UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

Doctoral thesis

Resource allocation and NFV placement in resource constrained MEC-enabled 5G-Networks

Advisor
Prof. Fabrizio Granelli

Author
Riccardo Fedrizzi

2023 - 35° Cycle

Abstract

The fifth-generation (5G) of mobile communication networks are expected to support a large number of vertical industries requiring services with diverging requirements. To accommodate this, mobile networks are undergoing a significant transformation to enable a variety of services to coexist on the same infrastructure through network slicing. Additionally, the introduction of distributed user-plane and multi-access edge computing (MEC) technology allows the deployment of virtualised applications close to the network edge.

The first part of this dissertation focuses on end-to-end network slice provisioning for various vertical industries with different service requirements. Two slice provisioning strategies are explored, by formulating a mixed integer linear programming (MILP) problem. Further, a genetic algorithm (GA)-based approach is proposed with the aim to improve search-space exploration. Simulation results show that the proposed approach is effective in providing near-optimal solutions while drastically reducing computational complexity.

In a later stage, the study focuses on building a measurement-based digital twin (DT) for the highly heterogeneous MEC ecosystem. The DT operates as an intermediate and collaborative layer, enabling the orchestration layer to better understand network behavior before making changes to the physical network. Assisted by proper AI/ML solutions, the DT is envisioned to play a crucial role in automated network management. The study utilizes an emulated and physical test-bed to gather network key performance indicators (KPIs) and demonstrates the potential of graph neural network (GNN) in enabling closed loop automation with the help of DT. These findings offer a foundation for future research in the area of DT models and carbon footprint-aware orchestration.

Keywords
5G, MEC, Network Slicing, VNE, Resource Allocation, Optimization, Machine Learning, Network Emulation

Activity Report

Research/Study Activities

During my doctoral program, I participated and actively contributed to various research projects funded by the EU and ESA. Additionally, I participated in the initial stages of developing project proposals. Here is an overview of the activities I carried out during this time:

- Contributed significantly to the proposal preparation of the ESA-funded DINOS5G project as the main contact and contributor from FBK.

- Participated in the research activities of the DINOS5G project as the main contact from FBK. Specifically, my contribution focused on the definition and development of the "Edge Node" which integrates satellite and 5G terrestrial networks to support smart maintenance and monitoring applications for railways.

- Collaborated on various research activities of the EU funded 5G-CARMEN project. As part of my contribution, I co-authored several project deliverables and actively participated in multiple project meetings. The primary research focus was on the orchestration and placement of virtual network functions.

- Participation in the research activities of the EU funded project 5G-ESSENSE as the main contact from FBK side.

Regarding the study activity, I participated to several courses including: "Research Methodology", "Academic Writing for the Sciences and Engineerings", "Computing in Communication Networks", "Answer Set Programming for Knowledge Representation" and "From research to business: a technology transfer approach".

List of Publications

The following peer-reviewed conference and journal papers were published within the timeframe of the doctoral program:

1. D. Harutyunyan, R. Fedrizzi, N. Shahriar, R. Boutaba and R. Riggio, "Orchestrating End-to-end Slices in 5G Networks," 2019 15th International Conference on Network and Service Management (CNSM), Halifax, NS, Canada, 2019
2. M. Centenaro, R. Fedrizzi and L. Vangelista, "Why Is Network Reselection an Issue for Cross-Border Vehicular Applications?," 2020 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE), Turin, Italy, 2020, pp. 1-6
3. R. Fedrizzi, R. Behravesh, C. Costa, and F. Granelli. "An adaptive GA-based slice provisioning method for vertical industries in 5G and beyond networks". Computer Networks, 218:109397, 2022.
4. R. Fedrizzi, C. E. Costa and F. Granelli, "A Framework to define a Measurement-Based Model of MEC Nodes in the 5G System," in IEEE Networking Letters, doi: 10.1109/LNET.2023.3262760.
5. R. Fedrizzi, A. Bellin, C. E. Costa and F. Granelli, "Building the Digital Twin of a MEC node: a Data Driven Approach", 2323 IEEE NetSoft Conference and Workshops (NetSoft), Madrid, Spain, 2023, (accepted for publication).

The research work that forms the basis of this thesis includes publications 1, 3, 4, and 5, as well as latest achievements which are in the process of being prepared for publication.
Apart from the aforementioned papers, the following publications were released prior to the official commencement of the doctoral program.

1. R. Ferrus, O. Sallent, T. Ahmed and R. Fedrizzi, "Towards SDN/NFV-enabled satellite ground segment systems: End-to-End Traffic Engineering use case," 2017 IEEE International Conference on Communications Workshops (ICC Workshops), Paris, France, 2017, pp. 888-893.

2. T. Ahmed,R.Ferrus, R.Fedrizzi, O. Sallent, N. Kuhn, E. Dubois, P. Gelard, "Satellite Gateway Diversity in SDN/NFV-enabled satellite ground segment systems," 2017 IEEE International Conference on Communications Workshops (ICC Workshops), Paris, France, 2017, pp. 882-887.

3. T. Ahmed, R. Ferrus, R. Fedrizzi and O. Sallent, "Towards SDN/NFV-enabled satellite ground segment systems: Bandwidth on Demand use case," 2017 IEEE International Conference on Communications Workshops (ICC Workshops), Paris, France, 2017, pp. 894-899.

4. Antonio Francescon, Giovanni Baggio, Riccardo Fedrizzi, Ramon Ferrus, ImenGrida Ben Yahia, Roberto Riggio, "X–MANO: Cross–domain Management and Orchestration of Network Services", in Proc. of IEEE NetSoft 2017, Bologna, Italy

5. Antonio Francescon, Giovanni Baggio, Riccardo Fedrizzi, Enrico Orsini, Roberto Riggio, "X-MANO: An Open-Source Platform for Cross–domain Management and Orchestration", in Proc. of IEEE OSSN 2017, Bologna, Italy

6. Fedrizzi, Riccardo, et al. "A novel geometric handover model for aerial 4G networks with WiFi-based X2 interface." Transactions on Emerging Telecommunications Technologies 28.3 (2017).

7. R. Fedrizzi, L. Goratti, T. Rasheed and S. Kandeepan, "A heuristic approach to mobility robustness in 4G LTE public safety networks," 2016 IEEE Wireless Communications and Networking Conference, Doha, 2016, pp. 1-6.

8. Rasheed, T., R. Fedrizzi, L. Goratti, and K. M. Gomez, "On the Feasibility of Handover over WiFi Backhaul in LTE-based Aerial-Terrestrial Networks", IEEE Wireless Communications and Networking Conference (WCNC): IEEE, April, 2014.

9. Goratti, L., R. Fedrizzi, T. Rasheed, R. Riggio, and K. M. Gomez, "VCell: Going Beyond the Cell Abstraction in 5G Mobile Networks",

IEEE / IFIP International Workshop on SDN Management and Orchestration (part of IEEE NOMS 2014 Conference), May, 2014.

10. Goratti, L., K. M. Gomez, R. Fedrizzi, and T. Rasheed, "A Novel Device-to-Device Communication Protocol for Public Safety Applications", D2D workshop at GLOBECOM, Atlanta-USA, IEEE, December, 2013.

11. R. Fedrizzi and T. Rasheed, "Cooperative Short Range Routing for Energy Savings in Multi-Interface Wireless Networks," 2013 IEEE 77th Vehicular Technology Conference (VTC Spring), Dresden, 2013, pp. 1-5.

12. Kandeepan, S.; Jayaweera, S.K.; Fedrizzi, R.; , "Power-Trading in Wireless Communications: A Cooperative Networking Business Model", Wireless Communications, IEEE Transactions on , vol.11, no.5, pp.1872-1880, May 2012.

13. Fedrizzi, R., K. M. Gomez, K. Sithamparanathan, T. Rasheed, and V. Saradhi, "Energy Aware Routing in Heterogeneous Multi-Hop Wireless Networks", Greenets Workshop 2012, 2012.

14. Riccardo Fedrizzi, Karina Mabell Gomez, SithamparanathanKandeepan, Tinku Rasheed, V. S. Chava, "Energy Aware Routing in Heterogeneous Multi-Hop Wireless Networks", IEEE ICC 2012.

15. C. V. Saradhi, R. Fedrizzi, A. Zanardi, E. Salvadori, G. Galimberti, A. Tanzi, G. Martinelli, O. Gerstel, "Regenerator Sites Selection Based on Multiple Paths Considering Impairments and Protection Requirements", NOC/OC&I, Jul 2011.

16. C. V. Saradhi, A. Zanardi, R. Fedrizzi, E. Salvadori, G. Galimberti, A. Tanzi, G. Martinelli, O. Gerstel, "A framework for regenerator site selection based on multiple paths," 2010 Conference on Optical Fiber Communication (OFC/NFOEC), collocated National Fiber Optic Engineers Conference, San Diego, CA, 2010, pp. 1-3.

17. C. V. Saradhi, S. Zaks, R. Fedrizzi, A. Zanardi and E. Salvadori, "Practical and deployment issues to be considered in regenerator placement and operation of translucent optical networks," 2010 12th International Conference on Transparent Optical Networks, Munich, 2010, pp. 1-4.

18. C. V. Saradhi, R. Fedrizzi, A. Zanardi, E. Salvadori, G. Galimberti, A. Tanzi, G. Martinelli, O. Gerstel, "Traffic Independent Heuristics for Regenerator Site Selection for Providing Any-to-Any Optical Connectivity", Proc. of IEEE/OSA OFC, Mar 2010.

19. D'Orazio, Leandro; Sacchi, Claudio; Fedrizzi, Riccardo; De Natale, Francesco G. B., "An Adaptive Minimum-BER Approach for Multi-User Detection in STBC-MIMO MC-CDMA Systems", Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE , vol., no., pp.3427-3431, 26-30 Nov. 2007

20. C. Sacchi, M. Donelli, L. D'Orazio, R. Fedrizzi, and F. De Natale, "A genetic algorithm-based MMSE receiver for MC-CDMA systems transmitting over time-varying mobile channels", Electronics Letters, accepted on January 9, 2007, in press. Publication forecast in February 2007.

21. C. Sacchi, L. D'Orazio, M. Donelli, R. Fedrizzi, F.G.B. De Natale, "A genetic algorithm-assisted semi-adaptive MMSE multi-user detection for MC-CDMA mobile communication systems", Proc. of the 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'06), Helsinki (SF), September 11-14, 2006.

Contents

List of Tables

List of Figures

Acronyms

4G	fourth-generation
5G	fifth-generation
5GC	5G core
AGA	Adaptive GA
AI	artificial intelligence
AMF	access & mobility management function
APP	application
BH	backhaul
CP	control plane
CUPS	control plane and user plane separation
DC	data center
DNN	Data Network Name
DT	digital twin
E2E	end-to-end
eMBB	enhanced mobile broadband
FH	fronthaul
GA	genetic algorithm
GCN	graph convolutional network
gNB	gNodeB
GNN	graph neural network
IMT	International Mobile Telecommunications
InP	infrustructure provider
ITU-R	International Telecommunication Union - Radiocommunication Sector
KNN	k-nearest neighbors
KPI	key performance indicator

LTE	long-term evolution
MC	multi-connectivity
MEC	multi-access edge computing
MILP	mixed integer linear programming
ML	machine learning
MLP	multilayer perceptron
mMTC	massive machine-type communication
MNO	mobile network operator
MSE	mean squared error
MVNO	mobile virtual network operator
NFV	network function virtualization
NR	new radio
NRMSE	normalised root mean squared error
NSI	network slice instance
O-RAN	open radio access network (RAN)
PF	polynomial fitting
PGW	packet gateway
PT	physical twin
QoS	quality of service
RAN	radio access network
RL	reinforcement learning
RRC	radio resource control
SBA	service-based architecture
SDN	software defined network
SFC	service function chain
SMF	session management function
SVR	support vector regression
TTI	transmission time interval
UE	user equipment
UP	user plane
UPF	user plane function
URLLC	ultra-reliable low-latency communication
vEPC	virtualised evolved packet core
VNE	virtual network embedding
VNF	virtualised network function

Chapter 1

Introduction

The ever-evolving telecommunication technologies are fueling the growth of service coverage and capacity, leading to the identification of new use cases and applications. However, many of these applications, including virtual reality, augmented reality, autonomous driving, and internet of things, come with stringent network requirements, which increases the complexity of network management and pushes the limits of available technologies. As a result, the telecommunication industry is constantly challenged to keep up with the demands of these new applications while ensuring efficient and reliable network performance.

In this context, this introduction chapter serves to contextualize the motivations and objectives of this study, and to outline the methodology that was followed to achieve those objectives. Additionally, the chapter highlights the contributions that the study makes to the field and provides an overview of the thesis structure.

1.1 Motivations and objectives

Mobile communication systems have lately witnessed an increasing demand in supporting a highly diverse range of service types. The necessity to support such a broad range of services calls a paradigm shift in mobile network design, moving from a pure connectivity provisioning approach to a service-oriented one.

The International Telecommunication Union - Radiocommunication Sector (ITU-R) is a specialized agency that allocates global radio spectrum and develops technical standards to ensure seamless connectivity in telecommunication networks. With the aim of targeting the new generation of International Mobile Telecommunications (IMT), the ITU-R has defined high-level specifications [1]. As shown in Figure 1.1, the demands of IMT-2020 have been classified by the ITU-R into three broad categories: enhanced mobile broadband (eMBB), massive machine-type communication (mMTC) and ultra-reliable low-latency communication (URLLC). These categories have different requirements. The automotive industry is an example of the diverging requirements that are expected to be addressed by 5G and beyond mobile networks. For instance, while automated driving assistance requires low user-plane latency (between $1 - 10$ ms) and extremely high reliability, in-car infotainment and remote diagnostics have high data rate requirements (up to 100 MB/s) and massive connectivity provisioning, respectively [2].

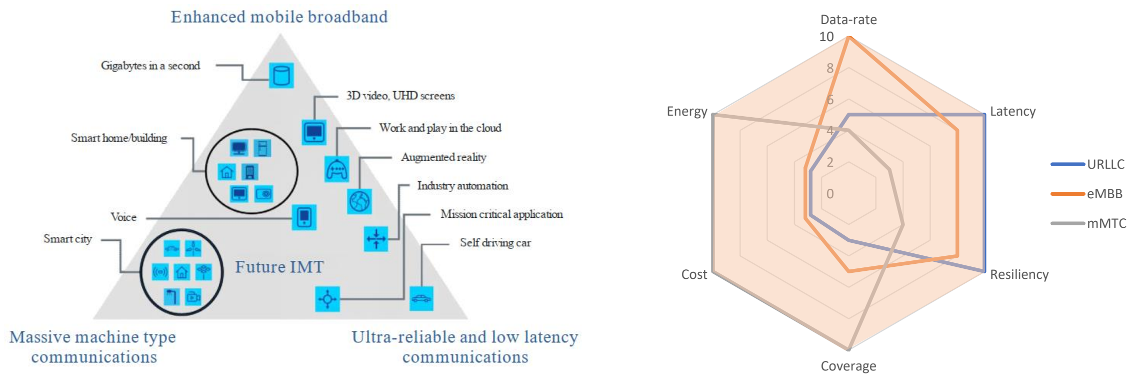


Figure 1.1: Service categories identified in the ITU vision (left), and exemplification of diverging service requirements for the three categories (right). [*Source for figure on the left: ITU-R M.2083-0: IMT Vision—Framework and overall objectives of the future deployment of IMT for 2020 and beyond*]

In order to embrace the ITU-R vision and to support an increasing demand of a wide range of **heterogeneous services**, 5G systems are reliant on multiple enabling technologies such as multi-access edge computing (MEC) [3], network function virtualization (NFV), and software defined network (SDN) among the others.

While NFV provides the ability to create dedicated logical functions and deploy them as virtualised network functions (VNFs) that can be chained to form a service function chain (SFC), SDN enables centralized control and management of the networking and communication between these VNFs. Conversely, MEC is an enabling technology that provides computing, networking, and virtualization resources close to the end-users. Using MEC technology, services can be provisioned in a considerably shorter time scale compared to the legacy computing paradigms such as cloud computing, where all the data should be transferred from the user to the cloud for processing. Moreover, the MEC technology mitigates the backhaul (BH) link utilization to a large extent through bypassing data transfer over the BH and serving the users locally by the MEC hosts nearby [4].

The emergence of the network slicing paradigm has been a game-changer for the evolution of mobile networks, particularly with the advent of 5G and beyond networks. This technology utilizes advanced **network softwarization** techniques, such as NFV and SDN, to divide the mobile network into logically separated networks. Each slice is designed to serve a particular type of service, enabling operators to offer customized and optimized solutions to customers with diverse requirements. The **distributed user-plane** deployment introduced in fifth-generation (5G) allows the slices to leverage edge computing, which enables the deployment of delay-sensitive applications closer to the network edge. By using MEC technology, applications can be processed with low latency, reducing network congestion, and improving overall network performance.

This technology offers a significant advantage by enabling mobile virtual network operator (MVNO) to tailor their network offerings to meet the specific requirements of different service types, such as low latency, high bandwidth, and reliability. By doing this, network slicing can tackle services with diverse requirements, alleviating the issue of supporting services with diverging requirements. This can ultimately lead to more efficient resource utilization, improved network performance, and enhanced user experience. However, while network slicing is a critical technology for the success of future mobile networks, it also poses additional challenges. An end-to-end (E2E) slice model spans different technological domains of the network such as radio access network (RAN), transport network, and 5G core (5GC) network functions. Upon receiving a slice request from a vertical, resources from different domains of the network shall be allocated to the slice to finally embed the service(s) onto the substrate network.

The softwarized network approach described above has the potential to reduce capital expenditure and operating expenses by enabling a high level of network programmability and flexibility to meet the heterogeneous service demand. However, it also presents some open challenges, including complex configuration, control, and management of such networks. One of the research challenges considered in this dissertation is to **develop optimization-based techniques to investigate efficient strategies for optimal slice provisioning while deploying a diverse set of service types**.

Over the past few years, there have been numerous initiatives and projects that have focused on developing proof-of-concept solutions for MEC, in line with ETSI standards[1]. Deploying MEC requires virtualization, and it can be implemented using various hardware platforms, including commercial off-the-shelf (COTS) devices [5]. MEC deployment architectures may include networks in different data centre locations, as well as smaller and more pervasive nodes that may have constraints in terms of costs and computational capability. Additionally, the MEC ecosystem is expected to be highly heterogeneous, with varying performance from device to device due to factors such as hardware, software, virtualization technology. As a result, service providers may face a **heterogeneous MEC landscape** in the future, depending on the depth and density of the edge.

The field of service orchestration has seen a surge in interest over the past few years, leading to a plethora of research papers and literature proposing various solutions [6, 7, 8]. While the abundance of literature is encouraging, many of these proposals rely heavily on mathematical models and assumptions derived from network theory to develop optimization techniques. Unfortunately, this approach is often necessitated by the lack of comprehensive databases containing the necessary measurements needed to build a realistic model of the network components. However, optimization models developed using techniques such as mixed integer linear programming (MILP) often require significant assumptions that limit their effectiveness in modeling complex and heterogeneous environments accu-

[1] `1https://mecwiki.etsi.org/index.php?title=MECEcosystem`

rately. This limitation poses a significant challenge for researchers and practitioners looking to optimize service orchestration solutions, especially in dynamic and ever-changing environments. Consequently, there is a need for alternative approaches that can integrate real-world measurements into optimization models and better reflect the complexities of service orchestration systems.

For MEC systems to be sustainable and manageable, service providers need to adopt suitable models that allow them to dimension and provision their MEC nodes on the network fabric. Additionally, providers must understand how to enable MEC nodes to support applications and their key performance indicators (KPIs). Another critical factor to consider in distributed systems, particularly MEC, is power consumption [9]. Energy efficiency in MEC depends on multiple factors such as hardware, virtualization technology, and software used for deployment. In light of these concerns, this dissertation endeavors to address another research challenge, namely, **investigating measurement-based approaches to create a digital representation of the network**.

In a highly heterogeneous scenario, the **digital twin (DT)** with proper AI/ML solutions can play a crucial role in **automated network management**. It operates as an intermediate and collaborative layer, enabling the orchestration layer to better understand network behavior before making changes to the physical network. Consequently, investigating **how to leverage a DT to realize closed-loop automation** is another critical challenge addressed in this dissertation. Overall, these efforts aim to enhance MEC's performance, sustainability, and manageability, making it a valuable solution for mobile network operators.

1.2 Methodology

The first objective addressed in this thesis was to study the **challenge of coordinating the functions of MVNOs with the applications deployed in network slices**. This problem can be framed as a virtual network embedding (VNE) problem, where each virtual network corresponds to a logically separated network slice spanning various technological domains, such as RAN, transport, core, and services. To address this challenge, several existing works on VNE problems in literature were studied to identify gaps and weaknesses. Based on the state-of-the-art and best practices in the field, an **optimization problem was formulated as a MILP problem** and solved using a mathematical optimization solver called Cplex [10].

While optimization solvers such as Cplex are guaranteed to efficiently find the optimal solution to a problem, subject to constraints, they are computationally intensive and can bring scalability issues. Furthermore, handling non-linearities, which is often necessary in network optimization problems, increases complexity and sensitivity to parameters. This is because the VNE problem is known to be NP-hard and is often investigated using optimization techniques and heuristics.

As done in prior research not part of this thesis [11], heuristics are frequently considered in the literature as a viable alternative approach for resolving scalability issues in optimization problems. Heuristics offer a rapid and efficient solution to the problem at hand, and can be implemented as an alternative to more computationally intensive optimization techniques. However, they often lack in optimality and theoretical foundation. Instead of pursuing an optimization goal, they are based on assumptions that are done through logical reasoning, which could lead to a good solution. As a consequence, heuristics are often tailored for a specific problem and might work in specific scenarios and not in others.

Based on these considerations, **an investigation was carried out into the practical feasibility of optimization techniques that rely on genetic algorithm (GA) for resolving the previously formulated optimization problem**. The goal was to evaluate the effectiveness and efficiency of GA-based techniques in producing satisfactory solutions to the problem, with results compared against the optimal solution generated by the MILP model as the benchmark.

The choice of using GA to solve optimization problems is based on several factors. While GA does not guarantee to reach the global optima, its efficiency in exploring a large search space often leads to near-optimal solutions. Additionally, GA has shown higher flexibility than MILP techniques in handling non-linearities of the model, which is often a necessity in network optimization problems. Regarding the computational complexity, it is crucial to carefully evaluate the performance of both GA and MILP on a case-by-case basis to determine which approach is most appropriate for a specific problem instance. However, in general GAs are often faster than MILP for solving NP-hard problems, particularly for large problem instances or when the problem is complex and difficult to model as an MILP problem. Thus, the performance of GA in solving the optimization problem has been extensively studied through synthetic simulations to investigate its optimality, convergence time, and sensitivity to parameters.

As explained in the previous section, measurement-based approaches play a crucial role in investigating network management issues since they provide insights into the behavior and performance of various network components, including traffic and applications, in diverse and complex environments. In line with this, two test-beds were developed and are presented in this dissertation to support the research work. The first one is an **emulation testbed** that emulates a heterogeneous MEC-enabled 5G network. This test-bed enables the evaluation of MEC applications under realistic network conditions and allows for the collection of various performance metrics related to traffic flows, and the utilization of virtual resources. The **second testbed is a physical one** that focuses on power consumption measurement while also varying the virtualization technology used. By using this testbed, it is possible to evaluate the energy efficiency of MEC deployments under different conditions and gain insights into the most appropriate virtualization technology to use in specific scenarios. The two testbeds have been used to create a measurement dataset and investigate the performance of MECs based on different KPIs, such as CPU consumption, achieved data-rate, and power consumption.

In the initial stage of this investigation, various **regression mechanisms**, such as k-nearest neighbors (KNN), support vector regression (SVR), and polynomial fitting (PF), were employed on the physical testbed measurements to analyze and study the characterization of MEC KPIs. These techniques were utilized to explore the variations in KPIs with respect to different virtualization technologies. The objective of this analysis **was to assess the feasibility of building a digital representation of a MEC node, known as a DT**. The DT would replicate the behavior of the actual MEC node subject to a certain amount of networking and computation demand. To validate the approach, the predicted KPIs were compared with real measurements which were held out as a test set. By doing so, the feasibility of building an accurate digital representation of a MEC node, in the form of a DT, were assessed in various conditions emulating scenarios with partial information available.

Through our observations, it was found that regression mechanisms are efficient in predicting the behavior of a MEC node when considering aggregated demands. However, when faced with multiple demands that may have a different impact on the MEC behavior compared to a few demands generating the same amount of aggregated requests, it is unclear how to efficiently leverage these mechanisms. In order to address this issue, **graph neural networks (GNNs) have been explored** as a viable solution. GNNs provide a powerful tool to handle a varying number of requests arriving at the same MEC by representing them as a graph. By leveraging the graph structure and the relationships between the nodes, GNNs can capture the varying impact of each individual demand on the overall MEC behavior, enabling accurate predictions even in the case of varying demand profiles.

GNN are particularly well-suited for modeling and analyzing the complex and interdependent relationships within communication networks, as they are able to naturally incorporate and process the graphical structure of such networks. By capturing the underlying network behavior, GNN can provide a more comprehensive and scalable representation of the network, which can be used to study networks of varying sizes and topologies. Through a comparison of predicted and actual KPIs, valuable insights were obtained into the reliability and accuracy of the DT, which has the potential to inform decision-making for a range of network management tasks. The findings also highlighted the potential of GNN-based DT in facilitating closed loop automation, where network management processes are automated and continuously refined based on real-time network performance data.

1.3 Contribution and Structure of the Thesis

With reference to the challenges and objectives motivated in Section 1.1, at a high level, the contributions of this thesis can be divided in three parts.

1. Investigate efficient strategies for optimal slice provisioning while deploying a diverse set of service types.

2. Develop relevant environments in order to experiment on measurement-based approaches to create a digital representation of the network.

3. Study the DT as a crucial component in automated network management systems.

With reference to the first challenge, the main contributions in this thesis are summarised as follows:

- Leverage GA optimization by intelligently acting on the strategies to generate new solutions and the cross-over and mutation operator, in order to guarantee the algorithm convergence, which otherwise is not achieved due to the high complexity of the problem. The Adaptive GA (AGA) approach is proposed to learn a better generation of new solutions with the aim to improve optimality and convergence time.

- The E2E delay is modeled considering the transport, propagation, and processing delays while ensuring the network resources are not over-provisioned and the E2E service requirements are met for all the services of the verticals. In particular, the processing delay of each VNF implementing the service and control plane (CP)/user plane (UP) functions, is modeled assuming a container-based deployment, where fractions of CPU can be reserved, and including a parameter accounting for the virtualization overhead.

- Piece-wise linearization is used to model the processing and transport delays. While this keeps the problem linear, it also allows to generalize the system to any other non-linear delay model.

- We introduce and study Dedicated and Shared user plane function (UPF) as two slice provisioning approaches to satisfy the services requested by the verticals.

