Agile and Scalable Design and Dimensioning of NFV-Enabled MEC Infrastructure to Support Heterogeneous Latency-Critical Applications

by

Lina Abou Haibeh

A thesis submitted to the Ottawa Carleton Institute for Electrical and Computer Engineering in partial fulfillment of the requirement for the doctoral degree

in

ELECTRICAL & COMPUTER ENGINEERING

University of Ottawa

Ottawa, Ontario, Canada

ABSTRACT

Mobile edge computing (MEC) has recently been introduced as a key technology, emerging in response to the increased focus on the emergence of new heterogeneous computing applications, resource-constrained mobile devices, and the long delay of traditional cloud data centers. Although many researchers have studied how the heterogeneous latency-critical application requirements can interact with the MEC system, very few have addressed how to deploy a flexible and scalable MEC infrastructure at the mobile operator for the expected heterogeneous mobile traffic.

The proposed system model in this research project relies on the Network Function Virtualization (NFV) concept to virtualize the MEC infrastructure and provide scalable and flexible infrastructure regardless of the underlying physical hardware. In NFV-enabled networks, the received mobile workload is often deployed as Service Function Chains (SFCs), responsible for accomplishing users' service requests by steering traffic through different VNF types and virtual links. Thus, efficient VNF placement and orchestration mechanisms are required to address the challenges of the heterogenous users' requests, various Quality of Service (QoS) requirements, and network traffic dynamicity.

This research project addresses the scalable design and dimensioning of an agile NFV-enabled MEC infrastructure problem from a dual perspective. First, a neural network model (i.e., a subset of machine learning) helps proactively auto-scale the various virtual service instances by predicting the number of SFCs required for a time-varying mobile traffic load. Second, the Mixed-Integer Linear Program (MILP) is used to create a physical MEC system infrastructure by mapping the predicted virtual SFC networks to the MEC nodes while minimizing deployment costs. Numerical results show that the machine learning (ML) model achieves a high prediction accuracy of 95.6%, which demonstrates the added value of using the ML technique at the edge network in reducing deployment costs while ensuring delay requirements for different latency-critical applications with high acceptance rates. Due to the exponential nature of this MILP formulation, we also propose a scalable bender decomposition approach with near-optimal results at a significantly reduced design and dimensioning cost.

Numerical results show the viability of the bender decomposition approach in its proximity to the optimal dimensioning cost and in its reasonable solution time.

SOMMAIRE

L'informatique mobile de pointe (MEC) a récemment été introduite en tant qu'une technologie clé, en réponse à l'attention accrue portée à l'émergence de nouvelles applications informatiques hétérogènes. Elle a été développée pour résoudre les problèmes liés aux appareils mobiles à ressources limitées et aux temps de traitement plus long des centres de données cloud traditionnels. Bien que de nombreux chercheurs aient étudié comment les exigences hétérogènes des applications critiques pour la latence peuvent interagir avec le système IMP, très peu ont abordé la manière de déployer une infrastructure MEC flexible et évolutive chez un opérateur mobile pour un trafic mobile hétérogène attendu.

Le modèle de système proposé dans ce projet de recherche repose sur le concept de virtualisation des fonctions réseau (NFV) pour virtualiser les ressources physiques du système MEC et fournir une infrastructure évolutive et flexible quel que soit le matériel physique sous-jacent. Dans les réseaux compatibles NFV, la charge de travail mobile reçue est souvent déployée sous forme de chaînes de fonctions de service (SFC), chargées de répondre aux demandes de service des utilisateurs en dirigeant le trafic via différents types de VNF et liens virtuels. Ainsi, des mécanismes efficaces de placement et d'orchestration de VNF sont nécessaires pour relever les défis des demandes hétérogènes des utilisateurs, des diverses exigences de qualité de service (QoS) et de la variation du volume de trafic réseau.

Ce projet de recherche aborde la conception évolutive et le dimensionnement d'un problème d'infrastructure MEC agile compatible NFV dans une double perspective. Tout d'abord, un modèle de réseau neuronal (c'est-à-dire un sous-ensemble d'apprentissage automatique) aide à mettre à l'échelle automatiquement et de manière proactive les différentes instances de service virtuel en prédisant le nombre de SFC nécessaires pour une charge de trafic mobile variable dans le temps. Deuxièmement, un programme linéaire à nombres entiers mixtes (MILP) est utilisé pour créer une infrastructure de système MEC physique en affectant les réseaux SFC virtuels prédits aux nœuds MEC tout en minimisant les coûts de déploiement. Les résultats numériques montrent que le modèle d'apprentissage automatique (ML) atteint une précision de prédiction élevée de 95.6 %, ce qui démontre l'efficacité de l'utilisation de la technique ML au niveau du réseau périphérique pour réduire les coûts de déploiement tout en

garantissant les exigences de délais de propagation pour différentes applications critiques à latence élevée et à forts taux d'acceptation. En raison de la nature exponentielle de cette formulation MILP, nous proposons également une approche de décomposition évolutive avec des résultats quasi optimaux à un coût de conception et de dimensionnement considérablement réduit. Les résultats numériques montrent la viabilité de l'approche de décomposition Binder dans sa proximité avec le coût de dimensionnement optimal et dans son temps de résolution raisonnable.

ACKNOWLEDGMENTS

Words cannot express my gratitude to my supervisor (Prof. Mustapha Yagoub) and my co-supervisor (Dr. Abdallah Jarray) for their invaluable patience and feedback. They generously provided knowledge and expertise and this endeavor would not have been possible without their great support. Additionally, I would like to thank Ismael Martinez (Ph.D. Candidate) for his help in the decomposition approach modeling, implementation, and coding.

Lastly, I would be remiss in not mentioning my family, especially my parents and siblings. Their belief in me has kept my spirits and motivation high during this process. I would also like to thank my little kid Ryan for his patience in staying away from him for long hours working on my thesis and for all the entertainment and emotional support I got from this little cutie.

DEDICATION

First, I dedicate my dissertation work to my little kid, Ryan, who adds more color and joy to my life.

I also dedicate this dissertation to my family and many friends. Special gratitude to my loving parents for their encouragement and push for tenacity rang in my ears. I will always appreciate all they have done.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Greek Symbols

b	Virtual Link Bandwidth
c	Computational Resource Cost
d	Physical Link Delay
E	Substrate Link
F	Predicted Service Function Chains
G	Directed Graph of NFV-Enabled MEC Physical Infrastructure
G_f	Virtual Network for SFC Request
I	Virtual Instances
K	Virtual Links between VNFs in an SFC Request
N	MEC Nodes
P	SFC Request Types
R	Computational Resource
s	Computational Resource Capacity
T	VNF Types
V	VNFs in an SFC Request
β	Substrate Link Bandwidth Capacity
Π	Physical Paths in NFV-Enabled MEC Physical Infrastructure
μ	Processing Rate for MEC Node
∂	E2E Delay Threshold for SFC
λ	Traffic Arrival Rate

ABBREVIATIONS

AI	Artificial Intelligence
APs	Access Points
AR	Augmented Reality
ARIMA	Autoregressive Integrated Moving Average
BD	Benders Decomposition
BSs	Base Stations
CAPEX	Capital Expenses
CFS	Customer Facility Service
CN	Core Network
DCs	Data Centers
DNN	Deep Neural Network
E2E	End-to-End delay
eNB	E-UTRAN Node B
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
FMC	Follow-me cloud
FMCC	Follow-me cloud Controller
FN	False Negative
FP	False Positive
FW	Firewall
GPS	Global Positioning System
GPU	Graphical Processing Unit
HetNet	Heterogeneous Network

HSS	Home Subscriber Server
IDPS	Intrusion Detection and Prevention System
IoT	Internet of Things
IP	Integer Programming
ISG	Internet Specification Group
LSTM	Long Short-Term Memory
LCM	Lifecycle Management
MC	MobiScud Control
MEC	Mobile Edge Computing
MILP	Mixed Integer Linear Program
ML	Machine Learning
MNOs	Mobile Network Operators
MMC	Mobile Micro Cloud
MME	Mobility Management Entity
MPC	Model Predictive Control
NAT	Network Address Translator
NFV	Network Function Virtualization
NFVO	Network Function Virtualization Orchestrator
OPEX	Operational Expenses
OS	Operating System
OSS	Operation Support System
PaaS	Platform-as-a-Service
P-GW	Packet Gateway
PoC	Proof of Concept

QoE	Quality-of-Experience
QoS	Quality of Service
RAN	Radio Access Network
RAT	Radio Access Technology
RB	Resource Block
RIE	Radio Interface Equipment
RNN	Recurrent Neural Network
SCC	Small Cell Cloud
SCeNB	Small Cell E-UTRAN Node B
SCM	Small Cell Manager
SDN	Software-Defined Network
S-GW	Serving Gateway
SFC	Service Function Chain
SL	Supervised Learning
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
TS	Traffic Shaper
UE	User Equipment
URLLC	Ultra-Reliable Low Latency Communication
VIM	Virtual Infrastructure Manager
VM	Virtual Machines
VN	Virtual Network
VNF	Virtual Network Function

VNFM	Virtual Network Function Manager
VOC	Video Optimization Controller
VoIP	Voice Over IP
VR	Virtual Reality
VS	Virtualized Services
WMAN	Wireless Metropolitan Area Network
WOC	WAN Optimization Co

Chapter 1

INTRODUCTION

1.1 Motivation

The 5G era has witnessed tremendous growth in cellular subscriptions and mobile data traffic. Based on Ericsson's latest forecast for 2020–2026, global mobile subscriptions are expected to grow from 7.9 billion to 8.8 billion, and international mobile data traffic is expected to double [1]. Increasing mobile network coverage drives this significant increase in mobile data traffic, the massive deployment of Internet of Things (IoT) devices, and mobile broadband subscriptions. Therefore, the demand for traditional cloud-based services and real-time computing applications in mobile networks increases dramatically, e.g., video streaming, social networking, and online retail services. In addition, there is a high demand for new cloud services, such as mobile cloud games, remote control services for aircraft and ground vehicles, and services that handle manufacturing processes [2].

To meet the increasing demand for user requests in cellular networks, conventional cloud computing platforms (e.g., based on data centers) continuously expand their server capacities and thus improve the QoS they provide. However, for various reasons, new computing platforms and architectures are needed to scale better with the explosion in mobile service demands. For mobile users to access traditional cloud services, mobile traffic must pass through multiple stages, including mobile backhaul and possibly the mobile core operator, resulting in additional communication latency that can exceed the latency requirements of critical services. In addition, conventional cloud computing platforms require intensive computing and communication resources investments to improve the quality of the services provided in cellular networks, which further increases the cost of access to cellular networks [3]. Finally, network knowledge and user context information can help provide localized services to end users and enhance their Quality-of-Experience (QoE).

Mobile Edge Computing (MEC) is emerging as a promising technology to address these needs[4], where it can provide cloud computing capabilities by placing computing infrastructures close to mobile users at the network edge (i.e., Radio Access Network RAN). Leveraging the proximity of mobile end-users, MEC enables mobile devices to offload and run heterogeneous computing application requirements on various virtual machines deployed on physical mobile edge servers, meeting different QoS requirements [5]. Unlike centralized clouds, edge servers can coexist with cellular base stations (BSs) deployed at the edge servers [6].

To build a flexible MEC infrastructure, the concept of network function virtualization (NFV) is one possible approach that could be used to decouple network functions (NFs) from the underlying physical substrate network and transform them into software units known as virtual network functions (VNFs) [7]. The decoupling mechanism used in NFV brings the opportunity for Mobile Network Operators (MNOs) to efficiently upgrade their existing physical network to guarantee QoS for a wide range of mobile traffic with heterogeneous requirements. That clearly minimized the Capital Expenses (CAPEX) and the Operational Expenses (OPEX) for the MEC infrastructure deployment. The virtualization technique enables mobile providers to respond to the dynamics of function services while supporting new use cases such as heterogeneous IoT environments and intelligent transport. The received mobile workload is often deployed as a service function chain (SFC) in such an NFV-enabled network. It processes user requests by routing traffic through various VNF types and virtual circuits. Therefore, effective VNF placement and orchestration mechanisms are required to support heterogeneous user requirements, different QoS requirements, and network traffic dynamics [8].

The unpredictability of cellular traffic is another issue to consider when deploying a flexible MEC infrastructure [9]. Unlike traditional clouds, where mobile workloads are easily predictable, distributed MEC systems encounter workload variations based on time-varying traffic distribution and mobility patterns, leading to uneven workload distribution within the mobile network. As a result, the number of SFCs required to process the workload at any given time changes continuously to meet QoS requirements. To that end, a proactive auto-scaling mechanism can be proposed as a necessary step to quantify the required number of physical resources (i.e., compute and network resources) to be installed on the edge nodes at a given time and to deliver the promised cost reductions in deploying the MEC Systems. Furthermore, in order to accommodate the varying demands of users and handle the disparate requests received,

it is essential to tackle the issue of efficient SFC mapping and optimal VNF placement within the NFV-enabled MEC infrastructure [6].

A proper process of implementing a comprehensive MEC infrastructure is required to meet the requirements mentioned above. To achieve this goal, one must (i) maximize the utilization of computing resources at each edge node, (ii) provide mobile users with cost-effective heterogeneous real-time services, and (iii) ensure the quality of the experience for mobile users. Most of the research focuses on MEC resource management to support heterogeneous computing applications and to meet the requirements described above, assuming an existing MEC infrastructure [1].

