40 were relatively slight despite having twice as many iterations as FMCL_20. Concerning the performance of variable versus fixed MCLs, the former strategy provided similar or better costs than its fixed analogous for most examined use cases. This outcome was more remarkable for the second group of MCLs, which had greater MCLs than the other. Overall, the worst performance was provided by VMCL_1020 since this variant performed fewer iterations and, therefore, poorer scans of the solution space.



(**a**) NS_3T with restart-stop ($P_r = 3$ and $P_s = 3$).

(**b**) NS_3T without restart-stop.

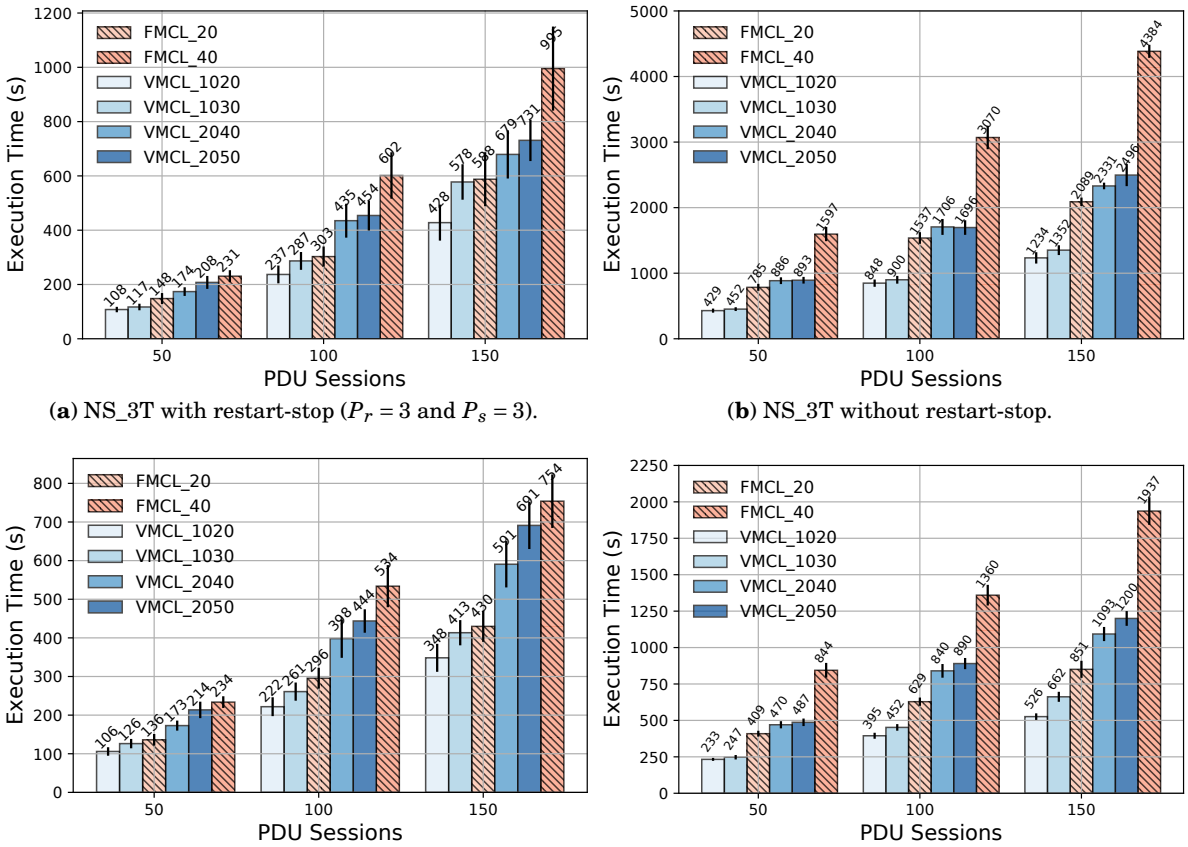
(**c**) NS_ST with restart-stop ($P_r = 3$ and $P_s = 3$).

(**d**) NS_ST without restart-stop.

Figure 5.9: SA output cost for different MCL strategies with and without restart-stop.

Figure 5.10 represents the running time of the SA algorithm for all analyzed use cases. As expected, the value of this metric increased with the MCL. Nonetheless, the proposed VMCL solutions required shorter execution times than their corresponding fixed variants. This was most notable in the second group of MCL values, as well as SA without restart-stop. For these combined use cases, VMCL_2040 decreased the overall execution time of the algorithm between 25% and 42 % for NS_ST and around 43% for the NS_3T neighbor solution compared to FMCL_40.

Overall, the combination of restart-stop with VMCL did not degrade the algorithm's final



(**a**) NS_3T with restart-stop ($P_r = 3$ and $P_s = 3$).

(**b**) NS_3T without restart-stop.

(**c**) NS_ST with restart-stop ($P_r = 3$ and $P_s = 3$).

(**d**) NS_ST without restart-stop.

Figure 5.10: SA execution time for different MCL strategies with and without restart-stop.

solution quality. In fact, we obtained better results when using restart-stop than when excluding it, as evidenced by the above-described experiments.

5.4.4 UPC Performance Evaluation

In this subsection, we analyze the performance of the proposed solutions in terms of normalized cost, execution time, average E2E delay, number of open servers, and deployed UPFs. Additionally, we describe a variant of the classical SA, which generated neighbor solutions by introducing only type-one changes. This baseline, called CSA_T1, had an FMCL of 20 iterations per temperature value and always used the best current solution as the restart point. In contrast, the SA-UPC algorithm adopted the VMCL_1030 and the proposed restart method. We considered a restart period of three consecutive temperatures without updating the best solution and a stopping condition of three restarts. The proposed PC-UPC heuristic was used for the SA initial solution. Regarding the exact solution approach (ILP model), we present simulation results for up to 100 active PDU sessions since it could not solve the problem for higher numbers of sessions within a reasonable time.

5.4.4.1 Overall Cost

Figure 5.11 depicts the average total cost provided by all analyzed solutions for different values of PDU session requests. We observed that the total cost increased linearly with SFCRs. This outcome was expected since more servers and VNF instances must be activated to cope with increasing service demands. The proposed ILP model and the SA-UPC solution provided the best costs. Moreover, the conceived metaheuristic obtained almost identical results to the optimal one in both of its variants (i.e., SA-UPC_3T and SA-UPC_ST).

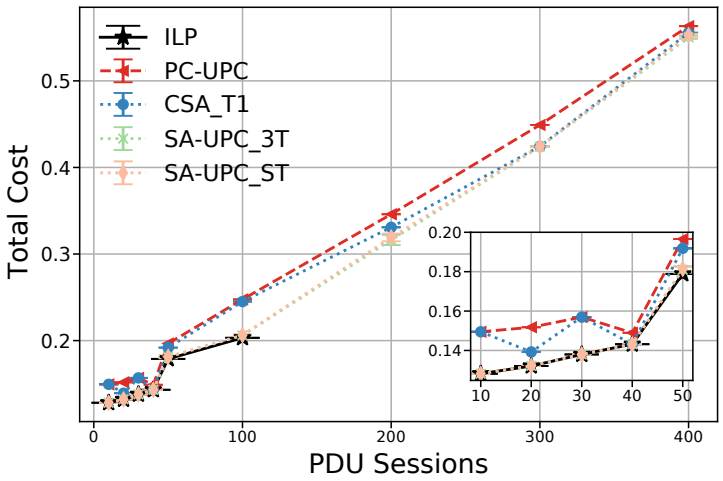


Figure 5.11: Total cost versus different numbers of SFCs.

The devised SA with the NS_ST mechanism provided similar results to NS_3T when using an IS with acceptable quality and the proposed restart-stop strategy. The PC-UPC had the worst performance in these experiments since it was selected as the initial point of the SA-based solutions. Nevertheless, its difference from the optimal was at most 22% in the worst-case scenario.

Additionally, the conceived SA-UPC solution provided cost reductions ranging from 2.3% to 17%, whereas the CSA_T1 improved the quality of the initial solution by only 8.5% in the best-case scenario. Furthermore, the improvement achieved by this method was mainly null or scarce, with less than 3%. In the cases in which the proposed SA had similar results to the baseline, the proposed heuristic, used as the initial solution, obtained near-optimal results (i.e., $S = 40$) or the minimum numbers of VNFs and open servers (i.e., $S = 300$ and $S = 400$) needed to fulfill the service's demand. Thus, the only possibility for improvement in those cases was the latency, which held minor importance for the objective function.

5.4.4.2 Running Time

The average running time required for each solution to solve the UPC problem is summarized in Fig. 5.12. As seen in the figure, the running time of the ILP model significantly differed from

that of the heuristic-based solutions. The execution time of the ILP model grew exponentially with the number of active PDU sessions. Moreover, for SFCRs higher than 100, the model could not converge toward the optimal solution due to the vast number of combinations. In contrast, the heuristic-based solutions' running time increased linearly with the number of active PDU sessions. In this respect, the PC-UPC heuristic provided the best computing time, which solved the problem within a couple of seconds (i.e., less than 10 s), with an average optimality gap below 13.5%. Additionally, the computation time of the baseline compared to the proposed SA was, on average, 2.8 and 5.5 times faster than SA-UPC_ST and SA-UPC_3T, respectively. However, this speed was achieved at the expense of fewer improvements regarding the quality of the final solution.

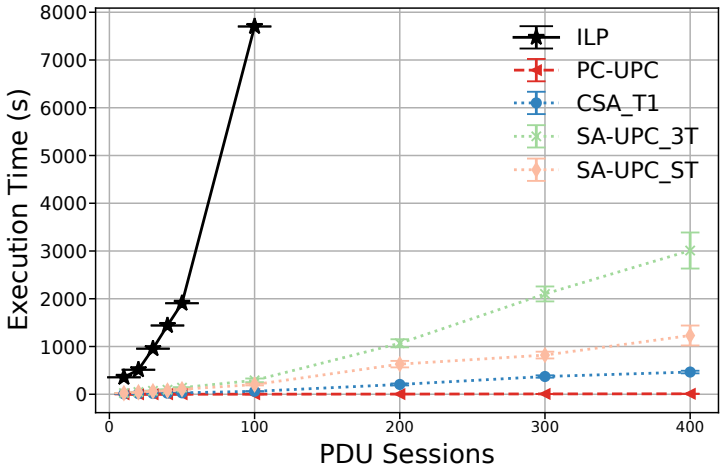


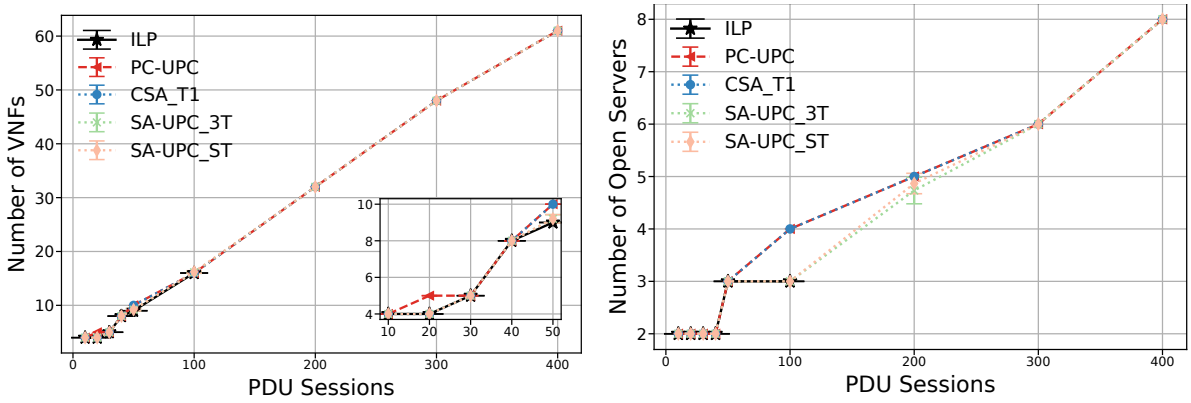
Figure 5.12: Running time versus different numbers of SFCs.

5.4.4.3 Other Metrics

In this section, we discuss other aspects of the designed solutions by describing the behavior of each term in the objective function (5.4).

Figure 5.13 shows the average number of active servers and deployed UPFs obtained by all analyzed solutions for different SFCRs. As seen in the figure, the PC-UPC heuristic provided solutions that differed from the optimal, at most, in the activation of one server or a VNF instance, but not both simultaneously. Moreover, both variants of the conceived SA-UPC obtained identical results to the mathematical model, outperforming the other heuristic approaches.

Similar results were attained regarding the average E2E delay parameter, as indicated by Fig. 5.14. Concretely, the SA-UPC approach was the one that most closely resembled the performance of the ILP model. Moreover, the NS_3T and NS_ST solutions behaved similarly for all analyzed SFCR sets. Furthermore, our proposed SA typically required lower average delays than the classical SA baseline, even in cases with fewer active servers or deployed UPFs.



(a) Number of active servers.

(b) Number of UPFs instances.

Figure 5.13: Numbers of active servers and UPFs versus different numbers of SFCs.

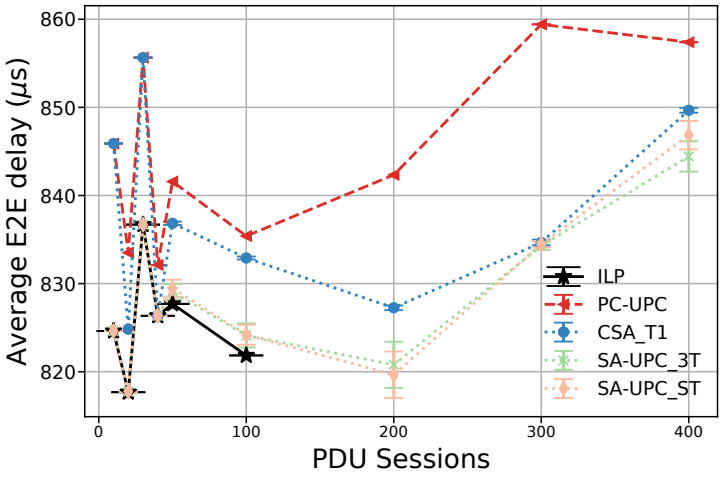


Figure 5.14: Average E2E latency versus different numbers of SFCs.

5.5 Conclusion

In this chapter, we investigated the 5G UPC problem. We described an ILP and two heuristic-based solutions to address this problem, which we referred to as PC-UPC and SA-UPC. Their main objective is to minimize expenditures associated with PDU session provisioning and to enhance QoS (i.e., network response time).

The mathematical model became computationally intractable as the size of the involved sets increased. As evidenced by the conducted experiments, the model's computational time grew exponentially with the number of PDU sessions until it was unable to determine a solution in a reasonable time when the number of SFCRs was higher than 100 in a medium-size 5G topology. We presented heuristic and metaheuristic-based solutions to overcome these limitations, introducing various mechanisms that enhanced their performance.

The simulation results revealed that the proposed solutions not only outperformed some

existing benchmarks but obtained near-optimal results in significantly less time than the exact solution when the problem scale was small. Specifically, the PC-UPC heuristic solved the problem within seconds with an average optimality gap of around 13.5%. Additionally, the SA algorithm allowed for results almost identical to the optimal solution, although at the expense of higher running times than the heuristic. We verified that the introduced modifications significantly improved the solution's performance, enabling it to provide near-optimal results regardless of the quality of the initial solution. Our proposed strategies provided the necessary flexibility and efficiency to meet 5G service stringent requirements, such as latency and device density, while reducing their deployment and operational costs.

CHAPTER 6

DYNAMIC UPF PLACEMENT AND CHAINING

This chapter is based on:

- **I. Leyva-Pupo**, C. Cervelló-Pastor, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic UPF Placement and Chaining Reconfiguration in 5G Networks," *Computer Networks*, vol. 215, p.109200, 2022.

This chapter addresses the problem of dynamic UPF placement and chaining reconfiguration (UPCR) in MEC environments to cope with user mobility while guaranteeing cost savings and acceptable QoS. We proposed an ILP model and a heuristic algorithm, called dynamic priority and cautions UPCR (DPC-UPCR) to solve the problem. These solutions aim to minimize multiple cost components involved in the UPCR procedure, such as VNF migration and session reassignments. The DPC-UPC algorithm seeks to reduce the solution time complexity to extend its applicability to online scenarios. The algorithm is based on the SFCR-mapping procedure presented in Chapter 5 and incorporates two strategies (i.e., partial unmapping of SFCRs and an improvement phase) to enhance the solution's efficiency.

Moreover, a decision-making mechanism based on OST referred to as the optimal scheduling of the reconfiguration (OSR) is provided to determine the optimal time to readjust the UPC configuration. This decision is made according to instantaneous values of sessions with latency violations, a pre-established QoS threshold, and an expected reconfiguration cost.

The rest of the chapter is organized as follows. Section 6.1 describes the system model and the solutions for the UPCR problem. Section 6.2 provides a scheduling mechanism to decide the optimal reconfiguration time. Next, Section 6.3 investigates the performance of the proposed solutions, while Section 6.4 presents the chapter's conclusions.

6.1 UPF Placement and Chaining Reconfiguration

This section first presents the UPCR problem, the system model, and the used notation. Next, it provides an exact solution and a heuristic-based approach to tackle this problem. These solutions aim to optimize deployment and operational expenditures associated with reconfiguration events.

6.1.1 Problem Statement

The existence of multiple UPFs performing different functionalities in a PDU session data path adds extra complexity to the UPF placement problem. We had to account for UPF order and inter-dependency requirements under stringent service latency demands and edge nodes' limited resources. This requires higher deployment and operational costs, as more EN and UPF instances must be deployed to fulfill service requirements such as latency, bandwidth, and user density. Furthermore, frequent session relocations in MEC ecosystems are likely to occur due to highly mobile users and UPF's smaller service areas. Costs and session relocations can be reduced by decreasing the number of UPF instances and consolidating their deployment on a small subset of edge servers. However, this approach may provoke QoS deterioration and higher routing costs.

In dynamic environments, where user positions and demands change over time, the reconfiguration of the UPF placement and chaining setup may be needed to ensure QoS and guarantee efficient usage of resources. Therefore, we addressed the problem of determining the best UPC configuration that allows reduced expenditure while ensuring 5G service requirements.

6.1.1.1 System Model

We model the 5G network as a directed graph G(N, E), where N and E denote the sets of network nodes and links, respectively. The set of nodes is formed by three subsets: access nodes (N_r), aggregation points (N_a), and server nodes (N_c). For the VNF placement, the server nodes are analyzed as candidate locations since they provide virtualization resources required to host VNF instances. The parameter C_c represents the resources, such as memory and CPU, available at a candidate location $c \in N_c$. A physical link $(u,v) \in E$ is characterized by bandwidth capacity ($\beta_{u,v}$) and latency ($d_{u,v}$). This latency comprises the link propagation delay and the processing time of the underlying transmission nodes. Additionally, we consider a set of pre-calculated paths (P), where each path $p \in P$ is identified by its two endpoints (n,m) and an ID (h). This ID helps to distinguish different paths between the same pair of nodes. Moreover, the binary indicator $H^p_{u,v}$ specifies a link mapping (u,v) to a path $p \in P$.

We denote the available VNF types as T, where $t \in T$ represents a specific VNF type. Every VNF type has an associated processing capacity (C_t), a processing time (d_t), and a maximum number of instances that can be deployed (I_t). Additionally, the set F represents those VNF instances that were deployed during previous placement events, where (i,t) identifies the *i-th*

instance of type t in F. Table 6.1 summarizes the used notation for sets and parameters associated with physical and virtual networks.

Table 6.1: Notation for physical and virtual network elements in the UPCR problem.

Notation	Description
N	Set of all nodes
N_r	Set of access nodes
N_c	Set of candidate locations (e.g., MEC servers)
N_a	Set of aggregation points
E	Set of physical links
P	Set of paths between all network nodes
$P_{n,m}$	Set of paths between nodes n and m, $(n,m \in N)$
T	Set of all types of available VNFs
F	Set of deployed VNF instances
C_c	Resource capacity at candidate location $n \in N_c$
C_t	Resource capacity of VNF of type $t \in T$
$\beta_{u,v}$	Bandwidth capacity of link (u,v)
$d_{u,v}$	Latency associated with link (u,v)
d_p	Latency associated with path $p \in P$
d_t	Processing delay of VNF of type $t \in T$
I_t	Maximum number of instances of type $t \in T$
Ψ	Cost component

We consider a set of mobile devices connected to the (R)AN through their nearest access node ($n_r^s \in N_r^s$) that are requesting PDU sessions. The set of PDU service requests is denoted as S. A PDU session $s \in S$ is compounded by an ordered set of VNF services (F_s) and demands certain processing capacity (C_s), bandwidth (B_s), and E2E service delay (L_s) to be mapped. We represent the properties of a given SFCR $s \in S$ using a tuple $(n_r^s, F_s, C_s, \beta_s, L_s, B_s, T_s^{f,t}, O_s^{f,g,b}, Q_s^{f,b})$, where parameters B_s, $T_s^{f,t}$, $O_s^{f,g,b}$ and $Q_s^{f,b}$ denote the number of branches in an SFCR, the type of each VNF, and their order and presence in the branches forming the SFC, respectively. Table 6.2 provides the used notation for the SFCR representation.

Table 6.2: Notation for sets and parameters associated with SFCRs

Notation	Description
S	Set of PDU sessions (SFCRs)
N_r^s	Set of access nodes (N_r) per PDU sessions (S)
F_s	Set of VNFs forming SFCR $s \in S$
n_r^s	Access node ($n_r \in N_r$) of SFCR $s \in S$
$\|F_s\|$	Number of VNFs forming SFCR $s \in S$
C_s	Computing resources required by SFCR $s \in S$
β_s	Bandwidth capacity required by SFCR $s \in S$
L_s	E2E latency requirement of SFCR $s \in S$
B_s	Number of aUPFs (branches) in SFCR $s \in S$

Table 6.3 describes the binary variables and parameters used for the UPCR problem solution.

Table 6.3: Notation for decision variables and binary parameters (Param.) involved in the UPCR problem.

Notation	Var.	Param.	Description
w_n	x		1 if candidate node $n \in N_c$ is open
r_s	x		1 if PDU session $s \in S$ was reassigned during the reconfiguration.
$v_{i,t}$	x		1 if as a result of the placement reconfiguration there is a new instance $i \in I_t$ of VNF type $t \in T$
$x_{i,t,n}$	x		1 if instance $i \in I_t$ of VNF type $t \in T$ is deployed on node $n \in N_c$
$m_{i,t,n',n}$	x		1 if the i-th instance of VNF type $t \in T$ was migrated from node n' to node n ($n', n \in N_c$)
$z^{f,s}_{i,t,n}$	x		1 if VNF $f \in F_s$ of SFCR $s \in S$ is mapped to instance $i \in I_t$ of VNF type $t \in T$ located at node $n \in N_c$
$a^{f,s}_n$	x		1 if VNF $f \in F_s$ of SFCR $s \in S$ is assigned to node $n \in N_c$
$y^{f,g,s}_p$	x		1 if path $p \in P$ is used to route traffic between VNFs f and g ($f, g \in F^+_s$) of SFCR $s \in S$
$\delta^{f,g,s}_{n,m}$	x		1 if VNFs f and g ($f, g \in F_s$) of SFCR $s \in S$ are mapped on nodes n and m ($n, m \in N_c$), resp.
ξ_t	x		1 if there is at least one new VNF of type $t \in T$ deployed in the network
$\bar{X}_{i,t,n}$		x	1 if instance $i \in I_t$ of VNF type $t \in T$ was placed on node $n \in N_c$ before the reconfiguration
$\bar{A}^{f,s}_n$		x	1 if VNF $f \in F_s$ of SFCR $s \in S$ is hosted in node $n \in N_c$
$H^p_{u,v}$		x	1 if physical link $(u,v) \in E$ is mapped to path $p \in P$
V^t_n		x	1 if node $n \in N$ supports VNFs of type $t \in T$
$T^{f,t}_s$		x	1 if VNF $f \in F_s$ in SFCR $s \in S$ is of type $t \in T$
$O^{f,g,b}_s$		x	1 if VNF f goes just before VNF g in branch $b \in B_s$ of SFCR $s \in S$
$Q^{f,b}_s$		x	1 if VNF $f \in F_s$ is present in branch $b \in B_s$ of SFCR $s \in S$

6.1.2 Model: Optimal UPF Placement and Chaining Reconfiguration

The main objective of the optimal UPCR (O-UPCR) solution is to minimize deployment and operational costs associated with the UPF placement and chaining during reconfiguration events. To this aim, we consider multiple cost components:

- Node activation cost (C_{act}): The cost associated with the activation of a server.

$$C_{act} = \sum_{n \in N_c} \Psi^n_a \cdot w_n \tag{6.1}$$

- VNF deployment cost (C_{dep}): The cost related to instantiating new VNFs (e.g., software license cost).

$$C_{dep} = \sum_{i \in I_t} \sum_{t \in T} \Psi^t_d \cdot v_{i,t} \tag{6.2}$$

- VNF running cost (C_{run}): The cost of running VNF instances at a given candidate location (e.g., power consumption).

$$C_{run} = \sum_{i \in I_t} \sum_{t \in T} \sum_{n \in N_c} \Psi^{t,n}_r \cdot x_{i,t,n} \tag{6.3}$$

- VNF migration cost (C_{mig}): The cost of migrating a VNF instance from one location to another. Thus, we define it in terms of the set of already deployed VNF instances (F).

$$C_{mig} = \sum_{(i,t)\in F} \sum_{n'\in N_c} \sum_{n\in N_c} \Psi_m^{t,n',n} \cdot m_{n',n}^{i,t} \tag{6.4}$$

- Routing cost (C_{rou}): The cost of routing traffic between the VNF services forming the PDU sessions. This expression can also improve network response time since it includes the propagation delay of the virtual links that form the SFC data path.

$$C_{rou} = \sum_{f,g\in F_s^+} \sum_{s\in S} \sum_{p\in P} \Psi^{s,p} \cdot d_p \cdot y_p^{f,g,s} \tag{6.5}$$

- Session reassignment cost (C_{rea}): The cost for reassigning PDU sessions during the UPC recalculation. This cost is measured as a penalty that the service provider must pay for interrupting user sessions or exceeding the service delay requirement. We consider a session to be reassigned if at least one of its requested VNF services has been relocated from its previously assigned server.

$$C_{rea} = \sum_{s\in S} \Psi^s \cdot r_s \tag{6.6}$$

The O-UPCR solution aims to determine the optimal UPC arrangement, thereby minimizing the total costs incurred during reconfiguration events. To achieve this goal, we express its objective function as a linear combination of the cost components mentioned above. Additionally, we introduce weight factors (α_i) to reflect the relative importance of each component. Thus, the objective function of the O-UPCR model can be expressed as follows:

$$\begin{aligned}\text{Min: } & \alpha_1 \cdot \sum_{n\in N_c} \Psi_a^n \cdot w_n + \alpha_2 \cdot \sum_{i\in I_t} \sum_{t\in T} \Psi_d^t \cdot v_{i,t} + \alpha_3 \cdot \sum_{i\in I_t} \sum_{t\in T} \sum_{n\in N_c} \Psi_r^{t,n} \cdot x_{i,t,n} + \\ & \alpha_4 \cdot \sum_{(i,t)\in F} \sum_{n'\in N_c} \sum_{n\in N_c} \Psi_m^{t,n',n} \cdot m_{n',n}^{i,t} + \alpha_5 \cdot \sum_{f,g\in F_s^+} \sum_{s\in S} \sum_{p\in P} \Psi^{s,p} \cdot d_p \cdot y_p^{f,g,s} + \alpha_6 \cdot \sum_{s\in S} \Psi^s \cdot r_s \end{aligned} \tag{6.7}$$

A set of constraints must be satisfied to generate feasible solutions. We group these constraints into VNF, path, reconfiguration, QoS, and capacity restrictions.