With reference to the second challenge, the main contributions in this thesis are summarised as follows:

- A test-bed for emulation has been created to evaluate heterogeneous MEC systems. The emulation test-bed collects a range of performance metrics related to traffic flows and the utilization of virtual resources.
- With the testbed, MEC applications can be effectively tested under realistic network conditions, emulating an end-to-end 5G network.
- A physical test-bed was created with a focus on measuring power consumption while varying the virtualization technology. This allows to evaluate the energy efficiency of MEC deployments under different conditions, and provides insights into the most appropriate virtualization technology to use in specific scenarios.

With reference to the third challenge, the main contributions in this thesis are summarised as follows:

- The data-driven DT representation of the performance of MEC nodes is a virtual model that mirrors the behavior of a physical MEC node, based on data collected from the test-bed.
- Machine learning (ML) techniques have been applied to improve the accuracy of the digital twin representation for different populations of training sets, enabling more accurate predictions of the performance of MEC nodes.
- The use of GNNs has been investigated as a potential solution for improving the DT representation, taking into account impairments due to the underlying network and variations in the number of service requests.

- The potential of GNN-based DTs has also been studied in the context of closed-loop automation, where network management processes can continuously adapt and optimize based on real-time network performance data, improving overall system efficiency and reliability.

The three challenges described earlier are addressed in Chapters 2, 3, and 4 of this thesis. Finally, Chapter 5 summarizes the key findings of the work and provides concluding remarks.

Chapter 2

Mathematical models for Slice Provisioning in 5G and Beyond Networks

2.1 Motivations

Section 1.1 introduces network slicing, which plays a crucial role in enabling 5G and beyond 5G networks. This technology leverages virtualization technologies like NFV and SDN to divide the mobile network into multiple logical networks, each designed to cater to a specific service type. By utilizing NFV, it becomes feasible to deploy logical functions as dedicated VNFs. Conversely, SDN facilitates centralized control and management of networking and communication between these VNFs, enabling them to be chained together to form a SFC. Figure 2.1 illustrates the concept of end-to-end service provisioning. We remark that the provisioning of connectivity is usually the job of network operators providing the end-to-end UP and CP functions to the end-users. In this context, suitable mechanisms should be in place to ensure effective end-to-end service provisioning. Considering a neutral host model, the job of an infrustructure provider (InP) is to embed and maintain service requests satisfying the application requirements.

5G undertakes an end-to-end transformation spanning different domains of the network, including the RAN, the transport, and the core network. While in the RAN domain different technological enablers have been introduced with 5G new radio (NR) such as flexible numerology [12] and

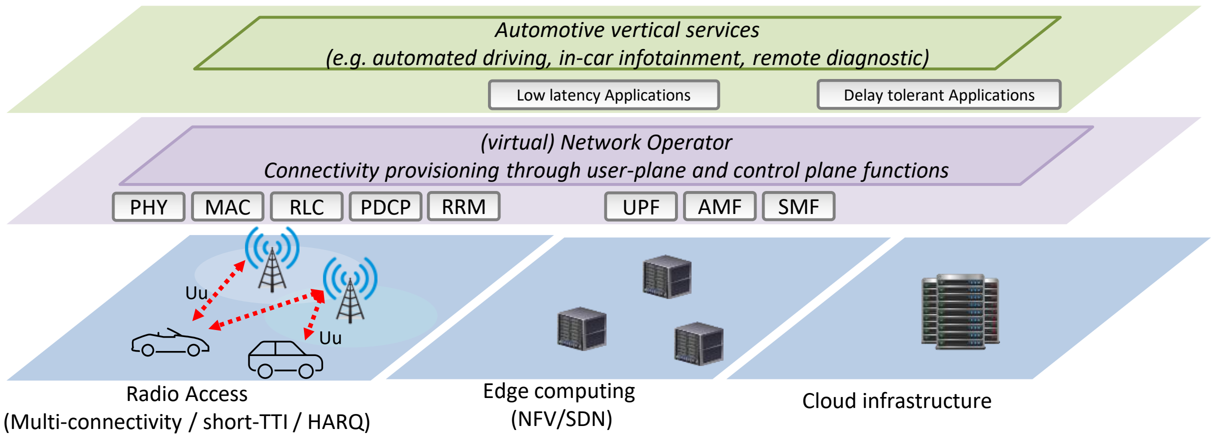

Automotive vertical services
(e.g. automated driving, in-car infotainment, remote diagnostic)
Low latency Applications
Delay tolerant Applications
(virtual) Network Operator
Connectivity provisioning through user-plane and control plane functions
PHY
MAC
RLC
PDCP
RRM
UPF
AMF
SMF
Uu
Uu
Radio Access
(Multi-connectivity / short-TTI / HARQ)
Edge computing
(NFV/SDN)
Cloud infrastructure

Figure 2.1: End-to-end slice provisioning to verticals, spanning different technological domains of the network.

multi-connectivity (MC)[13, 14], to name a few, the Core domain undertakes a radical transformation by introducing control plane and user plane separation (CUPS). Thus, the CP, whose main components are the access & mobility management function (AMF) and the session management function (SMF), is detached from the UP allowing a distributed
acupf deployment close to the users and contributing to latency reduction. As a rule of thumb, considering an E2E delay requirement of 10ms, the speed of light in optical fiber, and 9.5ms as the E2E delay budget including transmission, channel access, queuing and processing, the distance between the application and the gNodeB (gNB) - base station in 5G - cannot be more than 50km. Thus, a flexible deployment is essential to deploy low latency applications close to the end-user, while delay-tolerant applications can be kept in the cloud.

However, the E2E embedding of network slices onto the substrate network is a nontrivial task that needs to carefully account for several factors such as satisfaction of service requirements, providing slice isolation, and guaranteeing efficient use of the non-uniform network resources. Embedding network slices onto the substrate network becomes more challenging, especially when it comes to the network edge, where the computing and network resources are considerably scarce and costly, and the orchestration should be performed in a way that requirements of the slice consumers are met and at the same time resources are used in the most efficient manner.

In this work, the network slice embedding problem is formulated as a multi-objective optimization problem. By acting on weighting parameters, the network operator is allowed to select the desired trade-off between computing, bandwidth, and VNF migrations costs while always satisfying the service QoS requirements in terms of delay and throughput. Then we explore the applicability of GA-based techniques as a practically viable solution and using an optimal MILP model as the benchmark.

2.2 Related Work

Network slicing has been considered a key enabler for 5G and beyond 5G networks to meet distinctive QoS requirements of different applications. In this regard, a considerable body of research has studied this problem focusing on different technological domains of the network. In [15, 16, 17] has studied the problem of network slicing in the RAN domain. In this context, an emerging trend is to split the responsibility of SFC orchestration and radio access control, as demonstrated by the open RAN (O-RAN) initiative [18]. While some works consider both SFC placement and radio access aspects like user association [19, 20], in this chapter, we mainly focus on workload placement and resource allocation problems.

Different from the above-mentioned studies, the works in [21, 22, 23, 24, 25] study the problem of network slicing in the core aiming at deploying virtualised evolved packet core (vEPC) functionalities. The concept of a flexible fourth-generation (4G)/long-term evolution (LTE) network deployment was later included in the design of the 5G system architecture. In [21], the vEPC concept is investigated, proposing SoftEPC for an on-demand placement of frequently used functions close to the users, while in [22] KLEIN is proposed to distribute vEPC load across different data centers. Further, architectural aspects of managing the mobile packet core in the SDN/NFV context have been investigated in [23]. In [26] and [27], a decoupling of CP and UP functions of the vEPC is proposed in order to improve the scalability. This concept has been included in the standardised service-based 5G mobile core network [28]. Service-aware packet gateway (PGW) placement is studied in [24] to minimise the operators' cost, while multi-domain aspects have been investigated in [29, 30] using coalition formation game to deploy vEPC components in federated clouds. In [25], the authors study the embedding of vEPC functions by means of a MILP formulation to minimise the cost of nodes and link usage while satisfying the latency constraints. A similar problem definition can be found in [31], but in addition, the authors leverage empirical data to introduce a virtualisation overhead as a source of additional delay. However, in both works, the processing delay does not consider the resources allocated to the VNFs, and service provisioning in isolated slices is not taken into account.

Regarding 5G networks, a number of studies demonstrated network slicing developing prototypes [32, 33, 34], and in [35] a proof-of-concept is introduced to perform live migration and scaling experiments. In terms of research studies, a workload placement problem for 5G state management functions is provided in [36] and [37] where authors focus on the trade-off between minimisation of the aggregated VNFs traffic and minimisation of the state transfer between state management functions. The Pareto optimal solution has been investigated based on the adaptive weighted sum approach [38]. Based on complex network theory, in [39], authors study the E2E slice embedding problem optimising the KPIs of different services. In contrast, a cross-domain scenario is considered in [40] to address the cost-optimal deployment and abstracting the 5G core slice as a set of SFCs. An E2E slice resource allocation scheme based on auction theory, is investigated in [41] and considering a pre-determined VNF placement across a 3-tier data centre network. A joint problem for edge node placement (i.e., network planning) and UPFs deployment have been proposed in [42], where an ILP framework and heuristics are proposed in order to minimise the VNFs deployment cost and UPFs relocations. However, the paper does not consider control plane functionalities, and only the propagation delay is accounted for service latency. In contrast, [11] considers an E2E 5G slice orchestration considering CP and UP functions placement and considering transport and processing delays based on links and VNFs load.

Optimal solvers are computationally expensive. For this reason, several works leverage MILP techniques and propose heuristics associated with it. However, while heuristics bring advantages in terms of execution time, they are usually tailored to a specific objective and have poorer performances in service rejections. Several meta-heuristics techniques have been proposed to address this issue, which allows pursuing the same goal of MILP formulations while reducing the computational complexity [43].

As recognised in [44], meta-heuristics algorithms other than GAs are either more prone to fall into local minima, leading to an excessive memory requirement, or more indicated for continuous optimisation problems. For this reason, in this work, we focus on GA-based approaches.

In [44], authors modify the GA algorithm process by performing mutations only if crossover does not return better results and adding a new solution only if a better one is found. The drawback of this approach is that new solutions, even if worst that the best one contained in the current population, can statistically lead to better individuals when applying crossover operator. An evolutive GA is proposed in [45] where the population is initialised leveraging the output of the previous time interval. However, this could lead to a lower population diversity making the algorithm more difficult to converge. In [46] a strategy for an on-the-fly correction of unfeasible solutions is proposed. However, correcting a solution to meet one constraint (e.g., CPU node capacity) could lead to unpredictable behaviours and the violation of another one (e.g., E2E latency). In [47], authors leverage Deep Learning techniques to predict vehicle mobility to apply a GA-based service orchestration then. However, the authors do not consider slicing and a full end-to-end service function chain in terms of network orchestration. In [48] authors propose two GA techniques to minimise the deployment cost and the number of migrations in two separate formulations. In contrast, our aim is to consider both aspects in one single formulation providing the user with the capability to decide on the placement behaviour by acting on cost parameters. Moreover, the previous approaches do not take into account the E2E delay service requirement.

Recently, several works applied reinforcement learning (RL) techniques in network orchestration problems [49, 50]. However, motivated by the results in [51] where GA techniques show a higher convergence speed with respect to RL, in this work, we focus on investigating the applicability of the former type of technique. While applying a GA approach similar to the one presented in [48], several convergence issues have been observed due to the high complexity of the problem. For this reason, we adopted a standard GA procedure as explained in section 2.4 in order to achieve algorithm convergence. Since the generation of a good solution is much dependent on the scenario, we also proposed an adaptive version in order to learn the best way to create new solutions, thus improving the algorithm optimality and reducing its convergence time.

Compared to the other works highlighted before, our problem formulation takes into account two slice provisioning approaches and namely Dedicated and Shared UPF, as explained in Sec. 2.3.1. The full E2E service chain, spanning from the RAN to the application and passing through the UP and CP functions, are modelled assuming a container-based technology and using a continuous, rather than discrete, CPU reservation mechanism. The network resources are reserved in order to meet the E2E delay requirements while jointly considering transport, propagation, and processing delay. In particular, for the transport and processing delay, we use the M/M/1 queuing model. While for the transport links, the delay is proportional to the aggregated traffic and the serving capacity (e.g., the achievable network interface throughput), the processing delay accounts for the VNF aggregated traffic, the reserved CPU and a virtualisation overhead. It is worth noticing that the method used to linearise the formulation allows the generalisation to any non-linear model relating the VNF processing delay with the CPU reserved to it, and the aggregated traffic handled. Similarly, the transport delay over a link can be generalised to any non-linear model relating the packet processing delay with the aggregated traffic.

2.3 Network Model

2.3.1 Scenario Description and Problem Statement

Network Slicing is natively supported by the 5G service-based architecture (SBA). It allows provisioning different logical and isolated networks for the deployment of services belonging to vertical customers. In a real-world 5G scenario, each vertical can request slices of different types (i.e., URLLC, eMBB and mMTC) with different quality of service (QoS) requirements (i.e., throughput and E2E latency). Network Slices are usually provisioned by the mobile network operators (MNOs) who deploy/instantiate the UP/CP functionalities on their network, allowing vertical users to connect to the network and access the services in cloud data centers (DCs). However, such a communication and computing paradigm is not sufficient by itself to respond to the stringent and diverging E2E requirements of the applications envisioned by the most demanding 5G use cases. In this work, we refer to a generic scenario where a network slice is dedicated to a vertical and UP, and CP functions are orchestrated together with the applications (APPs) deployed as VNFs.

In essence, each individual end-to-end network slice deploys the functionalities of a complete network, including specific network layer capabilities, operational parameters, and network characteristics. When a slice is deployed, it is referred to as network slice instance (NSI). Conventionally, one NSI is dedicated to a vertical, however the presence of services with diverging requirements, suggests dedicated resources be provisioned for each specific service type of a vertical [2] (e.g., an automotive vertical requesting services for automated driving, in-car infotainment, and remote diagnostic). In this work, we consider two slice provisioning approaches as depicted in Figure 2.2. The first approach is represented in green and called *Shared UPF*, while the second approach is shown in orange and called *Dedicated UPF*. In both cases, a network slice is dedicated to a vertical. However, in the Shared UPF approach, the same UP is used by all the application types, while in the Dedicated UPF approach, the UP is dedicated to a specific application type and deployed within specific SFCs, where each SFC has dedicated resources allocated and forms part of the end-to-end slice.

In our scenario, we deploy the CP functions in the network, considering them to be dedicated to a particular vertical. The rationale behind this approach is that each vertical might have the need for dedicated CP functionalities which are not usually provided to normal customers. For example, in the automotive vertical fast network re-selection solutions are needed to support a seamless transition of the users between networks of different operators [52]. Thus, in this work we consider virtualised CP functions to be included in the verticals's slice as this would ease the configuration and provisioning of interfaces that are tailored to a specific vertical and not supported in current networks due to technical, legal, and administrative challenges [53].

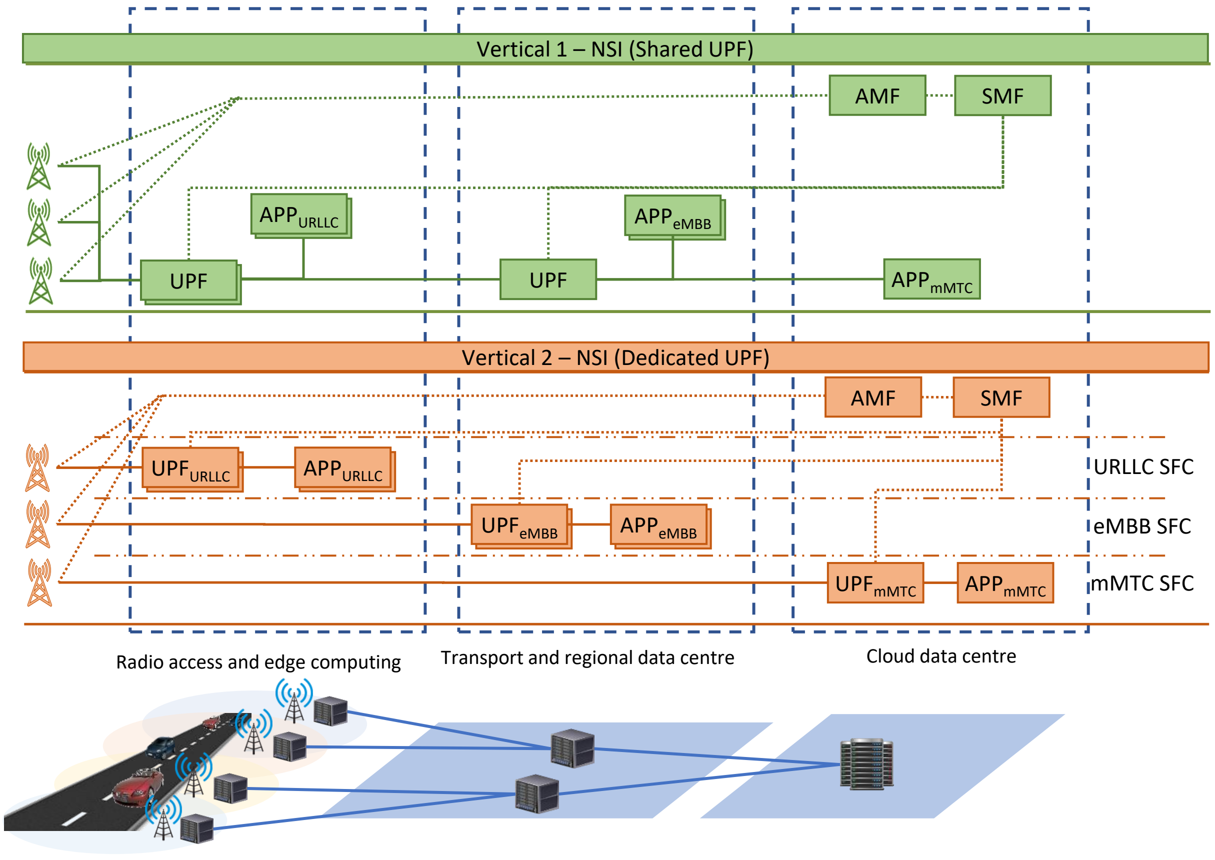


Figure 2.2: Sample of mobile network and the slice provisioning approaches. Green and orange represent the Shared and Dedicated UPF approaches, respectively. In both cases one network slice is provided to a vertical. Solid and dotted lines represent the UP and CP traffic flows.

After collecting all the requests from the verticals, the problem is to embed the SFCs and allocate the necessary link and node resources to satisfy the service requirements. The workload placement is performed pursuing different business logics, which is possible by minimizing a cost function given by the CPU and bandwidth usage. However, upon a change in the requests, already placed VNFs could be migrated while pursuing a specific cost minimization strategy. Since VNF migration is a costly and time-consuming operation, a VNF migration cost is also taken into account. The E2E service placement is formally stated as follows.

Given a substrate network modeled as described in Section 2.3.2, and a set of verticals requesting different services inside a network slice as described in Sec. 2.3.3.

Find a suitable placement of all the SFC requests assigning the necessary resources such that (*i*) the QoS requirements of each service is satisfied, and (*ii*) the network resources are used in the most efficient manner.

Objective: pursue a specific business logic defined as either (*i*) the minimization of the provisioning costs (i.e., CPU usage), (*ii*) minimization of the links bandwidth usage, or (*iii*) minimization of the VNFs migrations while changing the service requests.

Table 2.1: Substrate network parameters

Parameter	**Description**
$G(N, E)$	Mobile network graph.
N	All nodes/DCs in G.
N_g	Set of gNBs in G ($N_g \subset N$).
E	Set of substrate links in G.
$\mathcal{C}^e_{thr}$	Capacity of the link $e \in$ E.
$\mathcal{C}^d_{cpu}$	CPU resources available at the DC $d \in N$.
$\mathcal{C}^g_{thr}$	Maximum throughput supported by the gNB $g \in N_g$.
τ_g, τ_e	Transmission time over the air interface for gNB g and transmission time over transport link e.

2.3.2 Substrate Network Model

We assume a three-tier substrate network, composed of edge-DC, regional-DC, and cloud-DC, as illustrated in Figure 2.2. The substrate network model parameters are reported in Table 2.1 and explained below.

Let $G(N, E)$ be an undirected graph representing the substrate network, where N is the set of all nodes and includes both computing nodes (DCs) and 5G base stations (gNBs). It is assumed that each gNB $g \in N_g$ (being $N_g \subset N$) is co-located with one edge-DC. Each DC is equipped with a certain set of CPU resources $\mathcal{C}^d_{cpu}$ and a cost Λ_d associated with each CPU core. The closer the DC is to the edge, the lesser are resources and more costly to provide. As we go from the edge towards the cloud, resources become more abundant and cheaper to provide. We define E as the set of fronthaul (FH) and BH links interconnecting edge-DC to regional-DC and regional-DC to cloud-DC, respectively. A link $e^{m,n}$ between two DCs $m, n \in N$ is defined if they are directly connected. Each physical link has a certain capacity in terms of throughput $\mathcal{C}^{e}_{thr}$ and a cost Λ_e associated to the utilization of each Mbps of the link.

2.3.3 Slice and Service Request Model

This section presents the details of the slice and service request model employed in our work. The notations used in the model are reported in Table 2.2.

We consider an E2E slice model in which a slice spans different technological domains of the network (i.e. RAN, transport, and core). Upon receiving a slice request from a vertical, our model has to allocate resources from different domains of the network to the slice and then embed the service(s) onto the network. Each service is composed of two SFCs that are modeled as directed graphs. The first SFC defines the CP functions and the communication links between them that should be embedded onto the network. The CP functions provide controlling and management functionalities to other functions in the core. The second SFC to be mapped to the network includes the UP functions, which are responsible for processing the actual user traffic. In our model, the applications requested by the vertical are also part of the UP service chain as it is shown in Figure 2.2.

Let N_{slc} be the set of all slices requested by the verticals to be deployed in the network. For each $r \in N_{slc}$, let N^{r}_{ser} be the set of services requested inside that specific slice. Each requested service $s \in N^{r}_{ser}$ is defined by a service type, QoS requirements, including the maximum E2E latency ($\Delta^{r,s}_{up}$ for UP and $\Delta^{r,s}_{cp}$ for CP), the data traffic demand (ω^{r}_{s}), and a targeted geographical area included in the slice request. We define two sets of VNFs: the set $N^{r,s}_{vnf}$, defining the set of VNF types that are employed for the creation of the UP and CP SFCs of a specific service s; and the set N^{r}_{vnf}, which includes all the VNF types that are used within a slice containing the services requested by a vertical. Let's consider two services s_1 and s_2 belonging to the slice $r \in N_{slc}$. If the sets N^{r,s_1}_{vnf} and N^{r,s_2}_{vnf} contain the same identifier for a VNF type, the formulation accounts that those two services can share this VNF once instantiated. Assume both s_1 and s_2

Table 2.2: Mobile network slice request parameters

Parameter	**Description**
N_{slc}	Set of slices requested by the verticals, where each slice contains at least one service.
N^r_{ser}	Set of services requested inside the slices $r \in N_{slc}$.
N^r_{vnf}	Set of VNF types that are employed for the creation of services requested inside the slice $r \in N_{slc}$.
$N^{r,s}_{vnf}$	Set of VNF types that are employed for the creation of the service $s \in N^r_{ser}$ requested inside slice $r \in N_{slc}$.
$N^{r,s}_{l}$	Set of virtual links that define connection between VNFs in $N^{r,s}_{vnf}$ to realize the service $s \in N^r_{ser}$ requested inside slice $r \in N_{slc}$.
ω^r_s	Data rate demand of service $s \in N^r_{ser}$ requested inside slice $r \in N_{slc}$
$\Delta^{r,s}_{up}, \Delta^{r,s}_{cp}$	UP and CP E2E delay requirement for service $s \in N^r_{ser}$ requested inside slice $r \in N_{slc}$.

are services of type URLLC. Then, both sets N^{r,s_1}_{vnf} and N^{r,s_2}_{vnf} will include $\text{APP}_{\text{URLLC}}$ as an identifier for the application VNF. Once instantiated on a DC, this VNF can be leveraged by both s_1 and s_2 provided that their requirements are satisfied.

When we consider the Shared UPF approach the sets N^{r,s_1}_{vnf} and N^{r,s_2}_{vnf} will contain the same identifier for the UPF regardless of the service type. This means that the same deployed UPF can be used to implement the SFC of different services, even of a different type. Conversely, in the Dedicated UPF approach the sets N^{r,s_1}_{vnf} and N^{r,s_2}_{vnf} will contain a service-specific identifier for the UPF, such as $\text{UPF}_{\text{URLLC}}$, as shown in Figure 2.2. This means that a deployed UPF can be used to implement only the SFC of services belonging to the same category. Similarly, for the CP all the sets of VNF types $N^{r,s}_{vnf}$ belonging to the services in N^r_{ser} will include an unique AMF and SMF identifier indicating that the same VNFs implementing the SFC for the CP can be used by all the services of a vertical. The approach described above is exemplified in Figure 2.2. Finally, $N^{r,s}_{l}$ denotes the set of virtual links that are interconnecting the VNFs of the service s and realizing the UP chain (gNB – UPF – APP) and the CP chain (gNB – AMF – SMF – UPF). For each service s, the set $N^{r,s}_{l}$ refers to the same VNF identifiers used in $N^{r,s}_{vnf}$.

2.4 MILP Problem Formulation

We model the problem of SFC placement and resource allocation as an VNE problem, which is a well-known NP-hard problem and has been extensively studied by the literature [54, 55, 56]. The embedding process consists of two phases: node embedding and link embedding. While in the first phase we map the VNFs on the substrate network, in the latter, we construct the communication links between these embedded VNFs. Our proposed model enforces that for each of the considered types of resource (nodes and links) all the capacity constraints are respected for the model to be valid. We provide the details of the proposed MILP model followed by a GA-based algorithm.

2.4.1 MILP Objective Function

The defined VNE problem has been formulated using MILP techniques. In our model, we propose an objective function (2.1) with the goal of minimizing service provisioning cost, including the cost of CPU on the computing nodes, the cost of communication links, and the VNF migration cost.

$$
\begin{aligned}
min \sum_{d \in N} \sum_{r \in N_{slc}} \sum_{v \in N^{r}_{vnf}} \Lambda_d \zeta^{r,v}_{d} + \\
+ \sum_{r \in N_{slc}} \sum_{s \in N^{r}_{ser}} \sum_{l \in N^{r,s}_{l}} \sum_{e \in E} \Lambda_e \omega_{e'} \Phi^{l}_{e} + \\
+ \sum_{d \in N} \sum_{r \in N_{slc}} \sum_{v \in N^{r}_{vnf}} \Lambda_m M^{r,v}_{d}
\end{aligned}
\tag{2.1}
$$

As given in the objective function above, Λ_d, Λ_e and Λ_m are the resource usage costs on the DC $d \in N$, the costs per Mbps over the link $e \in E$, and the cost to migrate an already placed VNF, respectively.

Table 2.3: Binary and continuous decision variables.