1.2 Novelty and Contributions

The objective of this research endeavor is to develop a proactive, scalable, and cost-effective approach for the design and dimensioning of an NFV-enabled MEC infrastructure. The aim is to create a blueprint that can be easily implemented and deployed to support dynamic mobile data traffic and heterogeneous user requirements, while keeping the costs low.

To achieve this, we examine a radio access network of a mobile operator that does not have an existing MEC infrastructure at any of its base station locations. Our study defines a comprehensive MEC infrastructure design as the proactive selection of MEC node locations in the edge network and specifies the dimensioning of each MEC node, including computation resources such as storage, CPU, and memory, to be installed at a given time. The outcome of the dimensioning and design phase is a comprehensive plan for constructing a scalable MEC infrastructure that considers multiple attributes simultaneously, including cost, auto scaling, and optimal placement.

Our approach leverages advanced algorithms and machine learning techniques to enable efficient resource allocation and utilization, while also ensuring low-latency communication and seamless scalability to support heterogeneous latency-critical applications. We have conducted extensive simulations and experiments to validate the effectiveness of our proposed design and dimensioning approach in meeting the diverse needs of modern communication networks.

To the best of our knowledge, most of the existing literature on MEC dimensioning and design assumes either single-user equipment or workloads with homogeneous quality of service

(QoS) requirements (refer to section 3.1). Some works do consider the heterogeneity of mobile application traffic but fail to address traffic dynamics and MEC dimensioning. They assume an infrastructure in which the edge nodes are already deployed and receive a fixed amount of traffic over time. Therefore, the original contribution of this thesis is to present the first comprehensive study on a scalable, machine-driven, and cost-effective design and dimensioning scheme for an NFV-enabled MEC infrastructure. Our approach is based on predicted mobile traffic and intends to design a MEC infrastructure capable of processing upcoming heterogeneous requests from mobile users at various locations. Additionally, our method is scalable and allows for the optimal addition of new MEC nodes to the current infrastructure in response to a continuous increase in mobile users' requests over a specific period.

In this research project, we focus on an infrastructure consisting of MEC nodes that use NFV concept and are physically co-located with the base stations of mobile operators at various locations. We aim to optimize the design, dimensioning, and workload allocation to the MEC infrastructure. To accomplish this, we address and solve the problem of cost-efficient, machine-driven, scalable design and dimensioning of a comprehensive MEC-NFV infrastructure that can support heterogeneous SFCs while meeting different application latency requirements.

The primary contributions of this thesis can be summarized as follows:

- Designing an agile and flexible NFV-based MEC framework that supports the dynamic and heterogeneous nature of mobile traffic generated by diverse users, particularly in terms of computational resources and latency requirements. In this thesis, we propose a machine-driven system model that aims to reduce the deployment costs of a mobile operator while delivering an efficient and scalable comprehensive MEC infrastructure that caters to the requirements of various predicted computing applications. We validate the accuracy of our ML classifiers using various performance metrics, and our simulation results demonstrate the advantages of using ML methods as an initial step in dimensioning the MEC system in terms of cost and traffic acceptance ratio. This research emphasizes the importance of agility in designing an adaptable MEC infrastructure that can keep up with the dynamic mobile traffic while simultaneously providing efficient and cost-effective solutions to the mobile operator.
- Finding an efficient virtual instance mapping method is a significant challenge. To do so, we focus on the dynamic and flexible SFC mapping problem in this thesis. The service

chain is decomposed into a set of virtual nodes (i.e., virtual network functions or VNFs) interconnected by virtual connections in a specific order. This issue works in conjunction with a proactive auto-scaling mechanism that leverages scaling decisions to suggest an optimal VNF placement at any given time to minimize end-to-end delay. Our research highlights the importance of an efficient and dynamic VNF mapping technique in designing and scaling MEC systems while meeting the diverse requirements of different applications.

- Effective utilization of physical resources at edge servers is a critical consideration that should not be overlooked. To address this issue, we consider the shareability of virtual instances and resource overheads in the SFC mapping algorithm that calculates the resources required to process SFC requests while meeting latency requirements. Our simulations demonstrate the advantages of sharing virtual instances by different edge nodes in terms of dimensioning costs. We emphasize the importance of effective physical resource utilization at edge servers to reduce costs and enhance the performance of MEC systems.
- To address the computational complexity of the one-shot MILP problem in MEC system design, we propose a bender decomposition approach. This approach has been implemented and evaluated against standard benchmarks such as the Greedy Algorithm model, and our numerical results demonstrate its efficiency in reducing computation time. We show that our proposed approach can provide a practical and effective solution for optimizing MEC system design and dimensioning.

1.3 Thesis Objectives

The objective of this research project is to design and dimension a comprehensive NFV-enabled MEC infrastructure that can efficiently support a variety of latency-critical applications with varying requirements. To achieve this goal, several research objectives have been identified, including:

- Identifying the infrastructure requirements for a scalable and cost-effective MEC deployment, including the optimal placement and dimensioning of MEC nodes in the edge network.
- Leveraging machine learning techniques to predict mobile traffic patterns and computing

application demands, which can be used to optimize the MEC infrastructure design and dimensioning scheme.

- Developing an efficient and scalable virtual instance mapping method to minimize end-to-end latency and maximize resource utilization.
- Addressing the challenges of sharing virtual instances and minimizing resource overheads in the SFC mapping algorithm to further optimize the MEC infrastructure's performance.
- Proposing a bender decomposition approach to overcome the computation complexity of the MILP problem and reduce the computational overhead of the optimization process.

By achieving these research objectives, the proposed MEC infrastructure can be designed and dimensioned to provide a flexible and agile platform for supporting heterogeneous latency-critical applications with varying requirements.

1.4 Thesis Outline

In this research work, we outline the structure of six chapters. Following this introduction, Chapter 2 provides a comprehensive overview of the MEC architecture's key components and their interplay to achieve the MEC system's benefits, such as supporting low-latency real-time applications, reducing energy consumption, service cost, and facilitating user mobility.

Chapter 3 delves into the current research on MEC infrastructure design and dimensioning, highlighting the significance of deploying MEC systems on NFV infrastructure to maximize efficiency and obtain benefits. It discusses the current state of MEC dimensioning and design, identifies the shortcomings in current MEC dimensioning works, and reviews various auto-scaling approaches, including static and dynamic approaches, as well as service placement algorithms based on resource and service types for end-users. Finally, the chapter concludes by discussing MEC optimization methods that aim to achieve high performance.

In Chapter 4 of this thesis, we introduce our proposed comprehensive, agile and scalable MEC system model enabled by NFV. We define the design and dimensioning MILP problem statement, along with the parameters, constraints, and main objective of our model. We also address the scalability issue that arises in high-scale MEC systems by proposing the Bender

decomposition approach as an alternative to the one-shot MILP approach. Furthermore, we present the benchmarks used to validate the effectiveness of our novel dimensioning approach.

Chapter 5 of this thesis provides a detailed description of the MEC simulation setup, numerical evaluation, and results discussion. We use this chapter to validate the benefits and added value of our agile approach in improving the efficiency and scalability of MEC systems.

Finally, Chapter 6 provides a summary of the key findings of this research project and presents our future work in this area, including exploring the use of other optimization techniques, such as machine learning, to enhance the agility and scalability of MEC systems.

1.5 Related Publications

To date, the outcomes of the thesis are as follows:

Journals

- L.A. Haibeh, M.C.E. Yagoub, A. Jarray, “A Survey on mobile edge computing infrastructure: design, resource management, and optimization approaches,” *IEEE Access*, Vol. 10, Nº 2, pp. 27591-27610, February 2022.
- L.A. Haibeh, M.C.E. Yagoub, A. Jarray, “Machine-driven design and dimensioning of NFV-enabled MEC infrastructure to support heterogeneous latency-critical applications,” submitted to *Journal of Cloud Computing*, under review.

Conferences

- L.A. Haibeh, M.C.E. Yagoub, A. Jarray, "Agile design and dimensioning of MEC-NFV infrastructure to support heterogeneous service chains," *IEEE Global Communications Conference* (GLOBECOM 2022), Rio de Janeiro, Brazil, Dec. 4-8, 2022.

Chapter 2

MOBILE EDGE COMPUTING

2.1 MEC Overview

Mobile Edge Computing is an innovative cloud service delivery paradigm that brings cloud computing capabilities closer to mobile users, leveraging computing and storage resources at the edge of the mobile network. [10]. According to the European Telecommunications Standards Institute (ETSI), the MEC is characterized by low-end latency, local computing and storage resources, network awareness, and enhanced service quality provided by mobile operators. MEC requires seamless integration of both MNOs and the service providers in terms of architectural design and resource management [11]. The increasing demand for high-performance, computation-intensive applications and the need for MNOs to reduce time-to-market for new applications are the driving forces behind MEC's development. Therefore, the involvement of various stakeholders, including network service providers, MNOs, and mobile end-users, is necessary for the successful deployment of MEC.

Referring to ETSI white paper [12], MEC is differentiated by the following:

- On-premises: MEC can be deployed and run in an isolated environment where it has access only to its local computing resources and is completely separated from the rest of the network.
- Proximity: MEC servers are typically placed in proximity to mobile users. The close distance would make it possible for mobile operators to collect and store real-time information from mobile users and process it for different purposes, such as big data analytics and support location-aware services.
- Low latency: MEC systems can reduce propagation and communication latencies and avoid network congestion on front and backhaul network links. Thus, this will make it

possible for MEC to be a key enabler for latency-critical 5G applications while enhancing the content and service responsiveness time.

In this chapter, the MEC architecture is examined in terms of flexible resource management, different MEC solutions, popular MEC use cases, advantages, and disadvantages of MEC, as well as possible challenges that may arise.

2.2 MEC Use Cases

MEC is advantageous to all parties involved, such as mobile operators, service providers, and users. Based on [13], MEC can be classified into three main categories of use cases that generate profit for different stakeholders, as illustrated in Figure 2.1. The ensuing sections provide a comprehensive analysis of each use case category and emphasize critical service scenarios and applications.

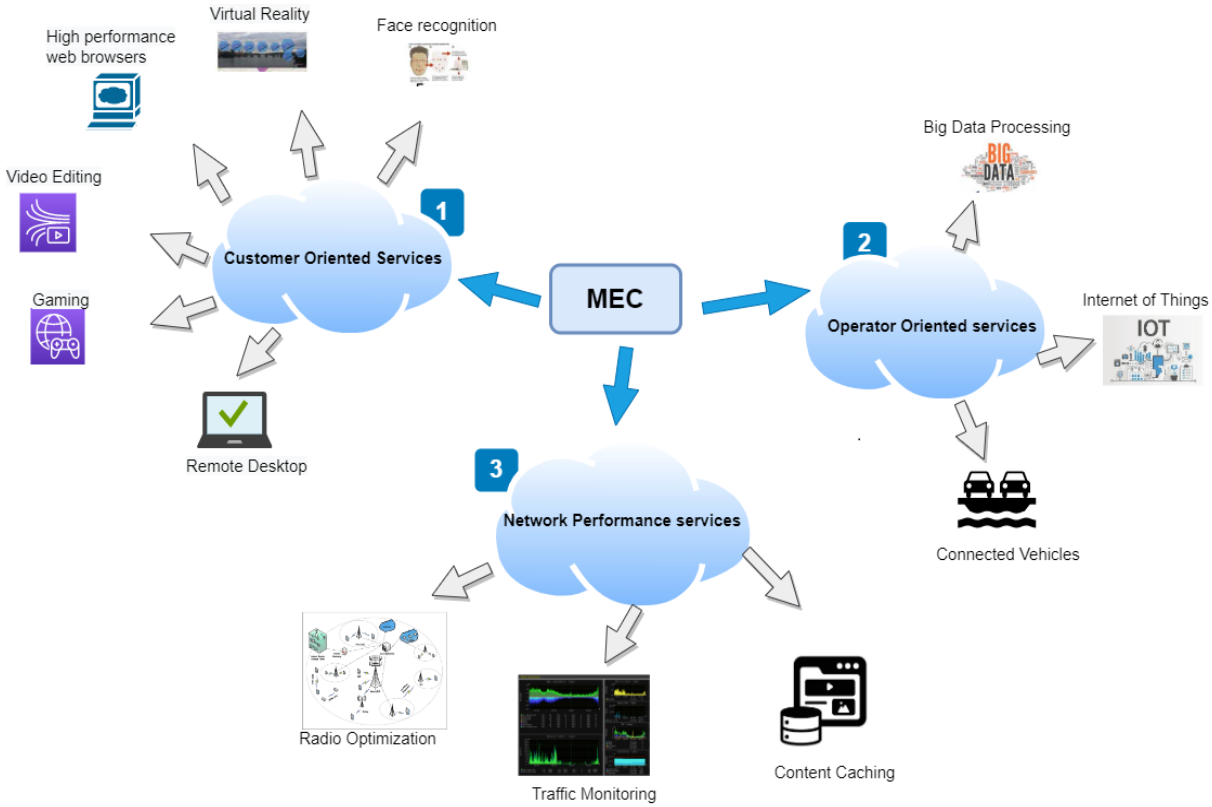


Figure 2.1 Different MEC Use Cases

2.2.1 Consumer-Oriented Services

The initial MEC use case category is focused on consumers and aims to provide direct benefits to end-users. The main advantage for users is the ability to offload computation, which enables the operation of new and emerging applications on user equipment (UE). An example of an application that benefits from cloud computing is an accelerated web browser, where the MEC system controls most of the browsing functions, including web content evaluation and optimized transmission [14].