VNF constraints: Inequality (6.8) restricts the maximum number of instances of a given type to be deployed according to the I_t parameter. Constraint (6.9) confines the placement of a VNF instance to a unique location in the network. Additionally, constraint (6.10) guarantees the deployment of VNF instances in open candidates that support the requested VNF type. Expression (6.11) forces the deactivation of MEC servers that are not hosting VNF instances.

$$\sum_{i\in I_t} \sum_{n\in N_c} x_{i,t,n} \leq I_t \qquad \forall t \in T \tag{6.8}$$

$$\sum_{n\in N_c} x_{i,t,n} \leq 1 \qquad \forall i \in I_t, \forall t \in T \tag{6.9}$$

$$x_{i,t,n} \leq w_n \cdot V_n^t \qquad \forall i \in I_t, \forall t \in T, \forall n \in N_c \tag{6.10}$$

$$w_n \leq \sum_{i \in I_t} \sum_{\forall t \in T} x_{i,t,n} \qquad \forall n \in N_c \tag{6.11}$$

Inequality (6.12) expresses that a VNF service forming an SFCR can be assigned to a VNF instance as long as this VNF has already been mapped in the infrastructure and is of the same VNF type. Additionally, constraint (6.13) ensures that a VNF service request $f \in F_s$ is served by only one VNF instance. Moreover, restriction (6.14) avoids the deployment of empty VNFs by ensuring that all the deployed VNF instances have assigned at least one SFCR.

$$z_{i,t,n}^{f,s} \leq x_{i,t,n} \cdot T_s^{f,t} \qquad \forall f \in F_s, \forall s \in S, \forall i \in I_t, \forall t \in T, \forall n \in N_c \tag{6.12}$$

$$\sum_{i \in I_t} \sum_{t \in T} \sum_{n \in N} z_{i,t,n}^{f,s} = 1 \qquad \forall f \in F_s, \forall s \in S \tag{6.13}$$

$$x_{i,t,n} \leq \sum_{s \in S} \sum_{f \in F_s} z_{i,t,n}^{f,s} \qquad \forall i \in I_t, \forall t \in T, \forall n \in N_c \tag{6.14}$$

The relationship between the binary variables $a_n^{f,s}$ and $z_{i,t,n}^{f,s}$ is established through expressions (6.15) and (6.16). Specifically, these two expressions state that if a VNF service request is mapped to a location, it has been assigned to a VNF instance deployed on that node.

$$a_n^{f,s} \leq \sum_{t \in T} \sum_{i \in I_t} z_{i,t,n}^{f,s} \qquad \forall f \in F_s, \forall s \in S, \forall n \in N_c \tag{6.15}$$

$$a_n^{f,s} \geq z_{i,t,n}^{f,s} \qquad \forall f \in F_s, \forall s \in S, \forall i \in I_t, \forall t \in T, \forall n \in N_c \tag{6.16}$$

Inequality (6.17) defines the anti-affinity property for VNFs of the same type (e.g., anchor UPFs) serving an SFCR. Concretely, it forces the deployment of VNF instances of the same type serving the same SFCR on different servers. Constraints (6.18) is specific for UPFs; it stipulates that anchor and IUPFs (iUPF with no BP or UL-CL functionality) assigned to the same PDU session must be mapped to different locations.

$$\sum_{f \in F_s} \sum_{i \in I_t} z_{i,t,n}^{f,s} \leq 1 \qquad \forall s \in S, \forall t \in T, \forall n \in N_c \tag{6.17}$$

$$\sum_{f \in F_s} \sum_{i \in I_t} z_{1,i,n}^{f,s} + \sum_{f \in F_s} \sum_{i \in I_t} z_{3,i,n}^{f,s} \leq 1 \qquad \forall s \in S, \forall n \in N_c \tag{6.18}$$

Path mapping constraints: Constraint (6.19) guarantees that a path exists in the specified direction (i.e., $f \rightarrow g$) between two consecutive VNFs in every branch forming an SFCR. Moreover, expression (6.20) limits the maximum number of assigned paths between any pair of VNFs to one. This avoids the existence of loops between successive VNFs that form an SFC. For these two constraints, we extend the set of SFCR constituent VNFs to include the requests' access node ($F_s^+ = F_s \cup n_r^s$). In this manner, the path between the access node and the first VNF in the chain is also mapped. Furthermore, inequality (6.21) ensures the mapping of consecutive VNFs to the endpoints of the established data paths. Similarly, constraint (6.22) is an adaptation of (6.21) to

include paths with an access node as one of their endpoints.

$$\sum_{p \in P} y_p^{f,g,s} \geq O_s^{f,g,b} \qquad \forall f,g \in F_s^+, \forall b \in B_s, \forall s \in S \tag{6.19}$$

$$\sum_{p \in P} y_p^{f,g,s} \leq 1 \qquad \forall f,g \in F_s^+, \forall s \in S \tag{6.20}$$

$$\sum_{p \in P_{n,m}} y_p^{f,g,s} \leq a_n^{f,s} \cdot a_m^{s,g} \qquad \forall f,g \in F_s, \forall s \in S, \forall n,m \in N_c \tag{6.21}$$

$$\sum_{p \in P_{n_r^s,m}} y_p^{n_r^s,g,s} \leq a_m^{s,g} \qquad \forall g \in F_s, \forall s \in S, n_r^s = N_r^s[s], \forall m \in N_c \tag{6.22}$$

Inequality (6.21) represents a non-linear constraint since it implies the product of two binary variables. However, we can express it in linear form using the following expressions:

$$\delta_{n,m}^{f,g,s} \leq a_n^{f,s} \qquad \forall f,g \in F_s, \forall s \in S, \forall n,m \in N_c \tag{6.23}$$

$$\delta_{n,m}^{f,g,s} \leq a_m^{s,g} \qquad \forall f,g \in F_s, \forall s \in S, \forall n,m \in N_c \tag{6.24}$$

$$\delta_{n,m}^{f,g,s} \geq a_n^{f,s} + a_m^{s,g} - 1 \qquad \forall f,g \in F_s, \forall s \in S, \forall n,m \in N_c \tag{6.25}$$

$$\sum_{p \in P_{n,m}} y_p^{f,g,s} \leq \delta_{n,m}^{f,g,s} \qquad \forall f,g \in F_s, \forall s \in S, \forall n,m \in N_c \tag{6.26}$$

Reconfiguration constraints: Expressions (6.27)–(6.30) help determine the type of changes produced during a reconfiguration event. Restriction (6.27) indicates the migration of a VNF instance when its location is different before and after the reconfiguration. This constraint is linear since $X_{i,t,n}$ is a binary indicator, not a variable. Regarding VNF new deployments, we define expressions (6.28) and (6.29). Mainly, they indicate the placement of new instances of a given type, as the activation of new VNF IDs and the incremental in the number of deployed instances with reference to the previous placement configuration. These two constraints promote VNF migration over new deployments when the deployment component is omitted from the optimization function in (6.7). Otherwise, deploying new instances could be preferred when optimizing migration effects.

$$m_{n',n}^{i,t} = x_{i,t,n} \cdot \bar{X}_{i,t,n'} \qquad \forall (i,t) \in F, \forall n,n' \in N_c; n \neq n' \tag{6.27}$$

$$v_{i,t} = \Big[\sum_{n \in N_c} x_{i,t,n} - \sum_{n \in N_c} \bar{X}_{i,t,n} \Big]^+ \qquad \forall i \in I_t, \forall t \in T \tag{6.28}$$

$$\sum_{i \in I_t} v_{i,t} = \Big[\sum_{n \in N_c} \sum_{i \in I_t} x_{i,t,n} - \sum_{n \in N_c} \sum_{i \in I_t} \bar{X}_{i,t,n} \Big]^+ \qquad \forall t \in T \tag{6.29}$$

Expression (6.30) indicates the reassignment of PDU sessions during the reconfiguration procedure. It stipulates that an SFCR has been reassigned when at least one constituent VNF has been relocated. We assume that the relocation of VNF service requests ($f \in F_s$) between VNF instances deployed inside the same server does not affect the session and service continuity. However, for more restrictive considerations, this constraint could be adjusted to express the r_s variable in terms of the variable $z_{i,t,n}^{f,s}$ instead of $a_n^{f,s}$.

$$r_s = 1 \Leftrightarrow |F_s| - \sum_{f \in F_s} \sum_{n \in N_c} a_n^{f,s} \cdot \bar{A}_n^{f,s} \geq 1 \qquad \forall s \in S \tag{6.30}$$

Constraints (6.28)-(6.30) are non-linear. Nonetheless, they can be reformulated in a linear form as follows:

$$(6.28) \Leftrightarrow \begin{cases} v_{i,t} \geq \sum\limits_{n \in N_c} x_{i,t,n} - \sum\limits_{n \in N_c} \bar{X}_{i,t,n} & \forall i \in I_t, \forall t \in T \\ 2v_{i,t} \leq 1 + \sum\limits_{n \in N_c} x_{i,t,n} - \sum\limits_{n \in N_c} \bar{X}_{i,t,n} & \forall i \in I_t, \forall t \in T \end{cases} \tag{6.31}$$

$$(6.29) \Leftrightarrow \begin{cases} \sum\limits_{i \in I_t} v_{i,t} = \xi_t \cdot (\sum\limits_{n \in N_c} \sum\limits_{i \in I_t} x_{i,t,n} - \sum\limits_{n \in N_c} \sum\limits_{i \in I_t} \bar{X}_{i,t,n}) & \forall t \in T \\ \xi_t \geq (\sum\limits_{n \in N_c} \sum\limits_{i \in I_t} x_{i,t,n} - \sum\limits_{n \in N_c} \sum\limits_{i \in I_t} \bar{X}_{i,t,n})/I_t & \forall t \in T \\ \xi_t \leq (I_t + \sum\limits_{n \in N_c} \sum\limits_{i \in I_t} x_{i,t,n} - \sum\limits_{n \in N_c} \sum\limits_{i \in I_t} \bar{X}_{i,t,n})/(I_t + 1) & \forall t \in T \end{cases} \tag{6.32}$$

$$(6.30) \Leftrightarrow \begin{cases} \sum\limits_{f \in F_s} \sum\limits_{n \in N_c} a_n^{f,s} \cdot \bar{A}_n^{f,s} \geq |F_s|(1 - r_s) & \forall s \in S \\ \sum\limits_{f \in F_s} \sum\limits_{n \in N_c} a_n^{f,s} \cdot \bar{A}_n^{f,s} \leq |F_s| - r_s & \forall s \in S \end{cases} \tag{6.33}$$

QoS constraints: We measure the system QoS in terms of service latency. This parameter is expressed as the combination of VNF processing times and the propagation delays between consecutive VNFs in every branch forming an SFCR. In this respect, constraint (6.34) guarantees that the E2E delay of a given PDU session does not violate its service latency requirement by mapping the constituent VNFs and data path to candidate locations and routing paths capable of fulfilling it.

$$2 \cdot (\sum_{f \in F_s^+} \sum_{t \in T} \sum_{n \in N_c} d_t \cdot Q_s^{f,b} \cdot T_s^{f,t} + \sum_{f,g \in F_s^+} \sum_{p \in P} d_p \cdot O_s^{f,g,b} \cdot y_p^{f,g,s}) + d_{DN} \leq L_s \quad \forall b \in B_s, \forall s \in S \tag{6.34}$$

Capacity constraints: Expressions (6.35), (6.36), and (6.37) refer to resource limitations in the edge servers, VNF instances, and physical links, respectively. Expressly, inequality (6.35) guarantees that the VNF instances deployed in a candidate location do not exceed the server's resources (e.g., CPU and memory). Similarly, inequalities (6.36) and (6.37) enforce the mapping of SFCRs to the VNF instances and links with enough processing capacity and bandwidth, respectively, to serve their demands.

$$\sum_{i \in I_t} \sum_{t \in T} C_t \cdot x_{i,t,n} \leq C_n \qquad \forall n \in N_c \tag{6.35}$$

$$\sum_{f \in F_s} \sum_{s \in S} C_s \cdot z_{i,t,n}^{f,s} \leq C_t \qquad \forall i \in I_t, \forall t \in T, \forall n \in N_c \tag{6.36}$$

$$\sum_{f,g \in F_s^+} \sum_{s \in S} \sum_{p \in P} \beta_s \cdot y_p^{f,g,s} \cdot H_{u,v}^p \leq \beta_{u,v} \qquad \forall (u,v) \in E \tag{6.37}$$

The binary nature of the variables is indicated as follows:

$$w_n, r_s, v_{i,t}, x_{i,t,n}, m_{i,t,n',n}, z_{i,t,n}^{f,s}, a_n^{f,s}, y_p^{f,g,s}, \delta_{n,m}^{f,g,s}, \xi_t \in \{0,1\} \qquad \forall f,g \in F_s^+, \forall b \in B_s, \forall s \in S, \forall i \in I_t, \forall t \in T, \forall p \in P, \forall n \in N \tag{6.38}$$

The linear form of the O-UPCR model can be summarized as follows:

Min:
$$\alpha_1 \cdot \sum_{n \in N_c} \Psi_a^n \cdot w_n + \alpha_2 \cdot \sum_{i \in I_t} \sum_{t \in T} \Psi_d^t \cdot v_{i,t} + \alpha_3 \cdot \sum_{i \in I_t} \sum_{t \in T} \sum_{n \in N_c} \Psi_r^{t,n} \cdot x_{i,t,n} +$$
$$\alpha_4 \cdot \sum_{(i,t) \in F} \sum_{n' \in N_c} \sum_{n \in N_c} \Psi_m^{t,n',n} \cdot m_{n',n}^{i,t} + \alpha_5 \cdot \sum_{f,g \in F_s^+} \sum_{s \in S} \sum_{p \in P} \Psi^{s,p} \cdot d_p \cdot y_p^{f,g,s} + \alpha_6 \cdot \sum_{s \in S} \Psi^s \cdot r_s$$

s. t.:

$$\sum_{i \in I_t} \sum_{n \in N_c} x_{i,t,n} \leq I_t \qquad \forall t \in T$$

$$\sum_{n \in N_c} x_{i,t,n} \leq 1 \qquad \forall i \in I_t, \forall t \in T$$

$$x_{i,t,n} \leq w_n \cdot V_n^t \qquad \forall i \in I_t, \forall t \in T, \forall n \in N_c$$

$$w_n \leq \sum_{i \in I_t} \sum_{\forall t \in T} x_{i,t,n} \qquad \forall n \in N_c$$

$$z_{i,t,n}^{f,s} \leq x_{i,t,n} \cdot T_s^{f,t} \qquad \forall f \in F_s, \forall s \in S, \forall i \in I_t, \forall t \in T, \forall n \in N_c$$

$$\sum_{i \in I_t} \sum_{t \in T} \sum_{n \in N} z_{i,t,n}^{f,s} = 1 \qquad \forall f \in F_s, \forall s \in S$$

$$x_{i,t,n} \leq \sum_{s \in S} \sum_{f \in F_s} z_{i,t,n}^{f,s} \qquad \forall i \in I_t, \forall t \in T, \forall n \in N_c$$

$$a_n^{f,s} \leq \sum_{t \in T} \sum_{i \in I_t} z_{i,t,n}^{f,s} \qquad \forall f \in F_s, \forall s \in S, \forall n \in N_c$$

$$a_n^{f,s} \geq z_{i,t,n}^{f,s} \qquad \forall f \in F_s, \forall s \in S, \forall i \in I_t, \forall t \in T, \forall n \in N_c$$

$$\sum_{f \in F_s} \sum_{i \in I_t} z_{i,t,n}^{f,s} \leq 1 \qquad \forall s \in S, \forall t \in T, \forall n \in N_c$$

$$\sum_{f \in F_s} \sum_{i \in I_t} z_{1,i,n}^{f,s} + \sum_{f \in F_s} \sum_{i \in I_t} z_{3,i,n}^{f,s} \leq 1 \qquad \forall s \in S, \forall n \in N_c$$

$$\sum_{p \in P} y_p^{f,g,s} \geq O_s^{f,g,b} \qquad \forall f,g \in F_s^+, \forall b \in B_s, \forall s \in S$$

$$\sum_{p \in P} y_p^{f,g,s} \leq 1 \qquad \forall f,g \in F_s^+, \forall s \in S$$

$$\sum_{p \in P_{n,m}} y_p^{f,g,s} \leq a_n^{f,s} \cdot a_m^{s,g} \qquad \forall f,g \in F_s, \forall s \in S, \forall n,m \in N_c$$

$$\delta_{n,m}^{f,g,s} \leq a_n^{f,s} \qquad \forall f,g \in F_s, \forall s \in S, \forall n,m \in N_c$$

$$\delta_{n,m}^{f,g,s} \leq a_m^{s,g} \qquad \forall f,g \in F_s, \forall s \in S, \forall n,m \in N_c$$

$$\delta_{n,m}^{f,g,s} \geq a_n^{f,s} + a_m^{s,g} - 1 \qquad \forall f,g \in F_s, \forall s \in S, \forall n,m \in N_c$$

$$\sum_{p \in P_{n,m}} y_p^{f,g,s} \leq \delta_{n,m}^{f,g,s} \qquad \forall f,g \in F_s, \forall s \in S, \forall n,m \in N_c$$

$$m_{n',n}^{i,t} = x_{i,t,n} \cdot \bar{X}_{i,t,n'} \qquad \forall (i,t) \in F, \forall n,n' \in N_c; n \neq n'$$

$$v_{i,t} \geq \sum_{n \in N_c} x_{i,t,n} - \sum_{n \in N_c} \bar{X}_{i,t,n} \qquad \forall i \in I_t, \forall t \in T$$

$$2v_{i,t} \leq 1 + \sum_{n \in N_c} x_{i,t,n} - \sum_{n \in N_c} \bar{X}_{i,t,n} \qquad \forall i \in I_t, \forall t \in T$$

$$\sum_{i \in I_t} v_{i,t} = \xi_t \cdot (\sum_{n \in N_c} \sum_{i \in I_t} x_{i,t,n} - \sum_{n \in N_c} \sum_{i \in I_t} \bar{X}_{i,t,n}) \qquad \forall t \in T$$

$$\xi_t \geq (\sum_{n \in N_c} \sum_{i \in I_t} x_{i,t,n} - \sum_{n \in N_c} \sum_{i \in I_t} \bar{X}_{i,t,n})/I_t \qquad \forall t \in T$$

$$\xi_t \leq (I_t + \sum_{n \in N_c} \sum_{i \in I_t} x_{i,t,n} - \sum_{n \in N_c} \sum_{i \in I_t} \bar{X}_{i,t,n})/(I_t + 1) \qquad \forall t \in T$$

$$\sum_{f \in F_s} \sum_{n \in N_c} a_n^{f,s} \cdot \bar{A}_n^{f,s} \geq |F_s|(1 - r_s) \qquad \forall s \in S$$

$$\sum_{f \in F_s} \sum_{n \in N_c} a_n^{f,s} \cdot \bar{A}_n^{f,s} \leq |F_s| - r_s \qquad \forall s \in S$$

$$2 \cdot (\sum_{f \in F_s^+} \sum_{t \in T} \sum_{n \in N_c} d_t \cdot Q_s^{f,b} \cdot T_s^{f,t} + \sum_{f,g \in F_s^+} \sum_{p \in P} d_p \cdot O_s^{f,g,b} \cdot y_p^{f,g,s}) + d_{DN} \leq L_s \qquad \forall b \in B_s, \forall s \in S$$

$$\sum_{i \in I_t} \sum_{t \in T} C_t \cdot x_{i,t,n} \leq C_n \qquad \forall n \in N_c$$

$$\sum_{f \in F_s} \sum_{s \in S} C_s \cdot z_{i,t,n}^{f,s} \leq C_t \qquad \forall i \in I_t, \forall t \in T, \forall n \in N_c$$

$$\sum_{f,g \in F_s^+} \sum_{s \in S} \sum_{p \in P} \beta_s \cdot y_p^{f,g,s} \cdot H_{u,v}^p \leq \beta_{u,v} \qquad \forall (u,v) \in E$$

$$w_n, r_s, v_{i,t}, x_{i,t,n}, m_{i,t,n',n}, z_{i,t,n}^{f,s}, a_n^{f,s}, y_p^{f,g,s}, \delta_{n,m}^{f,g,s}, \xi_t \in \{0,1\} \qquad \forall f,g \in F_s^+, \forall b \in B_s, \forall s \in S, \forall i \in I_t, \forall t \in T, \forall p \in P, \forall n \in N$$

6.1.3 Heuristic: Dynamic Priority and Cautious UPCR

The pseudo-code of the proposed heuristic, called DPC-UPCR, is provided in Algorithm 3. This algorithm has four main phases. First, data related to the current placement setup and SFC mapping are gathered (line: 2). For instance, the algorithm collects information about the location of each VNF, its available processing capacity, and resource utilization in the underlying infrastructure (i.e., servers and links). These data help determine the SFCR mapping and placement locations that imply fewer transformations and have the lowest impact on the overall reconfiguration cost. Next, the algorithm selects the set of sessions to be remapped (S_{remap}) during the placement reconfiguration (line: 4). This set can encompass all active PDU sessions in the system or a subset of them, as indicated by the parameter P_r. In the latter case, it prioritizes the sessions with worse QoS, those with latency violations, or those close to exceeding their service latency requirement. Afterward, the algorithm proceeds to release resources assigned to the selected sessions (line: 5). As part of this step, SFCRs assigned to underutilized VNF instances can also be remapped, implying an update of the set of selected sessions (line: 6). Subsequently, VNF instances can be removed, and active servers can be closed.

Second, the algorithm determines the set of possible locations for each unmapped SFCR. It starts by choosing the set of available servers to be used during the SFC remapping procedures

Algorithm 3: DPC-UPCR

Input: Percentage of additional SFCs to remap (P_r), Improvement attempts (F_i)

```
// Phase 1: Prepare variables and indicators
```

1 Initialize output variables and parameters
2 Gather information about previous placement and mapping configuration
3 **if** $P_r != 100$ **then**
4 $S_{remap} \leftarrow$ Select sessions ($s \in S$) to be remapped based on QoS and P_r
5 Release resources assigned to selected sessions
6 Update S_{remap} if required
7 **else** $S_{remap} \leftarrow$ S

```
// Phase 2: Find possible candidates and classify SFCRs
```

8 Select candidates with available capacity
9 **forall** $s \in S_{remap}$ **do**
10 Determine possible candidates
11 Classify SFCR s as critical or not according to its number of available candidates

```
// Phase 3: Map SFCRs
```

12 Sort $s \in S_{remap}$ according to established criteria ($sort_c$)
13 **while** $S_{remap} \neq \emptyset$ **do**
14 $s \leftarrow S_{remap}[0]$
15 SFCR-remapping procedure(s)
16 $S_{remap} \leftarrow S_{remap} - s$
17 **if** *mapping_success* **then**
18 $S_{map} \leftarrow S_{map} \cup s$
19 Update network and infrastructure resources
20 **if** *available servers changed* **then**
21 Update available candidates and determine criticism level for each SFCR
22 Sort selected SFCRs according to $sort_c$
23 **else** $S_{unmap} \leftarrow S_{unmap} \cup s$

```
// Phase 4: Improve solution and apply configuration
```

24 **if** $S_{unmap} = \emptyset$ **then**
25 Determine the cause of VNF temporal IDs if any
26 $cost_{best} \leftarrow$ Compute solution's cost
27 **while** $F_i \neq 0$ **do**
28 Make a copy of the current best solution
29 $sort_c, f_{remap} \leftarrow$Randomly select a sorting criterion and a VNF
30 $S_{remap} \leftarrow$ Sessions assigned to f_{remap}
31 Release resources assigned to selected sessions
32 Repeat steps 8–23
33 **if** $S_{unmap} = \emptyset$ **then**
34 Determine cause of temporal VNF IDs
35 $cost \leftarrow$ Compute solution's cost
36 **if** $cost < cost_{best}$ **then**
37 $cost_{best} \leftarrow$ cost
38 Update best solution
39 $F_i \leftarrow F_i - 1$
40 Update VNF temporal IDs if any
41 Apply mapping and placement configuration

(line: 8). A server is available when it disposes of enough resources to instantiate a new VNF or when its deployed VNFs can serve at least one unmapped SFCR. Once this step has been executed, the algorithm finds potential candidates for each SFC according to its service latency

demand (lines: 9–11). This information helps determine whether an SFCR is in a critical stage. An SFCR is considered critical if its number of possible candidates is lower or equal to the minimum number required to map its constituent VNFs.

Third, the algorithm maps the SFCRs. To this aim, the selected SFCRs (S_{remap}) are sorted according to the established criteria (line: 12). Specifically, the SFCR mapping order is decided according to their criticism level (critical or not), latency requirement, SFC length, access node location, and the number of potential candidates. This combination of parameters increases the possibility of successfully mapping the most demanding and critical sessions. After the mapping order of the specified sessions is established, the algorithm begins looping over them by choosing the most demanding one (line: 14).

The SFCR-remapping method determines the best mapping combination for every unmapped PDU session (line: 15). This method is a variant of the SFCR-mapping procedure presented in Section 5.3, with some minor modifications. The main difference between the procedures lies in the function used to evaluate the mapping cost of a VNF instance. Specifically, the SFCR-remapping procedure computes the mapping cost according to expression (6.7), whereas SFCR-mapping utilizes (5.4). Moreover, the remapping procedure does not consider the candidate's popularity when deploying a new VNF.

The SFCR-remapping procedure determines all possible candidates for each VNF service that forms the selected SFC. This step is required given that the latency budget and source nodes change when previous VNFs in the chain are mapped. Then, these candidates are classified as feasible or infeasible locations according to whether they satisfy the UPCR constraints (e.g., latency and anti-affinity). Additionally, the algorithm obtains the shortest virtual path with enough bandwidth to support the requested traffic flow and estimates the VNF remapping cost for each feasible candidate. Given that mapping a VNF instance to a different server may incur additional costs in terms of VNF migrations and SFC relocations; this procedure endeavors to maintain a similar configuration to the previous one as long as possible.

When selecting a server implies the relocation of a VNF service request, the procedure first checks if this node has available instances of the same type from previous placement events. If so, one of these instances is designated to provide the VNF service. Otherwise, a VNF migration or a new VNF deployment is required. Like the ILP model, the DPC-UPCR algorithm considers a new deployment when the current number of instances of a given type is higher than it was before the reconfiguration event. Otherwise, the algorithm assumes the migration of a VNF instance. Since several SFCs share the same VNF instance, and this decision is based on the mapping of the PDU session under analysis, further analysis is required to determine the cause of the change. The algorithm assigns a temporal ID to the instances, which helps determine the real cause of the VNF deployment (i.e., new instantiation or migration) in the selected candidate.