Variable	Description
$\Phi_g^{r,s}$	Binary variable indicating if a service $s \in N_{ser}^r$ belonging to a vertical $r \in N_{slc}$ is mapped to the substrate gNB $g \in N_g$
$\Phi_{v,d}^{r,s}$	Binary variable indicating if a service $s \in N_{ser}^r$ belonging to a vertical $r \in N_{slc}$ uses a VNF of type $v \in N_{vnf}^{r,s}$ deployed on the DC $d \in N$ to realise its SFC.
Φ_e^l	Binary variable indicating if the virtual link $l \in N_l^{r,s}$, composing the SFC of the service $s \in N_{ser}^r$ belonging to a vertical $r \in N_{slc}$, has been mapped to the substrate link $e \in E$.
$\zeta_d^{r,v}$	Continuous variable capturing, for each vertical $r \in N_{slc}$, the amount of reserved CPU for a VNF of type $v \in N_{vnf}^r$ deployed on the DC d.
$\delta_d^{r,v}$	Continuous variable capturing, for each vertical $r \in N_{slc}$, the computing delay of a VNF of type $v \in N_{vnf}^r$ which is deployed on the DC d.
δ_e	Continuous variable capturing, the delay incurred in transmitting over the network link $e \in E$.
$M_d^{r,v}$	Binary decision variable capturing if a VNF $v \in N_{vnf}^r$, belonging to the vertical $r \in N_{slc}$ and currently deployed on DC $d \in N$, it was previously deployed on another DC and thus have been migrated.

2.4.2 MILP Constraints

We consider three different business logic in this chapter. These can be obtained by modifying the objective function costs. The first one, dubbed *Obj-Cost*, tries to minimize the CPU usage cost: the closer the DC is to the gNB, the higher is the cost. The second objective, dubbed *Obj-Bwt*, tries to minimize the bandwidth consumption on the links. Finally, the third one, *Obj-Mig*, minimizes the VNF migrations while the scenario is changing with new incoming service requests. Regardless of the objective function, for a solution to be valid, our proposed MILP model needs to satisfy all the given constraints to ensure that (*i*) the correct services composition and allocation onto the substrate network are assured (*ii*) the network has enough resources to deploy the services, and (*iii*) the service QoS requirements are all met.

As mentioned earlier, our proposed method considers slice deployment in an E2E manner. Therefore, upon receiving a slice request by a vertical, for each service included in the request, our proposed model has to establish an SFC from a gNB in the region where the vertical defines as the origin point for the service to start. In this vein, let Ω_s be the set of candidate gNBs that can be selected to establish the RAN domain of the service. The set of candidate gNBs is built considering the euclidean distance between the gNB position $\vec{g}$ and the service request location $\vec{r}$, and the coverage radius of the gNB d_g.

$$\Omega_s = \left\{ g \in N_g : |\vec{g} - \vec{r}| \leq d_g \right\} \tag{2.2}$$

Notice this assumption is made for the sake of simplicity; however, it does not affect the validity of the approach. Indeed, the retrieval of the set Ω_s at the orchestrator can be a task performed by a RAN controller. Constraints (2.3) and (2.4) ensures that only one gNB $g \in N_g$ is utilized for each of the services $s \in N^r_{ser}$ inside the slice $r \in N_{slc}$, where that gNB should be a candidate gNB.

$$\sum_{g \in N_g} \Phi^{r,s}_g = 1 \quad \forall r \in N_{slc}, \quad \forall s \in N^r_{ser} \tag{2.3}$$

$$\sum_{g \in N_g \setminus \Omega_s} \Phi^{r,s}_g = 0 \quad \forall r \in N_{slc}, \quad \forall s \in N^r_{ser} \tag{2.4}$$

Constraint (2.5) ensures that each VNF $v \in N^{r,s}_{vnf}$ implementing the SFC of service $s \in N^r_{ser}$ should be deployed only once in a specific DC $d \in N$.

$$\sum_{d \in N} \Phi^{r,s}_{v,d} = 1 \quad \forall r \in N_{slc}, \quad \forall s \in N^r_{ser}, \quad \forall v \in N^{r,s}_{vnf} \tag{2.5}$$

To ensure the correct allocation of CPU captured by the variable $\zeta^{r,v}_d$ while keeping the problem linear, we need to introduce another decision variable $\Phi^{r,v}_d$. The variable $\Phi^{r,v}_d$ is equal to 1 if and only if there is at least one service using that VNF on that node. To do so we leverage the so-called *big-M* constraint as shown in constraint (2.6), where $\mathcal{M}_b$ is a big positive number.

$$\begin{cases} \sum_{s \in N^r_{ser}} \Phi^{r,s}_{v,d} - \mathcal{M}_b \Phi^{r,v}_d \leq 0 \\ \Phi^{r,v}_d - \mathcal{M}_b \sum_{s \in N^r_{ser}} \Phi^{r,s}_{v,d} \leq 0 \end{cases} \tag{2.6}$$

$$\forall r \in N_{slc}, \forall v \in N^r_{vnf}, \forall d \in N$$

The decision variable $\Phi^{r,v}_d$ is then used in order to ensure that CPU resources are reserved on a node only if a VNF is actually instantiated. This is done by constraint (2.7) leveraging again the *big-M* constraint and ensuring a virtualization overhead σ_v representing the minimum amount of resources used by the VNF when instantiated.

$$\begin{cases} -\mathcal{M}_b \Phi^{r,v}_d + \zeta^{r,v}_d \leq 0 \\ \mathcal{M}_b \Phi^{r,v}_d - \zeta^{r,v}_d + \sigma_v \leq \mathcal{M}_b \end{cases} \quad \forall r \in N_{slc}, \forall v \in N^r_{vnf}, \forall d \in N \tag{2.7}$$

Let $E^{\star d}$ be the set of links ending at node $d \in N$. Conversely, let $E^{d\star}$ be the set of links originating from the node d and arriving at any node directly connected to d. To enforce each virtual link to be mapped on a continuous path in the substrate network we use constraint (2.8) where $\overline{N}^{r,s}_l$ and $\underline{N}^{r,s}_l$ are the set of virtual links connecting a gNB and a VNF v, and two VNFs v_1 and v_2, respectively.

$$\sum_{e \in E^{\star d}} \Phi^l_e - \sum_{e \in E^{d\star}} \Phi^l_e = \begin{cases} \Phi^{r,s}_{d,v} - \Phi^{r,s}_g & \text{if } l \in \overline{N}^{r,s}_l \\ \Phi^{r,s}_{d,v_2} - \Phi^{r,s}_{d,v_1} & \text{if } l \in \underline{N}^{r,s}_l \end{cases} \tag{2.8}$$

$$\forall r \in N_{slc}, \quad \forall s \in N^r_{ser}, \quad \forall d \in N, \quad \forall l \in N^{r,s}_l$$

To account for the VNF migrations, the correct computation of the decision variable $M^{r,v}_d$ needs to be ensured. This is done by matching the current and previous embedding.

$$\begin{cases} P^{r,v}_d - \Phi^{r,v}_d - M^{r,v}_d \leq 0 \\ P^{r,v}_d + \Phi^{r,v}_d + M^{r,v}_d \leq 0 \\ -P^{r,v}_d + M^{r,v}_d \leq 0 \end{cases} \tag{2.9}$$

$$\forall r \in N_{slc}, \quad \forall v \in N^r_{vnf}, \quad \forall d \in N$$

where $P^{r,v}_d$ represents the previously placed VNFs, and is an input to the problem formulation.

To ensure that the service request mapping is done not exceeding the network resources, we define constraints (2.10), (2.11) and (2.12) accounting for the gNB capacity, DC CPU capacity, and network links bandwidth, respectively.

$$\sum_{r\in N_{slc}} \sum_{s\in N_{ser}^{r}} \omega_s^r \Phi_g^{r,s} \leq \mathcal{C}_{thr}^{g}, \quad \forall g \in N_g \tag{2.10}$$

$$\sum_{r\in N_{slc}} \sum_{v\in N_{vnf}^{r}} \zeta_d^{r,v} \leq \mathcal{C}_{cpu}^{d}, \quad \forall d \in N \tag{2.11}$$

$$\sum_{r\in N_{slc}} \sum_{s\in N_{ser}^{r}} \sum_{l\in N_l^{r,s}} \omega_s^r \Phi_e^{l} \leq \mathcal{C}_{thr}^{e} \quad \forall e \in E \tag{2.12}$$

The latency of the service requests is modeled in an E2E manner, considering communication and computing latency in the chain.

$$\tau_{proc}^{r,s} + \tau_{transp}^{r,s} + \tau_{prop}^{r,s} + \tau_g \leq \Delta_{up}^{r,s}, \tag{2.13}$$

where $\tau_{proc}^{r,s}$ is the VNF processing delay, $\tau_{transp}^{r,s}$ is the transport delay, $\tau_{prop}^{r,s}$ is the propagation delay and τ_g is the RAN delay. The aggregated traffic handled by a VNF ($T_d^{r,v}$), and over a link (T_e) can be computed as follows.

$$T_d^{r,v} = \sum_{s\in N_{ser}^{r}} \omega_s^r \Phi_{v,d}^{r,s}, \quad \forall r \in N_{slc},\ \forall v \in N_{vnf}^{r},\ \forall d \in N \tag{2.14}$$

$$T_e = \sum_{r\in N_{slc}} \sum_{s\in N_{ser}^{r}} \sum_{l\in N_l^{r,s}} \omega_s^r \Phi_e^{l} \quad \forall e \in E \tag{2.15}$$

Similar to the research study in [11], we consider a fixed delay for the RAN and the signal propagation over the air interface. Instead, we adopt the M/M/1 queue model for the VNF processing and transport delay over a link. Consequently, the processing delay of a VNF is computed as following.

$$\delta_d^{r,v} = \frac{1}{\rho_v(\zeta_d^{r,v} - \sigma_v) - T_d^{r,v}} \quad \forall r \in N_{slc},\ v \in N_{vnf}^{r},\ d \in N \tag{2.16}$$

where, for a VNF v, we consider ρ_v as the processing capacity per vCPU unit (i.e., Gbps/vCPU), and σ_v as the virtualization overhead (in vCPUs). Similarly, the transport delay on the network link e is computed as

$$\delta_e = \frac{1}{\mathcal{C}^e_{thr} - T_e} \quad \forall e \in E. \tag{2.17}$$

Notice that the introduction of this delay model introduces non-linearity in the model. To cope with this, we rely on piece-wise linearisation whose details are given in Appendix (2.4.3). It is worth noticing that while this method introduces some approximation by describing a non-linear function by means of sampling points x_i and y_i, it allows to implement any kind of delay model or even adopt machine learning techniques to learn the delay from the experience.

Let $\overline{N}^{r,s}_{vnf}$ and $\underline{N}^{r,s}_{vnf}$ be the subsets of $N^{r,s}_{vnf}$ including UP and CP VNFs, respectively. Similarly, let $\overline{N}^{r,s}_{l}$ and $\underline{N}^{r,s}_{l}$ be the subsets of $N^{r,s}_{l}$ including UP and CP virtual links, respectively. By rewriting Eq. (2.13), we can write Constraint (2.18) ensuring that the UP latency experienced by each service $s \in N^r_{ser}$ does not exceed the maximum latency $\Delta^{r,s}_{up}$:

$$\sum_{v\in\overline{N}^{r,s}_{vnf}} \sum_{d\in N} \delta^{r,v}_d \Phi^{r,s}_{v,d} + \sum_{l\in\overline{N}^{r,s}_{l}} \sum_{e\in E} \delta_e \Phi^l_e + \sum_{l\in\overline{N}^{r,s}_{l}} \sum_{e\in E} \tau_e \Phi^l_e + \tau_g \le \Delta^{r,s}_{up} \tag{2.18}$$

$$\forall r \in N_{slc}, \quad \forall s \in N^r_{ser}$$

Similarly, Constraint (2.18) ensure that the CP latency does not exceed the maximum acceptable latency $\Delta^{r,s}_{cp}$. In this case, we assume CP latency to have a negligible transport delay.

$$\sum_{d\in N} \sum_{v\in\underline{N}^{r,s}_{vnf}} \delta^{r,v}_d \Phi^{r,s}_{v,d} + \sum_{e\in E} \sum_{l\in\underline{N}^{r,s}_{l}} \tau_e \Phi^l_e + \tau_g \le \Delta^{r,s}_{cp} \tag{2.19}$$

$$\forall r \in N_{slc}, \quad \forall s \in N^r_{ser}$$

Clearly, constraints (2.18) and (2.19) are not linear as they contain multiplications between variables. In this regard we take the same linerazation process and linearize the equation, which is given in Appendix 2.4.3.

Nonlinear problems are not easy to control and are difficult and costly to solve. Their complexity lies in the high dependency between the system variables that make small variations in the initial data to produce great differences in the solution. For these reasons, we adopted several strategies in our approach to keep the problem formulation linear. However, this required the introduction of additional variables that exacerbated the search-space complexity, increasing the computational complexity.

As an alternative to combinatorial optimization techniques such as MILP, meta-heuristics techniques can be used. Being a trade-off between randomization and local search, they allow obtaining quality results in a reasonable time, even if reaching the optimal solution is not guaranteed. Several meta-heuristics can be considered to solve the VNE problem. However, as pointed out in [44], also they have their own drawbacks. Simulated annealing techniques are prone to fall in local minima; tabu search, even if superior in terms of global exploration, may lead to excessive memory requirements; swarm intelligence techniques like particle swarm optimization are more suitable for continuous optimization problems; and ant colony optimization is not suitable to predict the resources utilized in the network. In our work, we preferred to adopt evolutionary techniques for reaching near-optimal solutions: they allow handling non-linearities easily without any additional variables, thus allowing us to solve the exact same problem formulated in this section but with the expectation to drastically reduce the computational complexity.

2.4.3 MILP Constraints linearisation

The E2E latency is modeled as explained in section 2.4.1 which, however, is non linear and not directly applicable in solver for MILP formulations. To solve this issue, we use the piece-wise linearisation method as shown by constraints (2.20) and (2.21), where x_i and y_i define the sampling points approximating the function $y = 1/x$. Further, w_i is a continuous variable in the interval $[0, 1]$ over which the SOS type 2 constraint is imposed. This type of constraint, available in most of the MILP solvers like Gurobi [57] and Cplex [10], imposes that at most two consecutive values of w_i can be non-zero. As a consequence, constraints (2.20) and (2.21) are approximating the delay functions in constraints (2.16) and (2.17), respectively, while maintaining the problem linear.

$$\begin{cases} \forall i, w_i \in \text{SOS2} \\ \sum_{i=1}^{n} w_i = 1 \\ \sum_{i=1}^{n} w_i x_i = \rho_v(\zeta_d^{r,v} - \sigma_v) - T_d^{r,v} \\ \sum_{i=1}^{n} w_i y_i = \delta_d^{r,v} \end{cases} \tag{2.20}$$
$$\forall r \in N_{slc}, \quad \forall d \in N, \quad v \in N_{vnf}^{r}$$

$$\begin{cases} \forall i, w_i \in \text{SOS2} \\ \sum_{i=1}^{n} w_i = 1 \\ \sum_{i=1}^{n} w_i x_i = \omega_e - T_e \\ \sum_{i=1}^{n} w_i y_i = \delta_e \end{cases} \quad \forall e \in E \tag{2.21}$$

Constraints (2.18) and (2.19) are another source of non-linearity since they contain the variable multiplication $\delta_d^{r,v}\Phi_{v,d}^{r,s}$ and $\delta_e\Phi_e^l$. To cope with this issue, we introduce two variable $\delta_{v,d}^{r,s}$ and δ_e^l. The former accounts for the processing delay incurred by service $s \in N_{ser}^{r}$ on a VNF $v \in N_{vnf}^{r,s}$ that is on its SFC, while the latter accounts for the transmission delay incurred by the virtual link $l \in N_l^{r,s}$ of the service $s \in N_{ser}^{r}$ on a link e that is traversed by the SFC of service s.

$$\begin{cases} \delta_{v,d}^{r,s} \leq \Phi_{v,d}^{r,s} \\ \delta_{v,d}^{r,s} \leq \delta_d^{r,v} \\ \delta_{v,d}^{r,s} \geq \delta_d^{r,v} - 1 + \Phi_{v,d}^{r,s} \end{cases} \tag{2.22}$$
$$\forall r \in N_{slc}, \quad \forall s \in N_{ser}^{r}, \quad v \in N_{vnf}^{r,s}, \quad d \in N$$

$$\begin{cases} \delta_e^l \leq \Phi_e^l \\ \delta_e^l \leq \delta_e \\ \delta_e^l \geq \delta_e - 1 + \Phi_e^l \end{cases} \tag{2.23}$$

$$\forall r \in N_{slc}, \quad \forall s \in N_{ser}^{r}, \quad e \in E, \quad l \in N_{l}^{r,s}$$

The variables $\delta_{v,d}^{r,s}$ and δ_{e}^{l} are computed by constraints (2.22) and (2.23), and can be directly used to solve the non-linearity of constraints (2.18) and (2.19) by substituting $\delta_{d}^{r,v}\Phi_{v,d}^{r,s}$ with $\delta_{v,d}^{r,s}$, and $\delta_{e}\Phi_{e}^{l}$ with δ_{e}^{l}.

2.5 GA-based SFC Placement

The main idea behind applying GA-based algorithms to the VNE problem is to drastically reduce the very high computational complexity required for the MILP model while pursuing the same objective function. This type of evolutionary algorithm is based on a population of individuals, usually initialized randomly when starting the algorithm. Each individual represents a solution to the problem, which is referred to as a chromosome.[1] The population evolves generation by generation through applying crossover and mutation operators while pursuing minimization of an objective function, also dubbed fitness function[2]. The definition of the solution and its length, the configuration of the algorithm parameters, and the definition of the crossover and mutation operators need to be carefully designed in order to guarantee the convergence of the algorithm to reach a near-optimal solution.

In this section we propose two meta-heuristic algorithms based on the genetic algorithm approach: the *Dubbed GA (DGA)* and a more sophisticated version, the *AGA*, that adapts its behaviour with the variability of traffic demand. We also chose to treat the *Dedicated UPF* scenario only, since, as it will be discussed in section 2.6, optimal solutions obtained with MILP show that the *Shared UPF* approach requires higher computational complexity while not bringing benefits in terms of network resource usage. The *Dedicated UPF* scenario also has the advantage of allowing a simpler definition of chromosome. Moreover, in a realistic scenario, the *Dedicated UPF* approach allows a more flexible configuration of the SFCs tailored to service-specific KPIs. In the formulation of the problem, we assume that when an application is deployed on a node, an UPF is placed as well to

[1]We equally refer to chromosome, solution, and individual.

[2]We equally refer to fitness and objective function in the chapter.

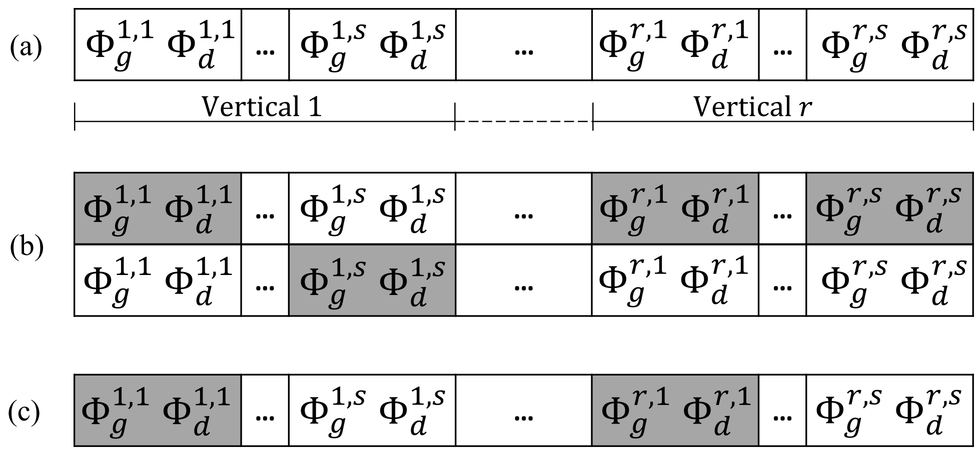

(a)
$\Phi_g^{1,1}$ $\Phi_d^{1,1}$ ... $\Phi_g^{1,s}$ $\Phi_d^{1,s}$ ... $\Phi_g^{r,1}$ $\Phi_d^{r,1}$ ... $\Phi_g^{r,s}$ $\Phi_d^{r,s}$
Vertical 1
Vertical r
(b)
(c)

Figure 2.3: Representation of a chromosome (a) and exemplification of crossover (b) and mutation (c) operators.

present the data packets to the application. While UP functions should be distributed to ensure the service KPIs, we assume CP functionalities are always placed in the cloud-DC. This assumption is justified by the results obtained from the MILP, always placing CP VNFs in the cloud-DC.

The GA procedure is presented in Algorithm 1 describing the iterative approach to evolve the population. In the rest of this section, we first describe our approach towards defining the service placement problem and the generation of new solutions. Secondly, we explain how the fitness function is calculated starting from a solution. Finally, we detail the procedure implemented for the GA procedure followed by the description of the AGA approach.

2.5.1 Service Placement

The chromosome is composed by an array of integer values describing the SFC placement for each service. As illustrated in Figure 2.3(a), the pair $[\Phi_g^{r,s}, \Phi_d^{r,s}]$ indicates that a service request s belonging to slice request r is associated to the gNB g and the DC d, respectively. To generate this associations we rely on the probability to select a gNB g and a DC d computed by rescaling Eq. (2.24) between 0 and 1,

$$\mathrm{P}^{r,s}_{g,d} = \frac{1}{\log(\Lambda_d)} \cdot \frac{1}{1 + d_{g-d}}, \tag{2.24}$$

where Λ_d is the node cost as for the MILP, and d_{g-d} is the number of hops for the shortest path between gNB g and a DC d. The aim of this strategy is to (i) give priority to nodes with lower cost, and (ii) avoid selecting associations resulting in longer paths with higher chances to violate constraints.

2.5.2 Fitness Calculation

The calculation of the fitness function is exemplified in steps 13-18 of Algorithm 1 where we have an individual as input to the function. Firstly, Φ^l_e is computed using the shortest path between the gNB and DC selected for each service. Consequently, δ_e can be computed as in (2.17). Recalling Eq. (2.13) and (2.18), the only unknown source of UP delay is the VNF processing time, which allows us to write

$$\tau^{r,s}_{proc} \geq \sum_{v \in N^{r,s}_{vnf}} \sum_{d \in N} \delta^{r,v}_d \Phi^{r,s}_d \quad \forall r \in N_{slc},\ \forall s \in N^r_{ser}, \tag{2.25}$$

where $\tau^{r,s}_{proc}$ is computed, for each service $s \in N^r_{ser}$, as the remaining latency budget given by $\Delta^{r,s}_{up}$ and subtracting the other known source of delay. Considering equations (2.16) and (2.25), the number of CPUs to allocate to each VNF in order to ensure the E2E latency can be computed as follows:

$$\zeta^{r,v}_d = \frac{1}{\rho_v}[2\Delta^{r,v}_d + T^{r,v}_d] + \sigma_v \tag{2.26}$$

where $\Delta^{r,v}_d = \max_{r,s}\{\Phi^{r,s}_d / \tau^{r,s}_{proc}\}$.

Similarly, we compute the CPU allocation for the CP VNFs. However, in this case $\tau^{r,s}_{proc}$ is derived from Eq. (2.13) as the remaining delay budget neglecting $\tau^{r,s}_{transp}$ in compliance with the MILP formulation.

It is worth mentioning that while inverting Eq. (2.25), we use the same piecewise linearisation reported in Appendix 2.4.3 to directly compare the GA and MILP approaches. Notice how we again rely on the same M/M/1 queue model used in the MILP case. However, also in this case the model

Algorithm 1 Genetic Algorithm

Input: $(G, N_{slc}, N^{r}_{ser}, N^{r,s}_{vnf}, N^{r,s}_{l})$
Output: SFC placement and resource allocation

```
 1: $Pop \leftarrow$ generate initial population according to $\mathrm{P}^{r,s}_{g,d}$ as in Eq. (2.24)
 2: $\forall I \in Pop$ call $fitness(I)$
 3: while $\nexists$ feasible solution & $N_{ss\_gen} < 10$ do
 4:     Select $2 \cdot |\mathcal{N}_{xov}| + |\mathcal{N}_{mut}|$ individuals
 5:     $\mathcal{N}_{xov} \leftarrow$ Run crossover operator over the first $2 \cdot |\mathcal{N}_{xov}|$ individuals
 6:     $\mathcal{N}_{mut} \leftarrow$ Run mutation operator over the latter $|\mathcal{N}_{mut}|$ individuals
 7:     $Pop \leftarrow \mathcal{N}_{elite} \cup \mathcal{N}_{xov} \cup \mathcal{N}_{mut}$
 8:     for $I \in \mathcal{N}_{xov} \cup \mathcal{N}_{mut}$ do
 9:         $I \leftarrow fitness(I)$
10:     end for
11: end while
12: Return: $I$ with minimum fitness
    Function $fitness(I)$
13: compute $\Phi^{l}_{e}$, $\zeta^{r,v}_{d}$, $M^{r,v}_{d}$
14: compute $C^{I}$ and $C^{r,s}$
15: if AGA is being executed then
16:     update $\mathrm{P}^{r,s}_{g,d}$ according to Eq. (2.27)
17: end if
18: Return: $I \leftarrow I \cup [C^{r,s}, C^{I}]$
```

can be generalised for using any technique returning the CPU allocation for VNF given a maximum allowed processing delay and a certain aggregated traffic. Finally, $M^{k,v}_{d}$ is computed following the same criteria used by the MILP in Eq. (2.9).

In step 14 of Algorithm 1, the fitness function computes C^{I} and $C^{r,s}$. C^{I} is the overall solution cost computed using Eq. (2.1), plus a penalty factor calculated using $\max\{\Lambda_d, \Lambda_e\}$ multiplied by the amount of resources allocated on nodes and links that exceed their capacity. Similarly, $C^{r,s}$ accounts for the same costs and penalty but only considering the resources related to the service s. The main difference between DGA and AGA is captured in steps 15-17 where the probability $\mathrm{P}^{r,s}_{g,d}$ is updated as described at the end of this section. Finally, step 18 appends C^{I} and $C^{r,s}$ to the individual and returns it.

2.5.3 GA Procedure

The GA procedure is exemplified in Algorithm 1. In steps 1 and 2, the initial population is generated and the fitness function is evaluated over the individuals. Steps 3-12 represent the main evolutionary loop which is terminated when at least one feasible solution exists (i.e. the penalty calculating C^I is zero) and the algorithm is in a steady state (i.e., no new better solutions for at least 10 generations).

In Step 4, the individuals are selected from the population to perform the crossover and mutation operator. This selection uses the well-known tournament selection where an individual is selected by randomly picking two individuals in the population and selecting the best one with minimum C^I.