In addition, computation offloading to MEC can be beneficial for augmented reality (AR) applications that require a significant amount of computing resources to evaluate the user's surroundings and respond rapidly. The user device may not have sufficient resources to support such applications. The feasibility of MEC for AR applications was demonstrated in a real-world testbed by the authors in [15] . Their findings showed that MEC reduced latency by up to 88% and saved energy. Computation offloading to MEC can utilize up to 93% of the UE.

Furthermore, the proximity of MEC to the user can benefit low-latency applications such as online gaming or video streaming. A new instance of a specific application can be launched at a suitable mobile edge host to reduce the application's latency and resource requirements on the user's mobile device.

2.2.2 Operator Services

The second use case category involves services that provide benefits to MNOs and third parties. One example is collecting large amounts of data from users or sensors, which are pre-processed and analyzed in MEC before being sent to conventional central servers for further assessment. This can be useful for applications such as safety and security, such as monitoring a specific area.

Another application of the MEC is for IoT (Internet of Things) purposes [16]. IoT devices rely on various radio technologies and communication protocols that require a delay-sensitive aggregation point to manage multiple protocols, message distribution, and computation. The MEC serves as an IoT gateway that can aggregate and provide IoT services to highly distributed base stations, enabling real-time application responses.

2.2.3 Network Performance Services

The third use case of MEC involves optimizing the performance of MNOs and enhancing the overall user experience. An important application in this category involves the coordination of radio and backbone networks. Currently, when one part of the network experiences a degradation in capacity, it affects the entire network, leading to a suboptimal performance. By leveraging MEC, real-time analytics applications can provide valuable insights into the network's traffic requirements. The optimization application can then adjust the traffic flow per application or reroute it as needed, thus improving the network's performance and user experience.

Another strategy to enhance network performance involves local content caching at the mobile edge, which can reduce congestion on backhaul links. MEC applications can store the most popular content in the region, allowing users to access it without transferring it over the backbone network [17].

The MEC can contribute to improving radio network performance and mitigating backhaul network issues. By gathering and analyzing relevant data from users' devices at the MEC server, the scheduling process can become more effective. Moreover, MEC can enhance mobile video delivery by using Transmission Control Protocol (TCP) throughput guidance. TCP faces difficulty adapting to quickly changing radio channel conditions, which results in inefficient resource utilization. Therefore, the MEC analytic application can transmit an estimate of real-time throughput to the backend video server to align the application-level coding with the approximate throughput, as suggested in[18].

2.3 MEC Solutions

In this section, a range of concepts are discussed pertaining to computation at the edge of mobile networks. The initial focus is on different MEC solutions that allow for computation to be performed in closer proximity to the users’ equipment.

2.3.1 Overview of MEC Concepts

This subsection outlines various MEC concepts that could facilitate the seamless integration of cloud capabilities into mobile network architecture. The fundamental principles of small cell cloud (SCC), mobile micro cloud (MMC), fast-moving personal cloud (FMC), and

CONCERT are briefly introduced, along with the necessary network architecture enhancements and modifications required for the implementation of each MEC concept.

2.3.1.1 Small Cell Cloud (SCC)

In 2012, the TROPIC project proposed a concept called small cell cloud (SCC), which aims to enhance small cells (SCeNBs) like femtocells, microcells, and picocells with extra computing and storage resources [19]. The SCC employs the network function virtualization (NFV) approach to create a cloud-enhanced pool of SCeNBs that can supply adequate computing power for user devices, particularly for latency-sensitive services. Given the anticipated widespread deployment of SCeNBs in future mobile networks, SCC can effectively cater to users' computation needs.

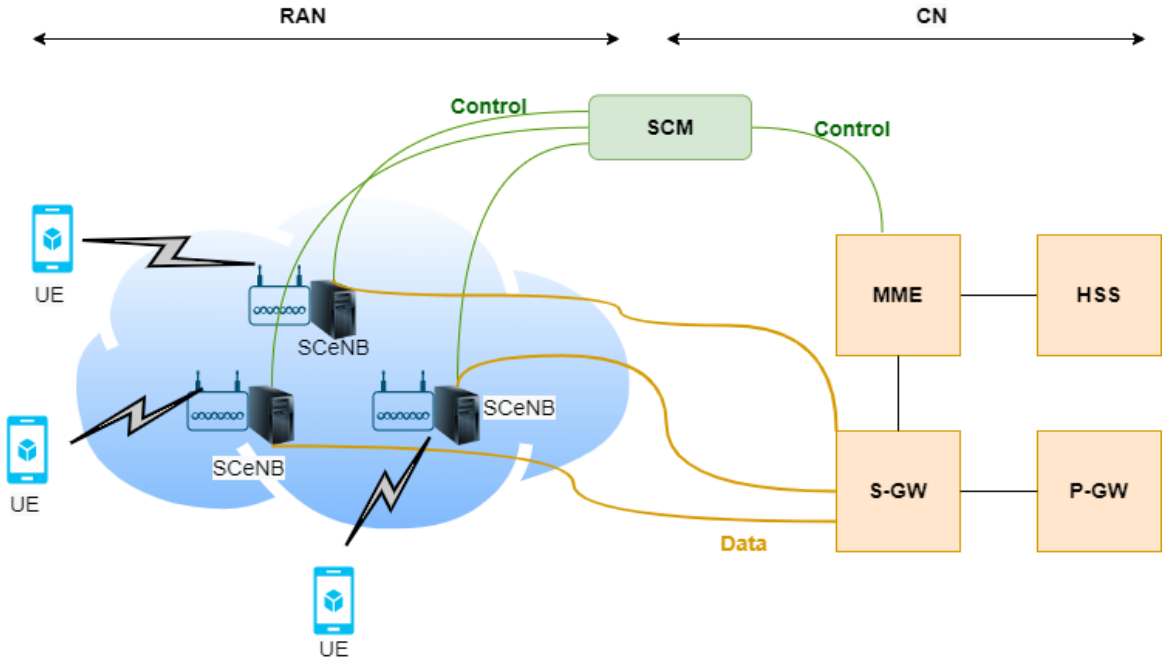


Figure 2.2 Centralized SCM

To effectively integrate the SCC concept into mobile network architecture, a new entity known as the small cell manager (SCM) is introduced to control the SCC [19]. The SCM manages the computing and storage resources made available by the SCeNBs in a dynamic and elastic manner, given that SCeNBs can be turned on and off at any time. To improve service delivery, the SCM is aware of the main cluster context and decides when and where to deploy a new edge node or relocate a continuing computation. Virtual Machines (VMs) hosted on

SCeNBs are used to virtualize computing resources. The deployment of the SCM is a crucial aspect of the SCC's architecture, as shown in Figure 2.2. The SCM can be implemented centrally within the RAN as a standalone SCM, near a group of SCeNBs, or as an extension to an MME. Additionally, the SCM can be deployed in a hierarchical manner, with a local SCM (L-SCM) or a virtual L-SCM (VL-SCM) managing the storage and computing resources of nearby SCeNB clusters, while a remote SCM (R-SCM) located in the CN has access to the resources from all SCeNBs connected to the CN (as illustrated in Figure 2.3).

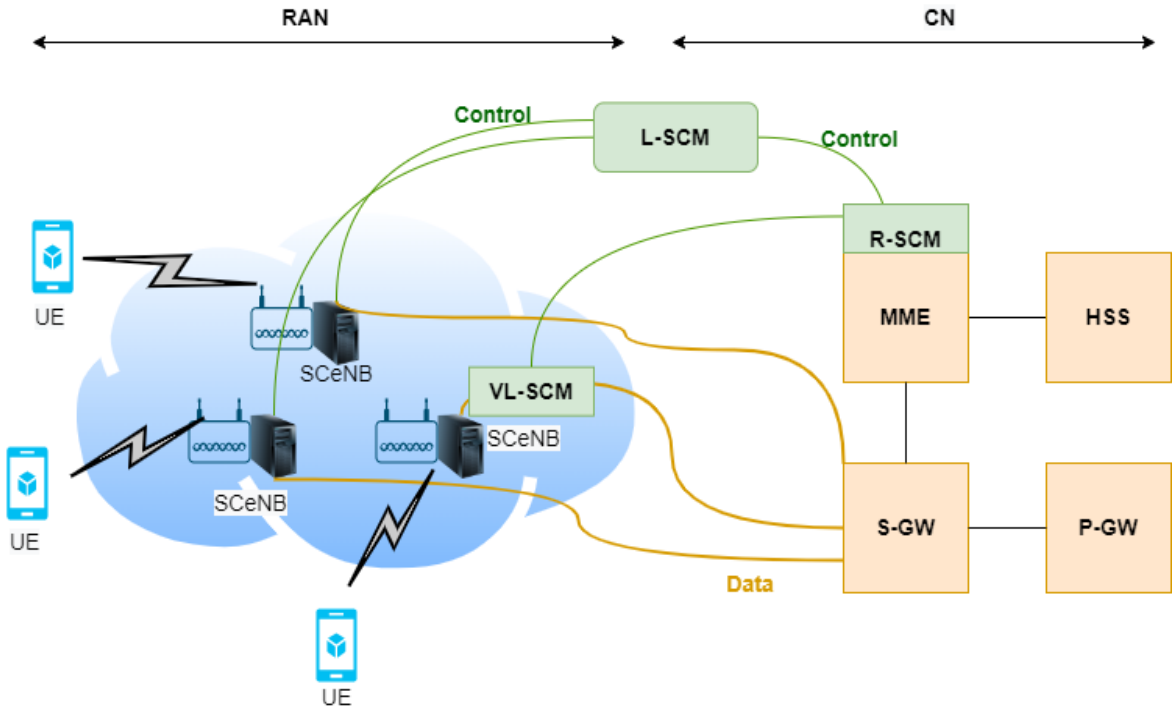


Figure 2.3 Distributed Hierarchical SCM

2.3.1.2 Mobile Micro Clouds (MMC)

The concept of the mobile micro cloud (MMC) was introduced in [20], aiming to provide users with instant access to cloud services with minimum latency. Unlike the SCC that utilizes interworking clusters of small cells to provide computation resources, the MMC enables user devices to access the computation resources of a single cloud instance, which is usually directly connected to a base station, as shown in Figure 2.4 [20]. The MMC does not require any control entities within the network, and the control is fully distributed, similar to the VL-SCM solution for the SCC. To ensure seamless VM migration among the MMCs and network reliability, the MMCs are directly or indirectly linked via broadband services.

2.3.1.3 Fast Moving Personal Cloud (MobiScud)

The MobiScud architecture, as introduced in [21], integrates cloud services with mobile operators using software-defined networking (SDN) and network function virtualization (NFV) technologies, while maintaining compatibility with existing mobile networks. Unlike the SCC and MMC approaches, the MobiScud does not place cloud resources directly at the SCeNB or eNB access nodes. Instead, it places cloud resources in operator clouds located within or near the RAN, as depicted in Figure 2.5.

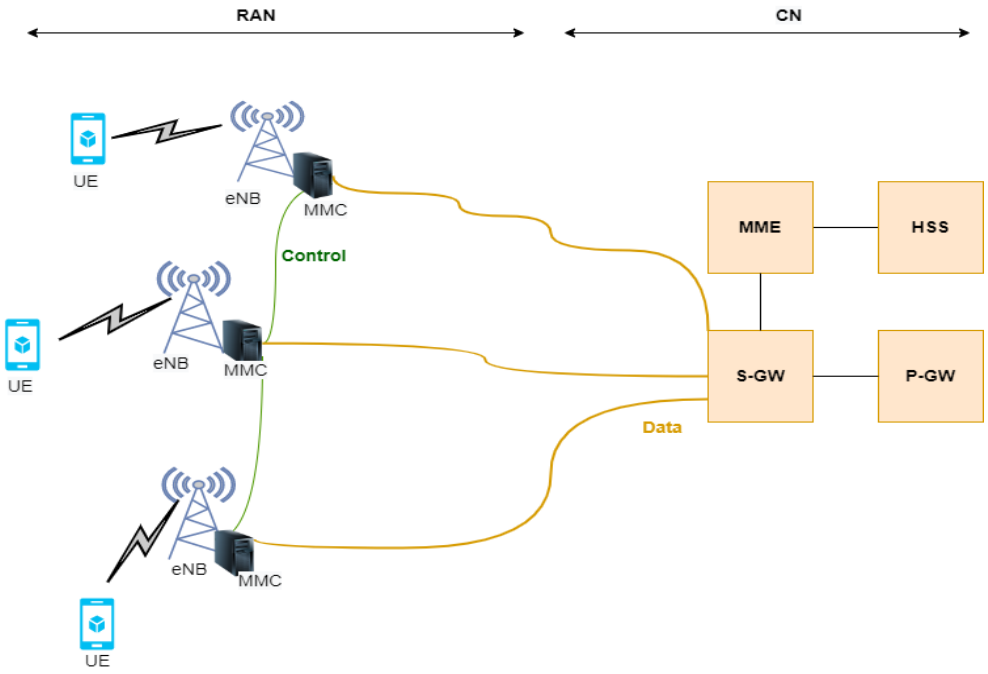


Figure 2.4 MMC Architecture

Despite the differing placement of cloud resources within the mobile network, the MobiScud, like the SCC and MMC, is designed to be extensively distributed, enabling users in the vicinity to access cloud services.