At the end of the evaluation process, the algorithm maps a VNF instance to the candidate location, implying the lowest reconfiguration cost. This process continues until all VNFs in the

SCF have been evaluated or no feasible candidate is left. Once the SFCR-remapping procedure has finished, the algorithm removes the selected session from the S_{remap} set, and the session is classified as mapped or rejected (lines: 16–23). Successfully mapping an SFC requires updating the network and the infrastructure's available resources (lines: 17–22). Moreover, the algorithm updates the subset of available candidates and verifies the criticism level for each SFCR when the number of available candidates varies. A change in the number of available candidates implies that the algorithm reorders the remaining SFCRs (line: 22). When no feasible candidate is found for mapping a VNF service, the SFCR-remapping procedure is interrupted, and the SFCR is classified as unmapped (line: 23).

The last phase of Algorithm 3 is optional, and its execution depends on the input parameter F_i (lines: 27–39). This phase's main objective is enhancing the quality of the generated solution by introducing some modifications. It first determines the cost of the current placement configuration and establishes this cost as the best one. Then, it performs F_i improvement attempts (line: 29). Each improvement attempt begins with the random selection of a sorting criterion for the SFCRs' mapping, as well as a VNF instance from the set of VNFs ($f_{remap} \in F$) (line: 29). Our algorithm offers three sorting strategies. One is the strategy previously mentioned in phase 3, which is used to generate the initial reconfiguration solution. Another is a variant of the above-mentioned sorting criterion in which the order of the number of candidates and access node location parameters are exchanged. The other strategy only considers service latency requirements to determine the mapping order. Furthermore, different approaches can be defined as desired by the service provider.

Regarding VNF selection (f_{remap}), the only condition is that the VNF be different from the one previously analyzed. These two approaches promote diversification in the generated solutions, thus reducing the chances of obtaining consecutive improvement attempts with similar outcomes. Next, the algorithm proceeds to release the resources assigned to the sessions mapped to the selected VNF (S_{remap}) and generates a new solution by remapping them. Similar to the initial solution, it investigates the cause of temporal IDs for the feasible generated solutions (i.e., $S_{unmap} = \emptyset$) before determining the solution cost. Then, it compares the obtained cost with the best one found thus far, and an update process occurs when a better solution is detected (lines: 33–38). At the end of this phase, the algorithm removes any temporal VNF IDs and applies the best placement and chaining configuration.

6.1.3.1 Complexity Analysis

In this subsection, we analyze the time complexity of the DPC-UPCR solution in depth. For each reconfiguration event, the main processes involved are variables and indicators preparation, candidates determination and classification of SFCRs, SFCR mapping, and solution improvement.

The time complexity of the first phase of Algorithm 3 depends on the selection of partial or full unmapping of the SFCR set. When no partial unmapping is selected, the complexity of this process

is $\mathcal{O}(1)$. By contrast, when $P_r \neq 100$, the complexity increases due to the execution of lines 4–6. Concretely, the selection of S_{remap} has complexity $S \cdot \log_2 S + S^{'}$ where S_{remap} denotes the subset of sessions with latency violation along with the additional percentage obtained from P_r. In the worst-case scenario, $S_{remap} \approx S$. However, the main goal of selecting a partial unmapping approach is to reduce the number of sessions under analysis during the reconfiguration procedure; thus, typically, $S_{remap} < S$. In addition, for each $s \in S_{remap}$, the resources assigned to each constituent VNF and link (F_s and P_s) are released. Thus, this step introduces $S_{remap} \cdot (F_s + P_s)$ complexity. Finally, the update of S_{remap} due to the existence of low-loaded instances requires iteration over each deployed VNF instance ($f \in F$) as well as the release of resources used by their assigned SFCRs. Hence, the time complexity of this phase, in the worst-case scenario, can be summarized as $\mathcal{O}(|S| \cdot \log_2 |S| + |S_{remap}| \cdot (|F_s| + |P_s|) + |F|)$.

The second stage determines the available candidates and classifies SFCRs according to their criticism level. In this vein, it first requires the analysis of the capacities available in each candidate server ($|N_c|$). Moreover, in the case of no available capacity to instantiate a new VNF instance, it proceeds to check if any of its instantiated VNFs has some capacity to serve any SFCRs. The complexity of this step is indicated as C. It must be noted that C cannot be determined beforehand since it depends on several factors, such as the maximum capacity of a VNF, SFCR processing demands, and the maximum number of instances that it can host. Therefore, the complexity of this step is formulated based on C as $\mathcal{O}(|N_c| \cdot |C|)$. Next, all the SFCRs are analyzed to determine their possible candidates and criticism levels. In the worst case, assuming that the entire set of SFCRs is analyzed and that all the candidates are available, it requires $|S| \cdot |N_c|$ iterations. Thereby, the second phase of Algorithm 3 can be performed in time complexity ($|N_c| \cdot (|S| + |C|)$).

The third phase begins by sorting the SFCRs according to the established criterion. In the worst case, this can be done in time complexity $|k| \cdot |S| \cdot log_2 |S|$ where K indicates the number of parameters in the sorting criterion and S represents the complete set of SFCRs (i.e., $S_{remap} = S$). Afterward, an iterative process begins to remap every VNF and link, forming each SFCR. The SFCR-remapping procedure conditions the complexity of this while loop. From Subsection 5.3.3, it was shown that the SFCR-mapping approach follows the order of $\mathcal{O}(|N_c|^2 \cdot |M| \cdot |F_s| \cdot |B_s|)$ where $|S|$, $|N_c|$, $|F_s|$, and $|B_s|$ indicates the sizes of the sets of SFCRs, available candidates, VNFs and branches forming a service chain, respectively. In addition, M represents the runtime complexity associated with the candidate evaluation process. Similar to C, M cannot be determined beforehand, as it depends on several factors, such as servers and VNF capacities, UPF-specific constraints, and available paths. Hence, the complexity of this phase is formulated as a function of M. Thereby, the total complexity of this phase is $|S| \cdot (|k| \cdot log_2 |S| + |N_c|^2 \cdot |M| \cdot |F_s| \cdot |B_s|)$ which is the level of $\mathcal{O}(|S| \cdot |N_c|^2 \cdot |M| \cdot |F_s| \cdot |B_s|)$.

Finally, the improvement procedure determines the complexity of the last part of Algorithm 3. Each improvement attempt implies the analysis of the sessions assigned to the randomly selected

VNF instance (S_{remap}). For this set of sessions, its resources must be released, and phases 2 and 3 must be executed. It should be noted that the number of sessions assigned to a VNF instance is typically much lower than the size of the overall set of SFCRs (i.e., $|S_{remap}| << |S|$). Given that this process performs F_i number of improvement attempts, the complexity of this phase, in the worst scenario, becomes $\mathcal{O}(|F_i| \cdot |S| \cdot |N_c|^2 \cdot |M| \cdot |F_s| \cdot |B_s|)$. In contrast, when no improvement attempt is considered (i.e., $F_i = 0$), this phase's complexity is as simple as $\mathcal{O}(|F_{temp}|)$, where $|F_{temp}|$ indicates the size of the VNF set with temporal IDs. In the latter case, the complexity of this algorithm stage is negligible compared to phase three.

Based on the previous analysis, the time complexity of the proposed heuristic is mainly determined by the third and last phases of Algorithm 3, as these phases involve more iterative processes than the first two. Thus, the maximum run time of DPC-UPCR depends on the number of iterations over phase three ($F_i + 1$). In this regard, the overall complexity of the proposed heuristic can be formulated as $\mathcal{O}(|F_i| \cdot |S| \cdot |N_c|^2 \cdot |M| \cdot |F_s| \cdot |B_s|)$. However, the number of improvement attempts can be omitted in large-scale scenarios where the numbers of candidate locations (N_c) and PDU sessions are elevated (S). Concretely, the F_i parameter does not contribute as much to the problem size as these other parameters (i.e., $F_i << N_c$ and $F_i << S$). Similarly, the numbers of VNFs and branches ($|F_s|$ and $|B_s|$) forming the SFC topologies are, generally, significantly smaller parameters than S and N_c. Thus, the overall complexity of Algorithm 3 can be reduced to $\mathcal{O}(|S| \cdot |N_c|^2 \cdot |M|)$. From this expression, we note that the complexity of the DPC-UPCR is strongly dependent on the size of the considered sets for SFCRs (S) and candidate locations (N_c). For this reason, the envisioned DPC-UPCR heuristic is a polynomial-time algorithm.

6.2 Dynamic Scheduling for the UPCR

This section introduces the OSR scheduler mechanism to execute UPF placement and chaining readjustments dynamically according to desired QoS levels.

6.2.1 Problem Statement

Like in the SSR solution presented in Chapter 4, the OSR deals with the problem of determining the optimal time to initiate a UPC reconfiguration procedure so that adverse effects of reevaluation events are minimized, and QoS is kept under acceptable values.

Given a UPC arrangement obtained as a result of an initial deployment or a reconfiguration event, in which all PDU sessions were mapped according to their service requirements (e.g., latency), we have to consider some degradation in the QoS over time (refers to Section 4.2.1 for more details). Moreover, we assume that at each time instant t, an agent measures the offered QoS, which has been defined in terms of the number of sessions with latency violations ($L_t = \sum_{s \in N_s} I_t^s$, where $I_t^s = 1$ if the service delay of PDU session $s \in S$ exceeds its service requirement). This

metric is desired to be as small as possible, being the QoS optimal when the latency requirement of all the sessions is satisfied (i.e., $L_t = 0$).

We also assume that at each time interval, the system can tolerate a maximum number of sessions with latency violations, denoted as Θ, where $\Theta > 0$, without needing to activate a reconfiguration event. Namely, the offered QoS is considered acceptable, and placement and chaining reconfiguration is not required as long as the value of the selected QoS metric is below the established threshold. Conversely, the QoS is considered degraded, and the UPC configuration must be readjusted when the Θ threshold is overpassed. Thus, the reevaluation time must be selected so that the value of the L_t metric is as close to the pre-established upper bound as possible without exceeding it. In this way, UPCRs can be delayed or even avoided while the QoS is maintained at acceptable levels. In other words, we have to determine when the system is about to exceed the established threshold to activate a reconfiguration event in advance and avoid QoS deterioration.

To facilitate the decision process, we present the cost function (6.39). This expression is defined as a function of the number of sessions with latency violations and the maximum QoS tolerance threshold Θ. If the number of sessions with latency violations is below the established threshold, no readjustment is needed, and the service provider avoids paying its users an amount of money proportional to the expected number of users with good QoS that will be affected due to the reconfiguration event ($\Psi \cdot (\mathbb{E}[S_r] - L_t)$). In contrast, when the Θ threshold is exceeded, the placement and chaining configuration must be readjusted, thereby incurring an expected reconfiguration cost $\mathbb{E}[C_{rec}]$.

$$Y_t(L_t) = \begin{cases} \Psi \cdot (\mathbb{E}[S_r] - L_t) & \text{if } L_t \leq \Theta \\ \lambda \cdot \mathbb{E}[C_{rec}] & \text{if } L_t > \Theta \end{cases} \tag{6.39}$$

where the Ψ is a cost component and λ is a weight factor to adjust the importance of the reconfiguration cost.

Our objective is to determine the time instance t when to stop observing the QoS parameter (L_t) and proceed to readjust the UPF placement and chaining configuration. Specifically, we aim to determine the optimal stopping rule that minimizes the expected loss function (6.39).

Problem 2. *Given a sequence of observations defined by L_t, an upper bound Θ of the accepted tolerance on the QoS and an expected reconfiguration cost $\mathbb{E}[C_{rec}]$, determine the optimal decision epoch t^*, which minimizes the cost function Y_t:*

$$\inf_{t \geq 0} \mathbb{E}[Y_t(L_t)] \tag{6.40}$$

6.2.2 Optimal Scheduling of the UPCR

In this subsection, we derive an optimal stopping rule to determine the optimal time at which to stop observing the L_t parameter and activate the UPCR. This stopping rule is based on the

1-SLA rule, which for stopping problems aims to minimize an expected loss, as follows:

$$t^* = \inf\{t \geq 0 : Y_t \leq \mathbb{E}[Y_{t+1}|\mathbb{F}_t]\} \tag{6.41}$$

where $\mathbb{F}_t$ is the σ-fields generated by the observations $L_1, L_2, \ldots, L_t$. Specifically, it represents the knowledge of the random variable L_t up to time t.

Theorem 4. *Given an upper bound (Θ) upon which the system QoS is considered degraded and a sequence of observations regarding the number of sessions with latency violations $L_1, \ldots, L_t$ with reference to the last optimal UPF placement and chaining configuration (i.e., $L_0 = 0$), the optimal stopping time (t^*) for the problem stated in* (6.40) *is:*

$$t^* = inf\{t \geq 0 : \Psi \cdot (\mathbb{E}[S_r] - L_t) \leq \lambda \cdot \mathbb{E}[C_{rec}] + (\Psi \cdot \mathbb{E}[S_r] - \lambda \cdot \mathbb{E}[C_{rec}]) \cdot \sum_{l=0}^{\Theta} P(L = l)\} - \Psi \cdot \sum_{l=0}^{\Theta} l \cdot P(L = l) \tag{6.42}$$

Proof. Given a time interval t where $L_t \leq \Theta$, the conditional expectation of Y_{t+1} at the next stage is given by:

$$\begin{aligned}
\mathbb{E}[Y_{t+1}|L_t \leq \Theta] &= \mathbb{E}[\Psi \cdot (\mathbb{E}[S_r] - L_{t+1})|L_t \leq \Theta, L_{t+1} \leq \Theta] \cdot P(L_{t+1} \leq \Theta) + \mathbb{E}[\lambda \cdot \mathbb{E}[C_{rec}]|L_t \leq \Theta, \\
&\quad L_{t+1} > \Theta] \cdot P(L_{t+1} > \Theta) \\
&= \mathbb{E}[\Psi \cdot (\mathbb{E}[S_r] - L_{t+1})|L_{t+1} \leq \Theta] \cdot P(L_{t+1} \leq \Theta) + \mathbb{E}[\lambda \cdot \mathbb{E}[C_{rec}]|L_{t+1} > \Theta] \cdot (1 - P(L_{t+1} \leq \Theta)) \\
&= \Psi \cdot (\mathbb{E}[S_r] - \mathbb{E}[L_{t+1}|L_{t+1} \leq \Theta]) \cdot P(L_{t+1} \leq \Theta) + \lambda \cdot \mathbb{E}[C_{rec}] \cdot (1 - P(L_{t+1} \leq \Theta)) \\
&= (\Psi \cdot \mathbb{E}[S_r] - \lambda \cdot \mathbb{E}[C_{rec}]) \cdot \sum_{l=0}^{\Theta} P(L = l) + \lambda \cdot \mathbb{E}[C_{rec}] - \Psi \cdot \sum_{l=0}^{\Theta} l \cdot P(L = l)
\end{aligned}$$

Thus, by comparing the current cost, $Y_t(L_t) = \Psi \cdot (\mathbb{E}[S_r] - L_t)$, with the one expected at the next time interval, we find that the UPC setup must be readjusted at the first time instance t such that $\Psi(\mathbb{E}[S_r] - L_t) \leq \mathbb{E}[Y_{t+1}|L_t \leq \Theta]$. In other words, stopping is optimal when the current loss is equal to or less than the expected cost at the next stage. ■

Based on Theorem 1, presented in Section 4.2.2, the stopping rule provided in (6.42) must be monotone to guarantee the optimality of the 1-SLA rule.

Theorem 5. *The 1-SLA rule defined in* (6.42) *is optimal for the OSR problem and minimizes the expected loss defined in* (6.39)*.*

Proof. In order the problem be monotone, the difference $\mathbb{E}[Y_{t+1}|L_t \leq \Theta] - Y_t(L_t)$ must be non-decreasing with L_t. This condition is satisfied since the right side of expression (6.42) remains constant, and its left side is non-increasing over L_t as long as L_t is below the predefined QoS threshold ($L_t \leq \Theta$). Thus, the problem is monotone, and the 1-SLA rule provided in (6.42) is optimal. When $L_t > \Theta$, we must stop immediately and proceed to readjust the UPF placement and chaining configuration. ■

6.3 Evaluation and Results

This section describes the conducted experiments to investigate the performance of the proposed solutions and analyzes the obtained results. First, it introduces aspects of the simulation setup, such as network topology and parameters. Next, Subsections 6.3.2 and 6.3.3 investigate the performance of the designed methods for the UPC reconfiguration. Finally, Subsection 6.3.5 evaluates the effectiveness of the OSR mechanism.

6.3.1 Simulation Setup

Figure 6.1 provides an overview of the considered network topology. It represents a 5G medium-scale scenario comprising 121 ANNs and 13 APs and edge servers. The gNBs connected through the APs and had an inter-site distance of 500 m and 200 m for gNBs located in urban and dense areas. Additionally, the ENs were co-located with the APs and had a service area with a maximum radius of 1 km. Some of these ENs had already deployed UPFs, for which the initial placement and chaining configuration was determined using the PC-UPC solution presented in Section 5.3. The PC-UPC algorithm was run by prioritizing the optimization of node activation and UPF deployment costs over the routing term (i.e., $\alpha = \beta = 0.4$, and $\gamma = 0.2$).

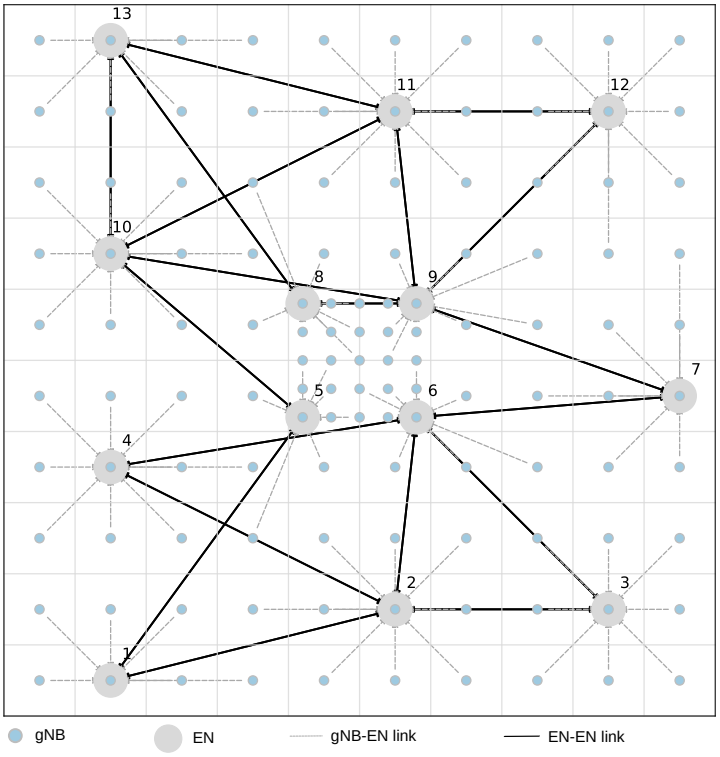


Figure 6.1: 5G access network topology in a MEC ecosystem.

For the service demand, we considered three types of SFCRs, formed by one to three UPFs. PDU session requirements, such as service latency, processing demand, and bandwidth, were

randomly generated based on the parameters specified in Table 6.4. Additionally, the number of connected users varied in the interval of [5–1000], and each one was assumed to have one active PDU session. To simulate user mobility, we selected the CityMob [149] mobility pattern generator with a Manhattan model ($m = 2$). Moreover, we developed a Python-based program to manage user location and assigned gNB and UPFs over the entire simulation time. Table 6.5 provides the simulation parameters utilized by the CityMob program.

Table 6.4: Simulation parameters for the UPC problem.

Notation	**Description**	**Value**
S	Number of PDU sessions (SFCRs)	[5–1000]
β_s	Bandwidth requirement (Mbps)	[1, 10, 50]
C_s	Processing demand (CPU)	[0.01, 0.05, 0.1]
L_s	Latency requirement (ms)	[0.95, 1]
N_r	No. of access nodes	121
N_c	No. of candidate locations (ENs)	13
E	Number of links	172
P	Number of shortest paths	1742
$\beta_{u,v}$	Bandwidth capacity of links (Gbps)	10
C_c	Resource capacity of a server (CPU)	40
C_t	Resource capacity of a VNF (CPU)	2
I_t	Maximum number of instances	+3[1]
d_r	RTT delay in the RAN (μs)	500
d_t	Processing time of UPFs (μs)	50
T_{ap}	Processing time of AP (μs)	5
d_{DN}	Processing time of DN (μs)	100
-	Prop. delay in optical links (μs/km)	5

[1] For each VNF type, the maximum number of instances was determined by adding three extra instances to the minimum number of VNFs required to meet SFCR's demands.

Table 6.5: Simulation parameters used in CityMob.

Notation	**Description**	**Value**
m	Mobility model	2[1]
n	Number of users	1000
t	Simulation time (s)	36000
s	Maximum speed of the users (m/s)	15
d	Distance between streets or block sizes (m)	100
w x d	Dimensions of the grid (km^2)	5x5
a	Number of accidents	0

[1] The m parameter takes numeric values to indicate the mobility model (e.g., $m = 2$ for the Manhattan model).

To evaluate the solutions, we considered two sets of weight factors. The first set (weight_set_1) assumes similar weights for all the terms in the objective function except for the reassignment cost, which is omitted. The second weight set (weight_set_2) considers that all cost components

need to be optimized and equally important.

6.3.2 DPC-UPCR Solution Performance

In the following section, we analyze the impacts of the partial SFC unmapping and improvement attempts on the performance of the proposed heuristic. In particular, we focus on the average reconfiguration cost and computation time metrics for different numbers of active PDU sessions (i.e., $S = 50$, $S = 100$, and $S = 200$). Every sample contains 10 reconfiguration events performed by a periodic scheduling mechanism with a reconfiguration period of 30 minutes.

6.3.3 DPC-UPCR Performance

6.3.3.1 Partial Unmapping

We performed these experiments by unmapping an extra percentage of SFCRs apart from those with latency violations at the reconfiguration moment, as indicated by the P_r parameter. This additional percentage of unmapped sections was formed by SFCs close to exceeding their service latency requirement. In addition, sessions assigned to UPF instances with low capacity utilization (i.e., below 20%) were also unmapped. Moreover, we did not consider any improvement phase for the heuristic solution for these experiments.

Figure 6.2 represents the average reconfiguration cost for various percentages of additional unmapped SFCs and both sets of weight factors. This figure shows how the behavior of the cost function changed notably with the considered cost components. More precisely, for weight_set_1, which is represented in light-blue, the overall reconfiguration cost decreased with the additional percentage of unmapped sessions. All examined sets of SFCRs obtained the highest costs when remapping only sessions with latency violations at the reconfiguration moment. In contrast, they had the lowest reconfiguration costs when reconfiguring the UPC with a full unmapping of the SFCR set ($P_r = 100$).

This outcome was expected since there were more combinations when all the sessions were unmapped. Therefore, the possibility of finding better UPF locations, mapping, and chaining was higher. The latter was sustained by the reassigned session behavior, which increased in value with the P_r parameter as indicated in Fig. 6.3(**a**). The relocation term was not reflected in the overall reconfiguration cost since this set of weights omitted it from the objective function. Moreover, greater values of P_r also allowed further improvements in the average system response time despite significant reductions in the total number of deployed UPFs, as illustrated in Figs. 6.3(**c**) and 6.3(**b**).

The behavior of the cost function was more complex for the second set of weights, which are represented in the figure in dark-blue, as it did not immediately seem to follow a clear trend related to P_r (see Fig. 6.2). In contrast to our initial assumptions, a partial unmapping resulted in lower reconfiguration costs than a full one in cases such as $S = 50$ and $S = 100$. This behavior was

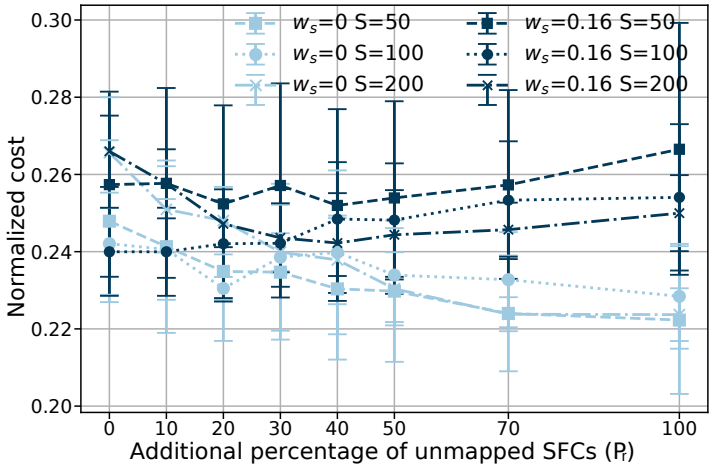
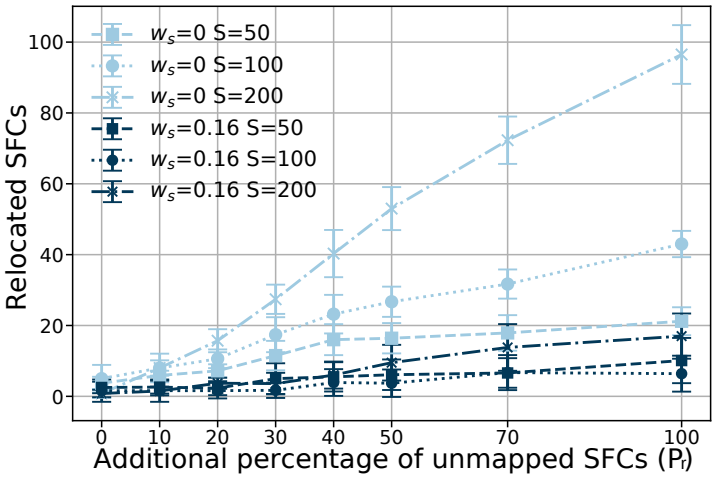
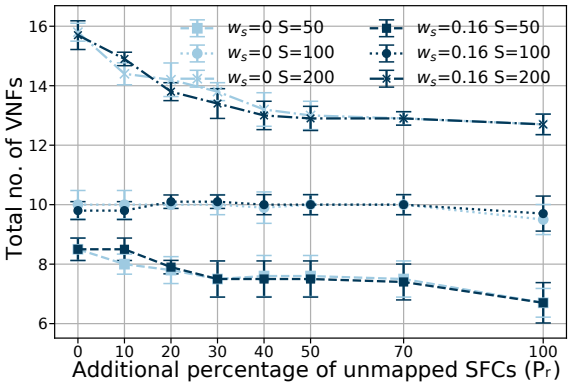


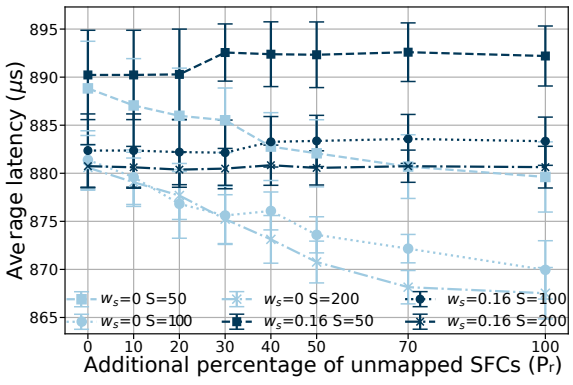
Figure 6.2: Average reconfiguration cost versus percentage of partially unmapped SFCs (P_r).



(a) Relocations



(b) Number of UPFs instances.



(c) Average E2E delay.

Figure 6.3: Performance of DPC-UPC solution in terms of relocated sessions, deployed VNF instances, and average latency for different percentages of partial unmapping.

conditioned by an increase in the number of relocated sessions with the percentage of unmapped sessions (see Fig. 6.3(**a**)). For the set formed by 200 PDU sessions, the cost function decreased for values of $P_r \leq 40\%$ and slightly increased for $P_r \geq 50\%$ due to higher variations in the number of reassigned sessions.