In Step 5, the set $\mathcal{N}_{xov}$ is filled with new individuals using the crossover operator. For every pair of input solutions the crossover operator merges them selecting, for each service, the placement association [$\Phi_g^{r,s}$, $\Phi_d^{r,s}$] contained in one of the input solutions according to the probability $P^{r,s}$. This probability, is computed by rescaling C^I and $C^{r,s}$ in the interval $[0, 1]$ and multiplying them. The rationale behind, is to combine the two input solutions by selecting the placement associations balancing the cost to place a service with the overall solution cost. This operator is exemplified in Figure 2.3(b), where the selected placement associations contained the two individuals are highlighted in grey.

In Step 6, the set $\mathcal{N}_{mut}$ is filled with new individuals using the mutation operator. For each selected individual, the mutation operator randomly selects N_m pairs [$\Phi_g^{r,s}$, $\Phi_d^{r,s}$] and finds a new placement association accordingly to $\mathrm{P}_{g,d}^{r,s}$. Where N_m is an input to the problem and is expressed as a percentage over the genome length. This operator is exemplified in Figure 2.3(c), where the placement associations highlighted in grey are the ones selected for mutation.

In Step 7, the new population is generated considering a set of elite individuals $\mathcal{N}_{elite}$ and the new generated individuals in $\mathcal{N}_{xov}$ and $\mathcal{N}_{mut}$. The set $\mathcal{N}_{elite}$ is composed by the 5% of the best individuals in the previous generation.

Finally, the creation of one generation is terminated by calling the fitness function over the new individuals in Step 9.

2.5.4 Adaptive GA (AGA)

The drawback of selecting placement associations following the strategy described in Eq. (2.24), is that the algorithm's performance may vary depending on the traffic demand. We propose a modified version DGA aiming at solving the issue by learning the probability $\mathrm{P}^{r,s}_{g,d}$ while the population is evolving. To learn this probability we use the iterative approach shown in Eq. (2.27).

$$\mathrm{P}^{r,s}_{g,d}(t+1) = \mathrm{P}^{r,s}_{g,d}(t) + \alpha(R + \gamma \cdot max\{\mathrm{P}^{r,s}_{g,d}(t)\} - \mathrm{P}^{r,s}_{g,d}(t)), \tag{2.27}$$

where α is the a learning rate, γ is a forgetting factor, and R is a reward function. In step 16, every time a new solution is produced by either the crossover or mutation operators, Eq. (2.27) is evaluated by computing the reward R as $(C^I_{prev} - C^I)/C^I_{prev}$, where C^I_{prev} and C^I are the fitness value of the previous and current solution respectively. For the crossover operator, C^I_{prev} is the fitness value of the best solution in input.

2.6 Performance Evaluation

Table 2.4: Parameters of the E2E services in each slice of a vertical.

Request type	**UP/CP lat.**	**CP events**	**Data rate**	**# of req.**
URLLC	10/10 ms	$700 - 900$ $[ev/h]$	$10 - 50$ Mbps	$1 - 3$
eMBB	20/100 ms	$400 - 600$ $[ev/h]$	$100 - 200$ Mbps	$1 - 2$
mMTC	100/500 ms	$80 - 120$ $[ev/h]$	$1 - 2$ Mbps	$5 - 10$

Table 2.5: Cost parameters for the objective function.

Objective type	**edge-DC**	**regional-DC**	**cloud-DC**	**FH**	**BH**	**Mig**
Obj-Cost	1e6	1e3	1	1	1	1
Obj-Bwt	1	1	1	1e3	1e6	1
Obj-Mig	1e3	100	1	1	1	1e6

This section provides the simulation environment details and compares the proposed MILP methods for the Shared, and Dedicated UPF approaches. After that, we compare the proposed GA methods with the optimal solution obtained from the MILP formulation.

2.6.1 Simulation Environment

Our considered mobile network is composed of 7 nodes, out of which one is a cloud-DC, two are regional-DCs, and four are edge-DCs. Four gNBs are assumed to be co-located with the edge-DCs, and there is one gNB co-located per edge-DC. The CPUs available at the nodes are 64, 24, and 4 for cloud-DC, regional-DC, and edge-DC, respectively. Each regional-DC is connected to the cloud-DC via a BH link with 2 Gbps of capacity. Conversely, each edge-DC is connected to a regional-DC with a FH link of 1 Gbps of capacity. Each gNB is assumed to support a maximum throughput of 1 Gbps. Further, we introduce a propagation delay of 0.2ms for FH and 1.5ms for BH links, which respectively corresponds to a distance of about 40km and 300km considering the signal propagation in optical fibers. Finally, for sake of simplicity we consider a RAN delay (τ_g) equal to a fixed transmission time interval (TTI) of 1ms. The representation of the network graph is reported at the bottom of Figure 2.2.

The costs defined for the three objective functions are reported in Table 2.5. We consider a fixed VNF migration cost for the sake of simplicity. However, it is possible to consider specific migration costs for one VNF type, as well as specific costs to migrate a VNF from one specific node to another. This would allow, for example, to consider higher VNF migration costs between edge-DCs, where migrations are expected to be more problematic in terms of processing and time. For the CP functions, we consider a CPU cost of $1e4$, 100, and 1 to place them at edge-DC, regional-DC, and cloud-DC, respectively, while disregarding the cost for bandwidth consumption. This is justified since the traffic generated by the CP is negligible compared to the traffic generated by UP.

In the simulations, each vertical is requesting a set of services accordingly to the parameters reported in Table 2.4. We assume 10 simulation intervals, and in each interval, a new vertical joins the scenario making new service requests to be deployed. For each service type, a vertical requests a random number of services (i.e., service coverage) within the intervals specified in the last column of Table 2.4. Overall, at iteration 10 there are 120 service requests to be deployed. The E2E service latency and throughput requirements are taken from [2] and compliant with 3GPP Releases 14-16. The serving capacity of an UPF and an APP is assumed to be 1Gbps for each allocated CPU. Conversely, for the CP VNFs we assume that each service is generating a number of CP events as reported in Table 2.4, while an AMF and an SMF can respectively serve 125000 and 250000[ev/h] [58]. The traffic generated by the CP is calculated assuming that for each CP event 6 packets are transmitted in average with a packet length of 192 Bytes.

The simulation experiments are performed in Matlab and Cplex [10] as the back-end MILP solver. The reported results are taken from the simulations conducted on a standard laptop with Intel Core i7-8665U CPU @ 1.90GHz $\times$ 8 with 16 GB RAM.

2.6.2 MILP Results

This section presents the results acquired from the proposed MILP and GA-based methods. First, we discuss the MILP results in terms of network load and VNF migrations for both the Shared and Dedicated UPF approaches (Figure 2.4, 2.5, and 2.6). Next, we provide more detailed information about the VNF placement and resource allocation pattern achieved by the proposed MILP in comparison with the proposed GA-based approaches (Figure 2.7). Specifically, we compare the optimal solution obtained from the Dedicated UPF approach with the DGA and AGA by looking at the overall CPU utilization for the three service types in the different computing nodes, namely the edge, the regional, and the cloud DCs. Since the aim of Figures 2.4, 2.5, 2.6, and 2.7 is to observe the placement and network load behaviour while increasing the number of verticals in the scenario, the results are referred to one representative scenario. On the contrary, the results presented in the other figures are averaged over 20 simulation runs using the parameters reported in Table 2.2 to generate the traffic demand in the scenario. It is worth to mention that in Figure 2.7 we report the

CPU utilization for each service type taking into account both UPF and application VNFs. This is done for a better comparison of the placement behavior for different service types since we focus on the Dedicated UPF approach where the UP is dedicated to each service type. Furthermore, we also report the overall CPU utilization induced by the CP function.

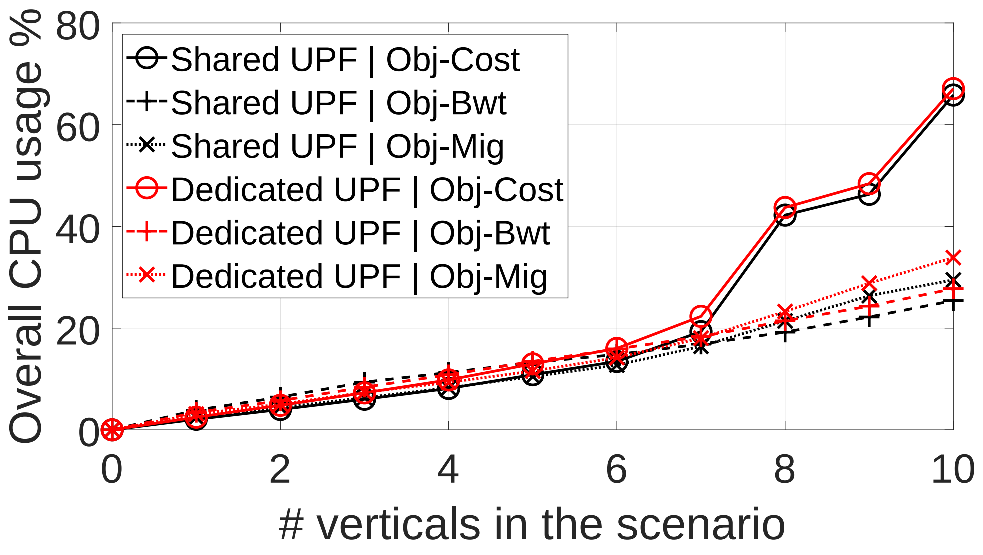


Figure 2.4: CPU utilization for the MILP Shared and Dedicated UPF.

CPU Utilization

Figure 2.4 depicts the evaluation results derived from the proposed MILP objectives in terms of overall CPU utilization in the network. The overall CPU utilization is computed as the number of CPU cores allocated to the VNFs divided by the total available CPUs in the network.

As illustrated in Figure 2.4 the Shared and Dedicated UPF approaches behave similarly in terms of CPU utilization for all the objective functions. Although the cost minimization objective (*Obj-Cost*) demonstrates the highest CPU utilization in percentage, the allocated resources are the cheapest resources available, which is shown in Figure 2.7(a). The main reason behind utilizing more resources by the *Obj-Cost* objective is that this objective function always prefers to utilize the cheapest resource available, which in our case can be found in the core. Figure 2.7(a), presents that regardless of the service type and as long as the latency requirements

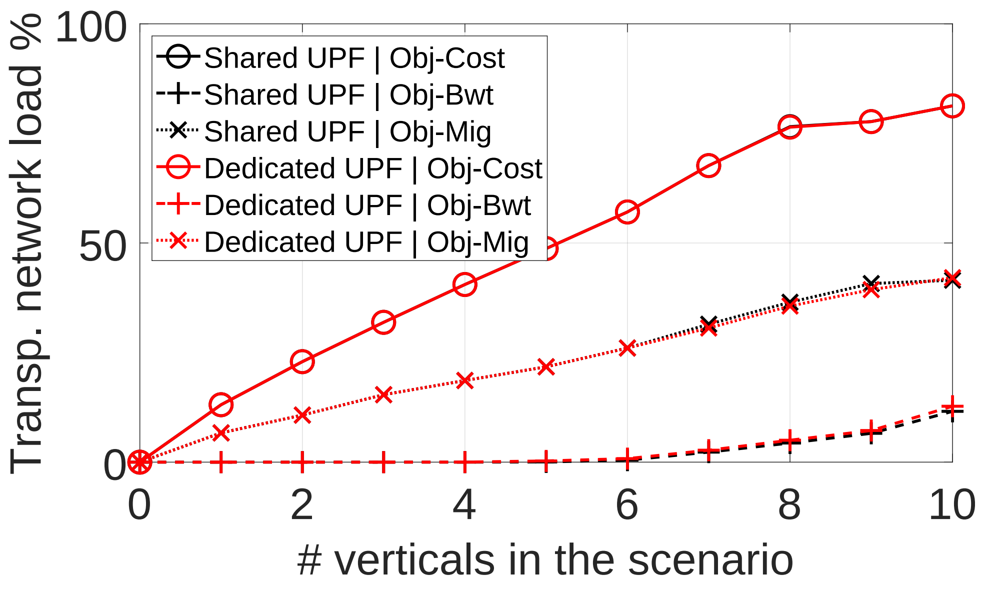


Figure 2.5: Link utilization for the MILP Shared and Dedicated UPF.

of the services are respected, all the services are hosted in the core DCs. However, gradually, with the increase in the number of services hosted in the cloud-DC, the transport delay for the services increased. By reaching around eight verticals making slice/service requests, the *Obj-Cost* objective cannot place all the services at the core. Therefore, in order not to violate the delay constraint, URLLC services are placed in the regional-DC, after iteration 8, and in the edge-DC, after iteration 9, which is when the cloud-DC resources are not sufficient to host all the services which would have minimised the overall objective cost.

Opposing to the *Obj-Cost* objective, the bandwidth minimization objective (*Obj-Bwt*) utilizes a smaller number of CPUs. This objective function does not account for the cost of CPU resources and only cares about the amount of link usage. Therefore, it tries to place all the VNFs at the network edge as long as there is enough resource available in order to avoid link utilization. The consequence of keeping the VNFs close to the network edge is less usage of the transport network and, simultaneously, a less required CPU allocation to meet the services' E2E delay requirement. As can be observed in Figure 2.7(b) all the services types are initially placed at the edge. Starting from iteration 5, when the resources at the edge-DCs

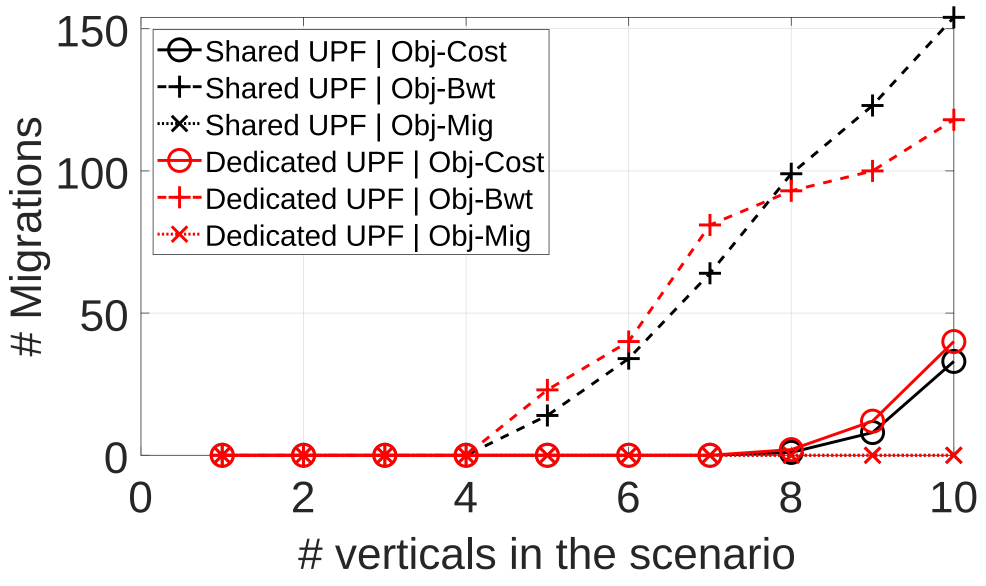


Figure 2.6: VNF migrations for the MILP Shared and Dedicated UPF.

are exhausted, the mMTC and URLLC services that can tolerate the latency requirements and have lower bandwidth requirements are placed at the regional-DCs. It is worth noticing that the VNFs for the CP are always placed in the cloud-DC because of their negligible bandwidth utilization.

As mentioned earlier, the *Obj-Mig* objective function aims to minimize the number of VNF migrations that happen in the network. In this regard, we define a VNF migration cost in the objective function in order to avoid excessive and unnecessary migrations. We also define a small cost for resource usage in order to avoid random behavior of the placement. On the one hand, the objective tries to balance the CPU cost (e.g., placement toward the cloud) with the link cost (e.g., placement toward the edge). On the other hand, the high VNF migration cost minimizes the number of VNF migrations that could happen while adding new services. Looking at Figure 2.7(c), initially both eMBB and URLLC services types are placed in regional-DCs, while mMTC is placed at the cloud-DC because the CPU cost has more influence than the cost for link utilization. While increasing the number of overall slice/service requests in the network, the new URLLC services are placed at the edge-DC in order to fulfill the E2E requirement while not migrating the already placed VNFs in the regional-DC. We can observe from Figure 2.4 that also with this objective, the Shared and Dedicated UPF approaches show similar behavior. Moreover, results show a less overall CPU utilization in the network compared to the *Obj-Cost* case - notice how URLLC is the highest source of CPU consumption when placed at the cloud-DC as shown in Figure 2.7(a).

Link Utilization

In Figure 2.5 we report the average link utilization in the network obtained from the MILP formulation using the Shared and Dedicated UPF approaches for all the objective functions. We can observe that both the approaches have very similar behavior in terms of link utilization, which is the consequence of a similar VNF placement and thus aggregated traffic on the transport links.

As expected, the *Obj-Cost* objective has the highest resource utilization both in terms of CPU usage, as mentioned earlier, and bandwidth usage since all the VNFs are preferably placed at the cloud-DC. Conversely, the *Obj-Bwt* objective shows the minimum link usage since the VNFs, and especially the ones to realize the SFCs of services with higher data-rate, are preferably placed at the edge-DCs without using the transport network. Indeed, for this objective the link utilization is visible after iteration 5 when, as it is obvious from Figure 2.7(b), the URLLC service is placed at the regional-DCs. This is feasible since the eMBB is placed at the edge-DCs to minimise the transport delay and the low transport network delay allows not to violate the stringent E2E latency requirement for URLLC. The *Obj-Mig* remains between the *Obj-Cost* and *Obj-Bwt* objectives, since it looks for a trade-off between CPU and bandwidth usage. Indeed, we can observe from Figure 2.7(c), that eMBB service types are placed at the regional-DCs and thus exploiting the FH resources. Conversely, eMBB service types are always placed in the cloud-DC and edge-DC in the *Obj-Cost* and *Obj-Bwt* objectives, respectively.

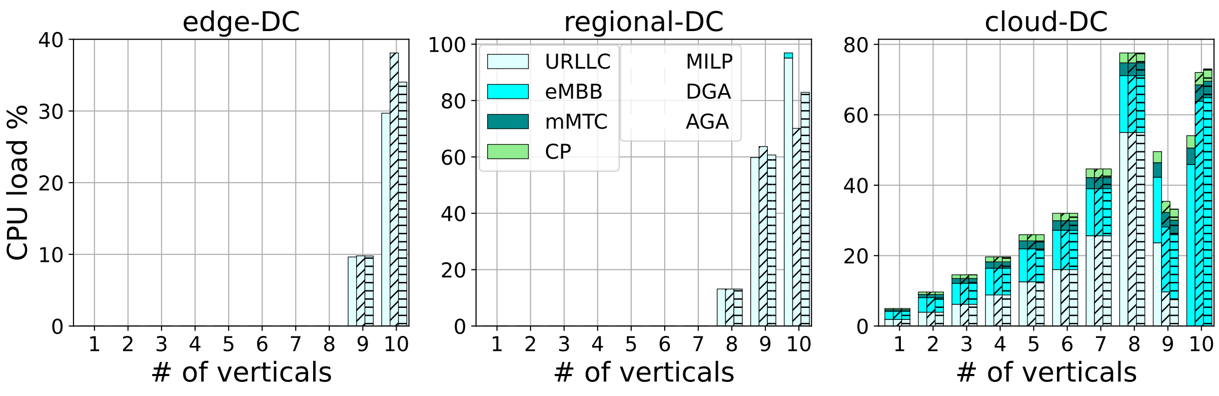
VNF Migration

A VNF is migrated when it moves from one hosting DC to another DC compared to the previous allocation. An excessive VNF migration leads to QoS degradation for the users in the network. When a VNF migrates from one DC to another, it requires a considerable time to be instantiated on the new host and can lead to unsatisfactory results. In this regard, we proposed an objective function that avoids excessive migration of VNFs in order not to violate QoS requirements of the end-users. As shown in Figure 2.6, the *Obj-Bwt* objective shows the worst performance since several VNFs needs to be migrated when the CPU capacity of edge-DCs is exhausted. Comparing shared/dedicated UPF approaches, notice how the latter initially shows higher migrations since, being the UP dedicated to each service, the number of deployed VNFs migrated is higher (embeddings 5-7). However, the number of migrations becomes lower in embeddings 8-10 since, with equal bandwidth utilization, and having independent SFCs for each service UP, results in more flexible deployment of the new required VNFs. In the *Obj-Cost* objective, the number of migrations rapidly increases in iterations 9 and 10, when URLLC services need to be placed toward the gNB to meet the E2E latency requirements. While pursuing the minimal cost for CPU usage, new arriving services demanding a high amount of CPU are preferably placed far from the edge. However, this could happen at the cost of migrating already deployed services toward the edge. Finally, the *Obj-Mig* objective has the minimum number of VNF migrations that happened in the network, effectively demonstrating the purpose of this business logic.

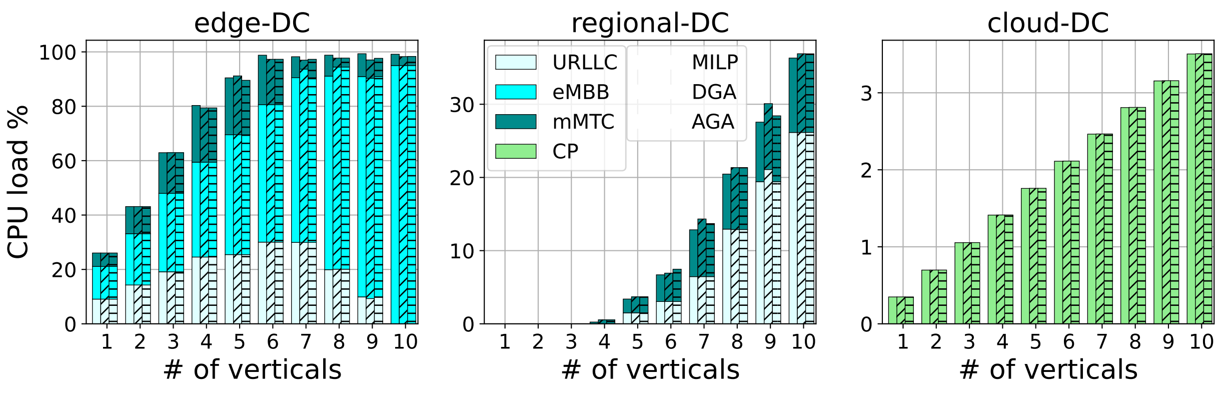
2.6.3 DGA and AGA Performance

To evaluate the convergence and the performance of the DGA and the AGA we compare their best solution against the optimal counterpart obtained by running the MILP formulation for the Dedicated UPF approach. We run the DGA and AGA using the same scenario as the MILP in terms of network, request parameters and objective function, and repeating the simulations 20 times using different random seeds.

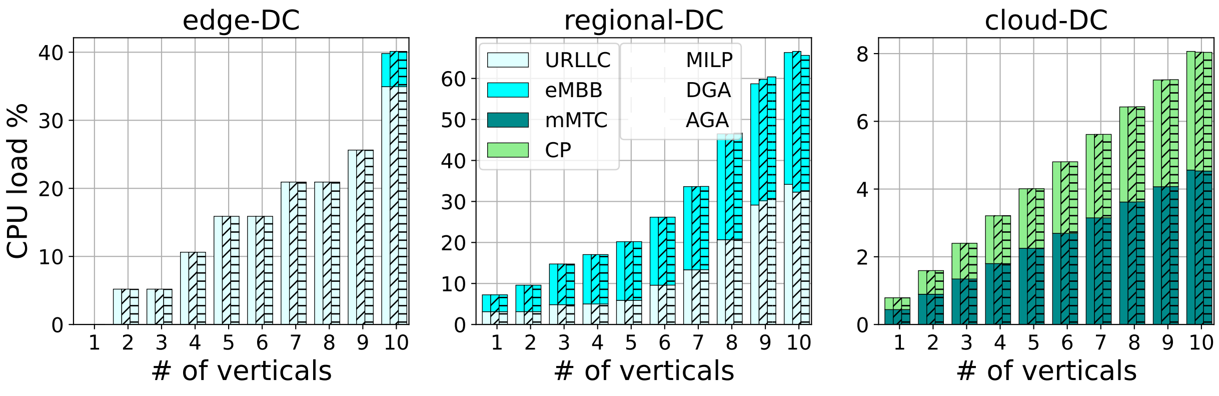
As shown in Figure 2.7(a), for the *Obj-Cost* the DGA and AGA follow the MILP behavior by first placing all the services at the core and then utilizing regional and edge DCs as the number of requests increases. It is worth noticing that in the AGA achieves a better placement compared to



(a) CPU utilisation with Obj-Cost.



(b) CPU utilisation with Obj-Bwt.



(c) CPU utilisation with Obj-Mig.

Figure 2.7: CPU allocated to the UP (i.e., UPFs and APPs) for each service type in each network layer using MILP (Dedicated UPF approach), DGA and AGA.

the DGA, which is closer to the optimal especially looking at the CPUs allocated in the edge-DCs and regional-DCs - the most costly ones. Results for DGA and the AGA are even more promising in case of *Obj-Bwt* and

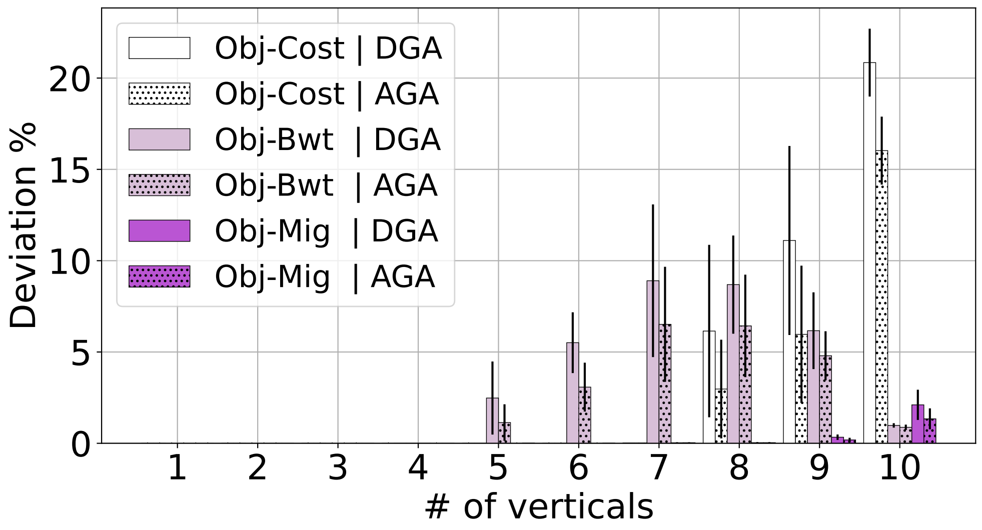


Figure 2.8: DGA and AGA objective relative error with respect to the MILP Dedicated UPF approach.

Obj-Mig. Indeed, looking at Figure 2.7(b) and 2.7(c) both the algorithms are very close to the optimal CPU allocation acquired by the MILP counterpart. Therefore, the results acquired from the evaluations demonstrate that DGA and especially our proposed AGA algorithm, follow very closely the optimal behavior.