Similar to the SCC and MMC, the MobiScud also introduces a novel control unit, the MobiScud control (MC), which interfaces with the MNO, SDN switches, and cloud servers. The MC has two main functions: (i) tracking control plane signaling messages exchanged between MNO elements to detect user activity; and (ii) managing and monitoring data traffic flows within SDN-enabled transport networks to facilitate application offloading and VM migration as the user moves through the mobile network [21].

2.3.1.4 Follow Me Cloud (FMC)

The concept of FMC, as introduced in [22], involves distributing cloud services across data centers (DCs) that follow the roaming of user equipment (UEs) throughout the network, much like the MobiScud. However, in contrast to prior Mobile Edge Computing (MEC) concepts, the FMC places computing power in the operator's central network rather than at the user level. The FMC leverages the decentralization of mobile operators' networks necessitated by growing user demands, instead of assuming a more centralized network deployment. This entails replacing the centralized CN of current network deployments with a decentralized one, as depicted in Figure 2.6 [22], with the data center situated alongside the distributed S/P-GWs for the operator's convenience.

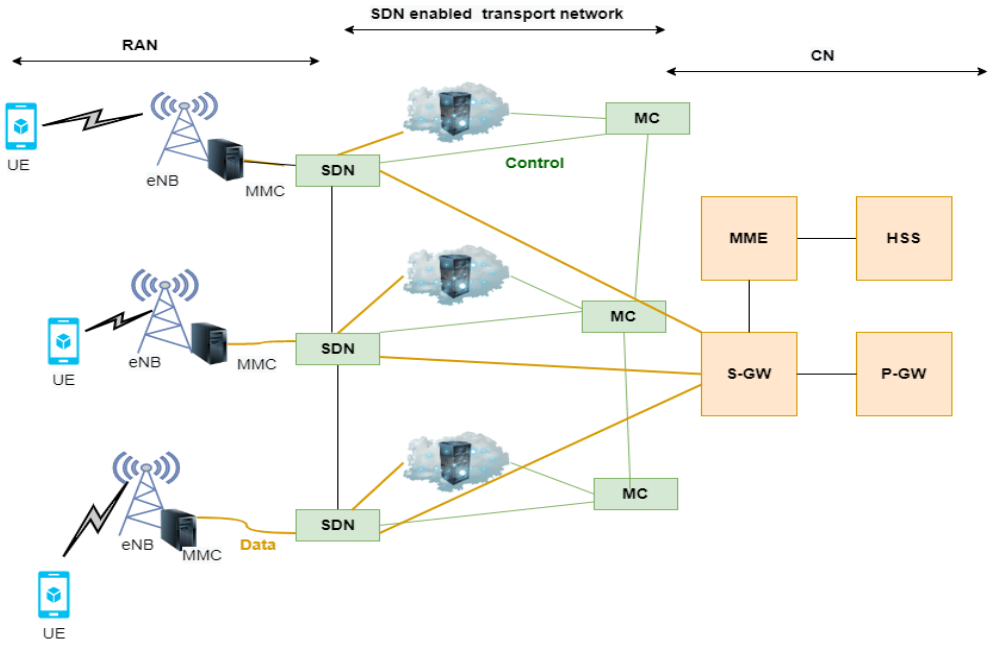


Figure 2.5 MobiScud Architecture

The FMC introduces new network units, including a DC/GW mapping unit and an FMC controller (FMCC). These units can be operational units that coexist with current network nodes or software that runs on any data center. The DC/GW mapping unit statically or dynamically connects distributed S/P-GWs to DCs based on various metrics such as position or number of hops between the data center and distributed edge networks. The FMCC handles the computation

resources of data centers and cloud services hosted on them, and it decides which data center should be associated with the user device via the cloud services. For better management [22], the FMCC can be implemented centrally, as shown in Figure 2.6, or in a hierarchical structure with a global FMCC (G-FMCC) and local FMCC (L-FMCC). Notably, the FMC can be decentralized by ignoring the FMCC, in which case the data centers coordinate themselves using a self-organizing approach.

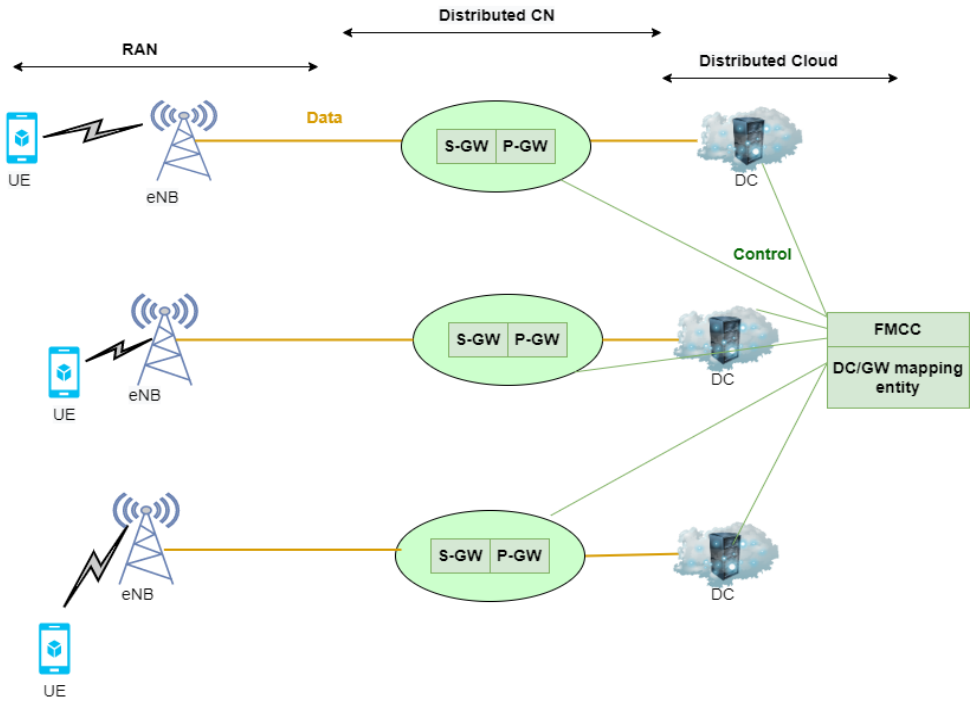


Figure 2.6 FMC Architecture

2.3.1.5 CONCERT

The concept of CONCERT proposed in [23] aims to combine cloud and cellular systems by utilizing NFV principles and SDN technology. Virtual resources are employed to represent the computing and storage resources used by traditional mobile communication and cloud computing services. The control plane is managed by a conductor that oversees the computing and communication cloud resources of the CONCERT architecture, and this conductor can be deployed centrally or hierarchically for scalability, similar to the SCC or FMC. The data plane consists of radio interface devices (RIEs) that physically represent the eNB, SDN switches, and computing resources, as shown in Figure 2.7. Unlike the earlier MEC concepts where the computation resources were distributed, the CONCERT suggests a hierarchical placement of

network resources for flexible and elastic network and cloud service management. Local nodes with low processing power are physically placed at the base station, and regional or central servers are employed when local resources are inadequate.

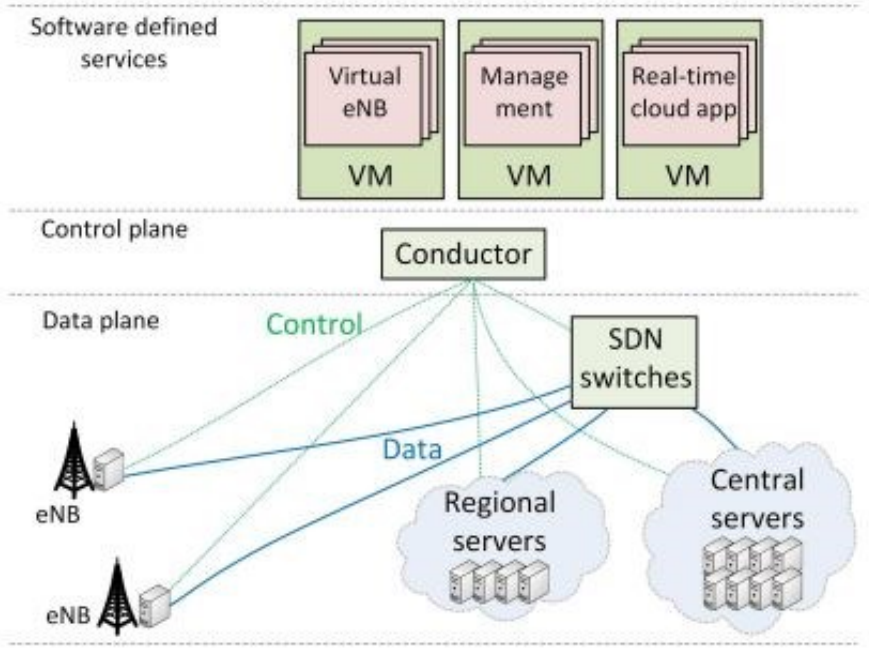


Figure 2.7 CONCERT [23]

2.4 MEC Architecture

Mobile Edge Computing (MEC) refers to the provision of computing and storage resources as mobile end-user services. MEC resources are typically deployed in close proximity to end-users and can be utilized at indoor and outdoor base stations, access points, and radio access networks that connect user equipment to the core network of the Mobile Network Operator (MNO) [11]. The specific deployment of MEC resources is influenced by various factors, including physical constraints, performance requirements, and the preferences of network operators. To illustrate the different deployment options for MEC, Figure 2.8 categorizes MEC components based on their functions and interactions in achieving MEC system objectives [24-29] . The first category is the MEC end-users, including mobile users, connected cars, and IoT devices. The second category is computing components that are used to handle end-user applications. Finally, the end-user and computing components are connected by communication components, the third category of MEC components.

1) Communication units include wireless or wired communication networks.

a) Wireless networks are considered the primary way to provide connectivity for end-users in the MEC. End-users can be either connected to an unlicensed radio spectrum (i.e., WiFi) or directly connected to base stations and mobile access points that operate on a licensed radio spectrum [25]. For example, 5G networks should provide enhanced support for bandwidth-intensive applications in MEC [26]. Additionally, 5G networks support Ultra-Reliable Low Latency Communication (URLLC), which is expected to achieve 1-millisecond latency and support applications that require strict latency requirements (e.g., process control and control services for drones) [27]. In addition to connecting end-users, wireless is considered an alternative to a wired backhaul, so wireless can also be used to communicate with MEC resources [28].

b) As for wired networks, the mobile backhaul network interconnects the various RAN components and the mobile core, establishes communication between the MEC components and the base stations or access points, and, finally, the end-users. The physical infrastructure consists of cables (i.e., fiber) and various forwarding and routing devices. Because transmission time is relatively short in wired environments, packet processing time on routers and switches constitutes a significant part of the delay [28].

In communication networks, network functions allow monitoring and managing communication resources. Typical networking functions include resource provisioning, orchestration, performance analysis, error detection, and load balancing.

Network Functions Virtualization is becoming increasingly popular in the telecommunications industry as it facilitates network management agility by virtualizing heterogeneous network functions provided by dedicated hardware [29]. Moreover, MEC-related network functions can also be virtualized [30].

2) Computing units

Computing units are another MEC component category that provides computing and storage resources to serve MEC end users. These components include the MEC nodes and orchestrator:

a) MEC nodes may be heterogeneous depending on the performance requirements and deployment conditions. These nodes can have out-of-the-box commercial servers, small form

factor servers, and purpose-built servers (for example, built-in artificial intelligence capability and high-volume storage capacity). MEC node consists of three components: Virtualized Services (VS) corresponding to the user application, a runtime environment that provides software support for running VS, and built-in hardware components. Due to the tight integration of MEC and mobile networks, MNOs can be owners and operators of MEC nodes. However, third-party service providers, such as property facility owners and cellphone tower owners, may have MEC nodes for their benefit, depending on the cost and complexity of deployment [31].

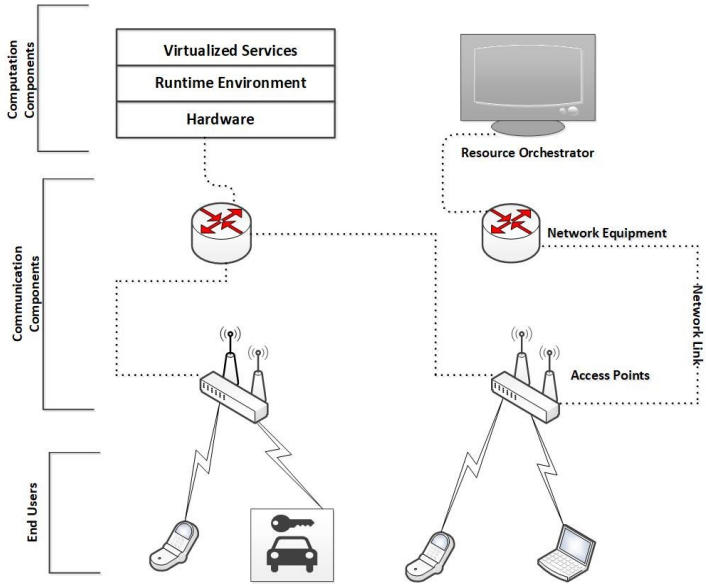


Figure 2.8 An illustration of the main MEC components

b) Resource orchestrators oversee MEC nodes to use their computing resources and provide the expected QoS efficiently. The main resource orchestrator functions include resource allocation, virtual network service placement, task scheduling, and software updates. Additionally, resource orchestrators may be responsible for detecting failures in the MEC and enabling failed schemes [32].