Nevertheless, unlike the other two SFCR sets, a complete unmapping required lower expenditures than when only SFCRs with poor QoS were reassigned ($P_r = 0$). By closely examining the relocation and cost functions, we noticed that both graphics had similar behavior when the number of sessions was small (i.e., $S = 50$ and $S = 100$). This is because the impact of session reassignments on the normalized cost function was more significant for small numbers of sessions. This can be better appreciated for $S = 100$, which experienced scarce improvements in the number of deployed instances.

As seen in Fig. 6.3(**c**), the second set of weights presented higher average latency values than the first set. In particular, for weight_set_2, the obtained delays rose with the P_r parameter, while the number of UPF instances usually decreased with P_r. Furthermore, unlike the results obtained for the other set of weights, considerably fewer sessions were relocated due to the consideration of this cost component in the objective function. Concerning the heuristic running time, Fig. 6.4 shows that for both sets of weights, this metric increased with the number of unmapped SFCRs (P_r), thereby requiring the highest execution time for complete UPF placement and chaining readjustment (i.e., $P_r = 100$).

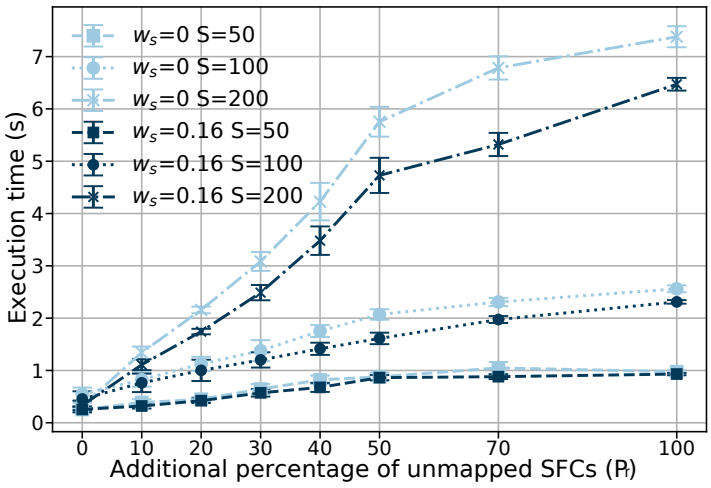


Figure 6.4: Average computing time of DPC-UPC heuristic versus different percentages of partially unmapped SFCs (P_r).

These results showed that partial rather than complete unmapping was a more appropriate option concerning the session reassignment cost and a small number of SFCRs. We obtained a relatively strong cost performance when considering partial unmapping of 30% to 50% of the sections. When omitting this cost component, a complete rather than partial unmapping of the SFCR set provided higher reductions in the reconfiguration cost. However, the complete

unmapping approach still required higher running times. Accordingly, a partial unmapping of the SFCR set may be adopted to reduce the computational time of the UPC reconfiguration events.

6.3.3.2 Improvement Phase

We assessed the impact of the improvement phase on the heuristic performance in terms of reconfiguration cost and execution time for different improvement attempts as well as a no-improvement phase (i.e., $F_i = 0$). Multiple samples (i.e., five) for the same F_i value were collected to reflect more stable behaviors. This was due to the random selection of the VNF instances and sorting criteria, which produce a different configuration at each iteration. The experiments were conducted by considering that all components in the objective function are equally important (i.e., weight_set_2).

Figure 6.5 represents the average reconfiguration costs for different SFCRs. Even in the simplest case (i.e., $F_i = 1$), the average reconfiguration cost was reduced with the introduction of the improvement phase. These reductions were more remarkable as the number of improvement attempts increased. For instance, the set formed by 50 PDU sessions experienced decrements between 1% and 6% for the studied values of the F_i parameter compared with the no improved version of the DPC-UPCR heuristic. For the other two SFCR sets, slighter reductions were achieved due to the high quality of their reconfiguration solutions. These reconfiguration solutions were close to the optimal ones, with an optimality gap of 5% (refer to Subsection 6.3.4 for more details).

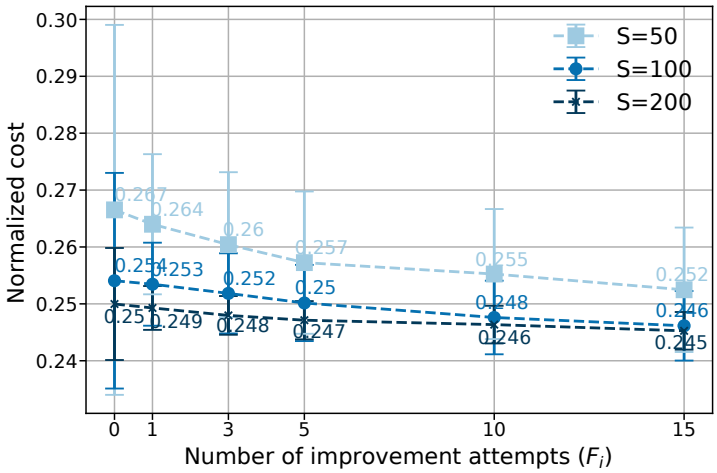


Figure 6.5: Average reconfiguration cost of DPC-UPC versus numbers of improvement attempts.

As shown in Fig. 6.6, these improvements in the solution cost came at the expense of higher execution times, which incremented with the rise of the F_i parameter. For instance, the execution times when selecting $F_i = 15$ were between four and seven times greater than the non-improved reconfiguration solutions ($F_i = 0$) for the selected SFC sets.

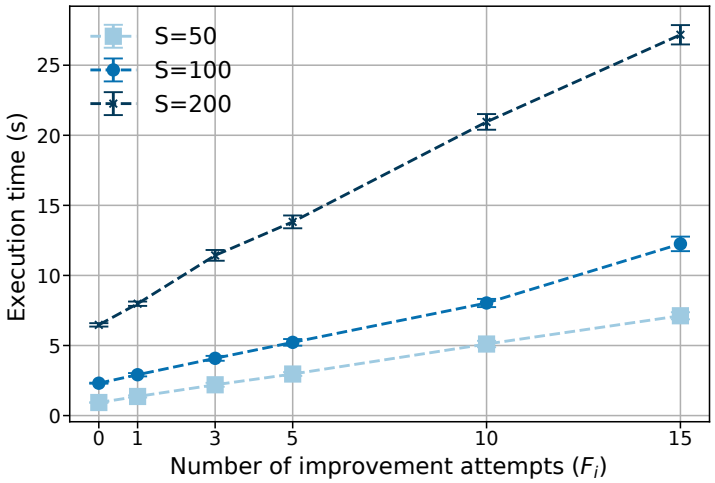


Figure 6.6: Average computing time of DPC-UPC versus numbers of improvement attempts.

6.3.4 UPCR Solutions Performance

In this subsection, we evaluate the performance of the presented solutions for the UPCR problem. Specifically, we consider two variants of the conceived DPC-UPCR heuristic. The first is a basic version with no improvement attempts, while the second applied 15 improvement attempts, which we refer to as BDPC-UPCR and IDPC-UPCR. Both solution approaches adopted a full unmapping of the PDU sessions. The optimal solutions (O-UPCR) were determined with a zero optimality gap.

First, we compared the efficiency of BDPC-UPCR with two baseline solutions referred to as Greedy- and Sorted Greedy-UPCR (i.e., G-UPCR and SG-UPCR). These solutions are greedy-based approaches that map each SFCR to the VNFs and candidates that further minimize the objective function presented in (6.7). The G-UPCR does not prioritize SFCR mapping, whereas the SG-UPCR obeys the same criteria as the designed heuristic. Moreover, they ignore the VNF mapping effects on the remaining VNFs that form the chain, which may cause the rejection of SFCRs. As discussed in Chapter 5, we extended both baselines to include a reassignment procedure to avoid rejections and ensure similar comparison conditions (refer to Subsection 5.4.2 for more details). Afterward, we analyzed the performance of BDPC-UPC and IDPC-UPC compared to the optimal approach.

The performance of all solutions was investigated in terms of overall normalized reconfiguration cost and average reconfiguration time for different demands of PDU sessions. Additionally, we examined the number of reassignment events when comparing the BDPC-UPCR with the aforementioned benchmarks. All cost components of the objective function were considered equally important (i.e., weight_set_2) for these experiments.

6.3.4.1 BDPC-UPCR versus Greedy Approaches

As seen in Fig. 6.7, the proposed heuristic, in its basic version (BDPC-UPC), always obtained the reconfiguration solution with the lowest cost. Its reductions in the reconfiguration cost were significant compared with the greedy-based baselines. More specifically, BDPC-UPC decreased the reconfiguration costs up to 24% and 21% concerning GH-UPCR and SGH-UPCR, respectively.

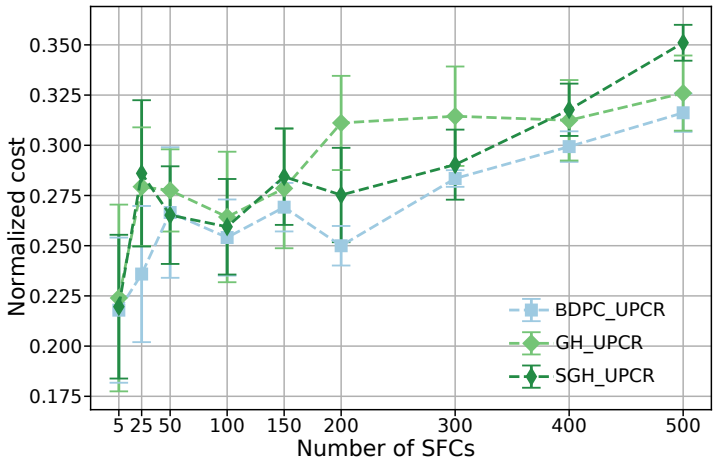
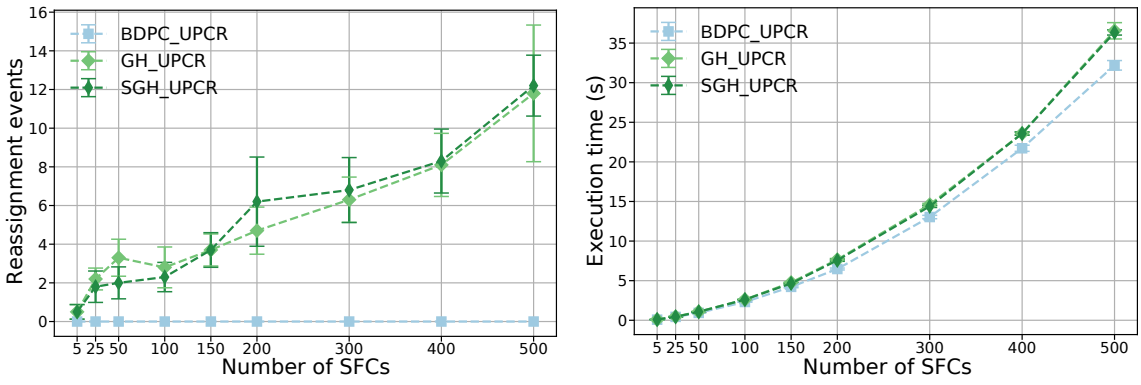


Figure 6.7: Reconfiguration cost of the BDPC-UPC heuristic versus greedy approaches.

Regarding SFCR reassignments, our solution approach remapped all examined SFCRs without executing the VNF reassignment procedure, see Fig. 6.8(**a**). Conversely, the benchmarks required the reassignment of some SFCRs, even in cases with small numbers of sessions. Generally, their number of reassignments increased with the size of the SFCR set. Additionally, Fig. 6.8(**b**) also shows that both benchmarks had similar computational times. Furthermore, their differences from the one provided by the BDPC-UPC approach were more distinctive as the size of the SFCR set increased. This was due to their execution of more reassignment events to avoid SFCR rejections. The results demonstrate that the proposed heuristic outperformed the baselines for all analyzed metrics.

6.3.4.2 UPCR Solutions

Figure 6.9 illustrates the average reconfiguration cost of the proposed solutions (i.e., O-UPCR, BDPC-UPCR, and IDPC-UPCR) for increasing demands of SFCRs. The performance of the BDPC-UPC approach was within 15% of the optimum, with an average optimality gap of 7.27%. IDPC-UPCR further narrowed this gap by incorporating the improvement phase and considering 15 improvement attempts (i.e., $F_i = 15$). This approach had an average optimality gap of 4.25% of the optimum, with a difference of 8.62% in the worst-case scenario (i.e., $S = 50$). Thus, it provided lower reconfiguration costs for values of $S \geq 100$ given the greater number of deployed VNFs and, consequently, the options for SFCR mapping. These outcomes indicated that the proposed



(a) Reassigned SFCRs. **(b)** Execution time.

Figure 6.8: Performance of the BDPC-UPC heuristic versus greedy-based approaches.

improvement method significantly enhanced the quality of the reconfiguration solutions with near-optimal results.

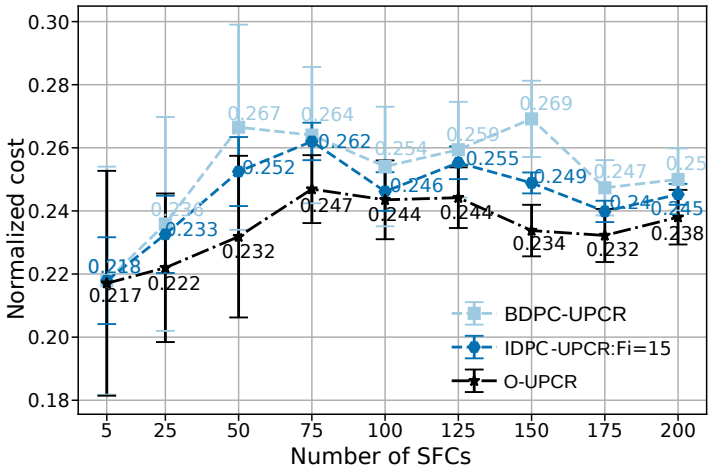


Figure 6.9: Average reconfiguration cost of the proposed solutions versus different numbers of SFCRs.

Figure 6.10 summarizes the average CPU execution time of the proposed solutions. As seen in the figure, the computational time increased with the number of PDU sessions for all solutions. The ILP model required the highest running time, whereas the basic heuristic solution was the fastest. Specifically, BDPC-UPC was between 4–9 and 10–22 times faster than its improved version and the mathematical model, respectively. Moreover, both heuristics scaled well with increasing numbers of SFCRs, while the exact solution's running time increased considerably until it was unable to solve the problem in a reasonable time for SFCRs greater than 200.

In terms of wall-clock time, the ILP model required several hours to conduct a reconfiguration event for instances of the problem with more than 150 SFCRs. In contrast, the heuristic approaches only needed a few seconds.

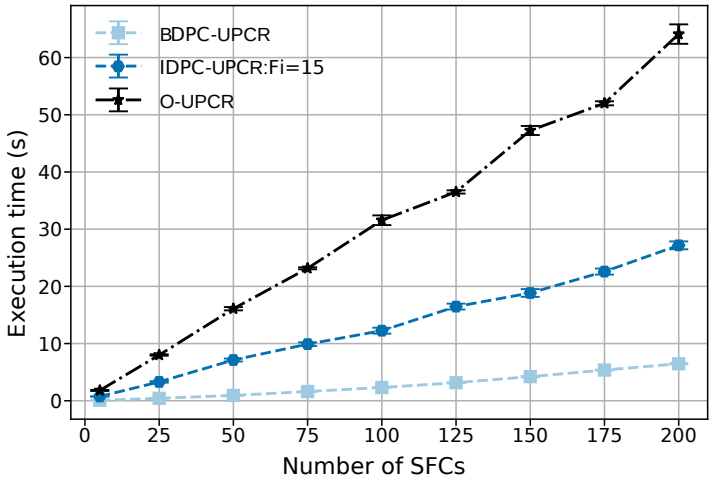


Figure 6.10: Average running time of the proposed solutions versus different numbers of SFCRs.

6.3.5 Dynamic Scheduling for the UPCR

In this subsection, we evaluate the effectiveness of the conceived scheduling mechanism by comparing it with the following benchmarks:

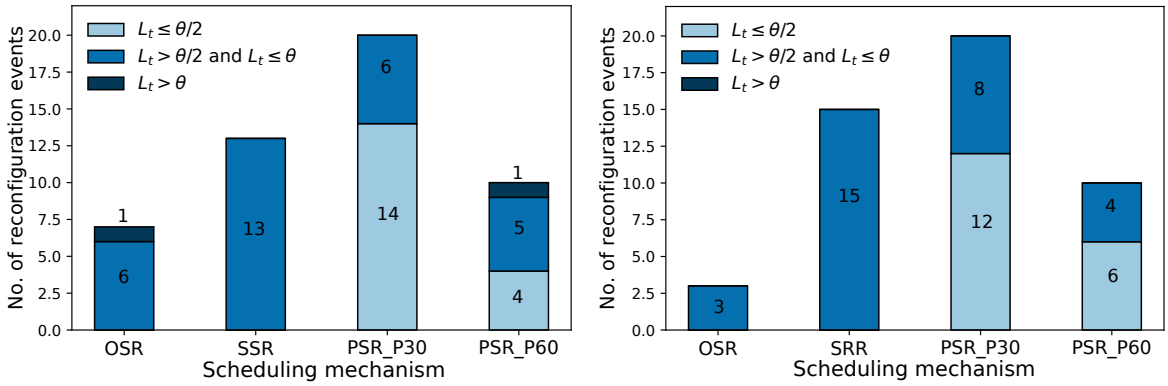
- *Periodic scheduling of the reconfiguration (PSR):* The UPC readjustment is executed periodically at fixed time intervals (i.e., every 30 and 60 minutes).
- *Skeptical scheduling of the reconfiguration (SSR):* The UPC is readjusted regarding a maximum threshold of allowed latency violations and an expected reconfiguration cost that is estimated in terms of the expected number of reassigned PDU sessions (see Section 4.2).

We ran the system for 10 hours for these experiments and measured the established QoS metric every minute until we gathered 600 samples. The performance of the scheduling mechanisms was investigated by considering several metrics of the system: the number of reconfiguration events, the number of sessions with latency violation at the reconfiguration moment, and QoS status. The number of relocated sessions and reconfiguration cost for the SSR and OSR models were estimated using the results obtained from the periodic reconfiguration with 30 minutes between reconfiguration events (PSR_P30) and a total of 1000 users with one active PDU session. For these mechanisms, we selected an upper bound on the QoS metric of 3% of PDU sessions with latency violations ($\Theta = 30$). The number of sessions with latency violations was modeled as a Poisson distribution with a mean of $\mu = 27$. This distribution was fitted on observed latency violations for a UPC configuration without readjustment for two hours.

To adjust the UPC configuration, we applied the proposed heuristic with a partial unmapping of 30% of additional sessions and one improvement attempt (i.e., $P_r = 30$ and $F_i = 1$). Furthermore, we conducted simulations by considering both sets of weight factors in the objective function (weight_set_1 and weight_set_2).

6.3.5.1 Reconfiguration Events

Figure 6.11 shows the number of reconfiguration events regarding the value of the established QoS metric (L_t) at the time of the reconfiguration procedures. We classified the number of sessions with latency violations into three groups (low, moderate, and high) with reference to the Θ threshold. To provide a more intuitive and legible representation, these groups are represented in different shades of blue (from light to dark).



(**a**) weight_set_1 (w_s=0)

(**b**) weight_set_2

Figure 6.11: Number of reconfiguration events with reference to QoS values at the reconfiguration moment for different scheduling mechanisms and sets of weight factors.

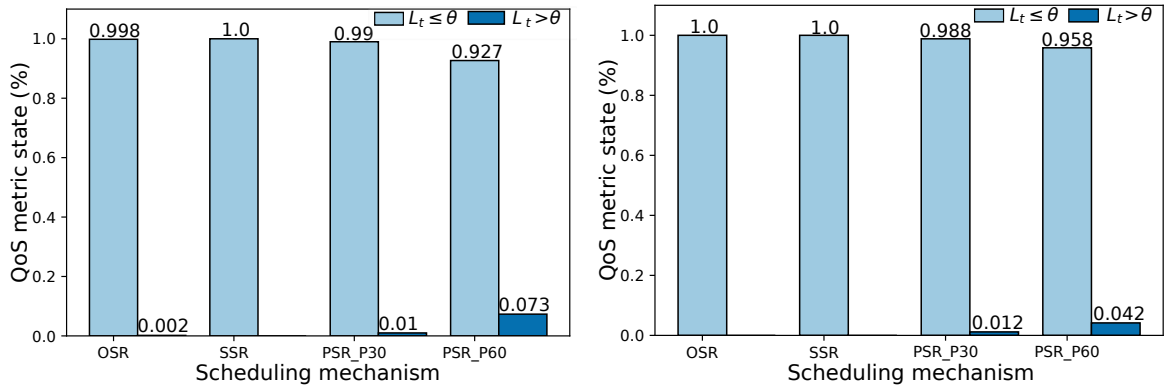
The experiments revealed that the proposed mechanism had the best performance. Precisely, it triggered the lowest number of reevaluation events and did it when the QoS threshold was close to being exceeded most of the time. Likewise, the SRR approach did not readjust the UPF placement and chaining configuration when the number of sessions with latency violations was low. However, it required significantly more reconfiguration events than the OSR solution. Specifically, SSR activated around twice and five times more reconfiguration events than the OSR mechanism for the first and second pair of weights, respectively.

The periodical approaches performed most of the reconfigurations when the number of sessions with poor QoS was low. Specifically, more than 60% and 40% of the reconfigurations for the PSR_P30 and PSR_P60 baselines, respectively, were triggered when $L_t \leq \frac{\theta}{2}$. The proposed OSR mechanism downsized the number of reconfiguration events between 65% and 85% and 30% and 70% compared to PSR_P30 and PSR_P60, respectively.

6.3.5.2 QoS Metric

Figure 6.12 summarizes the status of the QoS regarding the established threshold. The OST-based solutions (i.e., OSR and SSR) outperformed the periodic schedulers. For the OSR mechanism, the selected QoS metric was above the established threshold only once for the first set of weights and never for the second. Similarly, the SSR method always maintained under acceptable values the

system QoS although at the expense of more reconfiguration events than the OSR mechanism. Overall, they avoided QoS degradation for almost the entire simulation time in both cases.



(**a**) weight_set_1 (**b**) weight_set_2

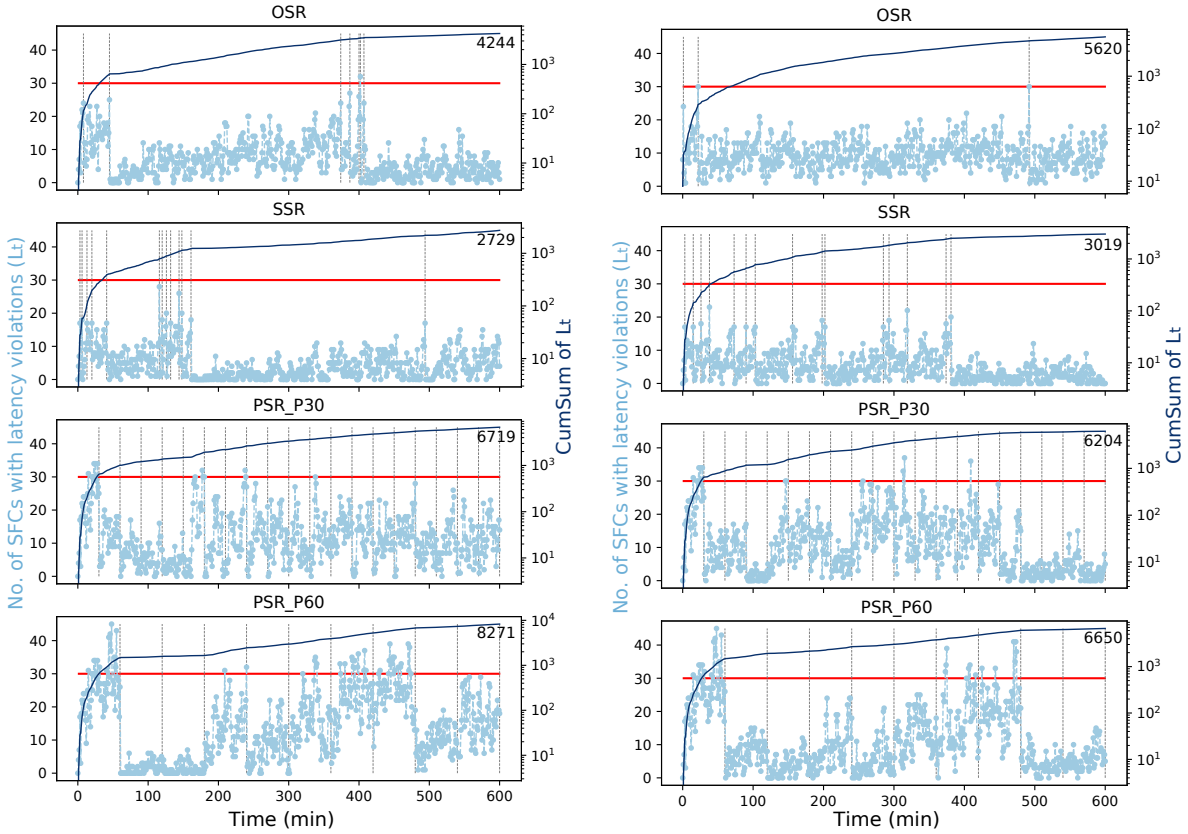
Figure 6.12: QoS status for different scheduling mechanisms and sets of weight factors.

In contrast, the periodic schedulers had the worst performance since they produced more violations of the established QoS threshold. The PSR_P30 resulted in a poor QoS for approximately 1% of the simulation time despite triggering the highest number of reconfiguration events for both sets of weight factors. Additionally, PSR_P60 had a more significant number of events with a poor QoS for around 7.3% of the simulation time for the first set and 4.2% for the second set.

Figure 6.13 depicts instantaneous and cumulative values of the number of sessions with latency violations over time for both sets of weight factors. It also illustrates the solution reconfiguration times using dashed grey lines. As seen in the figure, the OSR mechanism readjusted the UPC configuration when the number of sessions with a poor QoS was close to the established upper bound. The proposed OSR method only once exceeded the QoS threshold, for which a reconfiguration process was immediately activated. Furthermore, it did not trigger any reconfiguration event when the number of sessions with latency violations was small. Although the SSR mechanism showed similar behavior to the OSR, it produced more readjustment events due to its stopping rule, which initiated reconfigurations at lower values of the established QoS metric.

Inspecting the reconfiguration events revealed that both schedulers did not follow a fixed reconfiguration frequency since they depended on the values of the selected QoS metric, which is a random variable. The PSR approaches triggered the UPC reevaluation at fixed time intervals without accounting for the system QoS. For both sets of weight factors, the periodic approaches triggered these events independently of the QoS level, resulting in either unnecessary reconfigurations or poor QoS.

Concerning the cumulative sum of sessions with latency violations, the OSR mechanism had lower values than either variant of the PSR despite performing considerably fewer reconfigurations. Unlike the periodic approaches, it maintained the QoS parameter under acceptable values.



(**a**) weight_set_1 (**b**) weight_set_2

Figure 6.13: Number of sessions with latency violation and their cumulative sum per time instance for the DPC-UPC heuristic with partial unmapping (i.e., $P_r = 30$) and one improvement attempt ($F_i = 1$).