In Figure 2.8 we show the DGA and AGA deviation from the MILP optimal solution by plotting the relative error in percentage of objective function (2.1). The 95% confidence interval is also included. Notice how, before vertical #5 is added in the scenario, the GA approaches shows a negligible deviation from the optimal solution. Also, it is worth highlighting that in all the cases, both the GA approaches converged to a feasible solution, which means there is no service rejection.

As shown in Figure 2.8, the *Obj-Cost* has the highest deviation compared to the other two objective functions. This is because several placement combinations need to be evaluated to keep the VNFs toward the cloud while increasing the number of services and consequently evaluating the transport delay. However, both DGA and AGA follow the behaviour obtained by the optimal placement as shown in 2.7(a). Moreover, the AGA approach demonstrates 5% improvement compared to the DGA thanks to its ability to adapt to the scenario by improving the way the new solutions are generated.

In the *Obj-Bwt* objective, the GA-based approaches always show a relative objective error below 10%. Again, the AGA exhibits a behavior closer to the optimal, improving the DGA solution by a 2.5%. Notice how, the relative objective error increases for iterations 6, 7, and 8 and then decreases for iterations 9 and 10. This is because initially, for the GAs it is more difficult to find the optimal placement combination to accommodate the highest number of services at the edge-DC. Later, when the number of services increases and eMBB almost occupy edge-DC resources, it is easier for the crossover and mutation operators to converge close to the optimal solution by either discarding unfeasible solutions from the population or finding a good combination of services that can stay at edge-DC without exceeding the available CPU.

In the *Obj-Mig* objective, both DGA and AGA show very low objective relative errors and this result is supported by the very similar results in terms of CPU load as shown in Figure 2.7(c). However, looking at Figure 2.8 also in this case, the AGA is closer to the optimal value. Finally, even if not shown due to the lack of space, it is worth mentioning that both the DGA and AGA approaches achieve zero migrations similar to the MILP case. This is demonstrated by the low objective relative error obtained in Figure 2.8.

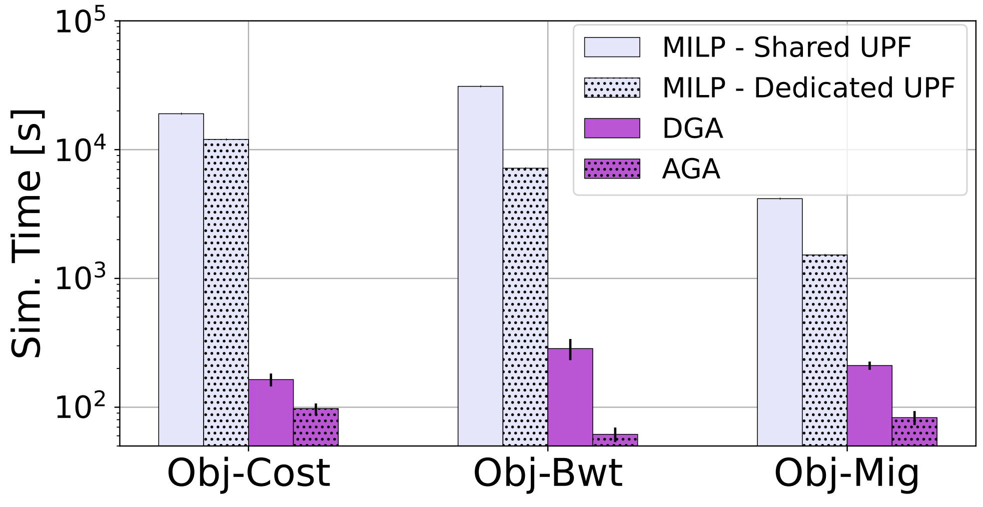


Figure 2.9: Algorithms execution time to embed 10 verticals.

Execution Time

As mentioned earlier, the main reason behind proposing the GA-based approaches is to reach solutions close to the MILP counterpart and at the same time mitigate the execution time to a great extent. In this regard, in this section, we present the results acquired from running the experiments and make a comparison between the GA-based approaches with the MILP solutions. Results in terms of execution time to embed 10 verticals in the network (i.e., when the search space is the widest considered in this study) are collected in Figure 2.9 where we compare the MILP formulation for both Shared and Dedicated UPF approaches, the DGA and the AGA. It is worth highlighting that the Shared UPF approach results in being more time-consuming than in the Dedicated UPF case. The reason is that by considering the branch-and-bound procedure of the MILP solver, it gets more complicated when different services having different KPIs are mapped on the same UP. As expected, both DGA and AGA outperform the MILP formulation reducing the execution time by 1 to 2 order of magnitude.

Interestingly, there is a low price to pay for this time reduction as demonstrated by the low deviation from the optimal solution (Figure 2.8) and the small difference in the CPU allocation (Figure 2.7). Further, the AGA approach behaves better than the DGA achieving better convergence in a much shorter time scale. Indeed, the ability of the AGA to adapt the way in which new individuals are mutated allows finding better solutions while spending less time in evolving the population generation by generation.

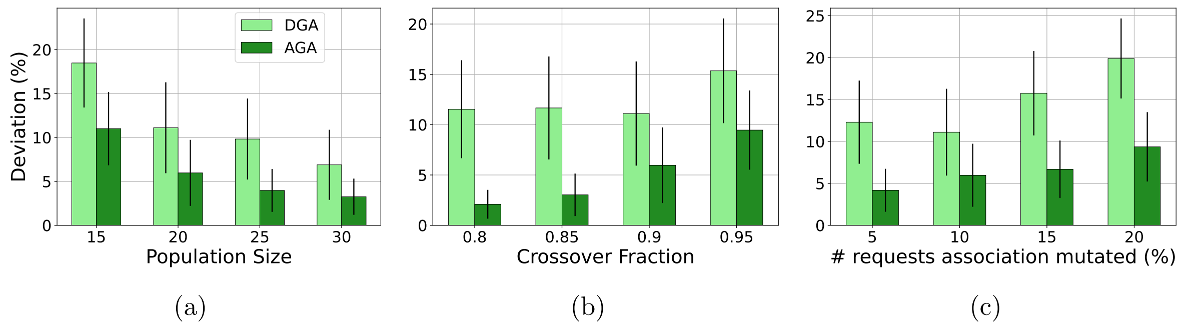


Figure 2.10: Objective function relative error for DGA and AGA varying the algorithm parameters.

DGA and AGA Sensitivity to Parameters

In order to investigate the performance of the DGA and AGA in terms of convergence, it is important to study their behavior varying the algorithm parameters. This is captured in Figure 2.10 which reports the deviation in percentage to the optimal solution and the bars represent the 95% confidence interval obtained by running the experiments 20 times. The results are related to the *Obj-Cost* objective, which has been shown to be the most difficult to solve.

In Figure 2.10(a) the objective relative error is shown by varying the number of solutions in the population. As it is expected, by increasing the population size, it is easier for both the GA approaches to find a good solution since there are more chances for the crossover and mutation operators to generate better solutions. However, in all the cases, the AGA is closer to the optimal by halving the objective error obtained by the DGA thanks to its capability to adapt the mechanism to generate new solutions.

The crossover fraction is usually a sensitive parameter for the GA since a lower value means a higher number of mutated individuals created at every generation. The behaviour varying this parameter is shown in Figure 2.10(b). In the DGA case, we increase the randomness in generating new solutions by lowering this value. In this case, the DGA performances are almost constant since the high crossover fraction allows the crossover operator to properly merge the individuals in the population and generate good solutions. However, we can see that the AGA outperforms the DGA thanks to its ability to learn how to generate new solutions.

In the mutation operator, N_m defines the number of mutated request associations when generating a new individual. In Figure 2.10(c) we show how the two GA approaches behave by varying N_m from 5 to 20% of the overall number of service requests in the scenario. It can be observed that, in general, a high value of N_m brings to worst performances since too much randomness is added in generating new solutions. However, also, in this case, the AGA outperforms the DGA, and it is more resilient to the variation of this parameter.

Chapter 3

Measurement-Based Model of MEC Nodes in the 5G System

3.1 Motivations

In the literature, several works can be found that propose orchestration techniques based on theoretical models for Edge Nodes' behaviour. Some provide promising preliminary approaches to compare different orchestration techniques in constrained cloud-edge scenarios. In [11] authors leverage simple delay models to characterise the VNF processing delay, which is then used to allocate CPU resources by imposing E2E service QoS constraints. More complex models are considered for the same objective in [59] and [6] using M/M/1 and M/D/1 queuing models, respectively. However, while enabling interesting preliminary evaluations, these models are not precise enough for modelling the behaviour of real edge nodes running actual services. To fill this gap, some authors started considering introducing experimental measurements in their models. A good example can be found in [31], where authors conducted experiments in a real setup and collected data useful to derive a model of the virtualisation overhead. This model was then adopted for defining a VNF placement algorithm. However, the drawback of this approach lies in its high dependency on the virtualisation technology used during experimentation. Focusing on a face recognition scenario, in [60], authors leverage experimental results to derive a theoretical model relating the CPU usage, network load and packet delay with the Edge node performance. The coexistence and parallel execution of the face recognition application in conjunction with other heterogeneous services were not considered, as well as the impact of UPF behaviours. Both these works give an idea of the behaviour of an application in a constrained scenario, but the findings are context-specific and cannot be easily generalised. In [61], the authors introduce Simu5G, a simulator of the 5G Core combined with physical edge hosts to evaluate and experiment MEC technologies. In this approach, the E2E 5G network is simulated by design and does not allow to emulate a MEC, which includes the virtualised UPF. OpenLEON is proposed in [62] as an E2E emulator spanning from mobile users to the edge data centre. To the best of our knowledge, this approach is one of the works closest to the one presented in this chapter. The authors leverage srsLTE and Containernet to design a realistic radio access implementation with a virtualised environment emulating a 3-tier edge data centre hosting the core network and services. Various applications are tested, focusing on analysing network performance in different LTE channel configurations.

The goal of this work is to enable the definition of a measurement-based model for studying MEC nodes and MEC-supported application performance and existing performance trade-offs without requiring detailed knowledge about the corresponding MEC node setup and hardware/software configuration. To this aim, in this section we propose two testbeds. One is emulated and enables the evaluation of MEC applications under realistic network conditions also emulating MEC nodes with different cumputing resources. The second test-bed is a physical one that focuses on power consumption measurement under different virtualization technologies.

3.2 Emulated Test-bed: Proposed Methodology

To study the performance of the applications deployed at the Edge and enabled by the MEC system, we designed a 5G-enabled resource-constrained MEC deployment on an emulation environment based on Open Source platforms. Comnetsemu is selected as the simulation platform, being capable of extending Mininet and Containernet with a nested virtualization approach based on DockerHosts. On top of the Comnetsemu network emulator UERANSIM and Open5GS are deployed to implement the 5G System, plus some software utilities to collect data and generate variable traffic and CPU load. The resulting architecture of the emulator environment is represented in Figure 3.2. ComNetsEmu enables the deployment of an SDN-enabled transport network connecting various hosts emulating the main components of the 5G network: the radio access network, the 5G CP, and the MECs. The scenario deployed for the performance evaluation is explained in this section and detailed in Figure 3.2.1.

Proper tools have been implemented to control the scenario, and to easily deploy the network, configure the RAN and the 5GC, and apply the desired APPs behaviours. Those tools implement functions to collect the status of the various network components (e.g., container CPU status through the Docker stats exposed through the Docker Client APIs). In parallel to that, APP tools have been implemented to emulate services running on the MECs and consumed by APP clients of the user equipments (UEs). Collection of the measurements is performed using the Publish/Subscribe infrastructure provided by Redis[1]. The code developed to deploy the emulation environment and to study MEC performance described in this chapter is freely available online [2] to facilitate the reproduction of the achieved results and to support further research activities on this subject.

In the remainder of this section, the system's building blocks and functionalities are described in more detail.

3.2.1 Emulated network deployment

Proper deployment of the 5G System should be supported by a sound environment integrating both SDN and NFV. Indeed, the Comnetsemu environment natively supports both SDN and NFV through a $docker-in-docker$ framework. All the network functions and applications are deployed within DockerHosts as resource-isolated hosts. Within a DockerHost, APPs and network functions can be emulated by either running processes directly or by deploying APPContainers exploiting the $docker-in-docker$ concept -

[1]`https://hub.docker.com/r/redislabs/redistimeseries`

[2]`https://github.com/RiccardoFedrizzi/networking_letter`

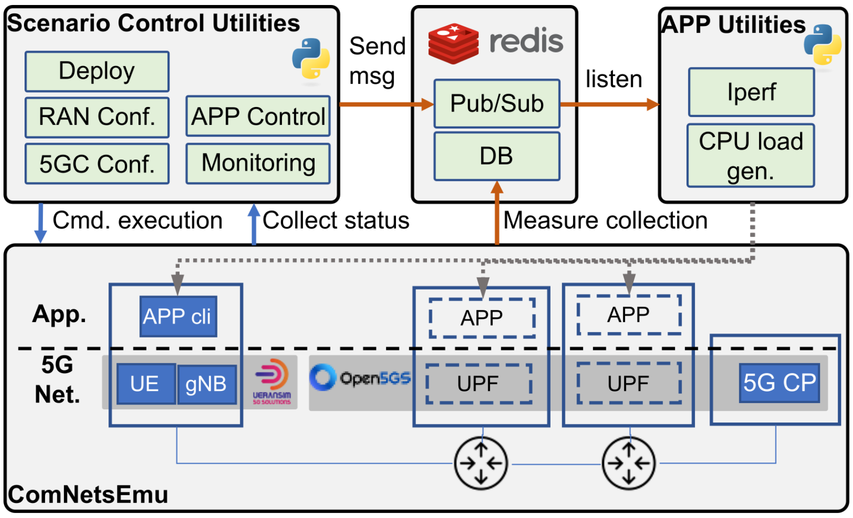


Figure 3.1: High-level architecture of the emulation environment.

see solid and dashed-line boxes in Figure 3.2, respectively. APPContainers add a second level of virtualisation to emulate containerised applications and network functions running on a resource-limited host.

Since by default Docker does not apply CPU limitations, we used Docker built-in functions to limit CPU utilisation for specific MEC Docker containers. We set two parameters to control the CPU allocation: the CPU period, CPUperiod, and the CPU quota, CPUquota. CPU quota specifies how much CPU time (in microseconds) the container can use per CPU period. After a container consumes all its CPU quota, it is throttled for the remainder of the CPU period.

To ease this process, we implemented a utility function whereby the CPU constraints of MEC can be set by specifying the maximum percentage of the overall system CPU that can be used by all the VNFs deployed in the specified MEC.

3.2.2 5G Core Network

In this study, an Open Source implementation of the 5GC[3] is used to deploy CP and the UP functions separately, and to create different Data Networks in each MEC to reach the APPs. To the best of knowledge of

[3]`https://open5gs.org`

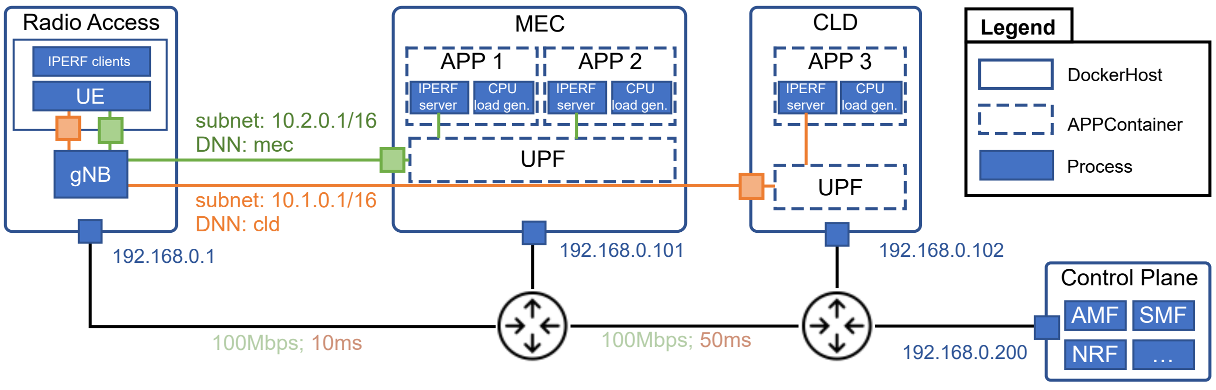


Figure 3.2: Emulation components and their integration.

the authors, such implementation of the 5GC allows emulating a realistic MEC environment, while avoiding monolithic 5GC deployment as in [62].

All the CP functions are started as processes inside a dedicated DockerHost for the sake of simplicity. Their configuration is done through YAML files which are updated on the fly based on the desired scenario. In particular, the SMF configuration specifies where to reach the UPFs, their Data Network Name (DNN) and the associated sub-network. Afterwards, an APPContainer running the UPF is deployed in each MEC node and configured to associate a new TUN interface, created on the fly and usually called ogstun, with a DNN and its sub-network.

Two different logical networks are created to allow reaching the APPs, depending on whether the service is provided through the MEC platform (MEC in Figure 3.2.1) or the cloud (CLD in Figure 3.2.1). The corresponding two networks are displayed in green and orange, respectively, in Figure 3.2.1. Then, the CP needs to be configured with the subscribers' information (International Mobile Subscriber Identity - IMSI, subscribed services, etc.) to allow a new UEs to connect.

3.2.3 5G Radio Access

UERANSIM[4], an open-source software simulator for the 5G UE and RAN (gNB), is used to simulate the radio access section of the 5G System. UERANSIM implements the main Radio Resource Control (RRC) procedures

[4]`https://github.com/aligungr/UERANSIM`

and the GTP protocol for the UP. It allows easing the deployment of an E2E 5G network in a self-contained emulation environment.

Being a simulator, UERANSIM does not implement the protocol stack below the RRC, preventing the emulation of scenarios in which lower layers are needed - however, this aspect is beyond the scope of this work. To avoid such limitation, it would be necessary either to use a physical RAN implementation (e.g., srsRAN), which requires rather costly external hardware and complicates the scenario definition, or more advanced network simulators (e.g., NS3), which typically do not support the required accuracy that an emulation environment might provide in terms of a realistic implementation of the UPF in the MEC and the corresponding workload. Both options are possible in the Comnetsemu environment, but the authors prefer to focus on a fully integrated and consistent scenario capable of running on a single common PC platform.

During the deployment phase, the gNB is deployed as a process in the RAN DockerHost. Once the gNB process is started, it establishes an NG Application Protocol (NGAP) connection with the AMF network function in the 5GC. The addition of a new UEs in the scenario can be easily automated thanks to dedicated control tools that have been implemented by the authors, which starts an UE and connects it by specifying a DNN. The DNN allows to discriminate between MECs or cloud connectivity, slice type, and the corresponding QoS parameters. Every time a UE connection is started, a new TUN interface is created, which allows using the 5G connection and reach the MEC APPs as shown in Figure 3.2.1. The SMF selects the IP address of the TUN interface within the DNN sub-network range. Each UE can have several active connections with different DNN and slice types. The IP addresses of an UE can be retrieved based on the triplet $[\mathrm{ID}, \mathrm{DNN}, \mathrm{SST}]$.

3.2.4 Emulated applications

To maintain generality while enabling a proper analysis of the MEC performance, we decided to classify applications into three categories: *communication intensive* applications, *computation intensive* applications and *mixed* applications.

Indeed, most MEC applications expected to fall either in the *communication intensive* category (i.e., requiring mainly a given throughput level, or offering a quality of service directly related to the achievable throughput level), *computation intensive* (i.e., requiring mainly CPU resources and performing mostly computation), or something in the middle (i.e., having multiple mixed KPIs). Indeed, the UPF used to steer the traffic to the MEC platform can be hosted and use resources from the platform itself, thus representing a clear example of a *communication intensive* service. To emulate the above application profiles, control utility functions have been implemented in the emulator in order to (*i*) control the data traffic between an UE and an APP, and (*ii*) emulate the APP computation by generating CPU load in its APPContainer.

In the next sections and overall in this chapter, we will assume that each MEC node resources should be shared among all the applications and functionalities running in such node (i.e., including the UPF and other "core" functionalities). The authors believe that this represents a more realistic situation of MEC deployment. However, the designed emulator is capable of supporting also other scenarios and enable different isolation of CPU or network resources.

Data traffic generation

Traffic generation is performed by using the Iperf network testing tool. Tests were performed by establishing a connection between an Iperf server instance running on each APP and listening on the ogstun TUN interface of the MEC, and the Iperf client on the UE side, which generates traffic towards the APP. A helper class has been defined to ease the traffic generation, which allows to select a specific slice, the protocol to use (TCP or UDP), the port, uplink/downlink traffic generation, the bit-rate to generate (in Mbps) and its duration.

CPU load generation

To generate the CPU load of an APP, we leverage the multiprocessing package to generate artificial CPU load by running one process for each online CPU core of the DockerHost hosting the APP. The amount of generated load is controlled by monitoring the container's CPU usage (using the psutil package) and regulating the rate of the artificial calculations to meet the CPU load requested. Since the psutil package returns the CPU load of the system, a re-scaling considering the resources allocated to the DockerHost is performed. By varying the CPU load requested over time, it possible to define APPs with dynamic CPU load.

3.2.5 Monitoring

To monitor the scenario and collect results we implemented control utilities to oversee the data-rate and CPU load monitoring of the desired hosts and applications. In the following, we explain more in detail how those monitors are implemented.

Data traffic monitoring

Throughput measures are collected for the overall traffic handled by a MEC as well as for each established Iperf session of the APPs. A process is deployed in each MEC node's container and listens on a dedicated Redis channel. After receiving the activation message, the process sends the throughput measures to the Redis database.

CPU Monitoring

Several tools are available for CPU monitoring (e.g., Glances[5]). However, since most of them leverage psutil, they incur the issue to disregard the containers' CPU constraints that are emulated (as previously explained). For this reason, we preferred to follow a lower level approach and leverage the Docker Client APIs to retrieve the container CPU load through the Docker stats. To realise this, we implement a utility function starting the CPU monitor for each container. Every time this function is invoked, a separate thread is started, which collects and tracks the container CPU status.

It is worth noticing that while measuring a MEC node we want to obtain its overall CPU usage. Conversely, measuring an APP CPU usage, the aim

[5]`https://nicolargo.github.io/glances/`

is to obtain its CPU usage over its hosting MEC node. For this reason, the measured CPU load of an APP is re-scaled with respect to the CPU resources allocated to its parent container.

3.2.6 Experimental Results

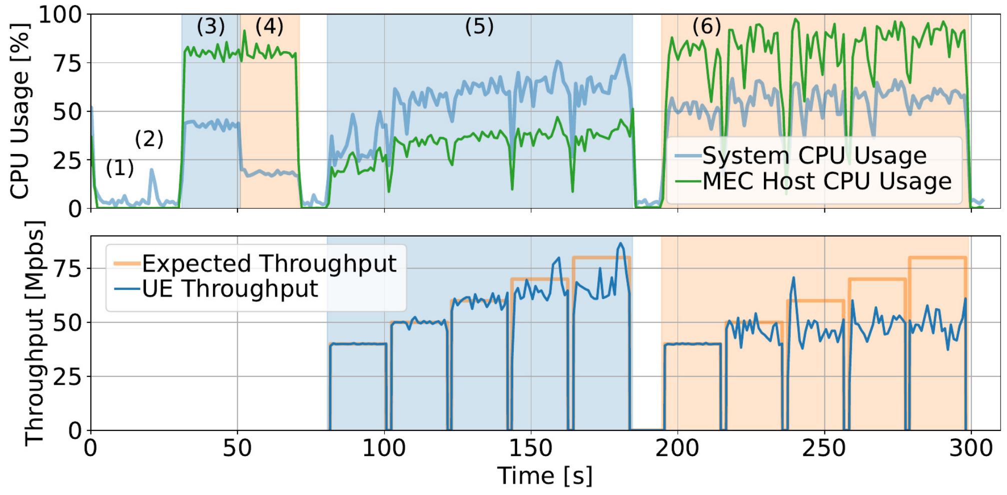
The environment used for the emulation consists of a VM with 2 CPU allocated on a standard laptop with 8 x Intel Core i7-8665U CPU @ 1.90GHz with 16 GB RAM. To calibrate our emulated environment with real hardware we use the Passmark[6] tool providing the *CPU mark*, a CPU performance measure. Figure 3.3(b) illustrates how to tune the MEC node resources to the desired value. For example, to tune MEC performance comparable to a RaspPi Model 3, allocate 37% of one CPU; for performance equivalent to three RasPis, allocate 90% of one CPU.

Scenario 1

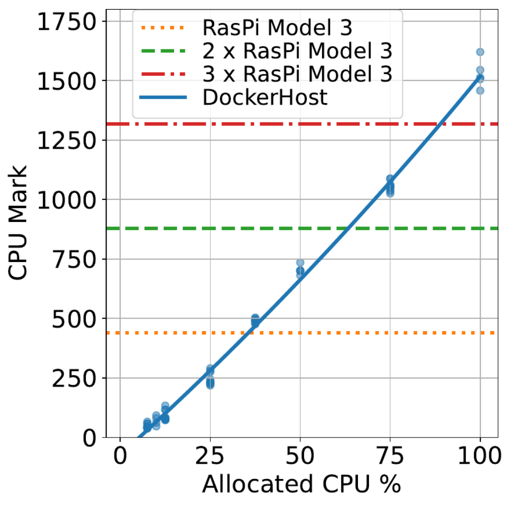
To test whether the system is working as expected and to understand the system's limitations, in this scenario, one UPF and one APP are deployed in MEC to emulate CPU load. Both applications exchange data traffic with one UE. Results are provided in Figure 3.3(a) where six zones are highlighted, corresponding to different operational setups of the system: (1) CP configuration, (2) UE deployment, (3) the APP is instructed to use 80% of the MEC CPU resources, (4) MEC container is forced to use a maximum of 37.5% of its allocated resources, (5) the UE generates from 40 to 70 Mbps of data traffic, with 1 full CPU allocated to the MEC node, (6) same as (5) but MEC resources are set to 37.5%. The corresponding throughput degradation starting from 50 Mbps of offered traffic clearly outlines the effect of the resources limitation.