2.5 MEC Benefits

In MEC systems, the computing resources are not centralized at the conventional cloud servers. They are distributed at the mobile network edges near the end-users. MEC infrastructure may include tens of large data centers and thousands of small ones collocated with cell towers and separated by less than 10 miles [33], as shown in Figure 2.9.

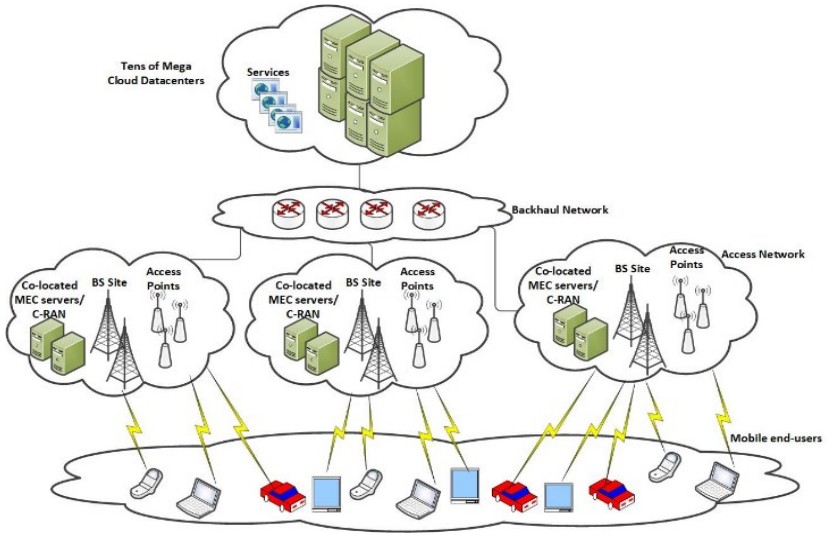


Figure 2.9 General MEC Architecture

This scalable and flexible architecture allows MEC to offer the benefits below:

- Highly distributed and heterogeneous resources: Edge Data Centers (DCs) in MEC are allocated in different geographical locations, and they vary in scale and type in terms of computing resources such as processing resources, storage resources, and network resources.
- Support real-time applications with low latency: MEC is an excellent choice for services requiring guaranteed QoS or low-latency communication, receiving high traffic from end-user devices, or needing extensive data analysis.
- Less mobile energy consumption: In MEC, compute-intensive applications can be offloaded from mobile devices to edge servers. Computation offloading will significantly reduce their energy consumption and prolong battery life.
- Mobility support: MEC can support mobile end users, including smartphones, IoT devices, and sensors, and provide seamless access to the network by changing their attachment points to the network as they move.
- Improving Security: The distributed deployment and the small-scale nature of the edge servers in MEC make them less vulnerable to security attacks. Moreover, MEC prevents uploading critical data to remote data centers. The IT administrator is responsible for maintaining the enterprise's authorization and access control rules and categorizing various service requests without needing an external unit.
- Interoperability with the traditional centralized cloud: In some cases, getting resources

from distant centralized clouds with much greater computing and storage capabilities is cheaper than MECs. So MECs should exploit resource allocation techniques to allocate end-users with computing resources with less latency, cost, energy consumption, and improved performance.

2.6 MEC Potential Challenges

In MEC systems, despite the various opportunities that MEC can offer, several potential challenges are needed to be investigated to allow network players such as mobile end-users, infrastructure providers, and MNOs to get an advantage from the edge services:

- Distributed resource allocation management: A multi-criteria resource management technique is required to handle the dynamic behaviors of the MEC system due to device mobility and the regular change in the computing application requirements [34]. Implementing a multi-objective resource management technique requires multi-criteria schedulers to be combined. This could be challenging due to the variable application types, heterogeneous MEC servers, various user requirements, and the different QoS requirements of the communication channels. Furthermore, the wireless channel would become congested with the massive increase in mobile devices, and users would compete fiercely for limited computing resources. This issue could be resolved using the centralized approach, but it has one drawback: high computing complexity and high reporting overhead. As a result, the centralized process is not convenient for distributed MEC systems [35]. Based on that, reliable and distributed MEC resource allocation schemes needed to be investigated.
- System interoperability and application portability: Depending on the users' locations and the technical requirements, physical nodes can be deployed at different locations within the MEC infrastructure. As a result, a new crucial challenge is MEC's transparent integration into the underlying existing infrastructure and interfaces without affecting the standard specifications of the core network and end device. According to [36], the ability of the MEC system to communicate with other system elements in the 5G network to manage users' workloads and get appropriate control information is a critical component of MEC integration. Furthermore, the application migration points to a requirement known as application portability to eliminate the need for software designers to create

multiple instances for various MEC frameworks.

- Reliability and mobility: Managing mobility and ensuring reliability in a dense environment is extremely difficult. First, when multiple small servers are used, user mobility can lead to frequent handovers, service outages, and overall poor network performance [37]. Second, mobile users can change their positions during the computation offloading time (e.g., vehicles, mobile devices). In such a case, mobile users may be unable to access the computation outcome after processing their request as they left the service area of the home serving node. As a result, reliable computational offloading methods are required for the task computation's success taking into account the dynamic changes in the number of offloading users [38]. Third, considering the dynamicity of wireless connections and user mobility, providing efficient edge computing services in mobile environments is extremely difficult. For example, real-time applications like Augmented Reality (AR) necessitate real-time response and reliable connectivity between the edge nodes and end-users. Furthermore, these requirements would not be fully met due to dynamic channel quality and intermittent connectivity.
- The synchronicity of decentralized MEC and centralized cloud: The conventional centralized cloud data centers, with massive computational and communication resources, can process big-data applications in real time and serve many users. However, the decentralized MEC infrastructure is highly desirable because it meets QoS user requirements and reduces latencies caused by traffic congestion and transmission delay[39]. It is advantageous to implement MEC hierarchically, consisting of three main layers: mobile user, edge-computing, and cloud-computing layers, like the Heterogenous Network (HetNet) architecture. The MEC provider also adds computational resources to the smaller E-UTRAN Node B (eNB), enabling HetNets to diversify radio transmissions and spread computing demands. It is observed that decentralized MEC may not have enough computing resources to process all user requests, raising concerns about providing latency-critical services. As a result, it makes sense to distribute the latency-critical computing applications to decentralized MEC servers while moving compute-intensive and delay-tolerant applications to the conventional remote cloud [40].
- Security and privacy: MEC system has security and privacy difficulties. First, MEC can coexist with various network equipment, so using traditional privacy and security

mechanisms used earlier in conventional cloud systems is inappropriate for MEC systems. Second, malicious leakers can perform overhead computation tasks. Thus, task offloading over wireless channels can be unsafe. The encryption on the user side and decryption on the destination server side should be used to secure the transfer of computing applications. This, however, can increase the propagation and execution delays and reduce application performance [41].

Despite all these challenges, there is a potential need for a Mobile Edge Computing system in many critical real-life applications scenarios. Chapter 3 highlights the main works in MEC design and dimensioning.

2.7 Conclusion

This chapter delves into the concept of Mobile Edge Computing (MEC) as a solution to address issues with centralized cloud computing. MEC involves deploying IT infrastructure at edge nodes to support mobile users' requests. The components of MEC architecture are discussed, highlighting their interaction and contribution to the benefits of the MEC system, such as distributed and heterogeneous resources, low-latency real-time applications, reduced mobile energy consumption, improved security, integration with centralized cloud, and support for mobility. However, the distributed and heterogeneous resources feature also poses potential challenges that have been explored in various works to enable different network players to benefit from MEC, including mobile end-users, infrastructure providers, and MNOs. The subsequent chapter examines works in the MEC framework, such as MEC virtualization techniques, MEC design and dimensioning, and service placement in MEC.

Chapter 3

MEC IMPLEMENTATION PROCESS OVERVIEW

3.1 Introduction

In edge computing, where and how to place the edge server is a vital issue. An appropriate MEC infrastructure implementation process is then required. To this aim, one needs to (i) maximize the used computation resources at each edge node, (ii) provide efficient real-time services to the mobile users, and (iii) guarantee the quality of experience for the mobile users.

To do so, we identified three stages to implement MEC infrastructure.

Initially, if there is no existing MEC infrastructure near the Base Station (BS) of the mobile network operator, the first stage of MEC infrastructure implementation is building a MEC infrastructure based on the workload predictions for the mobile traffic received from end-users such as mobile users, IoT devices, and connected vehicles at each BS location. For instance, the high mobile traffic received in specific BS locations can increase network congestion on the cloud and degrade network performance. Therefore, deploying localized MEC infrastructure in areas with high traffic can benefit conventional cloud providers, mobile core operators, and internet service providers.

The second stage involves the interaction between the MEC infrastructure and mobile end-users and the allocation of computational resources to meet their requirements. MEC resource management is a multi-objective process that involves deciding the location, type, and amount of computing resources to be assigned to each request, taking into account the network's dynamic properties and user mobility behavior. The resource management process is heavily reliant on other hardware or software architectures integrated into the MEC infrastructure, such as the MEC resource orchestrator, to ensure efficient node selection for edge computing.

In the third stage, the MEC infrastructure is optimized to meet the performance requirements of mission-critical applications. This involves identifying the Quality of Service

(QoS) parameters to optimize for each component of the MEC environment, including the service provider, infrastructure provider, and mobile end-user. The process must efficiently utilize computational resources and infrastructure to ensure optimal performance.

These three stages involve four major phases and are critical to the successful implementation of MEC infrastructure in mobile operator environments:

- *Phase 1*: Estimate the volume of mobile traffic that is received from mobile end-users at a particular BS location. Based on this estimate, a new MEC infrastructure can be designed and sized appropriately to cater to the traffic demand.
- *Phase 2*: Determine Virtualized Network Service (VNF) auto-scaling and placement mechanisms used in MEC-NFV infrastructure.
- *Phase 3*: Deploy the MEC framework with supplementary hardware and software components dedicated to MEC resource management and VNF scaling decision-making.
- *Phase 4*: Optimize the MEC infrastructure to achieve adequate QoS for end-user requests.

This chapter provides a comprehensive survey of the literature related to the key issue addressed in this thesis, namely, the design and dimensioning of MEC. It then examines the current state of research concerning the various mechanisms and methodologies that can be employed in the implementation of MEC.

3.2 MEC Design and Dimensioning- Literature Review

The present research delves into the concept of deploying a comprehensive, scalable, and agile MEC infrastructure, with a particular focus on dimensioning the infrastructure to support mobility and latency-sensitive computing applications for users. Numerous studies have been conducted to explore the utilization of existing edge infrastructures to achieve this goal. This section presents the various MEC dimensioning approaches that have been proposed, including those based on the Cloudlets Placement Approach, computational and communication resource requirements, network latency requirements, user mobility, and reliability. This work aims to contribute to the body of knowledge surrounding the deployment of scalable and agile MEC infrastructure by exploring the most effective dimensioning approaches to support latency-sensitive computing applications.

3.2.1 Cloudlets Placement Approaches

Several studies have proposed various approaches to address the issue of cloudlet placement in edge clouds, however, there exist some limitations and shortcomings in these approaches.

Jia *et al.* [42] have introduced a novel cloudlet placement algorithm that takes into account the allocation of mobile users in a wireless area network. Their approach involves two heuristic algorithms, namely a simple heavy-AP first algorithm and a density-based clustering algorithm. However, the simple heavy-AP first algorithm has a disadvantage of placing cloudlets solely at the access points with the highest user number, which can result in overburdening certain regions with cloudlets. On the other hand, the density-based clustering algorithm aims to alleviate the shortcomings of the simple heavy-AP first algorithm but cannot ensure an optimal solution. The proposed algorithm offers an innovative solution to the placement of cloudlets in a wireless area network. However, this method has a drawback of potentially causing congestion in certain areas with high user density. In contrast, the density-based clustering algorithm aims to mitigate this issue by clustering access points with similar user density and distributing cloudlets evenly among them. However, this method may not guarantee an optimal solution, as it prioritizes load balancing over response time.

Xu *et al.* [43] proposed a capacitated cloudlet placement algorithm in a wireless metropolitan area network (WMAN). The algorithm selects the strategic locations in the network to place K cloudlets based on the resource demands received by all user requests at a specific location. Although the algorithm aims to minimize the communication latency between mobile users and cloudlets, it is based on heuristics and does not provide an optimal solution.

Xiang *et al.* [44] presented a novel GPS-assisted adaptive cloudlet placement method for mobile applications. Their approach involves identifying cloudlet locations based on the gathering areas of mobile devices, aiming to enhance cloud service quality for dynamic context-aware mobile applications. While this method considers the mobility of users and dynamically adjusts cloudlet locations to improve service quality, it is heavily reliant on the availability and accuracy of GPS data, which may not always be reliable.