Therefore, PSR_P60 had the most significant number of sessions with latency violations since it was unaware of the QoS status and had the highest reconfiguration period.

6.4 Conclusion

This chapter investigated the problem of dynamic UPF placement and chaining reconfiguration. An ILP model and a heuristic-based solution were conceived to address this problem. These solutions minimize the overall reconfiguration cost while ensuring 5G stringent latency requirements and UPF and SFC specificities. The reconfiguration cost comprises multiple components, such as server activation, VNF migration, and SFC reassignment.

The DPC-UPC solution combines partial unmapping of SFCRs with an improvement phase to enhance the solution efficiency. Specifically, selecting a partial unmapping allows for faster running times, whereas the improvement phase enhances the reconfiguration costs at the expense of higher execution times. We compared its performance with two greedy-based algorithms and

the exact solution. The experiments revealed that DPC-UPC outperformed both heuristics concerning the analyzed metrics (e.g., reconfiguration cost and execution time). Furthermore, it provided near-optimal results in considerably less time than the baselines. Specifically, DPC-UPC determined solutions to the problem with average optimality gaps of 7.27% in its basic form and 4.25% for its improvement variant.

Furthermore, we devised a scheduler mechanism (i.e., OSR) based on OST to determine the optimal reconfiguration time. This scheduler made readjustment decisions according to instantaneous values of QoS (i.e., the number of sessions with latency violations), a maximum tolerance threshold, and the expected reconfiguration cost. Simulation results showcased significant improvements in the offered QoS and the number of reconfiguration events compared with the selected benchmarks. OSR provided the best QoS by keeping the QoS metric under the desired values for nearly the entire simulation. Moreover, it required the lowest number of reconfigurations compared to periodical schedulers, with reductions between 30% and 85%.

CHAPTER 7

INTELLIGENT SCHEDULING OF THE UPF PLACEMENT AND CHAINING RECONFIGURATION

This chapter is based on:

- **I. Leyva-Pupo**, and C. Cervelló-Pastor, "An Intelligent Scheduling for 5G User Plane Function Placement and Chaining Reconfiguration," Submitted to a Journal, 2022.

This chapter addresses the problem of determining the best time to readjust the UPC configuration to avoid QoS deterioration in the system. To this aim, we rely on machine learning (ML) techniques to predict the system QoS at a given time. These predictions help to decide, according to a pre-established QoS threshold, whether a UPCR is required. Specifically, we develop an ML-based framework called intelligent scheduling of the reconfiguration (ISR) to automate the decision process. This framework is responsible for data processing, model training, and UPCR decision and execution.

This chapter is organized as follows. Section 7.1 introduces the problem, while the conceived ML-based framework is detailed in Section 7.2. In Section 7.3, the performance of the proposed solution is investigated. Finally, Section 7.4 summarizes the main conclusions of this chapter.

7.1 Problem Description

In a set of 5G services and applications running close to the users at the network edge, service access is provided through a subset of UPFs, co-located with the edge data centers. As mentioned in previous chapters, theseUPFs may have different functionalities and may be chained together, thus forming different SFC topologies. Moreover, users are connected to the network and use

these services through UPFs. The service requests have been provisioned as a result of either an initial UPF placement and chaining configuration or a readjustment event by accounting for their service demands satisfaction and available infrastructure resources.

Initially, the QoS perceived by users is optimal, at least in terms of latency satisfaction. However, in dynamic scenarios, in which user demands and locations vary over time, the occurrence of QoS degradation events must be accounted for. For instance, user-perceived delays may increase due to their mobility toward less-available zones in terms of UPF capacities and due to an increase of the session data path length between the new access point and its anchor UPF.

Our main objective is to predict these QoS degradation events to reestablish the system QoS proactively through UPCRs. With the utilization of accurate ML-based prediction solutions, not only can the system QoS be kept under desired values, but frequent and unnecessary reconfiguration events can also be avoided. Similar to previous scheduling mechanisms, we measure the system QoS in terms of the instantaneous number of sessions with latency violations (L_t). We consider a session as having poor QoS when its perceived delay is higher than the one defined in its SLA. Thus, to ensure proper QoS levels, the E2E latency between users and their assigned data networks must be lower or equal to the service delay requirement ($L^s \leq L^s_{ser}$).

7.2 Solution Proposal: Intelligent Scheduling of the Reconfiguration

In this section, we describe our ML-based framework for the intelligent scheduling of the UPCR. The main goal is to anticipate reconfiguration decisions based on QoS predictions. Figure 7.1 provides an overview of the proposed framework and its main modules: data preprocessing, ML engine, and scheduler. Our conceived framework relies on supervised learning models –either regression or binary classification algorithms– to learn the relationship between the feature and labeled data during the training phase. In the following sections, we describe the framework's main modules and their interworking.

7.2.1 Data Preprocessing

The data preprocessing module performs data-related tasks: collection, processing, normalization, and feature selection (FS). It converts the raw monitoring data into time series samples suitable for training of the ML models (X_{t-1}, Y_{t-1}) and prediction (x_t).

This module collects and stores metric measurements from the virtual and physical networks, such as the number of deployed VNFs per type ($N_{u_{type}}$), VNF utilization level ($C_{u_{type}}$) and the number of active MEC servers ($N_{c_{act}}$). Additionally, it gathers information regarding the number of active PDU sessions (S), their service latency requirement, and user-perceived response time. These data are formatted as a multi-variate time series according to the specified sampling time (t_s). The module can also work with historical data to increase the number of samples.

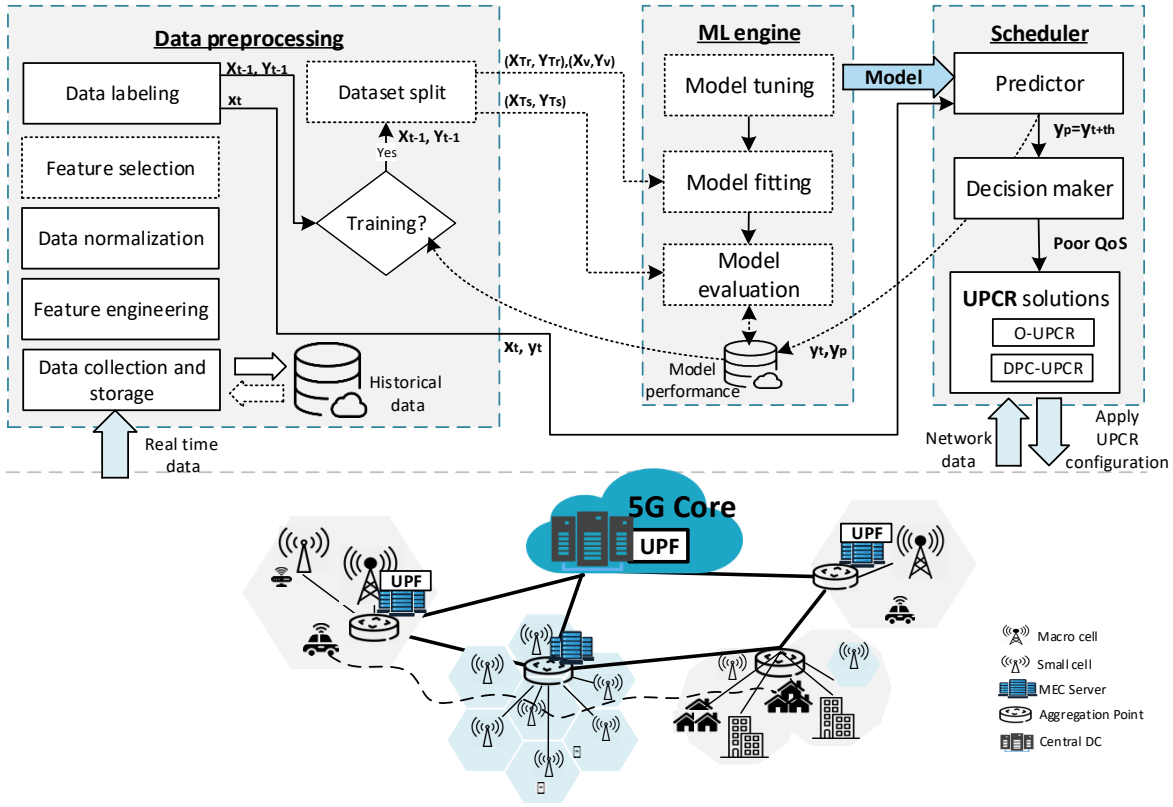


Figure 7.1: Architecture overview of the conceived ML-based framework for scheduling UPCRs based on QoS predictions.

Moreover, feature engineering is conducted to provide additional information about the system. For instance, the module extracts the maximum, minimum, and average values for parameters such as VNF utilization and user-perceived latency, which are multi-dimensional. Apart from these variables, another subset of features is built upon the collected data. This subset comprises aspects such as time since the last UPCR event ($T_{last_{reconf}}$) and change in the number of sessions with latency violation with reference to the last sampling time ($L_{t_{change}} = L_t - L_{t-1}$). It also includes the maximum, minimum, and average numbers of sessions with poor QoS between sample times ($L_{ts_{max}}$, $L_{ts_{min}}$, and $L_{ts_{aver}}$). To provide more information about the system, these three last parameters are constructed when the sampling time is greater than the collection time.

Once all input variables have been obtained, data normalization is applied to transform all the features into a similar scale (e.g., [0,1]). This technique avoids dimensional differences between the data, thereby ensuring a significant impact for all input variables and improving the convergence speed of the machine learning model.

Next, we select the most relevant features to predict the target variable. The FS sub-module implements several mechanisms belonging to the three main categories for feature extraction (i.e., filter, wrapper, and embedded methods). Currently, it supports the following methods: Pearson correlation, univariate selection, recursive feature elimination, random forest (RF), back-

ward elimination, forward elimination, lasso regularization, and principal component analysis. However, it can be easily extended to support other mechanisms.

Given that each FS method captures a particular relationship between the input and target variables, we adopt a combined approach to analyze multiple methods and thus provide more robust results. To this aim, we must specify the methods to be evaluated, as well as a decision threshold, which is upper bound by the number of selected techniques. The feature dataset is evaluated for every method, and a score of either zero or one is given to the input variables based on their significance. After assessing all specified FS techniques, the input variables with an overall method score greater than or equal to the decision threshold are selected as the independent variables for the ML module.

Furthermore, a data-labeling process occurs to convert the data into a supervised learning dataset. This is done by shifting the input variables backward according to the specified prediction horizon (t_h) and using the output variable (L_t) as the label at the proper time step (T_{samp}). Furthermore, this block determines the class labels for the binary classification methods. Namely, each sample is binarized according to the QoS value and the defined threshold (Θ). When the number of sessions with latency violation is above the established threshold, the QoS is considered deteriorated, as indicated by a label equal to 1. Otherwise, it takes the value of zero.

Additionally, this module splits the data into subsets for training procedures. This task is initially performed when launching the framework or when model retraining is required. This may be necessary based on predefined time intervals or model performance metrics that need to be defined by the system administrator along with the splitting proportions (T_r%:V%:T_s%) for training (T_r), validation (V), and test (T_s). Since the framework works with a time series, the dataset split must be done without shuffling to maintain the temporal relationship among consecutive observations. As a result, we obtain three subsets corresponding to training (X_{Tr}, Y_{Tr}), validation (X_v, Y_v), and test (X_{Ts}, Y_{Ts}) datasets.

Finally, this module outputs the processed data in a suitable form for the prediction procedure and model training. It returns the current time sample (x_t, y_t), which is passed to the scheduler block for predicting the QoS indicator at the specified time horizon ($y_p = y_{t+th}$). Additionally, the training, validation, and test datasets, up to the previous timestamp, are transferred to the ML engine when a model readjustment is required.

The FS and data splitting steps are not always required, as indicated through a dotted block representation. These two processes are executed when the framework is launched or the tuning and training procedures are activated.

7.2.2 ML Engine

The machine learning engine determines the ML algorithm and parameters that best fit the dataset. To determine this, we must stipulate the models we want to use. Moreover, we must specify the hyper-parameters to tune for each ML algorithm and their possible values. At least

one model must be specified, or already-tuned models can be loaded, which would avoid the tuning phase. This module currently supports the following algorithms, which have been implemented for classification and regression solutions:

- *Random forest (RF) [167]:* It is a supervised algorithm that uses an ensemble learning method for classification and regression problems. It consists of a set of decision trees forming a forest. For regression, the prediction output is the mean of the individual trees, while for classification, the output is the majority of the classes. These algorithms do not overfit since they work with average values. Moreover, they have a few hyper-parameters to tune. The most common are the number of trees, the maximum number of features, and the minimum number of leaves.

- *Support Vector Machine (SVM) [168]:* A supervised algorithm based on statistical learning theory. Support vector classification (SVC) separates samples into different classes by constructing hyper-planes in a multidimensional space. Hyper-planes are boundaries used to classify the samples, the dimensions of which depend on the number of features. Support vector regression (SVR) [169, 170] is a method derived from SVM to solve multi-variate regression problems. Overall, these two methods are characterized by two main parameters: the kernel and cost (C) functions.

- *Multi-layer Perceptron (MLP) [13]:* A type of neuronal network with at least three sequentially connected layers: an input layer, one or more hidden layers, and an output layer. Its hyper-parameters include the number of epochs and batch size, the optimizer, and the learning rate. It uses a supervised learning technique called back-propagation.

- *Bidirectional Long Short-Term Memory (BLSTM) [171]:* LSTM is a special kind of recurrent neuronal network (RNN) capable of learning long-term dependencies [172]. BLSTM trains two LSTMs on the input data. Specifically, it trains the input flows for each LSTM in different directions (backward and forward) to preserve past and future information.

When initializing the framework, the ML engine's first step is determining the optimal set of hyper-parameters that best fit the dataset. Therefore, the model tuning block builds a base model and performs a grid search over these parameters. Each model is characterized by specific parameters that must be carefully tuned to improve its performance. For instance, in the case of the SVM models, hyper-parameters such as gamma, penalty loss, and kernel function can be adjusted. In addition, for MLP and BLSTM-based models, a higher set of parameters must be tuned, such as activation function, the number of hidden layers, and neurons per hidden layer. The hyper-parameter tuning process may be a laborious and time-consuming task. Thus, it should be performed it offline. Additionally, manual exploration of the hyper-parameter values can be performed to reduce the number of possible combinations and, thereby, the models' tuning time.

For classification problems, the resulting class distribution may be highly imbalanced since the number of QoS degradation events are usually less common. However, an imbalanced distribution depends on several aspects, such as the established QoS threshold, the generated placement and chaining configuration quality, and user mobility patterns.

When working with classification problems, a standard optimization metric is accuracy. However, this can be a misleading performance indicator for imbalanced datasets since accuracy can be biased to the majority class [129]. That is, the model may show high accuracy but poor performance in detecting the minority class. To overcome this issue, we implement two techniques that reduce the effects of an uneven dataset distribution; we set the F1 score as a performance metric and modify the training algorithm to consider the classes' imbalanced distribution by defining the class weight parameter as balanced. Although other approaches can be implemented, such as re-sampling and synthetic data generation, we prefer these two due to their simplicity.

As a result of the tuning step, we obtain the parameter set that provides the best performance for each model. Then, the models are fitted with the selected combination of parameters. The training process can be repeated to adjust the model to data variations. The training times depend on the complexity of the ML model and its parameters. To save time, long training-time models can be fitted offline and loaded when ready.

The evaluation block assesses the performance of the models under study and selects the best when more than one is specified. Similarly to the other blocks composing this module, the evaluation process is not always required unless a model retraining based on performance has been indicated. In this case, it keeps track of the model performance based on the predicted and actual values of the target variable (QoS indicator) and a chosen metric. This metric value is sent to the preprocessing module to train decision-making.

Every model must be tuned, trained, and evaluated with the generated dataset to determine the best one. The scheduler module uses the selected model to predict the system QoS.

7.2.3 Scheduler

The scheduler is in charge of triggering the UPCR based on the predicted QoS of the system and a decision threshold (i.e., Θ). It consists of three sub-modules: the predictor, the decision maker, and the UPCR solution.

The predictor block loads the ML model generated by the ML engine and predicts the system QoS (y_p) at the specified time horizon based on the preprocessed data for the current sample (x_t). Depending on the model type (classifier or regressor), the predictor output can be a continuous or binary value. For regression models, the prediction corresponds to the value of the selected QoS metric at a given time horizon (i.e., the number of sessions with poor QoS). For classification models, the predictor output is a binary variable indicating the status of the QoS at the specified time horizon, where zero means good QoS and one indicates poor QoS. Then, the decision maker can determine whether a readjustment of the UPC configuration is required based on the

established QoS threshold and the forecasted QoS value. The output of a classification algorithm automatically indicates the action to be performed.

The final block of our framework is the UPCR solution which is in charge of readjusting UPF placement and session mapping configuration to cope with QoS degradation events produced due to user mobility. Its main goal is determining the best UPC arrangement that reestablishes the system QoS to optimal values ($L_t = 0$) and minimizes the reconfiguration costs. To this aim, it relies on the solutions provided in Chapter 6, referred to as O-UPCR and DPC-UPCR. Similar to the ML engine, this block can be extended to include more UPCR solutions. Heuristic methods are recommended for large-scale scenarios to improve solution efficiency.

To solve the UPCR problem, the UPCR requires information related to the system configuration, such as current VNF placement, SFC mapping, infrastructure utilization (node capacity and link bandwidth), and service demands. As a result of the solution execution, it returns a reconfiguration cost and a new UPC configuration.

7.3 Evaluation and Results

7.3.1 Simulation Setup and Data Generation

We considered a 5G medium-scale network topology composed of 121 access nodes and 13 MEC servers for the dataset generation. The MEC servers connected with the access nodes through aggregation points. Moreover, 1000 mobile users were assumed to be connected to the network, each with an active PDU session. These sessions had different SFC topologies and requirements (processing traffic, bandwidth, and latency) as specified in Subsection 6.3.1.

Upon initial user positions, a UPC configuration was setup by applying the PC-UPC algorithm and giving more importance to node activation and VNF deployment costs than the routing component in the objective function (i.e., $\alpha = \beta = 0.4$ and $\gamma = 0.2$). User mobility traces were generated using CityMob for a simulation period of 120 hours; see Table 6.5. During this time, the effect of user mobility over the UPC configuration was analyzed. Moreover, random UPCRs were conducted with reconfiguration times of 1800, 3600, and 7200 seconds to investigate the system QoS behavior over time under different conditions. For the UPCR, the DPC-UPC heuristic with a partial unmapping of 30% of sessions and three improvement attempts was applied by considering all the optimization terms as equally important.

The framework modules were implemented in Python along with the conceived solutions for the UPCR. We used Keras and sklearn libraries in Python to implement the neuronal network (NN)-based models and the RF and SVM models, respectively.

7.3.2 Data Preparation

We collected several metrics concerning the system QoS: the number of sessions with latency violations (L_t) and user-perceived delay (Lat), UPCR time (T_{reconf}), number of active sessions

(S), active servers ($N_{c_{active}}$) and deployed VNF instances per type ($N_{u_{type}}$), and VNF and server utilization ($C_{u_{type}}$).

The data generated by the system were processed and structured in time series format by the preprocessing module of our framework according to the specified sampling time and prediction horizon ($t_s = t_h = 60$ s). This module determined the values of every metric at each sampling time and added a time label (T_{samp}). Additionally, it extracted the maximum, minimum, and average values of multidimensional metrics, which are indicated by the max, min, and $aver$ sub-indexes in the variable name. Apart from these features, additional variables were built upon the collected samples. Table 7.1 summarizes all features (both collected and constructed) composing our data as well as their score, in descending order, obtained upon the combined evaluation of the FS methods.

Table 7.1: Features' score.

Feature	Score	Feature	Score	Feature	Score	Feature	Score
$T_{last_{reconf}}$	7	$L_{ts_{aver}}$	4	$N_{c_{open}}$	2	$C_{u_{aupf_{min}}}$	1
$L_{ts_{max}}$	6	Lat_{max}	4	$N_{u_{iupf_{ava}}}$	2	$C_{u_{aupf_{aver}}}$	1
L_t	6	$N_{u_{iupf_{all}}}$	4	$C_{u_{aupf_{imb}}}$	2	$C_{u_{iupf_{max}}}$	1
T_{samp}	5	$C_{u_{iupf_{aver}}}$	4	$C_{u_{miupf_{max}}}$	2	$C_{u_{iupf_{imb}}}$	1
$L_{t_{change}}$	5	$L_{ts_{min}}$	3	Lat_{min}	1	$C_{u_{miupf_{min}}}$	1
$N_{u_{aupf_{ava}}}$	5	Lat_{aver}	3	$N_{u_{aupf_{all}}}$	1	$u_{miupf_{imb}}$	1
$C_{u_{iupf_{min}}}$	5	$C_{u_{aupf_{max}}}$	3	$N_{u_{miupf_{all}}}$	1	$C_{u_{miupf_{aver}}}$	0
$L_{t_{all}}$	4	S_{reloc}	2	$N_{u_{miupf_{ava}}}$	1	S	0

For the feature extraction process, we applied the eight FS methods previously mentioned in Subsection 7.3.2. As seen in Table 7.1, the most relevant features were those related to QoS metrics and time (T_{samp} and $T_{last_{reconf}}$). Other metrics, such as the number of active PDU sessions, VNF utilization levels, and the number of deployed instances, were irrelevant since we maintained the number of connected users and their traffic demands. This approach was preferred to investigate the effects of user mobility on the UPC configuration and system QoS. For our experiments, we selected the three most dominant features (threshold score equal to six).

After performing FS, our processed data had three independent variables, each with 7200 samples. This dataset was decomposed into training, validation, and test subsets following the 60%:20%:20% proportion. The first 60% and 20% of samples in the time series were selected for training and validation, while the final 20% was used for testing. Moreover, we normalized the selected data within a range of [0,1] using the min-max scaling function.

7.3.3 Model Tuning and Evaluation Metrics

We applied grid search and manual search strategies to find the best hyper-parameters to boost the models' performance for our evaluation data. To this aim, we performed an extensive exploration by analyzing a wide range of hyper-parameter values.

As previously mentioned, each ML model has diverse hyper-parameters that must be carefully set. Tables 7.2 and 7.3 summarize the hyper-parameters and their range of values used to tune the models. These values are represented as a single list when common for both regression (R) and classification (C) algorithms. Otherwise, individual lists for each model type are provided. The tables also show the best combination of parameter values for the conceived models and selected features.

Table 7.2: Hyper-parameters values for RF and SVM-based regressor and classifier.

Parameters	**Values**	**Models**		**Parameters**	**Values**	**Models**	
		RF_R	**RF_C**			**SVR**	**SVC**
criterion	R:['mse', 'mae', 'poisson'] C:['gini', 'entropy',]	poisson	entropy	*C*	[0.001, 0.1,0.5,1,5, 10,50,100,500, 700, 1000, 2000,3000]	1000	0.1
max_depth	[25,50,100-500:100, 1000,2000,3000, None]	None	2000	*degree*	[1,2,3]	1	2
max_features	[1,2,3,'sqrt', 'log2', 'auto']	auto	3	*epsilon*	[0.0005, 0.001, 0.005,0.05,0.1]	0.1	—
min samples_leaf	[1-5:1,10,20, 25,50,100]	1	1	*gamma*	[0.05,0.1,0.5,1,5,10, 20,30,50, 100,200, 500, 'scale', 'auto']	200	0.1
min samples_split	[3,5,10,25,50,100, 200,300,500,1000]	300	2	*kernel*	['rbf','sigmoid', 'poly', 'linear']	poly	poly
number estimators	[1,5,10,50, 100-1000:100,3000]	10	1000	*tolerance*	[0.0001,0.001, 0.01,0.1,1]	0.001	0.01
				coef0	[0,0.0001,0.001, 0.01,0.1,1]	0	0

Table 7.3: Hyper-parameters values for the NN-based classifiers and regressors.

Parameters	**Values**	**Models**			
		BLSTM_R	**BLSTM_C**	**MLP_R**	**MLP_C**
activation	['softmax', 'softsign', 'linear', 'relu', 'elu', 'selu', 'tanh', 'sigmoid']	elu	sigmoid	elu	sigmoid
batch	C:[3,5,10,25,50,100-500:100] R:[10,50-500:50,700-1000:100]	100	5	200	3
dropout	[0-0.5:0.1]	(0.5)	(0.3,0.3,0.3)	(0, 0)	(0.5,0)
epochs	[50-500:50, 600-1000:100,1500]	1000	350	700	250
hidden layers	[1-5:1]	1	3	2	2
kernel initializer	['uniform','normal', 'random_uniform', 'random_normal', 'he_normal', 'he_uniform', 'glorot_normal', 'glorot_uniform',]	normal	glorot_normal	glorot_normal	he_uniform
loss	R:[MAE, MSE, huber_loss C:[binary_crossentropy (bc)]	MSE	bc	MSE	bc
lr	[0.5,0.1, 0.05,0.01, 0.005, 0.001 ,0.0005,0.0001]	0.0005	0.0005	0.0005	0.01
neurons	[50-300:50,700-800:100,1000]	(200)	(700,50,50)	(200,150)	(500,100)
optimizer	['adam', 'rms', 'sgd']	adam	rms	adam	rms

The architecture of the neuronal networks (MLP and BLSTM) consists of an input layer whose number of nodes depends on the number of selected features (i.e., three), a certain number of hidden layers with a given number of neurons, and an output layer formed by one neuron. The number of hidden layers and each layer's number of neurons, dropouts, and activation functions are depicted in Table 7.3. For the regression models, a linear activation function was used for the output layer, while a sigmoid function was used for the classifiers.

The tuning process was tedious and time-consuming since it is highly conditioned by the used ML algorithm and dataset.

We evaluated the performance of the regression models according to the following metrics:

- *Mean absolute error (MAE):* Expresses the average absolute difference between the actual and predicted values (error).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - y_i^p| \tag{7.1}$$

- *Root mean squared error (RMSE):* Measures the average magnitude of the errors of the actual value concerning the forecast.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i^p)^2} \tag{7.2}$$

- R^2 *score:* Measures how well-unseen observations are likely to be predicted by the model. In other words, it indicates the proportion of the predicted variable (y^p) that the independent variables can explain.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - y_i^p)^2}{\sum_{i=1}^{n}(y_i - \frac{1}{n}\sum_{i=1}^{n} y_i)^2} \tag{7.3}$$

  where n indicates the number of samples in the dataset and y_i and y_i^p are the actual and predicted values of the target variable.

MAE and RMSE are two of the most widely used metrics to measure regression models' performance. They provide a precise measurement of the prediction error since they have the same scale of the prediction variable. The lower these metrics are in value, the better the model performance. Conversely, the higher the R^2 score, the better the model, with 1 being its maximum value.

Additionally, the classification algorithms were evaluated based on the following metrics:

- *Accuracy (ACC):* Indicates the fraction of correct predictions (i.e., TP and TN) over the total number of observations.

$$ACC = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} \tag{7.4}$$

- *Precision (P):* The ability of a model to predict positive observations correctly. It indicates the proportion of positive observations correctly classified as positive.

$$P = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{TP_i + FP_i} \tag{7.5}$$

- *Recall (R):* Represents the capacity of a model to identify positive observations from all positive observations.

$$R = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{TP_i + FN_i} \tag{7.6}$$

- *F1 measure (F1):* Represents the weighted mean of precision and recall metrics. It gives a measure of the balance between precision and recall.