A more comprehensive performance evaluation under communication intensive APPs is shown in Figure 3.3(c) where we compare the throughput handled by the MEC with its CPU usage under different MEC resource constraints. As shown before, we can observe that when the MEC CPU resources are bounded to 100 and 70%, we reach the system limit as the

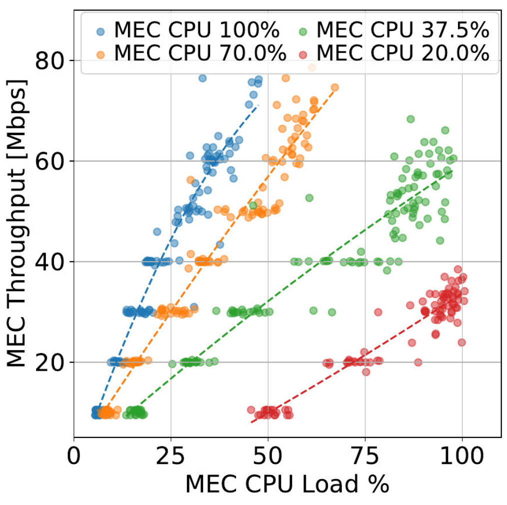
[6]`https://www.passmark.com/`



(a)



(b)



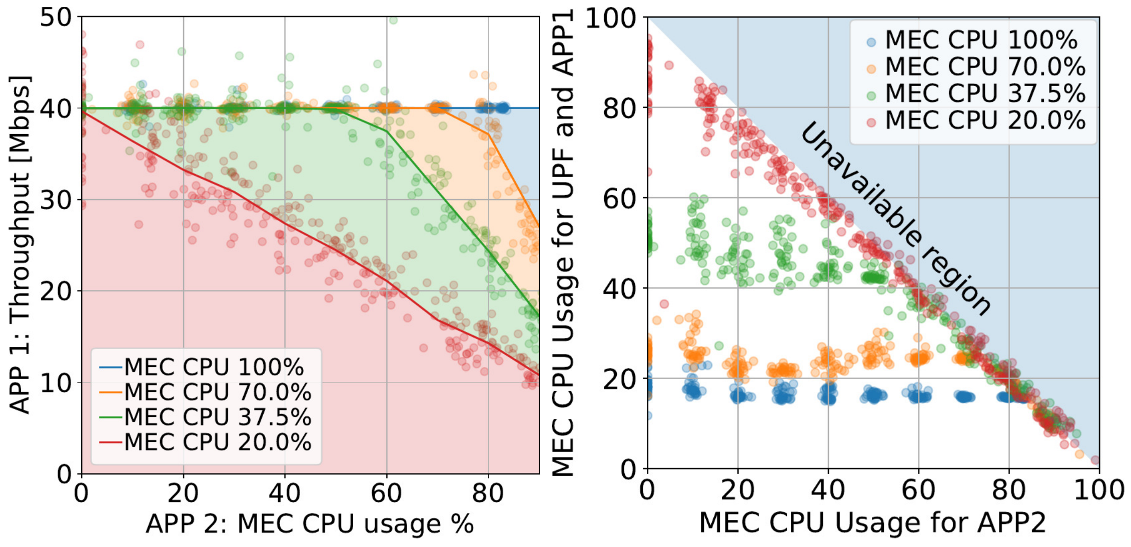
(c)

Figure 3.3: Performance evaluation of the emulated environment: (a) dynamic operating scenario;(b) performance comparison between DockerHost and RasPi Model 3; (c) throughput vs CPU load for variable resource availability (dots represent the measures, dashed lines represent the quadratic polynomial fitting).

throughput falls between 70 and 80 Mbps (the maximum level supported by the emulator deployed in a VM with 2 CPUs). Conversely, when the MEC CPU resources are bounded to 37.5 and 20% the throughput is limited by the scarcity of available MEC resources. For each MEC resource allocation, the relation between the achieved throughput and the CPU usage can be modeled with a quadratic polynomial (dashed curves in Figure 3.3(a)).

Scenario 2

To study the tradeoff between communication and computational intensive APPs, we deploy in MEC the UPF, an APP1 handling 40 Mbps of data traffic generated by one UE, and APP2 using from 0 to 90% of the MEC computing resources. We can observe from Fig 3.4(a) that, when 100% of the CPU is allocated to the MEC, the level of CPU usage of APP2 does not have impact on the achieved throughput. Conversely, with 20% of available CPU resources, we observe a relevant throughput decrease in APP1 as soon as APP2 starts computing tasks. This trade-off between CPU and data rate defines different areas of the resulting diagrams (Figure 3.4(a)): the shaded area under the curve related to a specific scenario (and corresponding resource limitations) represents the possible operating points achievable by the emulated MEC. Figure 3.4(b) effectively illustrates this treadeoff between CPU and data rate comparing the CPU usage of communication-intensive and computation-intensive VNFs. Specifically, the CPU usage of UPF and APP1 remains constant, as long as the requested CPU load from APP2 is low enough to avoid depleting MEC resources.



(a) (b)

Figure 3.4: Performance evaluation of a MEC hosting one throughput-intensive APP coexisting with a CPU-intensive APP.

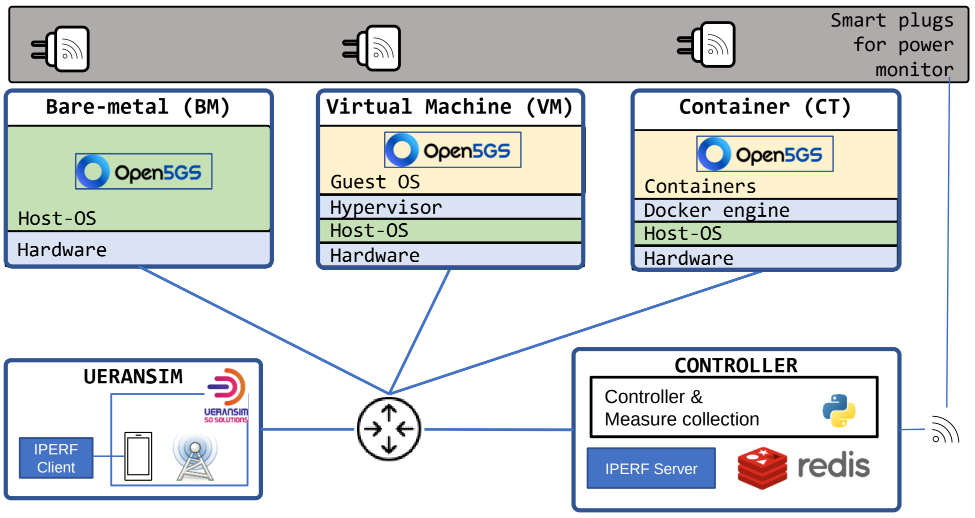


Figure 3.5: Testbed set-up for the proposed investigation.

3.3 Physical Test-bed: Proposed Methodology

For our experimentation we built a test-bed composed by 5 Intel Next Unit of Computing (NUC) connected via a fast switch, as shown in Figure 3.5. The rationale behind the setup is to demonstrate 5G-enabled MECs with different types of virtualization technologies. In each MEC we leverage the Open5GS[7] software to realise the 5G Core Network, which has been deployed with three configurations: BM, VM and CT. The three types of MECs, are realised with the same type of NUC which is equipped with a $i5-7260U$ CPU @2.20GHz. This is essential in order to have comparable results between the three types of MEC deployments. Conversely, the nodes to emulate the RAN and the controller are realised with two NUCs equipped with a $i5-1145G7$ CPU @2.60GHz. All the NUCs are connected toghether via a TL-SG105E five-ports Gigabit switch.

In the VM case, we utilize the QEMU hypervisor with KVM acceleration as it is widely accepted for its cost-effectiveness and high performance. On the other hand, for the CT deployment, we opt for Docker to create container hosting the 5GC functions. This decision was made to simplify the process and avoid unnecessary overhead that comes with other solutions like Kubernetes, since we only target a single MEC.

[7]https://open5gs.org

The RAN is established with the use of UERANSIM[8] which is an emulator designed to simplify the deployment of an E2E 5G network. Upon initiation, the UE process effectively establishes a data-plane connection between the UE and the selected Open5GS deployment. Although UERANSIM does not include the protocol stack below the radio resource control (RRC), making it unsuitable for scenarios where lower layers are needed, this limitation is deemed beyond the scope of this work. We thus resort to this solution for its simplicity.

In our experiments, the MEC nodes were subjected to various combinations of CPU and data-rate loads. The CPU load is generated using *stress-ng* allowing to set an overall CPU load, in percentage, within a NUC. Conversely, the data-rate is generated by deploying an *Iperf* client at the UE side and connecting it to the server in the controller node. Experiments are conducted by changing the CPU load from 0% to 90% and the data-rate from 100 to 700Mbps, with various experiments being performed to evaluate the performance of the system under different WPs.

The Controller node oversees the generation of experiments and the collection of measurements. It hosts an Iperf server for terminating the data traffic generated by an UE and flowing through the desired MEC. In each experiment we gather the relevant KPIs related to a MEC such as power consumption, overall data-rate, and the CPU usage. To measure the CPU usage and data-rate KPIs in the MEC nodes, we utilize the *psutil* Python package. After proper instruction from the controller, the MEC nodes report the collected measurement to the Redis[9] database hosted in the Controller. On the other hands, to measure the power consumption KPI each NUC hosting a MEC node is connected to a Meross MSS310 smart plug that monitors its power consumption. To gather those values, the Controller nodes queries the smart plugs via a WiFi connection and send them to Redis. All the measures are collected every second and stored in the Redis database, allowing us to have a centralized repository of all the gathered data. This makes it easier to analyze and interpret the results of the experiments.

[8]`https://github.com/aligungr/UERANSIM`

[9]`https://hub.docker.com/r/redislabs/redistimeseries`

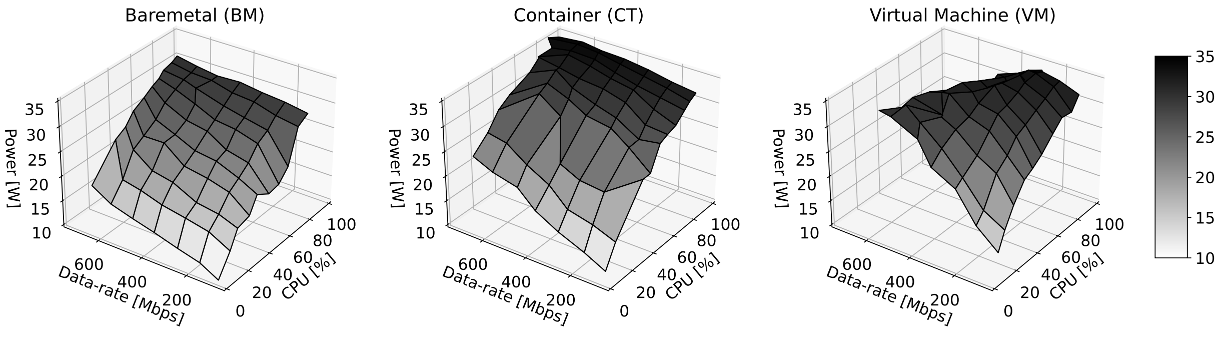


Figure 3.6: Relations between the measured KPIs for each type of MEC.

3.3.1 MEC performance and measured KPIs

We first analyze the power consumption and performance trade-offs in the evaluated MEC deployments, namely BM, VM, and CT. Each MEC was tested by setting WPs with CPU loads ranging from 0% to 90% and data-rates between 100 and 700 Mbps. For each WP, we gathered measurements every second within a 30-second time frame. The experiments are depicted in Figure 3.6, where each plot displays the relationship between the measured KPIs (power consumption, CPU load, and data-rate) for each type of MEC. The aim of this measurement campaign extends beyond investigating the impact of virtualization on MEC performance; it also aims to create a data-set to study the prediction performance of the MEC DT which is detailed in the next sections.

By examining the results under high data-rate and low CPU usage, we observe that the VM is the one consuming the highest amount of power, while the BM is the most efficient, as expected. The CT falls in between the two, with results that are close to those of the BM. Although it may not be clearly visible from the figure, it is worth mentioning that the higher power consumption in the VM can be attributed to its greater CPU usage in handling high data-rates. On the other hand, the BM MEC is the most efficient in traffic management, and the CT case represents a compromise between the two.

When the set WP demands higher CPU loads and high traffic, we see that the CT's power consumption is always greater than the BM. When the imposed demand is the highest which corresponds to 90% of CPU load and 700Mbps of traffic, the CT consumes about 5W more than the BM. However, it is still able to achieve the same BM data-rate without impairments. On the other hand, the power consumption for the VM appears to decrease significantly under high data-rate demand. This decrease is actually the result of impairments in the data-rate, which drops from the requested 700Mbps to an actual 250Mbps when the CPU and data-rate demand is the highest.

The lower data-rate achieved with the VM can be attributed to several factors. It is worth mentioning that the 5GC virtual machine is run using QEMU, a type-2 hypervisor, with KVM acceleration for improved performance. Despite the KVM acceleration, the QEMU VM adds an extra layer of virtualization, which can result in overhead and reduced performance. Furthermore, Docker utilizes a different networking approach compared to virtual machines, which can provide better speed and reduced latency due to leveraging the host's network stack, unlike virtual machines that use virtual network interface cards and incur additional overhead.

Chapter 4

Digital Twin for Network Orchestration

4.1 Motivations

The concept of a DT, which involves creating a virtual representation of a physical system, is gaining traction as a promising solution to address orchestration and power consumption issues in the MEC ecosystem. By developing a DT of an MEC node, it becomes possible to simulate its behavior, analyze its performance, and optimize its operations in a virtual environment before implementing changes in the physical system. Existing works on DTs primarily focus on analytically deriving the MEC DT model, which involves using mathematical reasoning and assumptions to model edge nodes' behavior in terms of KPIs while designing optimization algorithms. However, due to the high heterogeneity of the edge ecosystem, this approach may not be sufficient to address the orchestration challenges effectively. In this study, we propose a data-driven approach to gain insights into the behavior of MEC nodes and accurately predict their performance under varying workload demands. To the best of our knowledge, a data-driven approach that achieves an online and self-learning DT model is essential to cope with the MEC ecosystem's heterogeneity, including potential MEC reconfiguration.

In this work we envision a DT-enabled orchestration architecture as highlighted in Figure 4.1. Indeed, the DT plays a crucial role in managing the network if used for simulating possible what-if scenarios or analyzing

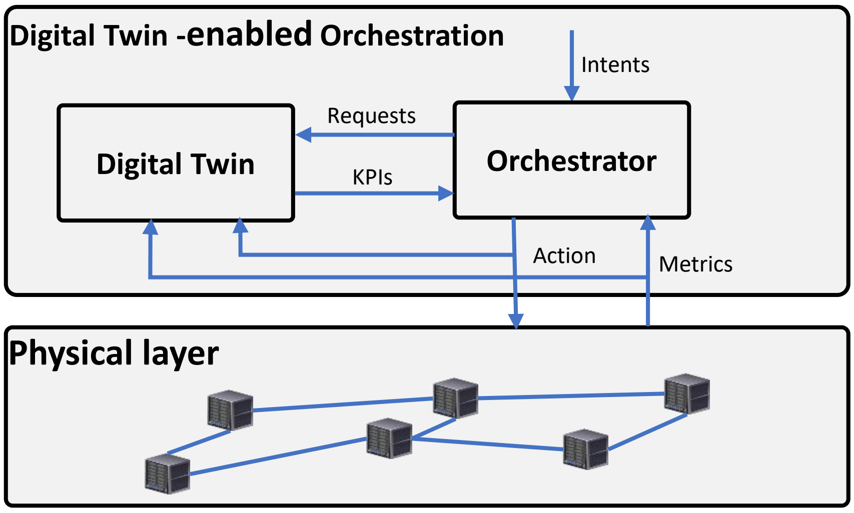


Figure 4.1: Conceptual architecture of the MEC Digital Twin -enabled orchestration envisioned in this work.

the potential impact of different strategies. Indeed, it allows the orchestration layer to take charge of network intents, i.e., services to be deployed, and to exploit the DT models to understand the network behavior before actually applying changes to the physical network. By introducing the DT in the learning loop of the orchestration, the orchestration layer receives predicted KPIs to evaluate the expected network behavior. After taking actions on the physical network, the orchestration layer informs the MEC DT and forwards relevant metrics to the twin, allowing it to learn a digital representation of the physical MEC performance and predict future requests.

4.2 Related works

Digital Twin and MEC

DT has gained a lot of attention recently as an effective tool for modelling simple as well as complex systems. The combination of MEC and DT can be leveraged to improve the network performance by optimizing the task offloading and resource allocation. Authors [63] define the digital twin edge network (DITEN) paradigm and show how it can be used to bridge the physical edge system and the digital space. In their work, they also provide a comprehensive survey.

In [64] a task offloading latency minimization problem while optimizing the transmit power of UEs is proposed. The DT model is derived analytically using mathematical reasoning and assumptions for modelling the processing capacity, latency and energy consumption. In [65], DT-assisted task offloading is modelled as a Markov decision problem. A mathematical optimization model is used to reduce the system delay and power consumption. In [66] a DT architecture to improve task offloading and task caching techniques is proposed. The aim is to minimise the E2E task offloading delay while bounding the energy consumption to a maximum value. Also in this work, the DT model is derived analytically using mathematical reasoning and assumptions to derive the DT predictions.

Power consumption measurements and modelling

In [9] an overview is presented summarizing the challenges, approaches and results of the state-of-the art research concerning the empirical measurements and analytical modelling of virtual entities power consumption in the telco cloud.

The performance of different virtualization technologies under multiple workloads have been analyzed by several studies, showing how each virtualization technology has it own specific advantages as well as a different impact on using of hardware resources. Containers have been proved to be more CPU and memory efficient compared to unikernels and virtual machines (although the lack of isolation might be a concern in specific use-cases)[67]. Container virtualization also registered a lower power consumption especially during the networking workloads [68]. A more extensive comparison can be found in [69], which analyses the power and energy consumption of four of the most adopted hypervisors and container engines across six hardware platforms and multiple workloads.

Other efforts focus on the design of models for the estimation of power consumption without relying on direct measurements. The survey in [70] benchmarks and evaluates the performance of 24 state-of-the-art power consumption models on different server architectures. The results highlight how interpolation and Support Vector Machine (SVM) have the lowest error for single and multi variable models respectively.

4.3 Regression Mechanisms for Digital Twin modelling

In this work, our focus is on developing a DT model of a MEC system following a measurement based approach.

We consider the aggregated data-rate and CPU demand as the input requests. Conversely, the performance KPIs of interest are the achieved data-rate, the CPU load, and the MEC power consumption.

In the first stage of our study, we conducted a measurement campaign on MECs with various virtualization technologies under different loads. To assess the MEC performance, we considered different workload profiles (WPs) by generating data traffic with varying CPU loads that mimic the computational demands of applications running on the MEC. Each combination of data-rate and CPU demand will be denoted as a WP. We simultaneously collected the relevant KPIs such as data-rate, CPU-load, and power consumption of the MEC. It's important to note that, in addition to network KPIs which play a crucial role in providing the required service QoS, the power consumption is also an important KPI to model in order to enable a carbon footprint aware orchestration.

In our measurement campaign we considered three type of MEC deployments where the 5G core network is either directly installed in the bare metal (BM), deployed within a virtual machine (VM), or deployed in a containerized environment (CT). As mentioned already, previous works already investigated the performance of different virtualization technologies in running specific applications. However, in this work the aim is to evaluate the power consumption in each scenarios as well in order to understand and model the trade-off between network KPI and power consumption.

In a next stage of our research, we focused on using ML to model the MEC DT. The goal is to compare the results of the learned model to the real measurements obtained in the measurement campaign, and evaluate its accuracy. To do this, we adopted three different ML algorithms, namely k-nearest neighbors (KNN), Support Vector Regression (SVR), and Polynomial Fitting (PF).

Regarding the KNN, the parameter specifying the number of neighbors has been set to $k = 5$. This value has been selected since appeared to be a sensible trade-off to prevent high bias and high complexity, happening with high k, and sensitivity to noise, happening with a low value of k.

For the SVR, we used Radial Basis Function (RBF) kernel which is commonly recognised as one of the most well-performing kernels in many cases. Moreover, we set the regularization parameter $C = 100$ since performing cross-validation it appears to be the best compromise to avoid under-fitting (low C) and over-fitting (high C).

The polynomial fitting method is proposed as an analytical model of the different MEC deployments. With the aim to provide a basis for further research, the collected data-set and the methodology and parameters of the polynomial fitting are provided in the Appendix 4.3 and available online. The goal is to encourage the replication of our results and facilitate the advancement of related research, leveraging the findings and insights obtained in this work.

These models are trained with the collected measurements to understand how they perform, and to determine which algorithm provides the best results.

Polynomial fitting

The process for developing the MEC analytical model using polynomial fitting is as follows. Let c and t be the requested WP for the MEC expressing the CPU and throughput demand, respectively. Let $\{\hat{c}, \hat{t}, \hat{p}\}$ be the predicted KPIs for the overall CPU load, throughput, and power consumption. To model the MEC behaviour with respect to each KPI, we define the polynomial equation in (4.1), where $p_{kpi}^{i,k}$ are the polynomial parameters as shown in Table 4.1.

$$F_{kpi}(c,t) = \sum_i \sum_k p_{kpi}^{i,k} c^i t^i, \quad \forall KPI \in \{\hat{c}, \hat{t}, \hat{p}\} \tag{4.1}$$

For each KPI, we use the non-linear least squares method to fit the set of observations with the non-linear equation (4.1). The derived analytical

Table 4.1: Polynomial fittings.

	k=0	k=1	k=2	k=3
i=0	$p_{kpi}^{0,0}$	$p_{kpi}^{0,1}$	$p_{kpi}^{0,2}$	$p_{kpi}^{0,3}$
i=1	$p_{kpi}^{1,0}$	$p_{kpi}^{1,1}$	$p_{kpi}^{1,2}$	0
i=2	$p_{kpi}^{2,0}$	$p_{kpi}^{2,1}$	0	0
i=3	$p_{kpi}^{3,0}$	0	0	0

models based and the measurement are available online[1]. It is worth to highlight that this model is valid within the considered range of WPs.

4.3.1 Experimental Results

This section provides the details about the experimentation proposed study. We first provide the details on the environment used for the performance evaluation, followed by a description on the experiments. Later, we provide details on the measured KPIs and the MEC performaces under different virtualization technology. Finally, we provide the details on the prediction performance of the MEC behaviour using the three types of ML algorithm to model the MEC system.

Digital Twin prediction under complete training data-set

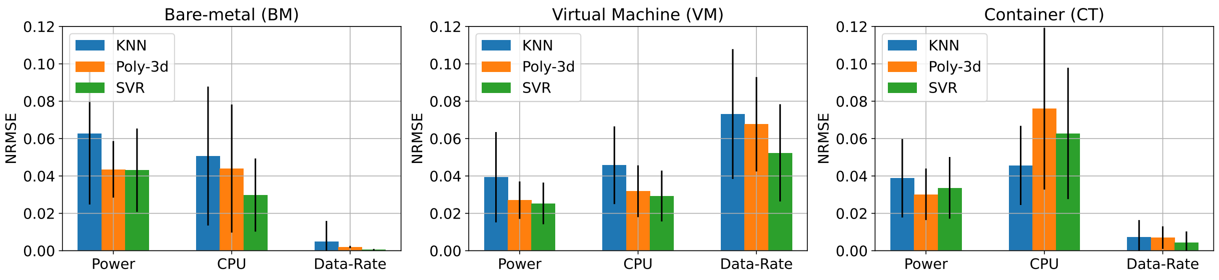


Figure 4.2: DT KPI prediction error in the different MEC deployments. The error bars represent the NRMSE standard deviation across the different WPs.

The aim of this analysis is to determine the accuracy of the DT in predicting the KPIs of the MEC deployments under different WPs. Building

[1]https://github.com/RiccardoFedrizzi/MEC_DT_model

upon the findings discussed in the prior section, we delve deeper to evaluate the forecasting accuracy of a DT using the data-set obtained from aforementioned measurement campaign. We evaluate three type of ML algorithms, namely KNN, RBF and PF. To do so, we use a subset of the collected data to train the DT by considering all the WPs but using 2/3 of the samples. To determine the effectiveness of the DT in accurately predicting the behavior of the MEC deployments, we use the remaining samples as a validation set.

The results of the DT prediction performance are depicted in Figure 4.2. The figure displays the Normalized Root Mean Square Error (NRMSE) for each predicted KPI for each MEC deployment. The NRMSE metric was chosen as it provides a percentage representation of the error relative to the absolute value of each KPI.

The low prediction error observed in all the cases highlights a good accuracy of all the three methods employed demonstrating their applicability to realise the DT prediction. However, our findings indicate that the SVR model performs better than KNN and PF models in the majority of cases. Nevertheless, the PF still displayed noteworthy performance, proving the viability of utilizing analytical models for further research in this area. Interestingly, compared to the other deployment scenarios, the prediction error for the data-rate in the VM scenario is higher. This is because of the data-rate degradation explained in the previous section and shown in Figure 3.6 which makes more challenging in the VM case to forecast the data-rate accurately.

Model prediction under incomplete data-set

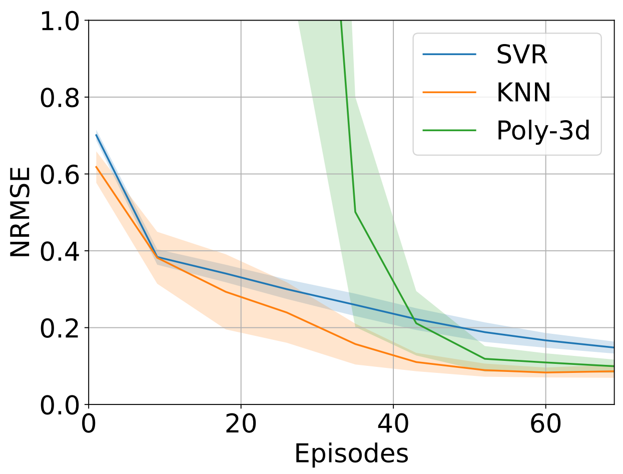
In this section, we examine the accuracy of the DT forecast when the data-set is sparse, as opposed to having a complete set of WPs available for training as was analyzed in the previous section. Training is conducted in several episodes, with measurements for a randomly selected WP being added for each episode, representing the variable measurements that might be acquired by the networking infrastructure. To evaluate the performance of the different ML algorithms, the results in terms of NRMSE were averaged over 20 trials, where in each trial a different WP was randomly selected as the test set, while the remaining WP were used for training. The goal of this approach is to understand the extent to which the different ML approaches are able to generalize the prediction when working with incomplete data, and to provide insight on how much the WP domain shall be explored before to achieve an acceptable prediction error.

In Figure 4.3(a), we report the results showing the NRMSE against the number of episodes. These results pertain to the experiments carried out on VM MEC. As expected, the error in prediction decreases as the number of episodes increases. Interestingly, while the SVR method outperformed the others techniques under complete training sets, the KNN algorithm achieved a better prediction accuracy when the training data is incomplete. This is because of KNN only relies on the proximity of the data points in the feature space, conversely the SVR is a model-based algorithm which potentially leads to biased or incorrect model fit under missing data.