However, these approaches in this subsection did not consider the dynamicity and heterogeneity of the workload when dimensioning edge clouds. Hence, there is a need for new

approaches that can address the limitations of the existing methods and consider the dynamic and heterogeneous nature of the workload in dimensioning MEC infrastructure.

3.2.2 Resource-based Dimensioning Approaches

Previous studies have extensively explored the dimensioning of MEC systems, aiming to identify the computational and communication resources required at each edge node to handle the workload efficiently. However, these studies have some limitations that our work addresses. For instance, some studies, such as [24, 50], only consider the number of MEC nodes required at each base station location, without accounting for the heterogeneity in requests and user mobility. On the other hand, studies like [52, 53] consider workload-based resource dimensioning but fail to incorporate the latency requirements of critical services or the deployment costs.

In their research, Takeda *et al.* [24] addressed the placement of a predetermined number of edge nodes in mobile networks, each linked to a set of BSs to offload computational tasks. They identified four sub-problems related to resource sizing, each with different trade-off targets for various performance metrics, such as maximum load of a single network link, the overall workload of a single MEC node, and the maximum network traffic. However, their study did not consider the heterogeneity of requests and user mobility, which are critical factors in MEC dimensioning. In contrast, our research proposes a novel approach that accounts for these factors to address the MEC dimensioning problem, potentially enhancing MEC system performance.

Wang *et al.* [45] formulated a resource sizing problem to optimize the placement of edge nodes that can balance the workload and end-user communication latency. Their research focused on constraining MEC node locations to base station locations and used an optimization solver to resolve the NP-hard problem. However, the study did not take into account the specific needs of latency-critical services, which is a significant factor in MEC dimensioning. In contrast, our work proposes an innovative approach that accounts for the specific requirements of latency-critical services in MEC dimensioning. This approach enables more effective and efficient MEC placement decisions, ultimately improving the performance of mobile networks for latency-sensitive applications.

Li *et al*. [46]proposed a top-down methodology for resource sizing in which each BS collects workloads from its users, regardless of the type of computational resource involved.

They formulated the resource sizing problem for placing MEC nodes to minimize energy consumption, which can be attributed to two sources: (1) energy consumption that occurs even when the server is idle, and (2) energy consumption caused by the hosting user application. As a result, they suggested an algorithm to optimize resource usage and reduce energy consumption at MEC nodes without considering the requirements of latency- critical services. However, our study considers these requirements in MEC resource sizing.

Zeng *et al.* in [47] presented a system usage model that considered traffic from both base stations and user requests. The authors made an assumption that the computational effort at each base station is proportional to its network traffic, which can be determined from historical data. They aimed to minimize the number of MEC nodes deployed, while ensuring the communication latency and the number of base stations that a MEC node can serve is within acceptable limits. However, the deployment cost, which is a crucial factor in MEC dimensioning, was not considered in their study. In our research, we aim to address this limitation by proposing a novel approach that minimizes the deployment cost as the main objective of a Mixed Integer Linear Programming (MILP) problem, while still satisfying the communication latency and workload requirements.

Zhao et al. in [48] presents a novel approach for jointly dimensioning and placing virtualized services in MECs. The authors propose a mathematical model that considers the dependability requirements of the services and the available resources in the MECs. They also propose a heuristic algorithm that aims to find an optimal placement and dimensioning of the services while satisfying their dependability requirements. One limitation of this approach is that it only considers a single objective function, which is the minimization of the overall cost of the services. This may not always be the most appropriate objective, as other factors such as service latency and scalability may also be important. Additionally, the proposed algorithm may not scale well to large-scale MECs, as it relies on exhaustive search and may become computationally expensive, while our approach addresses some of these limitations. We propose a comprehensive machine-driven and scalable approach to find an optimal placement and dimensioning of VNFs in an NFV-enabled MEC infrastructure. Our approach considers multiple objectives, including latency and cost. We also propose a scalable solution that can handle large-scale MECs efficiently.

X. Ma *et al*. in [49] presents a state-of-the-art approach for resource provisioning in MEC

environments. The authors propose a cost-efficient resource allocation scheme for dynamic requests in cloud assisted MEC scenarios. Their approach considers both the delay-sensitive user requirements and the resource availability of edge servers. The proposed approach utilizes a heuristic algorithm for provisioning and scheduling the resources in the MEC infrastructure. The authors evaluate their approach using simulations and demonstrate its effectiveness in reducing the overall cost of resource allocation while meeting the QoS requirements of users. Overall, this publication provides an important contribution to the field of MEC resource management, specifically in dynamic resource provisioning. However, like any other research work in this section, it does consider an existing MEC infrastructure and focuses on managing computing resources only.

3.2.3 Latency-based Dimensioning Approaches

Authors in [50, 51] considered network delay constraints when taking the MEC sizing decision. For instance, Ta *et al.* [50] proposed a technique for server placement in MEC environments considering the network latency requirements. They developed two heuristic solutions for placing the edge. This approach considers the user density and does not take into account other important factors such as network traffic and the available computational resources.

Kasi *et al*. in [51] also focused on MEC dimensioning by minimizing the latency between MEC nodes and BSs. They proposed three genetic and local search heuristics algorithms to solve the dimensioning problem. However, their approach does not consider the scalability and mobility of users, which is an important factor in MEC environments.

Gao *et al*. in [52] presents a solution to address challenges in large-scale mixed cooperative-competitive MEC environments to support low service latency for mobile users. Specifically, it proposes a decentralized computation offloading solution that addresses the on-demand requirements of heterogeneous tasks and improves task completion rates while reducing average service latency. The publication provides a novel solution that improves upon existing methods that focus more on static resource allocation, ignoring the on-demand requirements of heterogeneous tasks. While our approach involves optimization and allocation of resources, considering the dynamic and on-demand requirements of heterogeneous tasks, thereby improving the overall performance of the MEC infrastructure.

Cao *et al.* [53] proposes a scheme for minimizing response time in mobile edge-cloud computing by exploring the placement of heterogeneous edge servers. The authors use a heuristic algorithm to optimize the placement of edge servers based on the workload and communication characteristics of mobile devices. The scheme is evaluated using simulations, and the results show that the proposed approach can effectively reduce the response time and achieve load balancing. This work focuses on minimizing response time and improving performance in edge-cloud environments. However, the challenge of efficiently managing resources and workload balancing in a dynamic and heterogeneous environment remains a critical issue. One way to address this challenge is to use machine learning techniques to optimize resource allocation and workload management. Our approach of a scalable, machine-driven, and cost-effective design and dimensioning scheme for an NFV-enabled MEC infrastructure can address this challenge by leveraging machine learning techniques to optimize resource allocation and workload management while ensuring scalability and cost-effectiveness.

3.2.4 Mobility-based Dimensioning Approaches

The investigation conducted in [54] postulated a presumption of user mobility as a crucial factor responsible for the migration of workload between multiple MEC nodes. The researchers computed the feasible number of virtual machines that can be moved from a given edge node to another, while also accounting for the adequate computing resources to satisfy the resource requisites for service migration. Regrettably, the authors overlooked the implications of fluctuating user demands and network traffic on the MEC network, a parameter that holds considerable relevance in the context of dynamic network environments.

3.2.5 Reliability-based Dimensioning Approaches

In their inquiry presented in [30], the researchers directed their attention towards the reliability dimension of the MEC dimensioning problem, specifically examining the issue of apportioning a predetermined number of MEC nodes to cater to users within a given locality. To determine the number of MEC nodes that can be assigned to each user, the authors suggested the adoption of the reset power metric. The resource allocation conundrum was cast as a multipurpose integer programming (IP) problem, a computationally complex task classified as NP-hard. To mitigate this challenge, the authors devised an approximation algorithm to resolve

the issue at hand. However, a shortcoming of this method is its tacit assumption of a fixed number of MEC nodes allocated to serve users, which may not be the most favorable solution for dynamically changing user demand. Additionally, the reset power metric employed may inadequately capture the actual resource needs of users, resulting in suboptimal resource allocation.

Selvi *et al.* [55] proposes a dynamic resource allocation scheme for a 5G network that incorporates both Software-Defined Networking (SDN) and Edge Computing. The authors aim to improve the reliability of the system by reducing network congestion and packet loss. The proposed scheme allocates resources dynamically by considering the workload and available resources in real-time. The authors used simulations to evaluate the performance of the proposed scheme and compared it with other resource allocation schemes. The results show that the proposed scheme improves system reliability and reduces packet loss. However, the study did not consider the impact of mobility on resource allocation and did not evaluate the scheme in a real-world deployment.

3.2.6 Issues in the Existing Dimensioning Approaches

The current MEC design and dimensioning methods lack a proactive and auto-scaled approach in determining the necessary MEC nodes and resource allocation for each node to handle user requests. These approaches assume that the workload of the MEC system remains fixed, which is not an accurate representation of a real-world environment. The use of static dimensioning methods may lead to ineffective resource utilization over time and a higher rate of blocked requests when resources are insufficient to process user requests at specific times.

3.2.7 Conclusion: Our Proposed Solution

This research work introduces a novel and comprehensive methodology for the design and dimensioning of MEC systems aimed at edge deployment in scenarios where no MEC infrastructure currently exists within the mobile operator's network. The proposed approach emphasizes the importance of service placement and the optimization of communication and computational resources necessary to dynamically process user requests. Unlike existing approaches that rely on a finite set of fixed resource dimensions for edge servers, this methodology recommends defining each edge server's dimensions as a continuous variable,

subject to resource capacity constraints. To accomplish this, NFV technology is employed to virtualize the MEC infrastructure, allowing for the seamless deployment and execution of VNFs on a virtualized platform, such as a virtual machine or container.

The proposed approach outlined in this thesis offers a novel solution to the challenges of VNF placement and resource allocation in MEC systems. By incorporating traffic prediction and threshold-based methods, this proactive auto-scaling approach enables the efficient placement of VNFs and the automatic allocation and release of resources to meet time-varying traffic and service latency requirements. An important benefit of this approach is its flexibility in resource allocation, which enables the system to dynamically allocate resources based on real-time needs, improving overall performance while reducing resource wastage. Moreover, this approach offers a comprehensive solution to the management of MEC systems by considering the entire system and the actual amount of computational and communicational resources required for user requests. The proposed approach leverages ML models for the prediction of SFCs and supports proactive auto-scaling of VNFs, which is essential for efficient MEC system management. While the impact of this approach on ML tasks may vary, the continuous definition of edge server dimensions provided by this approach could enable more efficient resource allocation for ML tasks and potentially reduce the overall cost and time required for ML training and inference.

In this chapter, we present a summary of various works on VNF auto-scaling and placement mechanisms that can be utilized in the design and dimensioning of MEC systems. Our objective is to identify the most appropriate approach for our research by thoroughly examining and reviewing the existing literature.

3.3 Service Placement in MEC- Overview

Service placement in the MEC system refers to the allocation of virtualized services on available MEC nodes. Each VNF represents a service instance that processes the end user's request. The service placement process considers the available computational and communication resources, depending on the service type and use case. VNF placement algorithms can categorize services into three types: (1) latency-sensitive services, (2) services for mobile users, and (3) service chains. The optimization of the VNF placement problem aims to reduce operation costs, resource utilization, service latency, and energy consumption. This section presents the policies of the used placement algorithms and the services available for

VNFs, which can be used in the MEC dimensioning process.

3.3.1 VNF Placement Policies

Several research proposals have been presented to address the issue of VNF placement in the MEC system. Some of these proposals primarily focus on fulfilling the computational resource requirements for placing VNFs [57-62], while others emphasize allocating communication resources, either in the wireless [56] or wired networks [57], to offer high data rate services in the MEC system.

3.3.1.1 VNF Placement Based on Computational Resources

In [58], the authors formulated a service placement problem to maximize the rewards for the service to the users. The authors showed that the proposed service placement problem is NP-hard and then presented an approximation algorithm using the set cover problem. In [59], the service placement problem was investigated to reduce the total cost of computing and communication resources. The authors proposed a heuristic based on a genetic algorithm. Since MECs are closely linked to cellular networks, it is interesting to investigate whether the combined consideration of service placement and RAN design can facilitate the integration of cellular networks and MEC systems. This problem was examined in [60] to compute a Pareto-optimal solution for network cost and service delay. The problem in question is NP-hard, and the authors proposed an algorithm based on Bender's decomposition to calculate approximate solutions. The algorithm breaks down the general problem into a master problem for computing service placement and a sub-problem for computing RAN configuration. The solutions for the master problem and the sub-problem have a lower and upper bound of the resulting solution, and the algorithm stops when the upper and lower bounds match. The results showed that the general approach could significantly reduce the overall cost compared to a non-optimized cloud RAN.

The authors of [61] suggested a decentralized service placement approach in which location decisions are made jointly by end-users and MEC service providers. MEC service placement is done in two stages. In the first stage, each user is assigned an algorithm to determine whether to perform tasks on a local server or MEC nodes, while in the second stage, the MEC nodes required to process the user service are computed. Then, based on the calculated workload and costs, the login algorithm determines the order of services for the service provider.

Joslio et al. in [62] proposed a decentralized workload of the algorithm with a bounded approximate relationship. They assumed that the end-users share communications and computing resources in the MEC system. In this approach, each user decides which MEC node will load their work to reduce the weighted amount of their power consumption and response time. In [63], the authors considered the same MEC system with users generating functions periodically. They proposed a decentralized algorithm to gradually allow new users to make task-loading and assignment decisions.