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{7.7}$$

  where TP and TN denote the correct identification of positive and negative observations (i.e., true positive and true negative), respectively. Additionally, FP and FN indicate the incorrect classification of positive and negative samples (i.e., false positive and false negative).

The higher the value of these four metrics, the better the modal performance. Their values range from 0 to 1 and are usually expressed in percentages. Accuracy is the most common evaluation metric in classification problems. However, in imbalanced datasets, it can be an unreliable metric. Thus, other performance parameters, such as precision, recall, and F1 score, are recommended for datasets with an uneven class distribution. As seen in Fig. 7.2, our dataset was highly imbalanced since the number of events with poor QoS was significantly smaller.

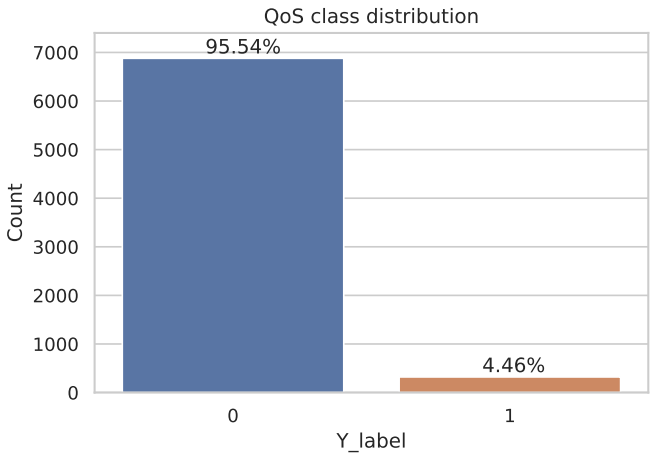


Figure 7.2: Class distribution of the target variable (i.e., QoS status).

Additionally, we also investigate the models' performance regarding their training time and the capability of the decision maker to determine the QoS status correctly. For the latter, we established a QoS tolerance threshold of 3% of sessions with latency violations; upon this value, the QoS of the system is considered deteriorated.

7.3.4 Regressor-based QoS Predictors Performance

This subsection investigates the performance of the regression models in terms of MAE, RMSE, and R^2 score metrics. Due to the stochastic nature of the models, we ran each model multiple times until we gathered 100 samples.

As seen in Fig. 7.3, the RF regressor (RF_R) had the poorest performance. It not only provided significantly worse MAE and RMSE scores but also more deviation between iterations. In particular, its average values for the MAE and RMSE scores were at least 40% and 70% higher, respectively, than for the other algorithms.

The SVR, BLSTM, and MLP regressors (i.e., BLSTM_R, MLP_R) had typical MAE values ranging from 2.90 to 2.95. Their RMSE metric's most common values were between 4.25 and 4.36. Regarding these models, the best performance was obtained by the BLSTM_R, followed by the MLP_R. Although both models had similar average values, the deviation in the results obtained by the BLSTM_R was smaller than the obtained by the MLP-based regressor. Conversely, the SVR had slightly higher MAE and RMSE values. However, it was the most precise regressor, with no difference in the obtained results between iterations.

By comparing the MAE and RMSE metrics, we found that all models had some variation in the magnitude of errors. The RF_R had the highest variance in individual errors, with an average difference of 3.34 between both metrics. For the other models, this value was around 1.4.

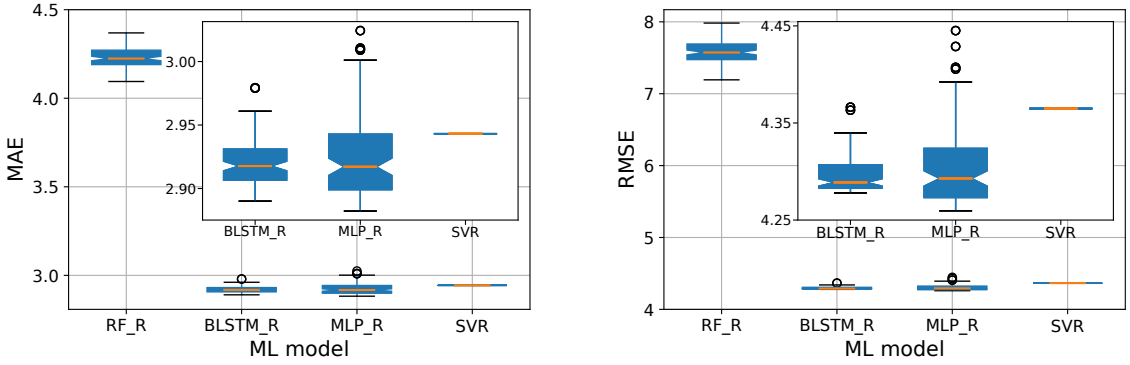
Regarding the R^2 score, all the regressors under study were able to explain more than 75% of the target variable from the independent variables, as seen in Fig 7.3(**c**). The regression model based on the RF strategy explained between 76% and 81% of the number of sessions with bad QoS at each time instance. The other regression algorithms significantly outperformed these results, providing average values equal to or above 93%. This showed their ability to explain a significant amount of variance.

7.3.5 Classifier-based QoS Predictors Performance

Figure 7.4 shows the obtained result for the classifiers under study regarding precision, recall, F1 measure, and accuracy. These results were obtained by running each model 10 times.

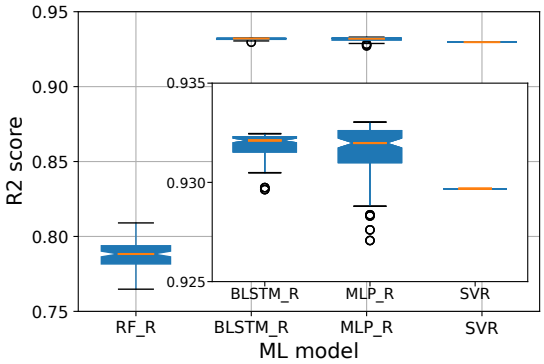
As seen in Fig. 7.4(**a**), all classifiers had a precision score above 80%, which indicated that they could correctly predict QoS violation events at least 80% of the time. The best performance was provided by the MLP and RF-based classifiers (MLP_C and RF_C), which had a mean precision of around 95%. In contrast, the BLSTM_C had the poorest performance, with an average precision of 85% and a high deviation with values ranging between 82% and 90%.

Regarding the recall metric, all algorithms had a score higher than 90%, with average values over 92%, as shown in Fig. 7.4(**b**). This showed that they correctly identified more than 90% of all events with poor QoS. However, the BLSTM classifier outperformed the other models with a typical score above 95%, followed by the SVC, which had a precision score rounding to 95%.



(a) MAE

(b) RMSE



(c) R^2 score

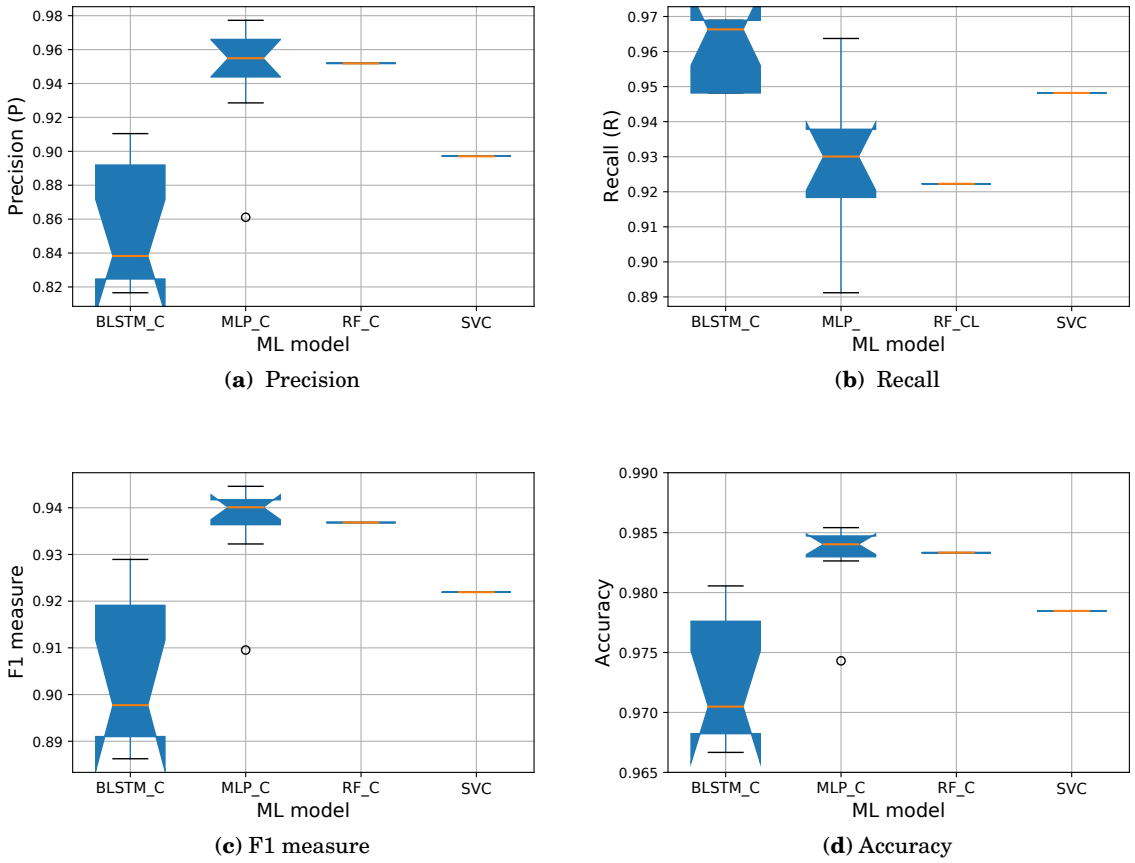
Figure 7.3: Performance of the regression models for QoS prediction.

As seen in Figs. 7.4(**a**) and 7.4(**b**), the models with the highest precision provided the lowest recall and vice-versa. Thus, these are opposite metrics; namely, a good performance in one of these metrics comes at the expense of the other. Thus, the F1 measure plays a crucial role in determining an equilibrium between both metrics and providing a more general view of the models' performance.

Figure 7.4(**c**) depicts the performance of the four classifiers concerning the F1 measure. All the models had a score superior to 89%, and the precision score highly influenced its behavior. The best performance was obtained by the MLP_C model, followed by the RF_C algorithm. Overall, both models provided an F1 score above 93%. Similar behavior in terms of accuracy is visible in Fig. 7.4(**d**), where both models outperformed the others with values higher than 98%.

7.3.6 Regression versus Classification Performance

In this subsection, we compare the performance of the regressors and classifiers in terms of correct prediction of the QoS status and their required training times. The predicted values of the



(a) Precision **(b)** Recall

(c) F1 measure **(d)** Accuracy

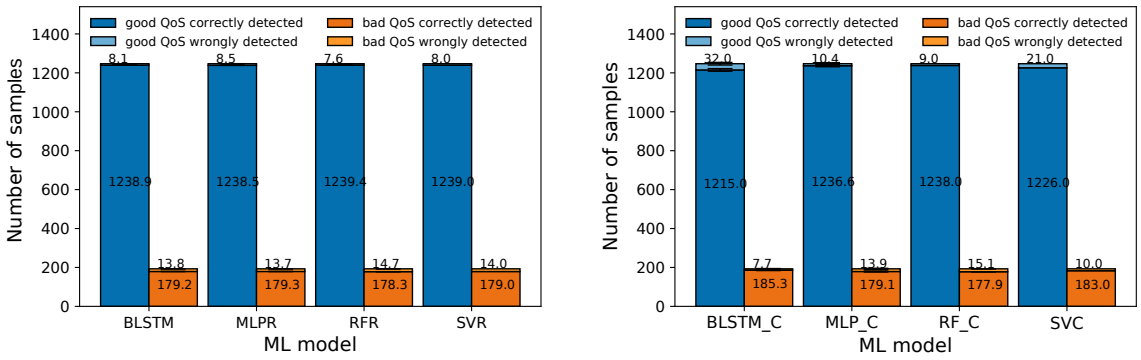
Figure 7.4: Performance of the classification models for QoS status prediction.

regression models were transformed into binary by comparing them with the established QoS threshold. This allowed us to compare both categories more evenly.

Figure 7.5 shows the QoS status predicted for each model. This metric was grouped into good and poor QoS samples, and each category was further divided regarding the prediction accuracy (i.e., correct or wrong identification). Figure 7.5(**a**) shows no substantial difference among the QoS status predicted by the regressors. However, BLSM_R and SVR provided slightly better results than the others. They correctly identified an average of 179 samples from the 193 observations with poor QoS in the test subset, which represented 92.7% of the total. Additionally, only eight out of 1247 samples with high-quality service were misclassified.

The difference among the classification models' predictions was more significant. The models with higher recall (BLSM_C and SVC) could correctly detect more samples with deteriorated QoS. However, this came at the cost of incorrectly predicting a higher number of events with acceptable QoS levels. In contrast, the MLP and RF-based classifiers correctly classified a higher number of observations with good QoS but detected fewer events with poor QoS.

By comparing both graphs, we concluded that, in general, the regression models had a



(**a**) Regressors (**b**) Classifiers

Figure 7.5: Prediction of the system QoS status inferred for the regression and classification models.

slightly better performance than the classifiers. They correctly predicted a higher ratio of samples with good QoS, thus reducing the number of unnecessary reconfiguration procedures. They also identified a considerable proportion of events with poor QoS.

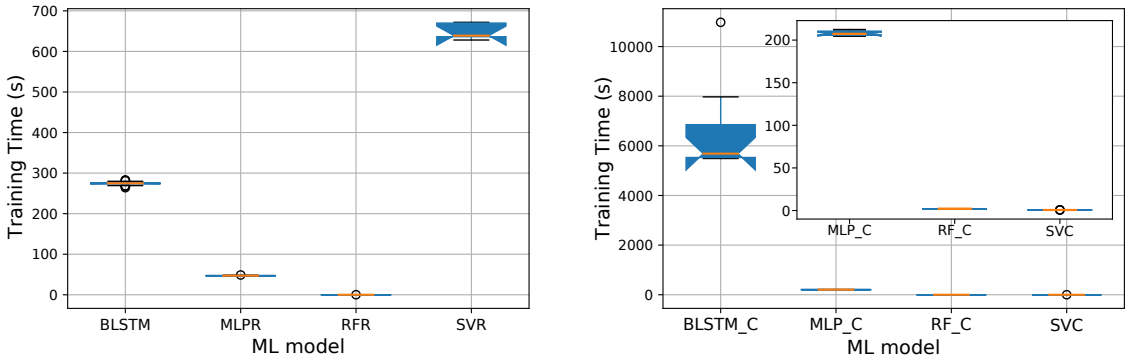
However, a disadvantage of the classification models is that they had to be tuned and retrained each time the QoS tolerance threshold (Θ) changed since this threshold determines the class distribution of the target variable.

Regarding the models' training time, see Fig. 7.6; notable differences were found between both ML categories (i.e., regression and classification) and among the models in each category. This was because this metric is highly dependent on the complexity of the model, which was different in each case. Overall, the BLSTM_C had the worst performance, with considerably higher training times than the rest, which made it difficult to tune. Regarding the regressors, the training time required by the SVR model significantly exceeded the other regressors. In general, the RF-based algorithms provided the best performance in terms of training time.

7.3.7 ISR Performance

This subsection discusses the performance of the proposed ISR strategy by comparing it to some of the scheduling mechanisms described in Chapter 6. Specifically, we used the OSR and periodic schedulers (i.e., PSR_P30 and PSR_P60) as benchmarks. We investigated their efficiency regarding four main aspects: the number of UPCRs and sessions with latency violation at the reconfiguration moment, the average reconfiguration cost, and the overall QoS. Unless specified for the metrics under study, we showed average results obtained by running each solution 10 times and using a 95% confidence interval.

We constructed our ISR by selecting the BLSTM_R model as the predictor based on the results discussed in the previous subsections. This ML could successfully determine most of the events



(a) Regressors **(b)** Classifiers

Figure 7.6: Training times required by the regression and classification models.

with poor QoS from the test dataset, as shown in Fig. 7.7. However, due to the unstable behavior of our target variable, the predictor could not always react as quickly as desired. Therefore, we considered the MAE of the predictions in the decision-making phase to reduce the likelihood of exceeding the Θ threshold. For these mechanisms, we maintained an upper threshold on the QoS metric of 3% of sessions with latency violations ($\Theta = 30$). For the OSR strategy, the number of sessions with latency violations was modeled as a Poisson distribution with a mean of $\mu = 21$. Each scheduler was evaluated for 24 hours with a sampling time of 1 minute. Additionally, we used a prediction horizon of 1 minute for the ISR mechanism.

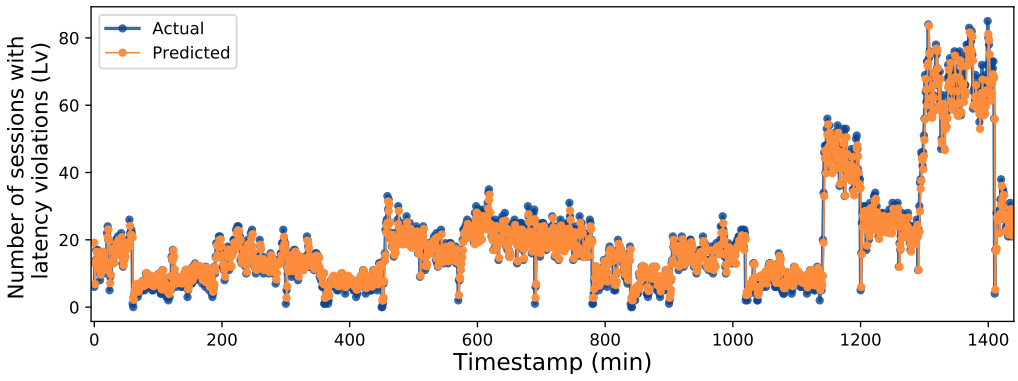
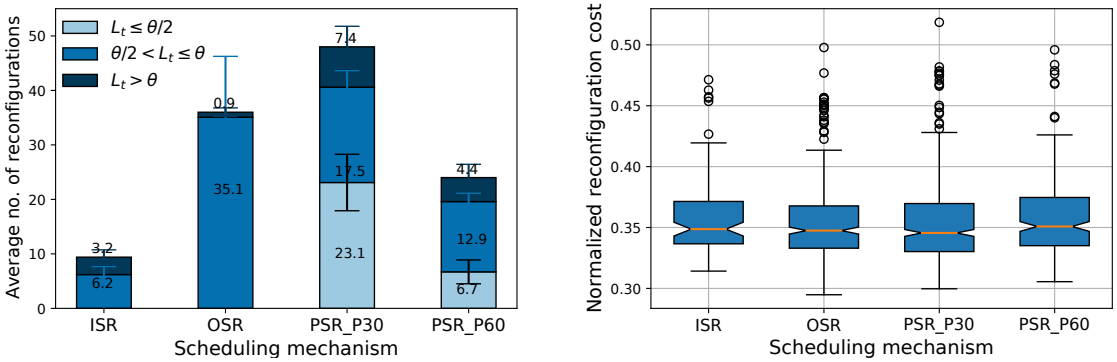


Figure 7.7: Actual versus predicted values of QoS provided by the BLSTM_R model.

7.3.7.1 Reconfiguration Events

Figure 7.8 depicts the performance of the four scheduling mechanisms in terms of the average number of reconfiguration events and cost.

As seen in Fig. 7.8(**a**), the ML-based scheduler (ISR) outperformed the other mechanisms,



(**a**) Average number of UPCR events.

(**b**) Average UPCR cost

Figure 7.8: Performance of the schedulers regarding the number of reconfiguration events and normalized UPCR cost.

providing significant reductions in the number of reconfigurations. Specifically, it decreased the activation of UPCRs by at least 72%, 79%, and 58% compared with the OSR, PSR_P30, and PSR_P60 schedulers, respectively.

The figure also shows the number of sessions with latency violations at the reconfiguration moment, which were categorized into three levels (low, medium, and high) in terms of the Θ threshold. The ISR and OSR solutions never triggered a reconfiguration when the QoS metric value was low ($L_t \leq \frac{\Theta}{2}$); most of these events were activated before exceeding the tolerance threshold. In fact, both schedulers had the lowest number of UPCRs executed under poor QoS conditions. In contrast, a significant fraction of the average reconfigurations performed by the periodic schedulers was unnecessary since the number of sessions with latency violations was far from exceeding the Θ threshold.

We also investigated the behavior of the UPCR cost function to determine whether the ISR performance was influenced by the resulting placement and chaining configuration. However, as seen in Fig. 7.8(**b**), this was not the case; overall, the UPCR events triggered by all solutions had a similar cost distribution.

7.3.7.2 System QoS

Figure 7.9 summarizes the overall status of the system QoS in terms of good ($L_t \leq \Theta$) and poor ($L_t > \Theta$) QoS levels. As expected, the solutions proposed in this thesis (OSR and ISR) had the best performance since they were aware of the system QoS. They kept the system QoS under desired values for almost the entire simulation time with a percentage of samples with good QoS values superior to 99.8%. The OSR strategy provided slightly better results than the ISR, but at the expense of increased reconfigurations.

The periodic schedulers provided inferior performance. They had many samples with deterio-

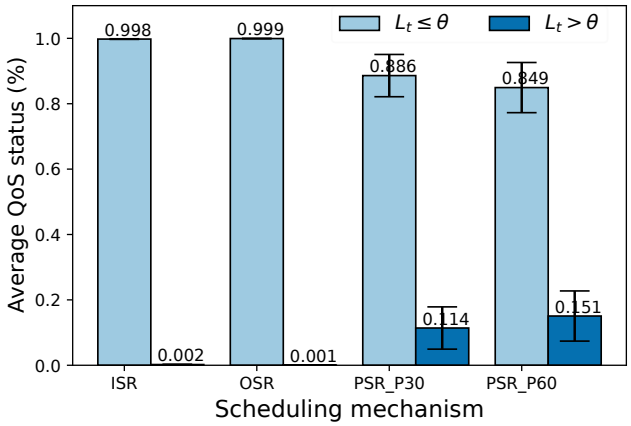


Figure 7.9: QoS status provided by the UPCR schedulers.

rated QoS and a significant deviation from the mean among experiments. The system resulted in poor QoS 11.4% and 15.1% of the simulation time when applying the PSR_P30 and PSR_P60 mechanisms, respectively. However, by comparing both strategies, we found no significant difference in their outcomes despite PSR_P30 executing twice as many reconfigurations as the PSR_P60.

Figure 7.10 displays an overview of the system QoS over time in terms of the instantaneous and cumulative number of sessions with latency violations for one experiment. Furthermore, it illustrates in dashed gray lines the UPCR times and the predicted values of QoS for the ISR mechanism.

As seen in this figure, the conceived schedulers kept the QoS metric under the established threshold most of the time, with a low number of samples exceeding the Θ threshold. In this case, these schedulers activated a UPCR to return the system QoS to optimal values. Moreover, they triggered a reconfiguration when the number of sessions with bad QoS was near the Θ indicator. By comparing both mechanisms, we found that the ISR substantially reduced the number of reconfigurations with reference to the OSR strategy. The ML-based scheduler makes predictions based on recent and broader system information, while the OSR was more conservative since it works with expected values. No fixed frequency was observed regarding their reconfiguration times since UPCR events depend on the QoS as conditioned by user mobility and the underlying placement and mapping configuration.

In contrast, the periodic schedulers had a high number of observations with poor QoS; typically, these events continued until the next reconfiguration period. Moreover, most of their reconfigurations were unnecessary since the system QoS was under acceptable values at their execution times.

Concerning the cumulative sum of sessions with latency violations, the OSR and ISR provided the lowest values. This was because they could keep the system QoS under the established

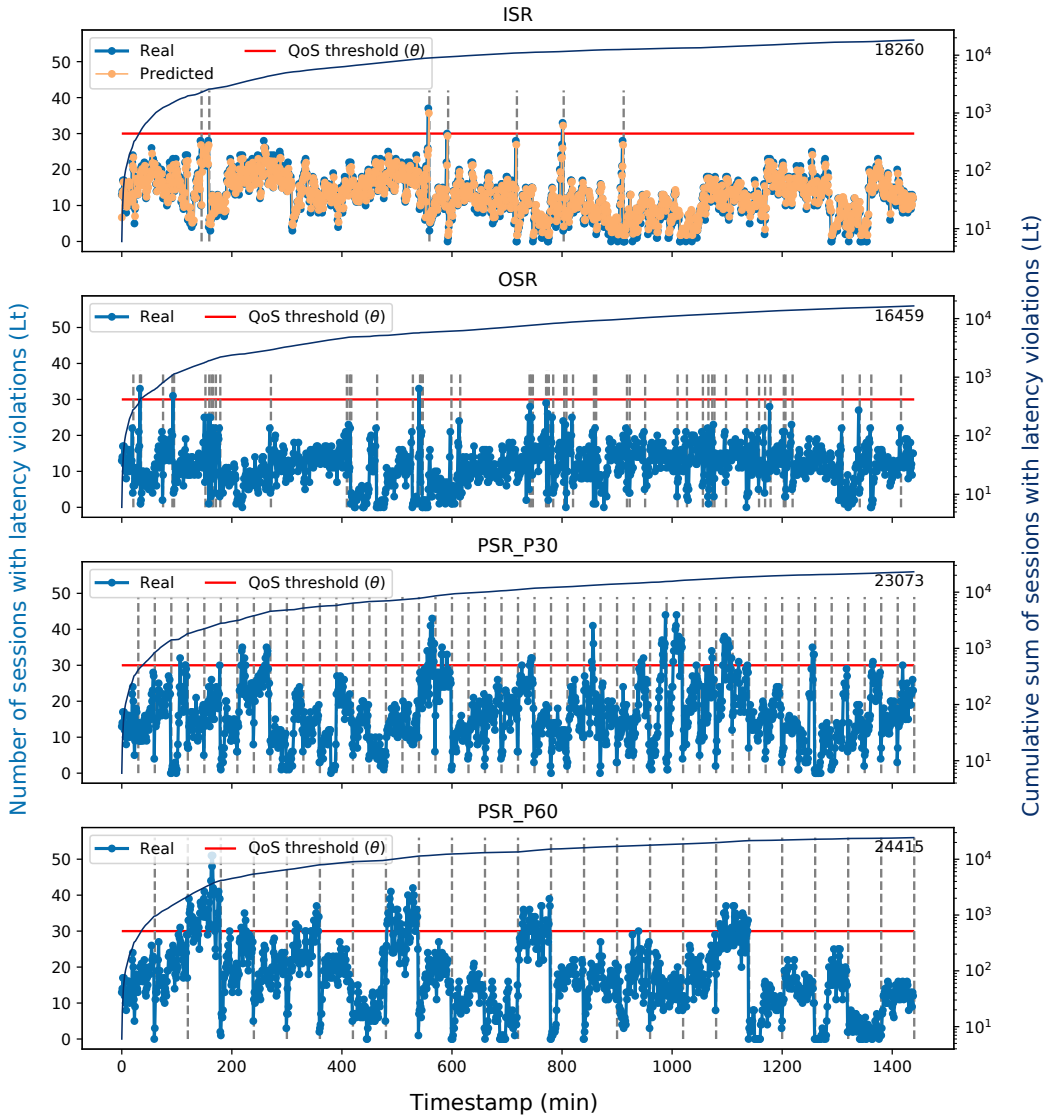


Figure 7.10: Number of sessions with latency violations and their cumulative sum over time.

threshold most of the time, with sporadic exceptions. Conversely, PSR_P30 and PSR_P60 had the worst performance, with a substantial number of sessions with poor QoS over time.