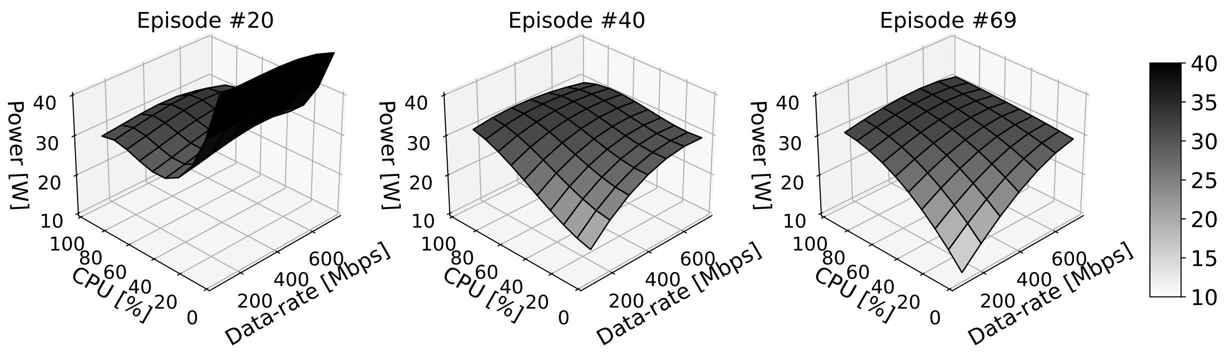
The evaluation of the PF method reveals a substantial prediction error until episode 40. This can be attributed to the polynomial fitting's tendency to produce extreme spikes when evaluating it outside the range covered by the training set. This is demonstrated in the plot of predicted power consumption for different episodes in Figure 4.3(b). If the training set is sparse, the polynomial fitting may be quite ineffective, as highlighted in episode 20. However, the prediction is significantly better as in episode 40 and eventually stabilizes later on. This underlines that the polynomial fitting method can be used to improve the accuracy of the DT representation, even if it requires a suitable number of samples to properly capture the characteristics of the Physical Twin. To encourage the usage of this concept to model networks and MEC in particular, the polynomial fitting model is shared online in the Appendix 4.3.

4.4 GNN-Based Digital Twin for orchestration

This section focuses on the application of graph neural network (GNN) for developing a DT and for assisting in orchestration tasks. We start by discussing the motivations and the problem statement behind this research. We then present a GNN-based model for realizing a DT, which leverages



(a)



(b)

Figure 4.3: Prediction error with incomplete training-set for the VM MEC. In (a) we show the NRMSE vs. the number of episodes, in each episode a new randomly picked WP is added to the training set. Considering one single run, in (b) we plot the predicted power consumption for three episodes.

the power of GNNs to capture the complex relationships and dependencies between the components of the edge network system. Finally, we present preliminary results on using the digital twin to orchestrate service allocations, which demonstrate the potential of GNN-based digital twins for optimizing system performance and reducing operational costs while maintaining the service KPIs.

Overall, this section highlights the potential of GNNs for developing DTs and for aiding orchestration tasks, and we propose several future research directions for further advancing this area.

4.4.1 Motivations and proposed methodology

In the previous section, we explored the use of regression mechanisms for building a digital twin of a MEC node. While this approach showed promising results in capturing the behavior of the MEC node, it did not fully capture the complex interactions between different components in a larger edge network ecosystem. This limitation underscores the requirement for a more advanced modeling approach that can effectively capture the intricate dependencies and interactions among various components in an edge network, while also accounting for any possible impairments that may arise due to the underlying network.

With the aim of addressing these issues, in this section, we propose the use of GNNs for building a DT of edge networks. GNNs are a powerful class of neural networks that can learn from graph-structured data and capture the complex relationships between different components in a system [71]. By modeling the edge network as a graph, we can use GNNs to learn a digital representation of the entire edge network ecosystem. This representation can capture the interactions between different components, as well as the possible impairments due to the underlying network, and enable us to predict the performance of the edge network in real-time. Optimization tasks reducing operational costs while maintaining the requests QoS can be then performed in the digital domain before to act on the physical network. In particular, in this work we focus on service allocation decisions to minimize operational costs without compromising the requested service-level KPIs.

In this section, we consider a scenario in which an edge network comprises a number of heterogeneous MEC nodes that are requested to support service requests. Each service request in this network is responsible for handling requests coming from UEs. We assume that each UE request is processed by a micro-service that is spawned in one of the MEC within the network. The amount of computational and communication load that a micro-service places on a MEC can vary significantly, depending on the QoS requirements of the initiated the service request. We assume each service request is composed of both data traffic and a certain number of operations per second, which need to be processed by the MEC.

The allocation of services to specific MECs, based on their capabilities and the QoS requirements of the services, can have a significant impact on overall operational expenditures (OPEX) and the potential for service quality violations. In this work, we propose the use of power consumption as the primary metric for OPEX. This is justified considering that energy consumption is one of the largest operational expenses in MEC ecosystems [72] [73].

The problem finding the optimal service allocation is defined as follows:

Given: the dynamic nature of the edge network scenario described above, with the number of requests varying over time and MEC servers with different performances and power consumption efficiency.

Find: a digital representation of the edge ecosystem (DT) which allows for the prediction of network behavior based on service placement decisions. Based on the DT predictions find the optimal service allocations in the digital domain, before to apply changes to the physical network.

Objective: maintain the requested QoS while minimizing the operational expenses of the MEC ecosystem.

4.4.2 Edge network scenario

In our study, we utilize the emulation test-bed, which is described in Section 3.2, to simulate a scenario with two MECs having different capabilities. Figure 4.4 illustrates the network architecture, where the two MECs host micro-services that support user requests. To emulate service requests, emulated UEs are used to generate data-traffic and request computation tasks to their associated micro-service. These micro-services are deployed as containers on the fly on a specific MEC. This approach allows us to build a data-set containing performance KPIs based on different possible combination of requests and service allocations.

In our experiments, we aim to improve the stability of the generated data-set by assigning a dedicated CPU to each MEC. However, MEC 1 has only half of a CPU dedicated, while MEC 2 has a full CPU dedicated to it. This configuration allows us to evaluate service allocation strategies within an heterogeneous MEC environment. Table 4.2 shows the request parameters for each service, which represent the generated demand that

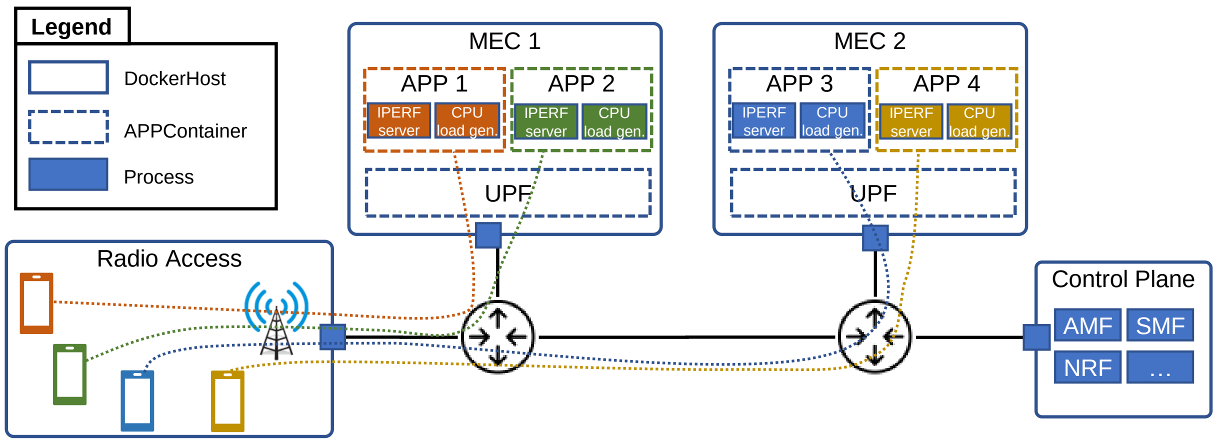


Figure 4.4: Representation of the edge network scenarioi implemented leveraging the emulation test-bed of Section 3.2.

needs to be maintained.

It is important to note that measuring the power consumption of each MEC it is not possible in an emulation environment, but this information is crucial for our research objectives. To address this challenge, we utilized the results from our physical test-bed experiments, which are detailed in Section 4.3. Specifically, we used the SVR model to estimate the power consumption of the emulated MECs while re-scaling the data-rates in order to meet the physical test-bed workload. To account for differences between MEC 1 and MEC 2, we assumed that MEC 1 consumes half the power of MEC 2 in the same operational conditions. This assumption is justified by the fact that MEC 1 has half of the computing capabilities of MEC2.

To collect the dataset, we follow a specific procedure. For each emulation interval, we (*i*) randomly decide whether to add or remove a service while maintaining their overall number within the interval specified in Table 4.2. Then, (*ii*) we randomly assign a service request and choose a MEC to allocate the service. Next, (*iii*) we run the emulation for 20 seconds and collect the measured KPIs. Finally, (*iv*) we compute the MECs power consumption KPI based on the SVR model as previously mentioned, and store the results of the iteration in the data-set.

The range of service request parameters was defined based on preliminary tests that demonstrated MEC 1 can achieve a maximum throughput of 30 Mbps, while MEC 2 can handle up to 60 Mbps. However, we observed

Table 4.2: Parameters for the emulated scenario.

Parameter	**Description**
Number of emulation intervals	1000
Emulation time for each interval	20 sec
Collected KPIs	MEC CPU load / Overall MEC data-rate / UE achieved data-rate
Measurement time interval	1 sec
Number of requests	2-5
Service demand: data-rate	2-10 [Mbps] (TCP)
Service demand: computing	0-300 [kOps]

impairments in one MEC when we increased the throughput handled by the other MEC due to congestion. We also observed issues with increasing the computing demand in the same MEC, which led to CPU overhead. These findings have been taken into account in order to derive the service demand range explained in Table 4.2

4.4.3 GNN-based Digital Twin

The GNN architecture used to build the DT is shown in Figure 4.5. This architecture takes an input graph composed of service requests and their connections. For each node representing a service request, the features are composed of the computing and communication requirements, represented by $\mathrm{R}_s = [R_s^{cpu}, R_s^{dr}]$. In addition, the MEC nodes in the graph are represented by the aggregated requests for each MEC, where M is the set of MECs in the scenario and S_m is the set of services allocated on MEC m. The MEC node features are calculated as in the equation below by aggregating the requests for each MEC.

$$\mathrm{R}_m = \left[\sum_{s \in S_m} R_s^{cpu}, \sum_{s \in S_m} R_s^{dr} \right] \quad \forall m \in M \tag{4.2}$$

Moreover, the edges of the graph are considered to be the combination of the edges representing the underlying network and the edges associating each service with a MEC.

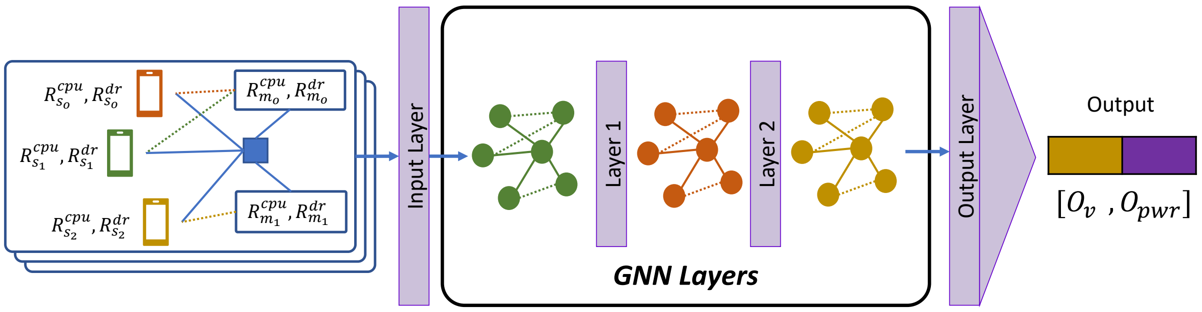

$R_{s_o}^{cpu}, R_{s_o}^{dr}$
$R_{s_1}^{cpu}, R_{s_1}^{dr}$
$R_{s_2}^{cpu}, R_{s_2}^{dr}$
$R_{m_o}^{cpu}, R_{m_o}^{dr}$
$R_{m_1}^{cpu}, R_{m_1}^{dr}$
Input Layer
Layer 1
Layer 2
GNN Layers
Output Layer
Output
$[O_v, O_{pwr}]$

Figure 4.5: Representation of the GNN architecture used to build the DT.

The GNN architecture is composed of an initial layer, two graph convolutional network (GCN) layers with 50 hidden channels, and a final multilayer perceptron (MLP) layer used for the graph prediction tasks. The initial layer takes the input graph as explained above. Then, the GCN layers perform convolution operations on the graph to extract relevant features. Both the GCN layer produces 50 hidden channel outputs as output, which are then passed through a non-linear activation function to produce a new set of hidden features. Finally, the MLP layer takes the hidden features as input and produce the final graph prediction output. The output is a vector of two elements, $[O_v, O_{pwr}]$ containing (i) whether there is a service violation, and (ii) the overall power consumption of the system.

As discussed in Section 4.4.2, we gather O_m^{cpu}, O_m^{dr}, and O_m^{pwr} for each MEC, representing the CPU load, achieved data-rate, and power consumption, respectively, for each sample in the dataset. To compute the GNN expected output and allow its training, for each entry in the data-set the GNN expected output is computed as follows.

A violation $(O_v = 1)$ occurs if either the CPU load of a MEC exceeds 90%, or the requested MEC data rate exceeds the observed value by a confidence interval of 2 Mbps.

$$\begin{aligned} O_v = 0 \quad & \text{if} \quad \begin{cases} O_m^{dr} > R_m^{dr} - 2\text{Mbps} \quad \forall m \in M \\ O_m^{cpu} < 90\% \quad \forall m \in M \end{cases} \\ O_v = 1 \quad & \text{otherwise} \end{aligned} \tag{4.3}$$

Conversely, the overall power consumption of the system is computed as the sum of the observed power consumption of each MEC unit, denoted

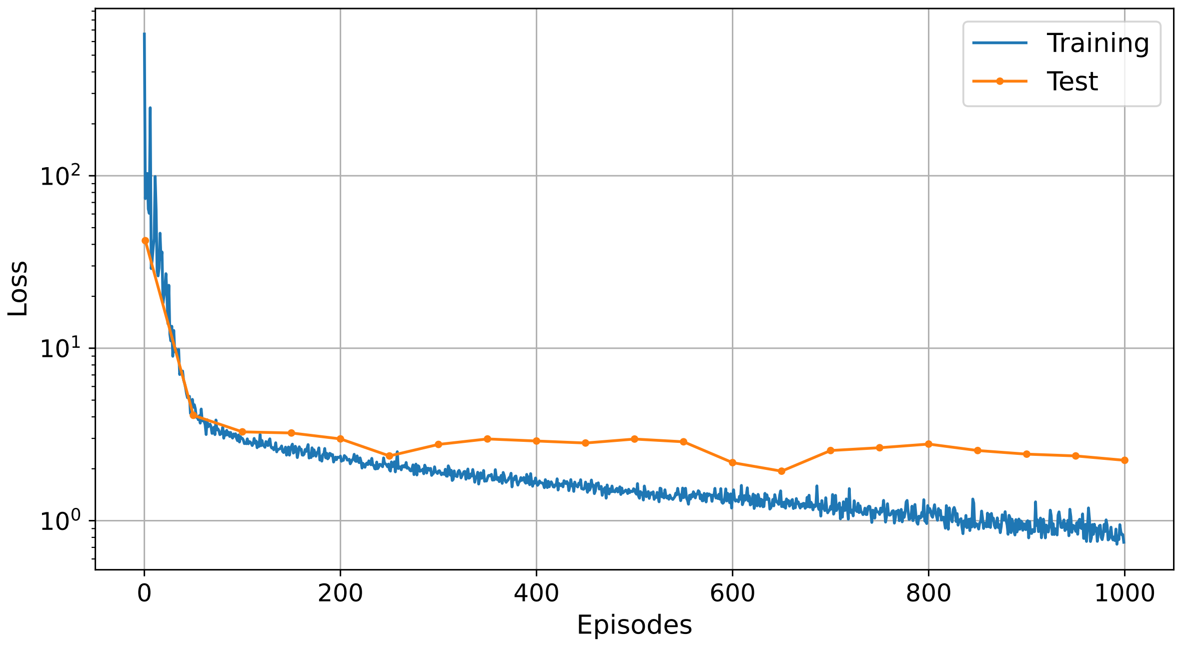


Figure 4.6: Evolution of training and test MSE loss over 1000 iterations. The loss over test set has been computed every 50 episodes.

by O_m^{cpu}.

$$O_{pwr} = \sum_{m \in M} O_m^{cpu} \tag{4.4}$$

We randomly select 700 samples from the overall dataset of collected measurements, and allocate 80% of them for training and the remaining 20% for validation. During the selection process, we ensure that a sufficient number of violations are included in the training set to enable the GNN to learn to recognize service violations effectively.

Figure 4.6 illustrates the changes in training loss over 1000 iterations, calculated as the mean squared error (mean squared error (MSE)) between the GNN output $[\tilde{O}_v, \tilde{O}_{pwr}]$ and the corresponding true values $[O_v, O_{pwr}]$ in the dataset. The results show that the final MSE for the train set is approximately 0.68, while for the test set it is 2.9. Notably, the test set loss demonstrates a consistent and flat behavior, which indicates robustness against over-fitting.

Figure 4.7(a) shows the normalised root mean squared error (NRMSE)

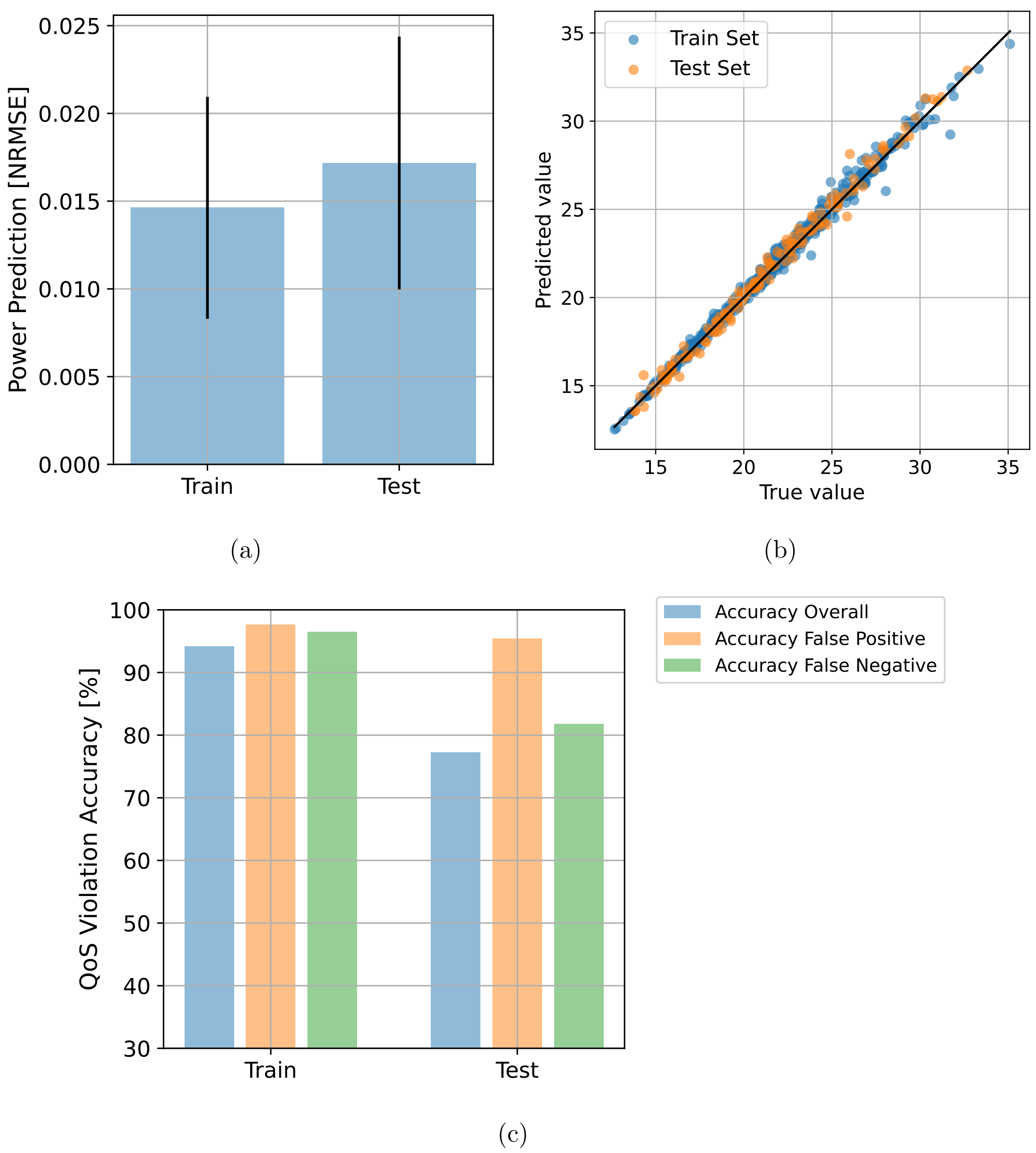


Figure 4.7: The Accuracy of the GNN DT prediction evaluated as: (a) NRMSE of the predicted power consumption, (b) true vs. predicted power consumption, and (c) accuracy of predicting QoS violations.

specifically for power consumption prediction on both the train and test sets. This metric is useful as it allows for a direct comparison with the amplitude of the power measures. The results indicate that the average prediction error for power consumption is around 1.5%, demonstrating the effectiveness of the model in accurately predicting power consumption.

This is further illustrated by Figure 4.7(b), which displays a scatterplot comparing the predicted values of power consumption to their corresponding true values. The graph indicates a strong linear correlation between the predicted and true values, confirming the effectiveness of the model's power consumption predictions.

Figure 4.7(c) presents the prediction accuracy for QoS violations, as computed by Equation (4.3). The model correctly recognizes QoS violations approximately 93% of the time in the train set, but this accuracy drops to below 80% in the test set. We also observe that the accuracy for false negatives is lower than that for false positives, which could lead to issues in correctly identifying QoS service violations. However, we argue that achieving an 80% accuracy on the test set is a significant accomplishment.

Finally, in Figure 4.7(c) we show the prediction accuracy over the QoS violations computed as in Equation (4.3). We can observe that while the QoS violation is correctly recognised the 93% of times over train set, this is not true for the test set, where the probability decreases below 80%. We also observe that the accuracy over the false negative is lower that the false positive, which might cause issues for not correctly recognising a QoS service violation. However, achieving 80% of accuracy in the QoS violations prediction over the test set is a good achievement that demonstrate the viability of this GNN-based approach.

4.4.4 Service allocation based on Digital Twin predictions

After the DT is trained using the measures from the physical twin (PT), it can be used to forecast a service placement layer that enables the implementation of various policies. These policies can be evaluated before applying them in the physical domain. The service placement decision loop shown in Figure 4.8 starts with the service placement policy layer receiving service requests. The policy can then query the DT to obtain predictions on the network behavior, as discussed in the previous section. Based on this information, the policy can decide to apply an action on the PT or explore other options.

In this section, we propose three heuristics for service allocation decisions:

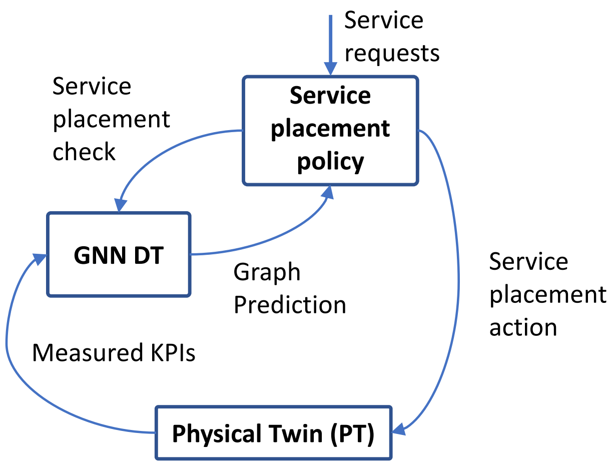
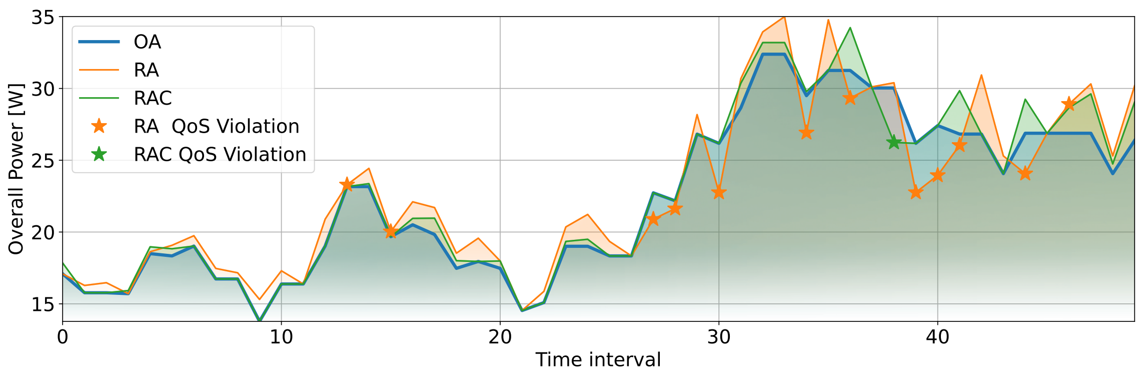


Figure 4.8: Envisioned service placement strategy leveraging the GNN DT.

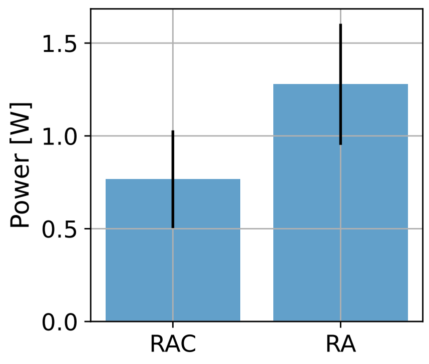
- The first one, dubbed OA, is an Optimal Allocation policy that evaluates all possible service allocations by querying the DT and returns the best one. Where the best one is the service allocation minimising the predicted power consumption, while avoiding service violations.
- The second one, dubbed RAC, is a Random Allocation policy with multiple Checks that first performs a random allocation and then checks the DT for possible service violations. If any violations are predicted, the policy performs up to N_{rac} retries to find a different valid service allocation.
- The third one, dubbed RA is a pure Random Allocation approach, which is considered for worst-case comparison purposes.