3.3.1.2 VNF Placement Based on Communication Resources

Within this context, we shall delve into the allocation of communication resources in both wireless and wired networks. The allocation of communication resources in wireless networks ensures that end-users receive achievable data rates tailored to their specific requirements. For example, the authors in [62] considered the possibility of hosting a group of services in the MEC to meet the mobile users' requirements, considering MEC nodes' computational and communication capabilities to reduce traffic congestion. In particular, the throughput of the MEC node is viewed as the number of available resource blocks (RBs) which meet the individual data rate requirements. Two sources are required to host a particular service: the first part is the power consumption of the main activities of the service.

In contrast, the latter aspect is directly correlated with the quantity of mobile users served. The scholarly article introduced a greedy algorithm for a solitary MEC node, in which each node chooses the services to be hosted based on the optimization of resource efficiency and utilization. Furthermore, it presented a decentralized strategy that relies on mapping and mutual preference between MEC nodes and services. The findings indicated that a cooperative approach to service and radio resource deployment can alleviate traffic congestion and substantially enhance computing resource efficiency.

Authors in [56] investigated the allocation of wireless communication resources for the offloading requests in MEC. They modeled the interaction between BS and end-users as a Stackelberg game. They proposed decentralized approximation algorithms concerning different operator resource allocation policies to minimize the completion time of task offloading.

When allocating resources in wired networks like mobile backhaul, joint consideration of service placement and communication resource allocation was proposed [57]. This approach

facilitates meeting end-user demand and places VNF in MEC. The authors formulated a multi-criteria problem to reduce the implementation cost and balance the workload on network links, subject to the latency requirement of individual services and available computing resources. The authors expressed the resulting problem as a MILP and proposed an effective heuristic algorithm.

In this section, we provide an overview of various VNF service placement approaches based on the computation and communication capabilities of edge servers. It is important to note that the studies discussed in this section presuppose the existence of the edge network infrastructure and the limitations of edge node capabilities when it comes to VNF placement. The following section describes the VNF placement methods for different services, such as those related to latency-critical services, mobility services, and service chains.

3.3.2 VNF Services

In the context of the MEC system, VNF placement can be leveraged to facilitate the provisioning of services that are sensitive to latency or user mobility, or that require the processing of a service chain involving multiple VNFs. Such placement strategies are pivotal in ensuring that the MEC system can meet the diverse service requirements demanded by end-users. as follows:

3.3.2.1 Latency-Critical Services

In MEC, service delay is impacted by the end-users' locations and the MEC nodes hosting each service. Many studies [64 - 66] have been conducted to solve the VNF placement problem with a target to minimize the latency for critical services. The approach in [64] solved the VNF placement problem to support latency-critical services by prioritizing the overall delay performance measurements, which minimizes delays as the primary goal of the VNF placement. In this approach, the service latency can be expressed as the summation of MEC nodes' communication and processing latency. For example, the authors in [65] intended to place multiple VNF instances to serve a set of users, considering the MEC node capacity. Authors in [66] modeled a MEC system with access to central clouds. The authors suggest allocating users' requests that need intensive resources on the cloud DC instead of the MEC system is better.

The E2E delay should include significant cloud delays in such a case. The authors formulated the NP-hard placement problem to reduce the average latency of service. The authors

proposed a brute-force algorithm that lists all possible placements to calculate the optimal solution. In these two approaches, the latency performance of the user services may not be guaranteed as their main objective is to minimize the overall latency of the edge computing system only without any threshold. This issue could be solved by taking the latency threshold of the user service as a constraint to solve the VNF placement problem. For instance, authors in [67] followed this approach to increase the number of hosted services in the edge computing system. The authors first proposed an algorithm to calculate the best solution without limiting the capacity of MEC nodes. In the case of qualified MEC nodes, the authors proved that the problem is NP-hard and followed the approximation approach to solve it. In [68], the authors observed the same approach while considering deploying a service to balance the workload between MEC nodes, subject to MEC node capacity limitations. The problem is proven to be NP-hard. The authors have suggested a solution using the tabu search, starting with the earliest possible solution and gradually improving the proposed solution by changing service events. The algorithm ends when the maximum number of rounds is reached.

3.3.2.2 Mobility Services

Due to the mobility of MEC users, adjusting VNF placements based on users' real-time locations can enhance the user experience [68]. Different studies [69-71] have used a common approach to solve the VNF placement problem for mobile users. This approach divides the time into successive time slots and assumes that the connection between mobile users and BS within each slot is stable and the user moves within BS coverage. Therefore, service placements for each location slot can be calculated, and services can be switched between different time slots to follow the user. In [69], the authors formulated the service placement problem with user mobility to minimize the average response time of the system. The resulting problem is an integer nonlinear programming problem, and the authors proved that the problem is NP-hard. The first step uses a genetic algorithm to determine whether service migration is necessary based on system delays and latency caused by service transfers. If the service is decided to be migrated, then a sub-problem, NP-hard, is used to calculate the new placement and is solved by optimization solvers. However, this work is unsuitable for latency-critical services, as it does not consider the user request’s latency requirements. In [70], the authors considered hosting virtual reality (VR) games for dynamic user groups to reduce communication and computing costs and

communication latency between users within each group. They considered the general pattern of user mobility in a time slot and proposed the predictive control (MPC) method to estimate the number and locations of users for each time slot and then place the services according to the prediction, which is proven to be NP-hard. This placement problem could be solved using an efficient algorithm that defines an approximation limit for the service placement for each time slot.

Another method of capturing user mobility is to model the cost-of-service migration to track user movement. Migration costs are proportional to the amount of network traffic and computational resources. This method was used in the time slot system [71], where the authors assumed the general idea that migration costs are proportional to the user's workload and inversely proportional to the distance between users and MEC nodes. The authors formulated a joint problem to enhance QoE while reducing service migration costs. Other approaches considered a system with infinite time intervals, such as in [53], where an online approach was suggested to solve the service placement problem to meet the demand for end-user by reducing the weight of computational latency, communication delays, and transfer delays. The authors first developed a dynamic service placement problem as a contextual multi-armed bandit problem. They then suggested using Thompson sampling to estimate users' expected performance for various location decisions. The authors considered communication delays as system parameters in this model and did not include bandwidth allocation.

3.3.2.3 Service Chain

A service chain in MEC means that the end-user request must be processed by multiple services that depend on each other. Inter-service dependency adds a new dimension to managing service chain placement. For instance, the end-to-end latency of a user request in the service chain should include communication latency between the dependent services and traffic flow. Some constraints can be relaxed to make it easier to develop solutions to address the complexities of service chain placement. For example, authors in [72] investigated service chain placement problems regardless of the MEC node capability constraint. They proposed a heuristic based on a local search and Hungarian algorithm to reduce the cost of computing and communication resources.

Additionally, the authors in [73] worked on the service chain placement in MEC to

balance the workload of nodes. The authors modeled each service chain as a graph, where each vertex and edge in this graph corresponds to the service or communication path. The resulting deployment problem is NP-hard, and the work suggested two solutions. The first solution focused on the best placement of a single service chain, and then the second solution took an online approach to place multiple service chains, each of which is a tree in the diagram.

Service chain placement can also be performed under the capability of MEC nodes, including available CPU cycles, memory size, and hard drive space. Works in [74, 75] modeled the capacity of MEC nodes in the service placement problem. For example, the authors in [74] proposed service chain placement to increase resource utilization efficiency. They suggested a polynomial solution based on graphs and the Hungarian algorithm. Furthermore, the authors of [75] studied the problem of placing a service chain to reduce deployment costs. They proposed a heuristic solution based on the genetic algorithm and suggested two strategies for applying it. One is to calculate the assignment of the service chain sequentially, and the other is to calculate the allocation of the service chain together. Simulation results showed that the latter strategy reduces the total cost and requires more execution time.

Other works solved the service chain placement for user requests with high traffic demand, where network traffic transfer paths should be calculated under network capacity, traffic load, and dependence between services limitations. This problem was studied in [76], where the authors considered the combined problem of service chain placement and flow distribution to optimize proper flow and energy consumption jointly. The resulting problem was expressed as an NP-hard model and solved using an approximate algorithm based on linear relaxation and estimation techniques. The proposed solution first calculates the placement of services and then calculates the optimal flow distribution. Statistical results showed that a combined computing and communication resources allocation is necessary to optimize efficiency for systems and services.

3.3.3 Conclusion

In this section, we present various VNF placement techniques based on the placement policy or service type. Our research employs the service chain placement approach to deploy SFC requests received from mobile users on edge servers, as it is the most convenient way to represent such requests. The placement of service chains involves mapping virtual functions and

links to the physical embedded network. It is a crucial step in MEC system dimensioning, as it determines the appropriate level of resources to be allocated to individual VNFs and links in the SFC request, taking into account the limitations of edge node capacity, bandwidth requirements, service latency requirements, and time-varying traffic. In the subsequent section, we provide an overview of various auto-scaling methods that could serve as a vital step in designing an efficient MEC system.

3.4 MEC Auto-scaling Approaches

In the previous section, we discussed the complexity that VNF placement and orchestration mechanisms pose for the operator. Mobile traffic variation, which is influenced by user distributions and mobility models, could cause uneven traffic distribution within the MEC system, leading to a frequent fluctuation in the number of required VNF instances placed at the edge server. To tackle these challenges, operators must adopt a proactive auto-scaling approach in a distributed MEC-enabled NFV infrastructure. This approach dynamically places a set of VNFs and automatically allocates/release resources to a VNF and adds/removes one or more VNF instances. The proactive auto-scaling approach combines traffic prediction and threshold-based methods to produce scaling decisions ahead of time.

The three-tier edge architecture presented in Figure 2.8 has exhibited MEC's advantages, including local and global service reliability and availability. Nevertheless, the MEC framework faces three significant trends, such as the rapid growth of end-user data, latency-critical needs arising from modern computing applications like augmented reality applications [36], and dynamic traffic patterns in 5G mobile networks. To address these challenges, numerous studies have been conducted to discover an intelligent edge between mobile devices and centralized cloud architecture by leveraging the NFV technique to virtualize the underlying MEC infrastructure. In the MEC-NFV architecture framework, MEC network functions are separated from their underlying hardware, deployed as virtual network functions, and run on VMs or containers within the MEC edge server. To support the varying workload dynamics, while considering the edge node capacity limitations and service latency-critical requirements, mobile network operators should deploy and manage a VNF auto-scaling mechanism at their edge nodes. Prior research on VNF auto-scaling can be classified into two groups: proactive and reactive approaches. The subsequent subsections describe each approach and present its current

research.

3.4.1 Reactive Autoscaling Approach

In the reactive auto-scaling approach, the threshold levels are statically pre-defined and are easily implemented. The thresholds are hard to choose to handle the dynamic traffic in 5G networks. In [77-79], the authors proposed VNF auto-scaling mechanisms based on static thresholds (lower and upper scale thresholds). Auto-scaling is triggered if the traffic load falls below or exceeds these thresholds. For example, the proposed VNF auto-scaling technique in [77] aims to efficiently utilize cloud resources and improve QoE cloud services. Carella et al. in [78] presented an Autoscaling Engine (AE) mechanism that dynamically adapts the network services provided by Telecom operators and increases the reliability, stability, and resource utilization of these services. Authors in [79] proposed threshold-based auto-scaling for VMs to dynamically scale the virtual instances based on the application's computing resources.

Similarly, in [80], Hung et al. introduced an auto-scaling algorithm to enable the automated provisioning and balancing of VM resources depending on the active session of the end-user application. The techniques mentioned above could lead to an oscillation in the scaling decision and, thus, unstable behavior that may affect overall system performance. Alternatively, [81] and [82] introduced queueing theory and reinforcement learning mechanisms into threshold-based auto-scaling techniques to improve the system's performance. Still, it stays reactive approaches with the same weaknesses.

In this subsection, we enumerate several auto-scaling studies that utilize the reactive approach. The use of pre-determined and fixed thresholds in the reactive approach makes it less suitable for the MEC dimensioning process, given the dynamic nature of traffic and the non-uniform distribution of mobile requests. In the subsequent subsection, we elaborate on works concerning the proactive auto-scaling approach, which is characterized by greater dynamism and agility.

3.4.2 Proactive Autoscaling Approach

In proactive scaling approaches, predictive methods enable systems to automatically learn and anticipate future demands to make auto-scaling decisions. Machine learning is one of the most widely used predictive techniques for workload forecasting in auto-scaling. By

introducing intelligent network operations into mobile edge architectures, ML enables the network system to achieve its goals automatically without human intervention. The system can respond and adapt to environmental changes and gain knowledge from its operational history based on the collected statistical data. The integration of ML intelligence minimizes costs and errors, enhances system efficiency, and simplifies resource management in scalable networks.

ML is a cutting-edge prediction technique that can enable MEC to surpass its current capabilities by building more sophisticated systems. By incorporating ML, MEC can offer more extensive computing resources and enhance its ability to provide high-quality service. The application of ML in MEC can lead to a wide range of benefits such as facilitating the use of mobile and satellite servers, fostering better cooperation between servers in disparate networks, and enabling network status prediction. Additionally, [83] has identified various ML-based solutions that can effectively address the computation and communication challenges that arise in MEC environments.