7.4 Conclusion

This chapter investigated the use of ML techniques to automate the execution of UPCR events. An ML-based framework called ISR was conceived to readjust the UPC configuration proactively to avoid poor QoS states due to network variations produced by users' mobility. This framework makes reconfiguration decisions based on QoS values/status predictions and an established QoS threshold.

We assessed the effectiveness of several ML models for forecasting the QoS status/values at the next timestamp accurately. Overall, all the models showed acceptable capabilities. However, the regressors (i.e., BLSTM, MLP, and SVR) provided slightly better results, correctly predicting more observations than the classifiers. Moreover, we observed that the performance of ML algorithms could change drastically based on the problem type (classification or regression). This was the case with the models built upon RF and BLSTM techniques. Conversely, MLP and SVM algorithms showed good performance in addressing classification and regression problems.

Additionally, we investigated the capabilities of the conceived ISR by employing a BLSTM_R model as the framework predictor. Compared to the established benchmarks, the devised ISR mechanism led to substantial improvements in the overall system QoS. In sum, the scheduling showed outstanding performance, executing the fewest UPCR events and keeping the QoS metric under established values for nearly the entire simulation time (i.e., 99.8%).

CHAPTER 8

CONCLUSION AND FUTURE WORK

This doctoral thesis investigated the 5G UPF placement problem (UPP) in MEC ecosystems from diverse perspectives, including static and dynamic scenarios and PDU sessions served by one or multiple UPFs. Several solutions (mathematical models and heuristic-based algorithms) were designed to determine the best UPF placement and service demand mapping configuration to minimize operational and capital expenditures and satisfy 5G service stringent requirements. Furthermore, various scheduling mechanisms were proposed to determine the best reconfiguration time based on the selected QoS metric (i.e., the instantaneous number of sessions with latency violations). Overall, the solutions presented in this work provided substantial cost reductions and QoS satisfaction. Extensive simulation experiments were conducted to validate their feasibility and effectiveness in attaining their design goals.

The following sections summarize the main contributions and findings of the presented thesis and outline some perspectives for future research directions.

8.1 Research Contributions

Chapter 3 discussed the design of UPF placement solutions to equip service providers and network operators with a set of tools to place 5G UPFs efficiently when planning the deployment of new services in MEC environments. Two exact solutions and a heuristic algorithm were proposed, the main target of which is to satisfy 5G service requirements (e.g., latency and reliability) while minimizing their associated deployment and operational costs. These solutions consider several aspects of the system, including user mobility, service latency, and reliability requirements, and network function available capacity. These aspects had not previously been jointly addressed in the literature, at least at their publication moment. The solutions also incorporate reliability constraints to address the placement problem of user plane network functions. Thus, this chapter

provided a strategy capable of providing resilience against multiple failures while reducing the number of backup UPFs. This was possible through the distinction of primary and backup UPF roles and through sharing the capacity of the backup instances among several access nodes assigned to different main UPFs.

Chapter 4 presented a formulation of the dynamic UPF placement and session mapping reconfiguration problem. In this regard, our contributions were two-fold: an ILP model that determines the optimal UPF placement and session mapping setup and a scheduling mechanism that decides the optimal time to trigger the execution of readjustment events.

The exact solution was formulated as a multi-objective optimization problem comprising various cost components related not only to the UPF placement (i.e., UPF deployment, operation, and migration) but also to the system QoS (i.e., network response time and session reassignment). By defining weight factors, network operators and service providers can select and specify the importance of their optimization objectives. This model is a flexible and valuable tool to evaluate, study, and analyze the impact of different optimization terms when reconfiguring the UPF placement. Moreover, it can also be applied in static scenarios by relaxing those terms that depend on previous time samples (i.e., migration and reassignment).

In addition, the proposed scheduling strategy seeks to maintain the QoS under desired levels and reduce the execution of frequent and unnecessary reevaluation events. Thus, it relies on the principles of optimal stopping theory (OST) to determine the optimal time to reconfigure the UPF placement according to a specified QoS tolerance threshold and the number of sessions with latency violations at each time instance. The simulation results showed that this mechanism is an effective, practical, and straightforward approach for managing UPF placement.

Chapter 5 described the UPP in depth by discussing the possibility of splitting UPF functionalities into different instances, which can be chained together as required by the PDU sessions (service type). This UPF specialization enables better load distribution and higher efficiency. However, it also introduces additional complexity to the UPP due to different UPF roles with diverse requirements that need to be chained in a specific order. Therefore, we modeled PDU sessions as service function chain requests (SFCRs), which had different topologies, by contemplating UPF placement specificities such as UPF order and unknown egress point location. In summary, this chapter defined the 5G UPP as a variant of the virtual network function placement and chaining (VNFPC) problem. Within this chapter, we described one exact and two approximated solutions (i.e., a heuristic and a simulated annealing metaheuristic) to tackle the UPF placement and chaining (UPC) problem in static scenarios. These solutions seek to minimize expenditures associated with the number of active candidate nodes, deployed VNFs, and traffic routing and thus consider several aspects of the system, such as available resources (i.e., link bandwidth, VNF and server capacities), service latency requirements, and VNF anti-affinity and order.

The approximated solutions incorporate several mechanisms that significantly improved their performance compared to the baselines. The heuristic algorithm avoids SFCR rejections

and reduces expenditures by prioritizing mapping the most demanding SFCRs and considering the impact of VNF mapping decisions on current and subsequent VNFs that forms the service request. The conceived simulated annealing solution incorporates several strategies, such as restart-stop, variable Markov chain length, and two in-house neighborhood functions, which significantly enhance the solution time and quality. These solutions provided substantial cost reductions while ensuring 5G service stringent requirements (e.g., latency and device density). Furthermore, they provided near-optimal results with significantly lower execution times than the exact solutions did.

In Chapter 6, we further discussed the UPC problem by proposing solutions for dynamic scenarios in which the placement and session mapping readjustment is required to cope with variable network conditions (e.g., traffic demands or user mobility). We described a multi-objective ILP model and a heuristic to solve the problem with the primary goals of minimizing multiple cost components involved in the reconfiguration procedure (e.g., VNF migration and session reassignments) and re-establishing QoS. The conceived heuristic adapts to dynamic scenarios the algorithm presented in Chapter 5. Additionally, it combines partial unmapping of SFCRs with an improvement phase to enhance efficiency. This combination provides high levels of flexibility since these parameters can be set based on the specified optimization objectives, which renders the algorithm suitable for online scenarios with diverse specificities. Furthermore, we designed an OST-based scheduling strategy to determine the UPC reconfiguration time properly. The scheduler makes UPC reevaluation decisions based on three parameters: the instantaneous value of the selected QoS metric, a QoS tolerance threshold, and an expected reconfiguration cost. This mechanism is very efficient in attaining its design goals since it can maintain the selected QoS metric under desired values and reduce the number of readjustment events.

In Chapter 7, we addressed the problem of determining the best time to readjust the UPC configuration using a machine learning (ML) approach. An ML-based framework called intelligent scheduling of the reconfiguration (ISR) was envisioned to execute UPCRs proactively by anticipating events with poor QoS. We studied several ML algorithms to predict either the system's QoS status or its value at a given time. A decision maker determines UPCRs by leveraging the predictor outputs and a pre-established QoS tolerance threshold. The simulation experiments showed the superiority of the conceived ISR strategy compared to OST-based and periodic mechanisms. The ISR reduced the number of reconfiguration events by at least 50% compared to the other schedulers, and it kept the system QoS under the established threshold most of the time, providing similar results to the optimal scheduling of the reconfiguration (OSR).

8.2 Future Work

This section provides research recommendations in consideration of the proposed solutions' scope and limitations. The work presented in this doctoral thesis can be extended in several directions,

as identified below:

Session and service continuity (SSC) mode consideration during UPF placement strategies: Different mechanisms have been introduced in the 5G system to mitigate the effects of user handovers on UPF relocations. For example, two new modes have been proposed for SSC: SSC modes 2 and 3. Unlike SSC mode 1 (traditional mode), in which the PDU session is likely to maintain the same UPF regardless of the radio access technologies (RAT), SSC modes 2 and 3 may trigger UPF relocations. Each mode has advantages and drawbacks; SSC mode 1 avoids frequent relocations but may not be proper for time-sensitive applications, in contrast to modes 2 and 3.

SSC modes 2 and 3 are appropriate in MEC systems in which data processing may be performed close to the users for a quicker response time [173]. Although SSC mode 2 makes more efficient usage of UPF resources during handovers by releasing the PDU session, this mechanism introduces additional delays. Thus, it may not be suitable for services under the URLLC category, where session continuity and low response time are mandatory. In contrast, SSC mode 3 guarantees service continuity but with a higher consumption of resources, as the existing PDU session is maintained in the source UPF until a new one is established. Thus, UPF placement and relocations and their effects on overall costs and QoS depend on the SSC mode configuration.

For instance, SSC mode 1 must keep the anchor UPF (aUPF) during the session lifetime, which avoids the occurrence of relocations. However, this may cause QoS degradation due to higher propagation times between the user access point to the network and the aUPFs. This situation can be alleviated by inserting intermediate UPFs (iUPFs) to reduce the network response time. In addition, SSC modes 2 and 3 require the presence of aUPFs near users' new access points with enough available capacity to satisfy their service demands. Moreover, PDU sessions with SSC mode 3 continue using their assigned resources (link and processing capacity) in the source UPFs until new sessions are established.

In this context, the design of UPF placement and management solutions aware of sessions' SSC mode is mandatory to provide seamless and cost-effective mobility. One approach to address this problem is reserving resources (links and VNF processing capabilities) at target locations. Given the uncertainty of users' future locations, these resources could be shared among a subset of possible PDU sessions, thereby reducing system utilization since not all users are likely to move to the same UPF service areas simultaneously. Moreover, these solutions should be combined with efficient mobility management and prediction techniques to determine and provision users' target access nodes and aUPFs proactively. Otherwise, reserving resources for every possible user location will lead to over-provisioning, as well as a dramatic increase in expenditures for service providers and network operators.

Network slicing (NS)-aware UPF placement: NS is one of the main features of 5G networks. It allows network operators to build multiple isolated virtual networks on a shared

physical network to accommodate various services and applications [174] with diversified requirements. In sum, NS can be defined as an E2E logical network running on top of a physical network that may or may not share resources with other NS while providing isolation and independent control and management. Therefore, our proposed placement solutions for either static or dynamic scenarios should be extended to contemplate the possibility of deploying UPF instances managed by different tenants and supporting multiple slices with different requirements (e.g., eMBB, URLLC and mMTC). For instance, these solutions could contemplate the existence of diverse slices that may cooperate by sharing resources or some common VNF instances used to offload traffic during traffic peaks or VNF failures. This approach could be more efficient in terms of resource utilization and provisioning since it could reduce frequent readjustment events and over-provisioning. Thus, a thorough evaluation of the impact of VNF sharing among slices on resource utilization and QoS for different VNF roles should be conducted.

However, placing and chaining VNF instances in a network-sliced environment differs from traditional placement solutions and imposes additional challenges to the placement problem. In particular, some network slices may require specific placement strategies (optimization objective and constraints), and their performance may affect the system QoS, resource utilization, and overall deployment and operational cost. Moreover, they may serve users with diverse mobility and traffic patterns requiring specific readjustment strategies.

Intelligent UPF placement and management: Further investigation regarding the applicability of ML and artificial intelligence (AI) technologies in different network domains, such as network planning and optimization, is necessary for realizing the potential of 5G and beyond networks. In particular, ML and AI solutions could be applied during planning phases to dimension and place VNF instances better and map PDU sessions optimally according to traffic demands and user mobility pattern forecasts. For example, the most popular access nodes at a given time horizon could be identified by accounting for user mobility patterns. Consequently, QoE could be maximized by placing VNFs closer to these locations and mapping PDU sessions to UPF instances capable of serving the users at current and future locations. Moreover, resource utilization and the system QoS could be improved by predicting how service demand will evolve over time and proactively taking proper actions (e.g., auto-scaling and readjustment events). Moreover, further study of ML solutions, such as reinforcement learning, convolutional networks, and federated learning, to predict the occurrence of service level agreement (SLA) violation events is highly recommended. Additionally, the design of more efficient mechanisms to perform machine learning-related tasks (e.g., feature selection, forecasting solution selection, tuning, and training) is required since the current mechanisms are time-consuming and highly dependent on the available data and problem under study.

Our conceived mechanisms for determining the best reconfiguration time rely on the number of sessions with latency violations. However, we believe that more robust and efficient solutions could be designed by combining this metric with others relevant to network performance (e.g.,

VNF utilization, user-perceived delay, and the number of handovers). In this manner, several aspects could be jointly optimized, such as resource utilization, QoS, and QoE.

Reliability-aware UPF placement and chaining: Although our solutions proposed in Chapters 3 and 4 contemplate service reliability requirements when determining the best UPF placement, they focus on the deployment of backup UPF instances and neglect the consideration of disjointed backup links between access nodes and aUPFs. Moreover, the UPC solutions presented in Chapters 5 and 6 do not contemplate the implementation of any reliability strategy (e.g., backup SFC data path or VNF protection schemes). This was to avoid further increase of their time complexity since additional decision variables and constraints must be defined.

Therefore, the design of reliable UPF placement and chaining solutions is an open research area that should be investigated to improve the user plane's robustness in 5G and beyond networks. The proposed models could be extended to consider redundant transmission on the N3/N6 interface by selecting disjoint data paths between the access nodes and UPFs that serve sessions with reliability requirements. However, this approach does not protect against VNF failures. Thus, the implementation of VNF protection schemes should also be addressed.

Real-world scenario evaluation: A common limitation of the proposed solutions concerns their performance evaluation and the study of dynamic network conditions, such as traffic variation and user mobility in real-world scenarios. This limitation was mainly due to the lack of data sets reflecting user behavior in 5G networks. Moreover, most data sets containing user mobility traces are not sufficiently large to train ML models or obtain the desired user density levels in a medium-scale scenario. Furthermore, some of these data sets do not have information regarding user locations at small or equally spaced sampling times, which hinders the study of users' mobility patterns and, therefore, the design of mobility management strategies.

Likewise, the implementation of the proposed solutions in real scenarios and their performance evaluation should be investigated. These solutions could be integrated into a placement module as part of the management and orchestration (MANO) framework or as an external placement engine accessible through standard APIs. The placement module/engine needs information regarding service characteristics and status, as well as VNF and infrastructure resources (link bandwidth, server capacity) to operate. This information can be collected by different system components, such as the MANO framework (e.g., monitoring and VNF manager modules) and Virtualized Infrastructure Managers (VIM), and can be exposed to the placement solver through an intermediate module. Additionally, the intermediate module communicates and translates the computed placement configuration to the orchestration module.

Nowadays, there is a wide variety of MANO platforms, including OSM [175], ONAP [176], and SONATA [177], some of which include their placement module. For example, OSM, which is open source, has a placement module (OSM PLA), which could be extended to incorporate the conceived heuristics. This approach will guarantee compatibility and integration with the OSM components responsible for VNF deployment and management (e.g., OSM RO and OSM LCM). Another

option could be the development of the placement solver as an external application, which avoids modifications in the source code of the OSM framework and enhances its portability. Moreover, the placement algorithms could also be used by in-house orchestration solutions or directly by VIMs (e.g., Kubernetes), either within their orchestration service or externally. However, aspects such as multi-site, multi-domain, heterogeneity of resources (computing, storage, and networking) and technologies (e.g., VIM), as well as network service modification in dynamic scenarios, must be analyzed.

APPENDIX A

APPENDIX A: PUBLICATIONS

- **I. Leyva-Pupo**, C. Cervelló-Pastor, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic UPF Placement and Chaining Reconfiguration in 5G Networks," *Computer Networks*, vol. 215, p.109200, 2022.

- **I. Leyva-Pupo**, and C. Cervelló-Pastor, "Efficient solutions to the placement and chaining problem of User Plane Functions in 5G networks," *Journal of Network and Computer Applications*, vol. 197, p. 103269, 2022.

- A. Llorens-Carrodeguas, **I. Leyva-Pupo**, C. Cervelló-Pastor, L. Piñeiro, S. Shuaib, "An SDN-Based Solution for Horizontal Auto-Scaling and Load Balancing of Transparent VNF Clusters," *Sensors*, vol. 21, no. 24, Dec 2021.

- **I. Leyva-Pupo**, C. Cervelló-Pastor, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic Scheduling and Optimal Reconfiguration of UPF Placement in 5G Networks," in *Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (ACM MSWIM'20)*, Alicante, Spain, Nov. 2020, Association for Computing Machinery, pp. 103-111, 2020.

- **I. Leyva-Pupo**, C. Cervelló-Pastor and A. Llorens-Carrodeguas, "Optimal Placement of User Plane Functions in 5G Networks," In *17th International Conference on Wired/Wireless Internet Communications, IFIP WWIC 2019*, Wired/Wireless Internet Communications, Lecture Notes in Computer Science, vol 11618. Springer, Cham, 2019.

- A. Llorens-Carrodeguas, C. Cervelló-Pastor and **I. Leyva-Pupo**, "A Data Distribution Service in a Hierarchical SDN Architecture: Implementation and Evaluation," In *28th*

International Conference on Computer Communication and Networks (ICCCN), Valencia, Spain, 2019.

- **I. Leyva-Pupo**, A. Santoyo-González, and C. Cervelló-Pastor, “A Framework for the Joint Placement of Edge Service Infrastructure and User Plane Functions for 5G," *MDPI Sensors*, vol. 19, no. 18, pp. 3975, 2019.

- **I. Leyva-Pupo**, C. Cervelló-Pastor and A. Llorens-Carrodeguas, "A framework for placement and optimization of network functions in 5G," In *XXXIII Simposium Nacional de la Unión Científica Internacional de Radio (URSI)*, Granada, Spain, 2018.

- A. Llorens-Carrodeguas, C. Cervelló-Pastor, **I. Leyva-Pupo**, J. M. López-Soler and J. Navarro-Ortiz, "The Role of Data Distribution Service in Failure-aware SDN Controllers," In *XXXIII Simposium Nacional de la Unión Científica Internacional de Radio (URSI)*, Granada, Spain, 2018.

- **I. Leyva-Pupo**, C. Cervelló-Pastor and A. Llorens-Carrodeguas, "The Resources Placement Problem in a 5G Hierarchical SDN Control Plane," In *Distributed Computing and Artificial Intelligence, Special Sessions, 15th International Conference. DCAI 2018*, Advances in Intelligent Systems and Computing, vol 801. Springer, Cham, 2018.

- A. Llorens-Carrodeguas, C. Cervelló-Pastor and **I. Leyva-Pupo**, "Software Defined Networks and Data Distribution Service as Key Features for the 5G Control Plane," In *Distributed Computing and Artificial Intelligence, Special Sessions, 15th International Conference. DCAI 2018*, Advances in Intelligent Systems and Computing, vol 801. Springer, Cham, 2018.

- A. Llorens-Carrodeguas, C. Cervelló-Pastor, **I. Leyva-Pupo**, J. M. Lopez-Soler, J. Navarro-Ortiz and J. A. Exposito-Arenas, "An architecture for the 5G control plane based on SDN and data distribution service," In *2018 Fifth International Conference on Software Defined Systems (SDS)*, Barcelona, Spain, 2018.

References

[1] D. Wang *et al.*, "5G-Advanced Technology Evolution from a Network Perspective 2.0(2022) ——Towards a New Era of Intelligent Connect X." Available online, URL: https://www-file.huawei.com/-/media/corporate/pdf/news/5g-advanced%20technology%20evolution%20from%20a%20network%20perspective(2022).pdf?la=en (accessed on 2 Nov. 2022), 2022.

[2] Angus Muirhead, "5G: enabling innovation in robotics and automation." Available online, URL: https://www.credit-suisse.com/sg/en/articles/asset-management/5G-enabling-innovation-in-robotics-and-automation-sg-202010.html (accessed on 2 Nov. 2022), 2020.

[3] Gemalto (Thales), "Introducing 5G networks-Characteristics and usages." Available online, URL: http://repositorioiri5g.iri.usp.br/jspui/bitstream/123456789/106/1/tel-5G-networks-QandA.pdf (accessed on 2 Nov. 2022), 2016.

[4] 5G Americas, "5G Network Transformation." Available online, URL: https://www.5gamericas.org/wp-content/uploads/2019/07/5G_Network_Transformation_Final.pdf (accessed on 2 Nov. 2022), 2017.

[5] B. Blanco *et al.*, "Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN," *Computer Standards & Interfaces*, vol. 54, pp. 216–228, 2017.

[6] G. Nencioni, R. G. Garroppo, and R. F. Olimid, "5G Multi-access Edge Computing: Security, Dependability, and Performance," *arXiv preprint arXiv:2107.13374*, 2021.

[7] 3GPP, "System Architecture for the 5G System (5GS); Stage 2 Version 16.4.0," Technical Specification (TS) 23.501, 3rd Generation Partnership Project (3GPP), March 2020.

[8] 3GPP, "Procedures for the 5G System (5GS); Stage 2 Version 16.4.0," Technical Specification (TS) 23.502, 3rd Generation Partnership Project (3GPP), March 2020.

[9] ZTE, "Full-Scenario UPF Deployment." Available online, URL: https://res-www.zte.com.cn/mediares/zte/Files/newsolution/Wireless/CCN/hexinwang/whitepaper/ZTE_Full-Scenario_UPF_Deployment_White_Paper.pdf (accessed on 2 Nov. 2022).

[10] G. S. S. Chalapathi, V. Chamola, A. Vaish, and R. Buyya, "Industrial internet of things (iiot) applications of edge and fog computing: A review and future directions," *Fog/Edge Computing For Security, Privacy, and Applications*, pp. 293–325, 2021.

[11] Y. Li, X. Ma, M. Xu, A. Zhou, Q. Sun, N. Zhang, and S. Wang, "Joint Placement of UPF and Edge Server for 6G Network," *IEEE Internet of Things Journal*, 2021.

[12] P. Fondo-Ferreiro, D. Candal-Ventureira, F. J. González-Castaño, and F. Gil-Castiñeira, "Latency Reduction in Vehicular Sensing Applications by Dynamic 5G User Plane Function Allocation with Session Continuity," *Sensors*, vol. 21, no. 22, p. 7744, 2021.

[13] T. Subramanya, D. Harutyunyan, and R. Riggio, "Machine learning-driven service function chain placement and scaling in MEC-enabled 5G networks," *Computer Networks*, vol. 166, p. 106980, 2020.

[14] D. Harutyunyan, R. Behravesh, and N. Slamnik-Kriještorac, "Cost-efficient Placement and Scaling of 5G Core Network and MEC-enabled Application VNFs," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 241–249, IEEE, 2021.

[15] T. Taleb and A. Ksentini, "Gateway relocation avoidance-aware network function placement in carrier cloud," in *Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems*, pp. 341–346, ACM, 2013.

[16] M. Bagaa, T. Taleb, and A. Ksentini, "Service-aware network function placement for efficient traffic handling in carrier cloud," in *Wireless Communications and Networking Conference (WCNC), 2014 IEEE*, pp. 2402–2407, IEEE, 2014.

[17] T. Taleb, M. Bagaa, and A. Ksentini, "User mobility-aware virtual network function placement for virtual 5G network infrastructure," in *2015 IEEE International Conference on Communications (ICC)*, pp. 3879–3884, IEEE, June 2015.

[18] A. Ksentini, M. Bagaa, and T. Taleb, "On using SDN in 5G: the controller placement problem," in *Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2016.

[19] A. Basta *et al.*, "Towards a cost optimal design for a 5G mobile core network based on SDN and NFV," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1061–1075, 2017.

[20] F. B. Jemaa, G. Pujolle, and M. Pariente, "QoS-aware VNF placement optimization in edge-central carrier cloud architecture," in *2016 IEEE global communications conference (GLOBECOM)*, pp. 1–7, IEEE, 2016.

[21] R. Behravesh, E. Coronado, D. Harutyunyan, and R. Riggio, "Joint user association and VNF placement for latency sensitive applications in 5G networks," in *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, pp. 1–7, IEEE, 2019.

[22] P. Roy, A. Tahsin, S. Sarker, T. Adhikary, M. A. Razzaque, and M. M. Hassan, "User mobility and quality-of-experience aware placement of virtual network functions in 5G," *Computer Communications*, vol. 150, pp. 367–377, 2020.

[23] A. Mouaci, É. Gourdin, I. LjubiĆ, and N. Perrot, "Virtual Network Functions Placement and Routing Problem: Path formulation," in *2020 IFIP Networking Conference (Networking)*, pp. 55–63, IEEE, 2020.

[24] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, and E. Gourdin, "Virtual function placement for service chaining with partial orders and anti-affinity rules," *Networks*, vol. 71, no. 2, pp. 97–106, 2018.

[25] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement and resource optimization in NFV and edge computing enabled networks," *Computer Networks*, vol. 152, pp. 12–24, 2019.

[26] G. Americas, "3GPP Releases 16 & 17 & Beyond." Available online, URL: `https://www.5gamericas.org/wp-content/uploads/2021/01/InDesign-3GPP-Rel-16-17-2021.pdf` (accessed on 2 Nov. 2022), 2021.

[27] "5G subscriptions grow less than expected, but data traffic skyrockets." Available online, URL: `https://on5g.es/en/5g-subscriptions-grow-less-than-expected-but-data-traffic-skyrockets/` (accessed on 2 Nov. 2022), June 2022.

[28] GSMA, "The Mobile Economy 2022." Available online, URL: `https://www.gsma.com/mobileeconomy/wp-content/uploads/2022/02/280222-The-Mobile-Economy-2022.pdf` (accessed on 2 Nov. 2022), 2022.

[29] Ericsson, "Ericsson Mobility Report June 2022." Available online, URL: `https://www.ericsson.com/49d3a0/assets/local/reports-papers/mobility-report/documents/2022/ericsson-mobility-report-june-2022.pdf` (accessed on 2 Nov. 2022), 2022.

[30] A. I. Salameh and M. El Tarhuni, "From 5G to 6G—Challenges, Technologies, and Applications," *Future Internet*, vol. 14, no. 4, p. 117, 2022.

[31] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications surveys & tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[32] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[33] Y. C. Hu *et al.*, "ETSI White Paper #11 Mobile Edge Computing - a key technology towards 5G." Available online, URL: https://www.etsi.org/images/files/etsiwhitepapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf (accessed on 2 Nov. 2022), 2015.

[34] F. Spinelli and V. Mancuso, "Toward enabled industrial verticals in 5G: A survey on MEC-based approaches to provisioning and flexibility," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 596–630, 2020.

[35] V.-G. Nguyen *et al.*, "SDN/NFV-based mobile packet core network architectures: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1567–1602, 2017.

[36] J. Pagé and J.-M. Dricot, "Software-Defined Networking for low-latency 5G core network," in *2016 International Conference on Military Communications and Information Systems (ICMCIS). IEEE*, 2016.

[37] U. C. Kozat *et al.*, "A new control plane for 5G network architecture with a case study on unified handoff, mobility, and routing management," *IEEE Communications Magazine*, vol. 52, no. 11, pp. 76–85, 2014.