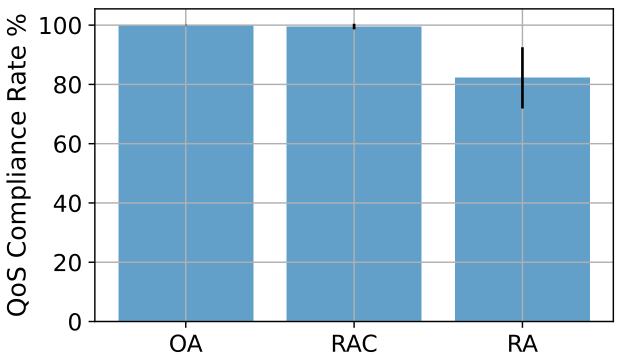
DT prediction evaluations

To evaluate the effectiveness of the three policies, we simulate a series of episodes. Each episode consists of 50 time intervals, during which the service requests are adjusted based on the procedure outlined in Section 4.4.2 for constructing the data set. We repeat the experiment 100 times and gather the predicted network KPIs results from the decision tree model. Figure 4.9(a) displays a comparison of overall power consumption during



(a)





(b) (c)

Figure 4.9: Comparison between OA, RAC, and RA over 100 experiments. In (a) we show a comparison of OA, RAC, and RA in terms of power consumption optimization over one episode with 50 different service requests. In (b) we show the power consumption overshoot of RAC and RA vs. OA on average over all the experiments. In (c) we compare the service QoS compliance in the case of the three policies.

one of the simulated episodes for the OA, RAC, and RA policies. Additionally, service violations are indicated with stars in the figure. As expected, the RA policy performs the worst as it randomly selects the initial service allocation. However, with only three retries, the RAC policy approaches optimality and exhibits minimal service violations. It is worth noting that the RA policy frequently produces solutions with lower power consumption than the OA policy. However, this comes at the expense of service violations, which are considered as service rejections.

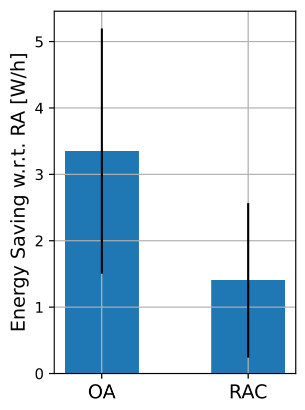
Figure 4.9(b) presents a comparison of power consumption between the RAC, RA policies against the OA. The results from 100 experiments are averaged and analyzed. On average, the RAC policy results in a power consumption overshoot of approximately 0.75W higher than the optimal. Conversely, the RA policy exhibits almost double the overshoot, with approximately 1.3W of overshoot. Figure 4.9(c) illustrates a comparison of QoS compliance rates among the OA, RAC, and RA policies. As anticipated, the RA policy displays the lowest QoS compliance rate of approximately 80%. Conversely, it is noteworthy that the RAC policy achieves a QoS compliance rate of nearly 100% with just three retries.

Performances of DT-based allocations on the PT

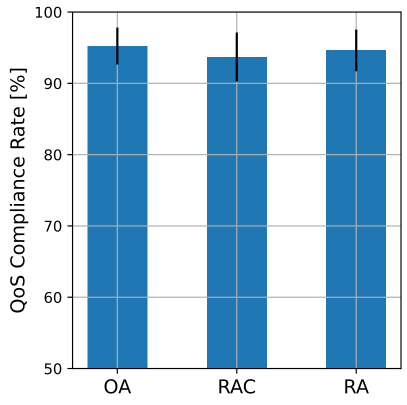
This paragraph provides a discussion on the comprehensive approach illustrated in Figure 4.8. To assess the effectiveness of the service allocations based on Decision Trees (DT), we conducted a series of 10 episodes, each consisting of 50 time intervals. During each time interval, multiple service requests (ranging from 3 to 5) were generated. Each service request comprised a data-rate between 0 and 6 Mbps and a computing request between 0 and 300 kOps. For for each time interval, we executed three service allocation policies, namely OA, RAC, and RA, to obtain the respective service allocations. Subsequently, the obtained service allocations were emulated to evaluate the performance of the DT-based service allocation against the PT.

Figure 4.10(a) illustrates the energy gain of OA and RA in comparison to the benchmark RA. OA demonstrates an energy consumption reduction of approximately 3.3W/h, whereas RAC shows a reduction of around 1.4W/h. Although these gains may not appear substantial, it is important to note that they were achieved within a limited scenario involving only two MECs. Furthermore, these reductions amount to approximately 10% of the maximum energy consumption of MEC 2.

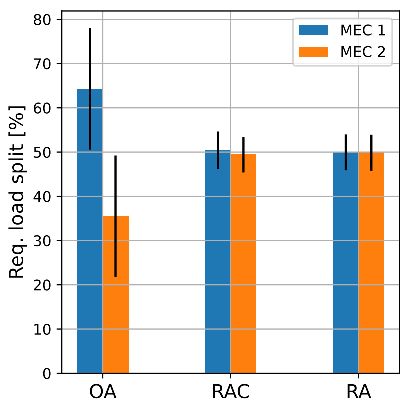
Referring to Figure 4.9(c), it can be observed that the decision output of the DT-based service allocation resulted in zero QoS violations for OA and demonstrated a high level of accuracy for RAC. However, when applied on the PT, as indicated in Figure 4.10(b), the actual service allocation decisions resulted in a similar rate of QoS compliance across all three policies. This can be attributed to the fact that, as depicted in Figure 4.7(c), the



(a)



(b)



(c)

Figure 4.10: Performance OA, RAC, and RA decisions applied on the PT and averaged over the emulated episodes. In (a) we show the energy gain achieved by OA and RAC over the RA. In (b) we show the service acceptance in the case of the three policies. In (c) we show for each policy how the overall requested demand is balanced between the two MECs of Figure 4.4.

DT has a higher tendency to generate errors in recognizing QoS violations, particularly by producing false negatives.

Figure 4.10(c) presents the load balancing among the MECs in the given scenario. OA prominently favors MEC 1 to achieve optimal energy efficiency, while RAC exhibits a slight preference for MEC 1 by making additional service allocation decisions only after a QoS violation is identified with the assistance of the DT. On the other hand, RA demonstrates its characteristic random behavior in load distribution.

It is important to note that the primary reason behind the higher energy efficiency observed in the OA case is the inclination to allocate services to MEC 1, which consumes less power compared to MEC 2. However, it should also be acknowledged that the preference of OA towards MEC 1 brings it closer to the boundaries of QoS violations. This is because MEC 1 is more constrained in terms of CPU resources compared to MEC 2. This behavior, combined with the imperfect prediction of QoS violations by the DT, explains why the OA decisions, when applied to the PT, do not achieve a perfect QoS compliance rate as we could expect by looking at the DT outcomes. Nevertheless, it is worth mentioning that all three policies exhibit a significantly high QoS compliance rate of around 95%.

Chapter 5

Conclusions

This thesis explores various aspects of the significant transformation that 5G mobile communication networks are undergoing. The purpose of this change is to facilitate the emergence of novel services like virtual and augmented reality, internet of things, and self-driving vehicles that are being demanded by emerging vertical industries. All these services need to coexist on the same network infrastructure, and this transformation aims to make that possible. In particular, the thesis focused on the emerging network softwarization techniques, which provide technical enablers such as network slicing and distributed user-plane. These techniques, along with MEC, are expected to play a crucial role in network automation of future networks and support multiple services with diverging requirements.

In Chapter 2 we investigated the problem to deploy different services requested by various verticals in 5G networks. To embed service applications, along with the UP and CP functions, within slices provided to each vertical industry while optimally allocating network resources, we formulated and solved a SFC placement problem. Two different approaches, namely Shared and Dedicated UPF strategies, were proposed to provision the UP functions to services of different types. Based on the MILP formulation results, both approaches provided similar network utilization performances, but the Dedicated UPF approach resulted in significantly lower execution times (between 1 and 5 hours). It was concluded that a network slice provisioning tailored to each service type, as advocated in literature, is a viable option. Additionally, three objective functions, namely Obj-Cost, Obj-Bwt, and Obj-Mig, were proposed for efficient service placement while pursuing different business logics. The Obj-Cost objective function resulted in high overall network utilization, whereas Obj-Bwt showed the lowest network utilization but exhausting edge-DC resources quickly. The Obj-Mig objective function is more suitable for achieving a compromise of network utilization, a balanced load across edge-DC, regional-DC, and cloud-DC, and minimization of VNF migrations. Alternative GA-based approaches were proposed to reduce the high computational cost of the optimization process and compared with the optimal solution provided by the MILP formulation. Although these methods resulted in sub-optimal solutions, simulations showed their effectiveness in achieving a good compromise between accuracy and computational effort. The proposed AGA resulted in up to 10% improvement in solution accuracy compared to its non-adaptive counterpart, while also exhibiting lower sensitivity to parameters and significantly reducing execution time.

The heterogeneity of the MEC ecosystem poses challenges in terms of network orchestration and automation, which demand measurement-based and model-free techniques. Motivated by this, two test-beds developed and presented in Chapter 3 using well-known open source software to analyze MEC performance in an end-to-end 5G system. The first test-bed is an emulation test-bed that enables the evaluation of MEC applications under realistic network conditions and the collection of various performance metrics related to traffic flows and virtual resource utilization. The second test-bed is a physical one that focuses on power consumption measurement while varying the virtualization technology used. Using these test-bed, it is possible to evaluate the energy efficiency of MEC deployments under different conditions and gain insights into the most appropriate virtualization technology to use in specific scenarios. The two test-beds have been used to create a data-set of measurements to investigate the performance of MECs based on different KPIs, such as CPU consumption, achieved data-rate, and power consumption. Results demonstrate that the emulated test-bed is effective in emulating heterogeneous MEC-enabled 5G networks and characterizing MEC performance trade-offs, and for this reason, the software has been made freely available online.

In Chapter 4, the two test-beds presented in the previous chapter were utilized to propose data-driven approaches to realize a MEC digital twin

(DT). Different regression mechanisms were employed to evaluate the feasibility of building a digital representation of a MEC node. The results showed that regression mechanisms are efficient in predicting MEC behavior, with an prediction error often below 5%. However, when multiple requests are made, it is unclear how to effectively use these mechanisms as some requests may impact MEC behavior differently than others. To address this issue, GNNs have been explored as a scalable solution also for capturing the underlying network behavior and providing a comprehensive representation of the network edge as a whole. A comparison of predicted and actual KPIs yielded valuable insights into the accuracy and reliability of the DT, which has the potential to inform decision-making for a range of network management tasks. The results also highlighted the potential of GNNs-based DTs in enabling closed-loop automation, where network management processes are continuously refined based on real-time network performance data. In particular, the GNN-based digital twin demonstrated high accuracy, achieving approximately 1.5% in predicting the overall power consumption of the edge ecosystem, and exhibiting good performance with an accuracy rate of around 80% in predicting QoS service violations. Furthermore, preliminary results indicate that the GNN-based digital twin is a viable tool for supporting service allocation tasks.

As a future endeavor, we aim to explore the feasibility of utilizing the GNN-based digital twin in realizing service allocation policies based on RL techniques. The digital twin's ability to accurately predict pertinent network KPIs can be leveraged to train offline a RL-based service allocation without to execute actions on the physical network. Moreover, self-learning loops can be envisioned in order to (*i*) automatically perform further training episodes for the DT based on the possibly observed deviations form the PT, and (*ii*) autonomously adapt the policy for service allocation without human intervention.

Bibliography

[1] M Series. IMT Vision–Framework and overall objectives of the future development of IMT for 2020 and beyond. *Recommendation ITU*, pages 2083–0, 2015.

[2] Mohammad Asif Habibi, Bin Han, Faqir Zarrar Yousaf, and Hans D. Schotten. How Should Network Slice Instances Be Provided to Multiple Use Cases of a Single Vertical Industry? *IEEE Communications Standards Magazine*, 4(3):53–61, 2020.

[3] ETSI. MEC in 5G networks. Technical Report ETSI White Paper No. 28, European Telecommunications Standards Institute, June 2018.

[4] Rasoul Behravesh, Davit Harutyunyan, Estefanía Coronado, and Roberto Riggio. Time-sensitive mobile user association and sfc placement in mec-enabled 5g networks. *IEEE Transactions on Network and Service Management*, 2021.

[5] Mobile Edge Computing (MEC); Deployment of Mobile Edge Computing in an NFV environment. Etsi gr mec 017 v1.1.1, 3GPP, 2018.

[6] Qiang Ye, Weihua Zhuang, Xu Li, and Jaya Rao. End-to-End Delay Modeling for Embedded VNF Chains in 5G Core Networks. *IEEE Internet of Things Journal*, 6(1):692–704, 2019.

[7] Peiying Zhang, Chao Wang, Neeraj Kumar, Weishan Zhang, and Lei Liu. Dynamic Virtual Network Embedding Algorithm Based on Graph Convolution Neural Network and Reinforcement Learning. *IEEE Internet of Things Journal*, 9(12):9389–9398, 2022.

[8] Anouar Rkhami, Tran Anh Quang Pham, Yassine Hadjadj-Aoul, Abdelkader Outtagarts, and Gerardo Rubino. On the use of graph neural

networks for virtual network embedding. *2020 International Symposium on Networks, Computers and Communications, ISNCC 2020*, pages 3–8, 2020.

[9] Etienne-Victor Depasquale, Franco Davoli, and Humaira Rajput. Dynamics of research into modeling the power consumption of virtual entities used in the telco cloud. *Sensors*, 23(1):255, 12 2022.

[10] Ibm ilog cplex optimizer. Accessed: 2023-04-13.

[11] Davit Harutyunyan, Riccardo Fedrizzi, Nashid Shahriar, Raouf Boutaba, and Roberto Riggio. Orchestrating End-to-end Slices in 5G Networks. *15th International Conference on Network and Service Management, CNSM 2019*, 2019.

[12] 5g nr radio resource control. 3gpp tr 38.331 version 15.4.0 release 15, 3GPP, 1 2019.

[13] Marie-Theres Suer, Christoph Thein, Hugues Tchouankem, and Lars Wolf. Multi-Connectivity as an Enabler for Reliable Low Latency Communications—An Overview. *IEEE Communications Surveys & Tutorials*, 22(1):156–169, 2020.

[14] M. Centenaro, D. Laselva, J. Steiner, K. Pedersen, and P. Mogensen. System-level study of data duplication enhancements for 5g downlink urllc. *IEEE Access*, 8:565–578, 2020.

[15] Matias Richart, Javier Baliosian, Joan Serrat, and Juan-Luis Gorricho. Resource slicing in virtual wireless networks: A survey. *IEEE Transactions on Network and Service Management*, 13(3):462–476, 2016.

[16] Ibrahim Afolabi, Tarik Taleb, Konstantinos Samdanis, Adlen Ksentini, and Hannu Flinck. Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. *IEEE Communications Surveys & Tutorials*, 20(3):2429–2453, 2018.

[17] Yasir Zaki, Liang Zhao, Carmelita Goerg, and Andreas Timm-Giel. LTE wireless virtualization and spectrum management. In *Proc. of IFIP WMNC*, Budapest, Hungary, 2010.

[18] O-RAN alliance. Accessed: 2021-12-15.

[19] Rasoul Behravesh, Estefanía Coronado, Davit Harutyunyan, and Roberto Riggio. Joint user association and vnf placement for latency sensitive applications in 5g networks. In *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, pages 1–7, 2019.

[20] Davit Harutyunyan, Rasoul Behravesh, and Nina 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)*, pages 241–249. IEEE, 2021.

[21] Faqir Zarrar Yousaf, Johannes Lessmann, Paulo Loureiro, and Stefan Schmid. Softepc - dynamic instantiation of mobile core network entities for efficient resource utilization. In *Proc. of IEEE ICC*, Budapest, Hungary, 2013.

[22] Zafar Ayyub Qazi, Phani Krishna Penumarthi, Vyas Sekar, Vijay Gopalakrishnan, Kaustubh Joshi, and Samir R Das. KLEIN: A minimally disruptive design for an elastic cellular core. In *Proc. of ACM SOSR*, Santa Clara, CA, 2016.

[23] Malla Reddy Sama, Luis M Contreras, John Kaippallimalil, Ippei Akiyoshi, Haiyang Qian, and Hui Ni. Software-defined control of the virtualized mobile packet core. *IEEE Communications Magazine*, 53(2):107–115, 2015.

[24] M. Bagaa, T. Taleb, and A. Ksentini. Service-aware network function placement for efficient traffic handling in carrier cloud. In *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2402–2407, 4 2014.

[25] A. Baumgartner, V. S. Reddy, and T. Bauschert. Combined virtual mobile core network function placement and topology optimization with latency bounds. In *2015 Fourth European Workshop on Software Defined Networks*, pages 87–102, 2015.

[26] Arsany Basta, Wolfgang Kellerer, Marco Hoffmann, Klaus Hoffmann, and Ernst-Dieter Schmidt. A virtual sdn-enabled lte epc architecture: A case study for s-/p-gateways functions. In *Proc. of IEEE SDN4FNS*, Trento, Italy, 2013.

[27] Arsany Basta, Wolfgang Kellerer, Marco Hoffmann, Hans Jochen Morper, and Klaus Hoffmann. Applying nfv and sdn to lte mobile core gateways, the functions placement problem. In *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, &#38; Challenges*, AllThingsCellular '14, pages 33–38, New York, NY, USA, 2014. ACM.

[28] 3GPP. System architecture for the 5g system. 3gpp ts 23.501 version 15.5.0 release 15, 3 2019.

[29] Miloud Bagaa, Tarik Taleb, Abdelquoddouss Laghrissi, Adlen Ksentini, and Hannu Flinck. Coalitional game for the creation of efficient virtual core network slices in 5G mobile systems. *IEEE Journal on Selected Areas in Communications*, 36(3):1–18, 2018.

[30] Miloud Bagaa, Tarik Taleb, Abdelquoddouss Laghrissi, and Adlen Ksentini. Efficient virtual evolved packet core deployment across multiple cloud domains. *IEEE Wireless Communications and Networking Conference, WCNC*, 2018-April:1–6, 2018.

[31] Dejene Boru Oljira, Karl Johan Grinnemo, Javid Taheri, and Anna Brunstrom. A model for QoS-Aware VNF placement and provisioning. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN 2017*, pages 1–7, 2017.

[32] Nakao, Akihiro and Du, Ping and Kiriha, Yoshiaki and Granelli, Fabrizio and Gebremariam, Anteneh Atumo and Taleb, Tarik and Bagaa, Miloud. End-to-end network slicing for 5G mobile networks. *Journal of Information Processing*, 25:153–163, 2017.

[33] Idelkys Quintana-Ramirez, Anthony Tsiopoulos, Maria A Lema, Fragkiskos Sardis, Luis Sequeira, James Arias, Aravindh Raman, Ali

Azam, and Mischa Dohler. The making of 5g: Building an end-to-end 5g-enabled system. *IEEE Communications Standards Magazine*, 2(4):88–96, 2018.

[34] Xueli An, Chan Zhou, Riccardo Trivisonno, Riccardo Guerzoni, Alexandros Kaloxylos, David Soldani, and Artur Hecker. On end to end network slicing for 5g communication systems. *Transactions on Emerging Telecommunications Technologies*, 28(4):e3058, 2017.

[35] Ioannis Sarrigiannis, Luis M. Contreras, Kostas Ramantas, Angelos Antonopoulos, and Christos Verikoukis. Fog-Enabled Scalable C-V2X Architecture for Distributed 5G and beyond Applications. *IEEE Network*, 34(5):120–126, 2020.

[36] Truong-Xuan Do and Younghan Kim. State management function placement for service-based 5g mobile core architecture. *Mobile Networks and Applications*, 24(2):504–516, 2019.

[37] Truong-Xuan Do and Younghan 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)*, pages 102–106, 2018.

[38] I. Y. Kim and O. L. De Weck. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and Multidisciplinary Optimization*, 29(2):149–158, 2005.

[39] Wanqing Guan, Xiangming Wen, Luhan Wang, Zhaoming Lu, and Yidi Shen. A service-oriented deployment policy of end-to-end network slicing based on complex network theory. *IEEE Access*, 6:19691–19701, 2018.

[40] Rami Addad, Miloud Bagaa, Tarik Taleb, Diego Leonel Cadette Dutra, and Hannu Flinck. Optimization model for cross-domain network slices in 5g networks. *IEEE Transactions on Mobile Computing*, 2019.

[41] Hassan Halabian. Distributed resource allocation optimization in 5g virtualized networks. *IEEE Journal on Selected Areas in Communications*, 37(3):627–642, 2019.

[42] Irian Leyva-Pupo, Alejandro Santoyo-González, and Cristina Cervelló-Pastor. A framework for the joint placement of edge service infrastructure and user plane functions for 5g. *Sensors (Basel, Switzerland)*, 19(18):3975, 9 2019. 31540093[pmid].

[43] Chun Wei Tsai and Joel J.P.C. Rodrigues. Metaheuristic scheduling for cloud: A survey. *IEEE Systems Journal*, 8(1):279–291, 2014.

[44] Fan Hsun Tseng, Xiaofei Wang, Li Der Chou, Han Chieh Chao, and Victor C.M. Leung. Dynamic Resource Prediction and Allocation for Cloud Data Center Using the Multiobjective Genetic Algorithm. *IEEE Systems Journal*, 12(2):1688–1699, 2018.

[45] Lidia Ruiz, Ramón J. Durán, Ignacio de Miguel, Pouria S. Khodashenas, Jose Juan Pedreno-Manresa, Noemí Merayo, Juan C. Aguado, Pablo Pavon-Marino, Shuaib Siddiqui, Javier Mata, Patricia Fernández, Rubén M. Lorenzo, and Evaristo J. Abril. A genetic algorithm for VNF provisioning in NFV-enabled cloud/MEC RAN architectures. *Applied Sciences (Switzerland)*, 8(12), 2018.

[46] Jiuyue Cao, Yan Zhang, Wei An, Xin Chen, Jiyan Sun, and Yanni Han. VNF-FG design and VNF placement for 5G mobile networks. *Science China Information Sciences*, 60(4):1–15, 2017.

[47] Anestis Dalgkitsis, Prodromos Vasileios Mekikis, Angelos Antonopoulos, and Christos Verikoukis. Data Driven Service Orchestration for Vehicular Networks. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4100–4109, 2021.

[48] Nahida Kiran, Xuanlin Liu, Sihua Wang, and Changchuan Yin. Optimising resource allocation for virtual network functions in SDN/NFV-enabled MEC networks. *IET Communications*, 15(13):1710–1722, 2021.

[49] Taihui Li, Xiaorong Zhu, and Xu Liu. An End-to-End Network Slicing Algorithm Based on Deep Q-Learning for 5G Network. *IEEE Access*, 8:122229–122240, 2020.

[50] Federico Mason, Gianfranco Nencioni, and Andrea Zanella. A multi-agent reinforcement learning architecture for network slicing orchestration. *2021 19th Mediterranean Communication and Computer Networking Conference, MedComNet 2021*, pages 1–15, 2021.

[51] Ziyi Xiang. *A comparison of learning for and reinforcement genetic algorithm autonomous driving.* PhD thesis, KTH Royal Institute of Technology, 2019.

[52] Marco Centenaro, Riccardo Fedrizzi, and Lorenzo Vangelista. Why is network reselection an issue for cross-border vehicular applications? In *2020 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE)*, pages 1–6, 2020.

[53] 5GPPP H2020 ICT-18-2018 Projects. 5G Trials for Cooperative, Connected and Automated Mobility along European 5G Cross-Border Corridors - Challenges and Opportunities. 10 2020.

[54] Mosharaf Chowdhury, Fady Samuel, and Raouf Boutaba. PolyViNE: policy-based virtual network embedding across multiple domains. In *Proc. of* {*ACM*} {*VISA*}, 2010.

[55] M. Chowdhury, M. R. Rahman, and R. Boutaba. ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on Networking*, 20(1):206 –219, 2 2012.

[56] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann De Meer, and Xavier Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys and Tutorials*, 15(4):1888–1906, 2013.

[57] Gurobi optimizer. Accessed: 2021-12-15.

[58] F. Z. Yousaf, P. Loureiro, F. Zdarsky, T. Taleb, and M. Liebsch. Cost analysis of initial deployment strategies for virtualized mobile core network functions. *IEEE Communications Magazine*, 53(12):60–66, 12 2015.

[59] Aris Leivadeas, George Kesidis, Mohamed Ibnkahla, and Ioannis Lambadaris. VNF placement optimization at the edge and cloud. *Future Internet*, 11(3):1–23, 2019.

[60] Andres Garcia-Saavedra, George Iosifidis, Xavier Costa-Perez, and Douglas J. Leith. Joint optimization of edge computing architectures and radio access networks. *IEEE Journal on Selected Areas in Communications*, 36(11):2433–2443, 2018.

[61] Giovanni Nardini, Giovanni Stea, and Antonio Virdis. Scalable real-time emulation of 5G networks with Simu5G. *IEEE Access*, 9, 2021.

[62] Claudio Fiandrino, Alejandro Blanco Pizarro, Pablo Jiménez Mateo, Carlos Andrés Ramiro, Norbert Ludant, and Joerg Widmer. openLEON: An end-to-end emulation platform from the edge data center to the mobile user. *Computer Communications*, 148(September):17–26, 2019.

[63] Fengxiao Tang, Xuehan Chen, Tiago Koketsu Rodrigues, Ming Zhao, and Nei Kato. Survey on digital twin edge networks (diten) toward 6g. *IEEE Open Journal of the Communications Society*, 3:1360–1381, 2022.

[64] Tan Do-Duy, Dang Van Huynh, Octavia A. Dobre, Berk Canberk, and Trung Q. Duong. Digital twin-aided intelligent offloading with edge selection in mobile edge computing. *IEEE Wireless Communications Letters*, 11(4):806–810, 2022.

[65] Tong Liu, Lun Tang, Weili Wang, Qianbin Chen, and Xiaoping Zeng. Digital-twin-assisted task offloading based on edge collaboration in the digital twin edge network. *IEEE Internet of Things Journal*, 9(2):1427–1444, 2022.

[66] Dang Van Huynh, Saeed R. Khosravirad, Antonino Masaracchia, Octavia A. Dobre, and Trung Q. Duong. Edge intelligence-based ultra-reliable and low-latency communications for digital twin-enabled metaverse. *IEEE Wireless Communications Letters*, 11(8):1733–1737, 2022.

[67] Rasoul Behravesh, Estefanía Coronado, and Roberto Riggio. Performance evaluation on virtualization technologies for nfv deployment in 5g networks. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 24–29, 2019.

[68] Roberto Morabito. Power consumption of virtualization technologies: An empirical investigation. In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pages 522–527, 2015.

[69] Congfeng Jiang, Yumei Wang, Dongyang Ou, Youhuizi Li, Jilin Zhang, Jian Wan, Bing Luo, and Weisong Shi. Energy efficiency comparison of hypervisors. *Sustainable Computing: Informatics and Systems*, 22:311–321, 2019.

[70] Leila Ismail and Huned Materwala. Computing server power modeling in a data center: Survey, taxonomy, and performance evaluation. *ACM Comput. Surv.*, 53(3), 6 2020.

[71] Weiwei Jiang. Graph-based deep learning for communication networks: A survey. 185:40–54, 2022.

[72] Changsheng You, Kaibin Huang, Hyukjin Chae, and Byoung-Hoon Kim. Energy-efficient resource allocation for mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, 16(3):1397–1411, 2017.

[73] Charu Guleria, Kaushik Das, and Aryabartta Sahu. A survey on mobile edge computing: Efficient energy management system. In *2021 Innovations in Energy Management and Renewable Resources(52042)*, pages 1–4, 2021.