[38] S. Jeon, D. Corujo, and R. L. Aguiar, "Virtualised EPC for on-demand mobile traffic offloading in 5G environments," in *Standards for Communications and Networking (CSCN), 2015 IEEE Conference on*, pp. 275–281, IEEE, 2015.

[39] M. R. Sama *et al.*, "Reshaping the mobile core network via function decomposition and network slicing for the 5G era," in *Wireless Communications and Networking Conference (WCNC), 2016 IEEE*, pp. 1–7, IEEE, 2016.

[40] Huawei, "Service-Based Architecture for 5G Core Networks." Available online, URL: https://img.lightreading.com/downloads/Service-Based-Architecture-for-5G-Core-Networks.pdf (accessed on 2 Nov. 2022), 2017.

[41] "Making 5G a Reality." Available online, URL: https://jpn.nec.com/nsp/5g_vision/pdf/wp2018ar.pdf (accessed on 2 Nov. 2022).

[42] N. Alliance, "NGMN Overview on 5G RAN Functional Decomposition." Available online, URL: https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2018/180226_NGMN_RANFSX_D1_V20_Final.pdf (accessed on 2 Nov. 2022), Feb. 2018.

[43] G. Americas, "New Services Applications with 5G Ultra-Reliable Low Latency Communications." Available online, URL: https://www.5gamericas.org/wp-content/uploads/2019/07/5G_Americas_URLLLC_White_Paper_Final__updateJW.pdf (accessed on 2 Nov. 2022), 2018.

[44] S. Rommer, P. Hedman, M. Olsson, L. Frid, S. Sultana, and C. Mulligan, "5G Core Networks: Powering Digitalization," 2019.

[45] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1409–1434, 2018.

[46] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vNF placement at the network edge," in *IEEE INFOCOM 2018- IEEE Conference on Computer Communications*, pp. 693–701, IEEE, 2018.

[47] P. K. Thiruvasagam, A. Chakraborty, and C. S. R. Murthy, "Latency-aware and survivable mapping of VNFs in 5G network edge cloud," in *2021 17th International Conference on the Design of Reliable Communication Networks (DRCN)*, pp. 1–8, IEEE, 2021.

[48] L. Yala, P. A. Frangoudis, and A. Ksentini, "Latency and availability driven VNF placement in a MEC-NFV environment," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2018.

[49] B. Yi, X. Wang, K. Li, M. Huang, *et al.*, "A comprehensive survey of network function virtualization," *Computer Networks*, vol. 133, pp. 212–262, 2018.

[50] S. Demirci and S. Sagiroglu, "Optimal placement of virtual network functions in software defined networks: A survey," *Journal of Network and Computer Applications*, vol. 147, p. 102424, 2019.

[51] J. Sun, Y. Zhang, F. Liu, H. Wang, X. Xu, and Y. Li, "A survey on the placement of virtual network functions," *Journal of Network and Computer Applications*, p. 103361, 2022.

[52] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, pp. 255–260, IEEE, 2015.

[53] K. Kawashima, T. Otoshi, Y. Ohsita, and M. Murata, "Dynamic placement of virtual network functions based on model predictive control," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 1037–1042, IEEE, 2016.

[54] I. Al-Surmi, B. Raddwan, and I. Al-Baltah, "Next Generation Mobile Core Resource Orchestration: Comprehensive Survey, Challenges and Perspectives," *Wireless Personal Communications*, vol. 120, no. 2, pp. 1341–1415, 2021.

[55] A. Alleg, R. Kouah, S. Moussaoui, and T. Ahmed, "Virtual Network Functions Placement and Chaining for real-time applications," in *2017 IEEE 22nd international workshop on computer aided modeling and design of communication links and networks (CAMAD)*, pp. 1–6, IEEE, 2017.

[56] X. Song, X. Zhang, S. Yu, S. Jiao, and Z. Xu, "Resource-efficient virtual network function placement in operator networks," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–7, IEEE, 2017.

[57] Z. Wang, J. Zhang, T. Huang, and Y. Liu, "Service Function Chain Composition, Placement, and Assignment in Data Centers," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1638–1650, 2019.

[58] Y. Liu, J. Pei, P. Hong, and D. Li, "Cost-efficient virtual network function placement and traffic steering," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[59] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On dynamic service function chain deployment and readjustment," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 543–553, 2017.

[60] L. Askari, A. Hmaity, F. Musumeci, and M. Tornatore, "Virtual-network-function placement for dynamic service chaining in metro-area networks," in *2018 International Conference on Optical Network Design and Modeling (ONDM)*, pp. 136–141, IEEE, 2018.

[61] J. Xia, Z. Cai, and M. Xu, "Optimized virtual network functions migration for NFV," in *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 340–346, IEEE, 2016.

[62] N. T. Jahromi, S. Kianpisheh, and R. H. Glitho, "Online VNF placement and chaining for value-added services in content delivery networks," in *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pp. 19–24, IEEE, 2018.

[63] J. Li, W. Shi, P. Yang, and X. Shen, "On dynamic mapping and scheduling of service function chains in SDN/NFV-enabled networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2019.

[64] Y.-T. Chen and W. Liao, "Mobility-aware service function chaining in 5G wireless networks with mobile edge computing," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[65] D. Zhao, G. Sun, D. Liao, S. Xu, and V. Chang, "Mobile-aware service function chain migration in cloud–fog computing," *Future Generation Computer Systems*, vol. 96, pp. 591–604, 2019.

[66] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2179–2192, 2018.

[67] Y. Liu, H. Lu, X. Li, and D. Zhao, "An approach for service function chain reconfiguration in network function virtualization architectures," *IEEE Access*, vol. 7, pp. 147224–147237, 2019.

[68] B. Li, B. Cheng, and J. Chen, "A Multi-Stage Approach for Virtual Network Function Migration and Service Function Chain Reconfiguration in NFV-enabled Networks," in *2020 IEEE International Conference on Web Services (ICWS)*, pp. 207–215, IEEE, 2020.

[69] G. Zheng, A. Tsiopoulos, and V. Friderikos, "Dynamic VNF chains placement for mobile IoT applications," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2019.

[70] L. Wang, M. Dolati, and M. Ghaderi, "CHANGE: Delay-Aware Service Function Chain Orchestration at the Edge," in *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC)*, pp. 19–28, 2021.

[71] Y. Gu, Y. Hu, Y. Ding, J. Lu, and J. Xie, "Elastic virtual network function orchestration policy based on workload prediction," *IEEE Access*, vol. 7, pp. 96868–96878, 2019.

[72] O. A. Wahab, N. Kara, C. Edstrom, and Y. Lemieux, "MAPLE: A machine learning approach for efficient placement and adjustment of virtual network functions," *Journal of Network and Computer Applications*, vol. 142, pp. 37–50, 2019.

[73] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 98–106, IEEE, 2015.

[74] M. C. Luizelli, W. L. da Costa Cordeiro, L. S. Buriol, and L. P. Gaspary, "A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining," *Computer Communications*, vol. 102, pp. 67–77, 2017.

[75] X. Wang, S. Zhang, and Y. Wang, "Cost-Aware and Delay-Constrained Service Function Orchestration in Multi-Data-Center Networks," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, IEEE, 2019.

[76] A. Leivadeas, G. Kesidis, M. Ibnkahla, and I. Lambadaris, "VNF placement optimization at the edge and cloud," *Future Internet*, vol. 11, no. 3, p. 69, 2019.

[77] M. Jalalitabar, Y. Wang, and X. Cao, "Branching-Aware Service Function Placement and Routing in Network Function Virtualization," in *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 1–6, IEEE, 2019.

[78] O. Alhussein, P. T. Do, Q. Ye, J. Li, W. Shi, W. Zhuang, X. Shen, X. Li, and J. Rao, "A virtual network customization framework for multicast services in NFV-enabled core networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1025–1039, 2020.

[79] B. Ren, D. Guo, Y. Shen, G. Tang, and X. Lin, "Embedding service function tree with minimum cost for NFV-enabled multicast," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1085–1097, 2019.

[80] A. Muhammad, I. Sorkhoh, L. Qu, and C. Assi, "Delay-sensitive multi-source multicast resource optimization in NFV-enabled networks: A column generation approach," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 286–300, 2021.

[81] M. Awad, N. Kara, and C. Edstrom, "SLO-aware dynamic self-adaptation of resources," *Future Generation Computer Systems*, vol. 133, pp. 266–280, 2022.

[82] G. Miotto, M. C. Luizelli, W. L. da Costa Cordeiro, and L. P. Gaspary, "Adaptive placement & chaining of virtual network functions with NFV-PEAR," *Journal of Internet Services and Applications*, vol. 10, no. 1, pp. 1–19, 2019.

[83] A. Baumgartner, V. S. Reddy, and T. Bauschert, "Mobile core network virtualization: A model for combined virtual core network function placement and topology optimization," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pp. 1–9, IEEE, Apr. 2015.

[84] A. Baumgartner, V. S. Reddy, and T. Bauschert, "Combined virtual mobile core network function placement and topology optimization with latency bounds," in *Software Defined Networks (EWSDN), 2015 Fourth European Workshop on*, pp. 97–102, IEEE, 2015.

[85] D. B. Oljira, K.-J. Grinnemo, J. Taheri, and A. Brunstrom, "A model for QoS-aware VNF placement and provisioning," in *2017 IEEE conference on network function virtualization and software defined networks (NFV-SDN)*, pp. 1–7, IEEE, 2017.

[86] D. Dietrich, C. Papagianni, P. Papadimitriou, and J. S. Baras, "Network function placement on virtualized cellular cores," in *2017 9th International conference on communication systems and networks (COMSNETS)*, pp. 259–266, IEEE, 2017.

[87] A. Gupta, M. Tomatore, B. Jaumard, and B. Mukherjee, "Virtual-mobile-core placement for metro network," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pp. 308–312, IEEE, 2018.

[88] C. Papagianni, P. Papadimitriou, and J. S. Baras, "Rethinking service chain embedding for cellular network slicing," in *2018 IFIP Networking Conference (IFIP Networking) and Workshops*, pp. 1–9, IEEE, 2018.

[89] M. Bagaa *et al.*, "Efficient virtual evolved packet core deployment across multiple cloud domains," in *Wireless Communications and Networking Conference (WCNC), 2018 IEEE*, pp. 1–6, IEEE, 2018.

[90] A. Patel, M. Vutukuru, and D. Krishnaswamy, "Mobility-aware VNF placement in the LTE EPC," in *2017 IEEE conference on network function virtualization and software defined networks (NFV-SDN)*, pp. 1–7, IEEE, 2017.

[91] A. Marotta and A. Kassler, "A power efficient and robust Virtual Network Functions placement problem," in *Teletraffic Congress (ITC 28), 2016 28th International*, vol. 1, pp. 331–339, IEEE, 2016.

[92] J. Prados-Garzon, A. Laghrissi, M. Bagaa, and T. Taleb, "A queuing based dynamic auto scaling algorithm for the LTE EPC control plane," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2018.

[93] C. H. T. Arteaga, F. B. Anacona, K. T. T. Ortega, and O. M. C. Rendon, "A scaling mechanism for an evolved packet core based on network functions virtualization," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 779–792, 2019.

[94] D. Basu, A. Jain, R. Datta, and U. Ghosh, "Optimized controller placement for soft handover in virtualized 5G network," in *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1–8, IEEE, 2020.

[95] J. Oueis, R. Stanica, and F. Valois, "Virtualized Local Core Network Functions Placement in Mobile Networks," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2019.

[96] W. Kiess and A. Khan, "Centralized vs. distributed: On the placement of gateway functionality in 5G cellular networks," in *2014 IEEE Global Communications Conference*, pp. 4788–4793, IEEE, 2014.

[97] A. Laghrissi, S. Retal, and A. Idrissi, "Modeling and optimization of the network functions placement using constraint programming," in *Proceedings of the International Conference on Big Data and Advanced Wireless Technologies*, pp. 1–8, 2016.

[98] F. Z. Yousaf, J. Lessmann, P. Loureiro, and S. Schmid, "SoftEPC—Dynamic instantiation of mobile core network entities for efficient resource utilization," in *2013 IEEE International Conference on Communications (ICC)*, pp. 3602–3606, IEEE, 2013.

[99] S. Patni, A. Hegde, and K. M. Sivalingam, "Load Balancing Techniques for Dynamic Gateway Selection in LTE Wireless Networks," *EAI Endorsed Transactions on Mobile Communications and Applications*, vol. 2, no. 7, 2016.

[100] M. Bagaa, T. Taleb, A. Laghrissi, A. Ksentini, and H. Flinck, "Coalitional game for the creation of efficient virtual core network slices in 5G mobile systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 469–484, 2018.

[101] C. Li, J. Li, Y. Li, and Z. Han, "Bayesian coalition formation game for virtual 5G core network functions," *IEEE Access*, vol. 7, pp. 29805–29817, 2019.

[102] L. Ma, X. Wen, L. Wang, Z. Lu, and R. Knopp, "An SDN/NFV based framework for management and deployment of service based 5G core network," *China Communications*, vol. 15, no. 10, pp. 86–98, 2018.

[103] T.-X. Do and Y. Kim, "State Management Function Placement for Service-Based 5G Mobile Core Architecture," *Mobile Networks and Applications*, pp. 1–13, Oct. 2018.

[104] T.-X. Do and Y. Kim, "Latency-aware Placement for State Management Functions in Service-based 5G Mobile Core Network," in *2018 IEEE Seventh International Conference on Communications and Electronics (ICCE)*, pp. 102–106, IEEE, July 2018.

[105] S. S. Shinde, D. Marabissi, and D. Tarchi, "A network operator-biased approach for multi-service network function placement in a 5G network slicing architecture," *Computer Networks*, vol. 201, p. 108598, 2021.

[106] D. Harutyunyan, N. Shahriar, R. Boutaba, and R. Riggio, "Latency and mobility-aware service function chain placement in 5G networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 5, pp. 1967–1709, 2020.

[107] J. Costa-Requena, A. Poutanen, S. Vural, G. Kamel, C. Clark, and S. K. Roy, "SDN-based UPF for mobile backhaul network slicing," in *2018 European Conference on Networks and Communications (EuCNC)*, pp. 48–53, IEEE, 2018.

[108] W.-E. Chen and C. H. Liu, "High-performance user plane function (UPF) for the next generation core networks," *IET Networks*, vol. 9, no. 6, pp. 284–289, 2020.

[109] U. Fattore, M. Liebsch, and C. J. Bernardos, "UPFlight: An enabler for Avionic MEC in a drone-extended 5G mobile network," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1–7, IEEE, 2020.

[110] D. Barach, L. Linguaglossa, D. Marion, P. Pfister, S. Pontarelli, and D. Rossi, "High-speed software data plane via vectorized packet processing," *IEEE Communications Magazine*, vol. 56, no. 12, pp. 97–103, 2018.

[111] K. Gökarslan, Y. S. Sandal, and T. Tugcu, "Towards a URLLC-Aware Programmable Data Path with P4 for Industrial 5G Networks," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, IEEE, 2021.

[112] F. Paolucci, D. Scano, F. Cugini, A. Sgambelluri, L. Valcarenghi, C. Cavazzoni, G. Ferraris, and P. Castoldi, "User Plane Function Offloading in P4 switches for enhanced 5G Mobile Edge Computing," in *2021 17th International Conference on the Design of Reliable Communication Networks (DRCN)*, pp. 1–3, IEEE, 2021.

[113] R. MacDavid, C. Cascone, P. Lin, B. Padmanabhan, A. Thakur, L. Peterson, J. Rexford, and O. Sunay, "A P4-based 5G User Plane Function," in *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*, pp. 162–168, 2021.

[114] C. Ge, D. Lake, N. Wang, Y. Rahulan, and R. Tafazolli, "Context-Aware Service Chaining Framework for Over-the-Top Applications in 5G Networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops*, pp. 50–55, IEEE, 2019.

[115] V.-M. Alevizaki, M. Anastasopoulos, A. Tzanakaki, D. Simeonidou, and U. Bristiol, "Dynamic Selection of User Plane Function in 5G Environments," in *25th International Conference on Optical Network Design and Modeling, ONDM 2021*, IEEE, 2021.

[116] S. Peters and M. A. Khan, "Anticipatory User Plane Management for 5G," in *2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2)*, pp. 9–15, IEEE, 2018.

[117] S. Peters and M. A. Khan, "Anticipatory Session Management and User Plane Function Placement for AI-Driven Beyond 5G Networks," *Procedia Computer Science*, vol. 160, pp. 214–223, 2019.

[118] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[119] E.-G. Talbi, *Metaheuristics: from design to implementation*.
John Wiley & Sons, 2009.

[120] H. Eiselt and C.-L. Sandblom, "Heuristic algorithms," in *Integer Programming and Network Models*, pp. 229–258, Springer, 2000.

[121] M. Abdel-Basset, L. Abdel-Fatah, and A. Sangaiah, "Chapter 10-metaheuristic algorithms: a comprehensive review. Computational intelligence for multimedia big data on the cloud with engineering applications," 2018.

[122] T. S. Ferguson, "Optimal Stopping and Applications." Available online, URL: `https://www.math.ucla.edu/~tom/Stopping/Contents.html` (accessed on 2 Nov. 2022), 2006.

[123] C. Anagnostopoulos and K. Kolomvatsos, "An intelligent, time-optimized monitoring scheme for edge nodes," *Journal of Network and Computer Applications*, vol. 148, p. 102458, 2019.

[124] J. Hu, G. Wang, X. Xu, and Y. Lu, "Study on Dynamic Service Migration Strategy with Energy Optimization in Mobile Edge Computing," *Mobile Information Systems*, vol. 2019, pp. 1–12, 2019.

[125] H. Wu, G. Wang, X. Xu, and J. Hu, "A cache placement strategy for energy savings in CCN," in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pp. 788–795, IEEE, 2018.

[126] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Communications*, vol. 24, no. 2, pp. 98–105, 2016.

[127] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to Pareto-optimal wireless networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1472–1514, 2020.

[128] U. Fattore, M. Liebsch, B. Brik, and A. Ksentini, "AutoMEC: LSTM-based user mobility prediction for service management in distributed MEC resources," in *Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 155–159, 2020.

[129] Z. Zaman, S. Rahman, and M. Naznin, "Novel approaches for VNF requirement prediction using DNN and LSTM," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2019.

[130] S. Schneider, N. P. Satheeschandran, M. Peuster, and H. Karl, "Machine learning for dynamic resource allocation in network function virtualization," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pp. 122–130, IEEE, 2020.

[131] 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.

[132] J. Bendriss, I. G. B. Yahia, P. Chemouil, and D. Zeghlache, "AI for SLA management in programmable networks," in *DRCN 2017-Design of Reliable Communication Networks; 13th International Conference*, pp. 1–8, VDE, 2017.

[133] N. Jalodia, M. Taneja, and A. Davy, "A Deep Neural Network-Based Multi-Label Classifier for SLA Violation Prediction in a Latency Sensitive NFV Application," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 2469–2493, 2021.

[134] Huawei Technologies Co., "5G Network Architecture A High-Level Perspective." Available online, URL: `https://www-file.huawei.com/-/media/corporate/pdf/mbb/5g_nework_architecture_whitepaper_en.pdf?la=en` (accessed on 2 Nov. 2022), 2016.

[135] Z. Ulukan and E. Demircioglu, "A survey of discrete facility location problems," *International Journal of Social Behavioral, Educational, Economic, Business and Industrial Engineering*, vol. 9, no. 7, pp. 2487–2492, 2015.

[136] A. B. Arabani and R. Z. Farahani, "Facility location dynamics: An overview of classifications and applications," *Computers & Industrial Engineering*, vol. 62, pp. 408–420, Feb. 2012.

[137] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, "Capacity-aware and delay-guaranteed resilient controller placement for software-defined WANs," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 991–1005, 2018.

[138] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware controller placement for software-defined networks," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pp. 672–675, IEEE, 2013.

[139] A. K. Singh, N. Kumar, and S. Srivastava, "PSO and TLBO based reliable placement of controllers in SDN," *International Journal of Computer Network and Information Security*, vol. 11, no. 2, pp. 36–42, 2019.

[140] Y. Bejerano, N. Immorlica, J. S. Naor, and M. Smith, "Efficient location area planning for personal communication systems," in *Proceedings of the 9th annual international conference on Mobile computing and networking*, pp. 109–121, ACM, 2003.

[141] N. Alliance, "Perspectives on Vertical Industries and Implications for 5G by NGMN Alliance." Available online, URL: `https://www.ngmn.org/fileadmin/user_upload/160922_NGMN_-_Perspectives_on_Vertical_Industries_and_Implications_for_5G_final.pdf` (accessed on 2 Nov. 2022), 2016.

[142] I. Parvez, A. Rahmati, I. Guvenc, A. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3098–3130, 2018.

[143] A. Abdulghaffar, A. Mahmoud, M. Abu-Amara, and T. Sheltami, "Modeling and evaluation of software defined networking based 5G core network architecture," *IEEE Access*, vol. 9, pp. 10179–10198, 2021.

[144] A. Santoyo-González and C. Cervelló-Pastor, "Latency-aware cost optimization of the service infrastructure placement in 5G networks," *Journal of Network and Computer Applications*, vol. 114, pp. 29–37, 2018.

[145] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola, *Pyomo–optimization modeling in python*, vol. 67.
Springer Science & Business Media, second ed., 2017.

[146] Gurobi Optimization, LLC, "Gurobi optimizer." Available online, URL: `https://www.gurobi.com/` (accessed on 3 Nov. 2022), 2021.

[147] M. Tanha, D. Sajjadi, and J. Pan, "Enduring Node Failures through Resilient Controller Placement for Software Defined Networks," in *Global Communications Conference (GLOBECOM), 2016 IEEE*, pp. 1–7, IEEE, 2016.

[148] J. Yan, S. Bi, and Y.-J. A. Zhang, "Optimal Model Placement and Online Model Splitting for Device-Edge Co-Inference," *arXiv preprint arXiv:2105.13618*, 2021.

[149] F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni, "Citymob: a mobility model pattern generator for VANETs," in *ICC Workshops-2008 IEEE International Conference on Communications Workshops*, pp. 370–374, IEEE, 2008.

[150] M. Ghaznavi, N. Shahriar, R. Ahmed, and R. Boutaba, "Service function chaining simplified," *arXiv preprint arXiv:1601.00751*, 2016.

[151] F. Y. Vincent, A. P. Redi, Y. A. Hidayat, and O. J. Wibowo, "A simulated annealing heuristic for the hybrid vehicle routing problem," *Applied Soft Computing*, vol. 53, pp. 119–132, 2017.

[152] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[153] A. Franzin and T. Stützle, "Revisiting simulated annealing: A component-based analysis," *Computers & operations research*, vol. 104, pp. 191–206, 2019.

[154] N. Azizi and S. Zolfaghari, "Adaptive temperature control for simulated annealing: a comparative study," *Computers & Operations Research*, vol. 31, no. 14, pp. 2439–2451, 2004.

[155] Y.-J. Jeon, J.-C. Kim, J.-O. Kim, J.-R. Shin, and K. Y. Lee, "An efficient simulated annealing algorithm for network reconfiguration in large-scale distribution systems," *IEEE Transactions on Power Delivery*, vol. 17, no. 4, pp. 1070–1078, 2002.

[156] S.-h. Zhan, J. Lin, Z.-j. Zhang, and Y.-w. Zhong, "List-based simulated annealing algorithm for traveling salesman problem," *Computational intelligence and neuroscience*, vol. 2016, pp. 1–12, 2016.

[157] M. M. Atiqullah and S. Rao, "Tuned annealing for optimization," in *International Conference on Computational Science*, pp. 669–679, Springer, 2001.

[158] L. Wang, R. Cai, M. Lin, and Y. Zhong, "Enhanced List-Based Simulated Annealing Algorithm for Large-Scale Traveling Salesman Problem," *IEEE Access*, vol. 7, pp. 144366–144380, 2019.

[159] F. Martinez-Rios and J. Frausto-Solis, "A simulated annealing algorithm for the satisfiability problem using dynamic Markov chains with linear regression equilibrium," *Simulated Annealing: Advances, Applications and Hybridizations*, p. 21, 2012.

[160] F. Martinez-Rios and J. Frausto-Solis, "An hybrid simulated annealing threshold accepting algorithm for satisfiability problems using dynamically cooling schemes," *WSEAS Transaction on computers*, vol. 7, pp. 374–386, 2008.

[161] X. Tian, J. Yan, Y. Yang, C. Xiao, and Q. Zhou, "Parameter identification of a nonlinear model using an improved version of simulated annealing," *International Journal of Distributed Sensor Networks*, vol. 15, no. 2, p. 1550147719832788, 2019.

[162] E. h. ADDOU, A. Serghini, and E. B. Mermri, "A hybrid algorithm based on simulated annealing and tabu search for k-minimum spanning tree problems," in *2019 5th International Conference on Optimization and Applications (ICOA)*, pp. 1–6, 2019.

[163] T. Yamada, B. E. Rosen, and R. Nakano, "A simulated annealing approach to job shop scheduling using critical block transition operators," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 7, pp. 4687–4692, IEEE, 1994.

[164] S.-W. Lin and F. Y. Vincent, "A simulated annealing heuristic for the multiconstraint team orienteering problem with multiple time windows," *Applied Soft Computing*, vol. 37, pp. 632–642, 2015.

[165] Y. Nakakuki and N. Sadeh, "Increasing the efficiency of simulated annealing search by learning to recognize (un) promising runs," in *AAAI*, pp. 1316–1322, 1994.

[166] N. Alliance, "Definition of the testing framework for the NGMN 5G pre-commercial networks trails version 3." Available online, URL: https://ngmn.org/wp-content/uploads/Publications/2019/190802_NGMN-PreCommTrials_Framework_definition_v3.0.pdf (accessed on 2 Nov. 2022), 2019.

[167] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[168] V. Jakkula, "Tutorial on support vector machine (svm)," *School of EECS, Washington State University*, vol. 37, no. 2.5, p. 3, 2006.

[169] H. Jmila, M. I. Khedher, and M. A. E. Yacoubi, "Estimating VNF resource requirements using machine learning techniques," in *International conference on neural information processing*, pp. 883–892, Springer, 2017.

[170] F. Zhang and L. J. O'Donnell, "Support vector regression," in *Machine Learning*, pp. 123–140, Elsevier, 2020.

[171] Y. Imrana, Y. Xiang, L. Ali, and Z. Abdul-Rauf, "A bidirectional LSTM deep learning approach for intrusion detection," *Expert Systems with Applications*, vol. 185, p. 115524, 2021.

[172] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[173] S. Faccin, H. Zisimopoulos, and H. Cheng, "Mechanism to enable optimized user plane anchoring for minimization of user plane relocation due to user equipment mobility," Oct. 15 2019.
US Patent 10,448,239.

[174] A. N. Toosi, R. Mahmud, Q. Chi, and R. Buyya, "Management and orchestration of network slices in 5G, fog, edge and clouds," *Fog and Edge Computing: Principles and Paradigms*, vol. 8, pp. 79–96, 2019.

[175] OSM, "Open Source MANO." Available online, URL: https://osm.etsi.org/ (accessed on 02 November 2022).

[176] ONAP, "Open Network Automation Platform." Available online, URL: https://www.onap.org/ (accessed on 02 November 2022).

[177] S. Project, "Sonata nfv platform." Available online, URL: https://www.sonata-nfv.eu/ (accessed on 02 November 2022).