Machine Learning plays a dual role in the context of edge computing which was presented in [84]:

ML introduces new capabilities to optimize MEC infrastructure: Distributed control and system orchestration, predictive network infrastructure maintenance, initiation of MEC resources, and proactive application offloading decision depending on the workload on each MEC node or end-user mobility across the MEC system.

Training ML: distributed edge platforms promptly provide massive amounts of local data to ML models. ML gets input from the underlying edge platform to enhance operational efficiency. This topic opens novel research directions to explore ML optimization approaches based on the workload information received from the edge system, including predicting the changes in the workload at the edge nodes to orchestrate the computational resources and task offloading in runtime. The most effective ML-based optimization approaches for MEC infrastructure are listed below.

In the context of proactive VNF auto-scaling in a distributed NFV-based edge infrastructure, authors in [82] proposed a dynamic mechanism like reinforcement learning to enhance NFV scaling policies based on dynamic thresholds. As it outperforms static approaches, it is still a reactive solution with the same flaws. For instance, Arteaga et al. [85] proposed an

adaptive scaling mechanism for NFV based on Q-Learning and Gaussian Processes. An agent uses it to manage performance variations better and implement an effective scaling policy strategy. This mechanism was validated through simulations in a virtualized Evolved Packet Core (EPC) case study, proving that it is more accurate than approximations.

On the other hand, [86] proposed a time series model to predict resource usage based on a historical dataset. Other authors, including Mijumbi et al. [87] and Mestres et al. [88], used ML models to handle the fluctuation in managing the amount of the virtualized resources assigned to specific VNF instances by anticipating resource requirements and thus improving the resource allocation algorithm efficiency. For example, a neural network-based model is used to predict the future resource requirements for each VNF instance based on the VNF forwarding graph topology information (VNFC) proposed in [87]. Each VNFC's topology information is derived by combining its previous resource utilization and the modeled effect on its neighborhood. The proposed approach was evaluated for real VoIP traffic traces, and the results showed a reduction in the dropped calls rate and an improvement in the call setup latency.

This section provides an enumeration of the scholarly works related to the agile autoscaling approach utilizing the ML algorithm. The ML-based autoscaling approach is well-suited to handle the heterogeneity and dynamicity in the mobile traffic in MEC networks, providing accurate auto-scaling decisions with low latency and high acceptance rates. Therefore, the proactive autoscaling approach using the ML algorithm is an essential component of our research work in the MEC dimensioning process. The ensuing section offers a synopsis of the forecasting techniques that could be applied to predict the time-series traffic.

3.4.3 Forecasting Techniques for Time-series Traffic

Nowadays, it is imperative to develop an intelligent planning tool for re-dimensioning mobile network capacities to mitigate blocking of user requests and degradation of quality of service, particularly in the face of rapid traffic growth and heterogeneous service demands. To ensure user satisfaction, mobile operators need to predict future mobile traffic demand and variations accurately and upgrade their network capacities and configurations accordingly.

Traffic forecasting in telecom networks is a time-series forecasting analysis problem that involves estimating how the observations in regularly observed time-series (e.g., hourly, daily, monthly, or annually) will continue over time. Selecting the most appropriate forecasting

methodology is crucial for minimizing the time, effort, and costs involved in the planning and re-dimensioning process.

Time-series forecasting techniques in the telecom domain can be classified into traditional and ML forecasting:

Traditional forecasting techniques perform well in cases where the traffic data have a finite and countable number of predictive variables. The conventional algorithms use predefined techniques and statistical models such as linear regression and Autoregressive Integrated Moving Average (ARIMA). The goal of traditional forecasting methods is mainly descriptive, aiming to analyze a univariate or multivariate dataset with finite, countable, and explainable predictors. ARIMA technique is used in many works to predict circuit-switched traffic with short-range dependencies for telecom operators at different time intervals (i.e., minutes, hourly, monthly) [95-97]. As the telecom networks evolved, the main traffic switches at operators were changed to use the packet switching technique instead. In packet switching, the mobile traffic becomes more dynamic and impacted by many abnormal events and network configuration changes than circuit-switched traffic. The ML approach predicts dynamic packet-switching traffic data with many variants as an alternative.

ML forecasting techniques use more complex features and predictive methods such as Supervised Learning (SL) to improve the accuracy of forecasts while minimizing the loss function for cases where we have a massive amount of data. Figure 3.1 shows the ML forecasting process followed by many research works.

Short- *and* long-term forecasting is needed for planning and dimensioning purposes in mobile operators. Short-term forecasting is essential to develop a proactive self-tuning resource management technique. Several works [89 - 90] proposed the ML model to forecast time-series data in the short term (seconds, minutes) using deep learning to solve the limitations of time-series analysis approaches to capture rapid fluctuations. The temporal variations in the traffic can be captured using recurrent neural networks based on Long Short-Term Memory (LSTM) units.

A neural network is a network of neurons that are organized in layers. The predictors (or inputs) form the bottom layer, and the forecasts (or outputs) include the top layer. There may also be intermediate layers containing “hidden neurons.” Figure 3.2 shows the neural network architecture known as a multilayer feed-forward network, where each layer of nodes receives

inputs from the previous layers.

The outputs of the nodes in one layer are inputs to the next layer. The inputs to each node are combined using a weighted linear combination. A nonlinear function then modifies the result before being output. The network is trained several times using different random starting points, and the results are averaged. For instance, in [90], the time-varying traffic data is divided into a regular grid, and a convolutional neural network is used to model spatial traffic dependencies among grid points in the short-term (i.e., minutes or hours). Other works use deep learning schemes to coarser time resolutions (i.e., an hour) to extend the forecasting horizon to several days [91].

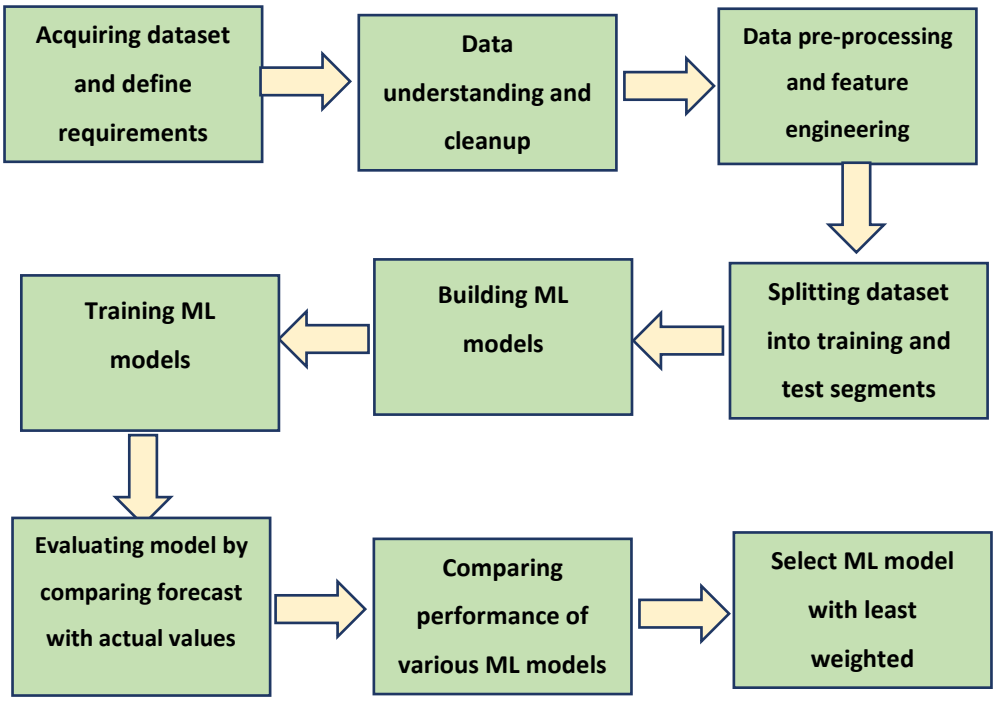


Figure 3.1 ML Forecasting Process

Long-term forecasting in the planning process is also required to predict the re-dimensioning actions, like deploying a new tower or server at a specific location. Thus, mobile traffic must be expected with much longer time horizons (i.e., months or annually). For this purpose, we should consider only the busy-hour measurements monthly to limit the number of historical data samples used for prediction. It has been proved in many studies that the short-time series used in long-term traffic forecasting are different from those that are used in short-term

traffic forecasting. In the long-term prediction, the impact of past measurements quickly diminishes after a few days or weeks due to the change in users' requirements and locations. However, it has been proven that supervised ML models, including LSTM, still outperform the traditional time series approach with these constraints [92] for long-term forecasting. We used the ML approach to forecast the traffic and predict the required number of SFCs deployed in this work.

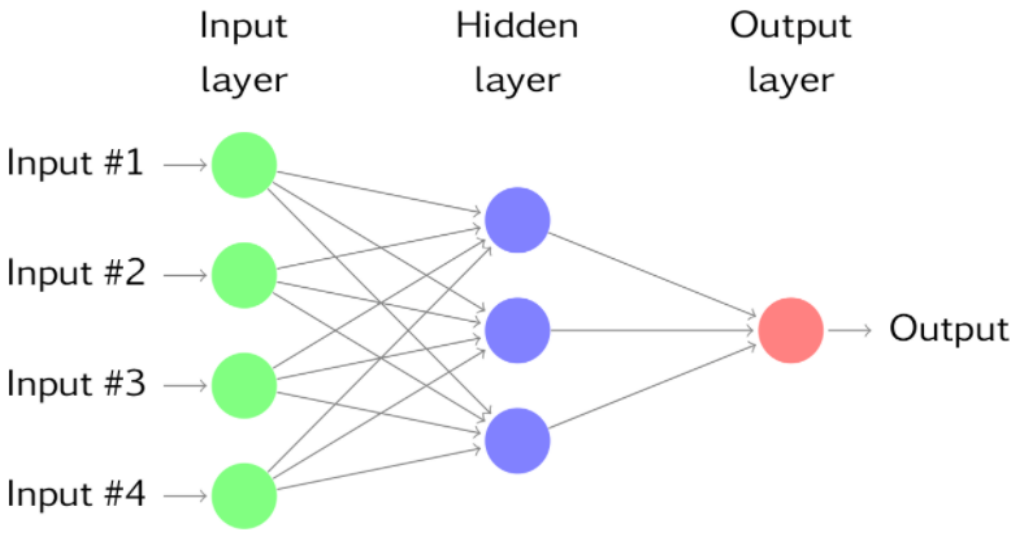


Figure 3.2 Neural Network Architecture

3.4.4 Conclusion

This section, we present a comprehensive review of the MEC auto-scaling approach, coupled with various forecasting techniques employed to predict time-series data. Our research employs a proactive auto-scaling approach using ML algorithm as a crucial component in the MEC design process. This approach is geared towards overcoming issues such as non-uniform user distribution, dynamicity, and heterogeneity in received SFC requests. The primary objective of utilizing ML in this context is to achieve long-term forecasting of MEC infrastructure design and dimensioning in mobile operators. The subsequent section delves into different optimization approaches that can be implemented to establish a highly efficient MEC framework with optimal performance.

3.5 MEC Optimization Approaches

ETSI's classification of MEC use cases comprises consumer-oriented, operator-oriented, and network performance-oriented services. These use cases need to be supported by MEC systems to facilitate a diverse range of new computing applications and services at edge nodes. The classification of MEC use cases is primarily dependent on the entities that can reap the benefits of the application advantages. For instance, the consumer-oriented services use case aims to directly provide benefits to end-users by enabling the execution of computation-intensive and latency-critical applications at edge nodes through computation offloading techniques, which allow mobile devices to leverage vast computational resources at edge nodes. The operator-oriented services use case provides benefits to mobile operators and third-party vendors by enabling them to allocate their applications and services utilizing the computational and storage resources at edge nodes, including extensive data analysis, active device location tracking, security, safety, and data analytics. Finally, the network performance-oriented services use case focuses on enhancing network operations, improving network performance, and QoE through local content caching at edge servers, such as optimizing video delivery for TCP.

MEC framework aims to provide high QoS to its users, particularly for latency-sensitive applications. Efficient utilization of resources and infrastructure is crucial to achieve the targeted performance requirements. Hence, various QoS parameters should be taken into consideration while designing optimization approaches. The selection of QoS parameters to be optimized depends on the stakeholder, whether it is the service provider, infrastructure provider, or end-user. Therefore, in this section, we will analyze current optimization approaches in the MEC system based on these perspectives and the QoS parameters being optimized.

QoS parameters examined in this section are Latency, Computing Resources (including processing, memory, and Bandwidth), Energy, and Combined Resources. Most of the works in this section used the VNF placement algorithm to formulate the optimization problem and then solve it to meet the required QoS. Table 3.1 shows a summary of the MEC framework optimization approaches.

The MILP approach was retained in this research work to find the optimal dimensioning cost to implement an agile NFV-enabled MEC system to support heterogenous SFC requests with different QoS requirements with high acceptance rates. More details about our MILP

approach are presented in Chapter 4.

3.6 Conclusion

The present chapter provides a comprehensive account of the entire implementation process for MEC, with the assumption of no pre-existing infrastructure. This process encompasses a critical examination and comparative analysis of extant works on MEC infrastructure design and dimensioning approaches, along with a delineation of the pertinent issues associated with them. Additionally, the chapter provides an overview of the latest works in VNF placement and autoscaling techniques and underscores the value they add as crucial stages in the MEC dimensioning process, in offering an efficient and agile solution to the main dimensioning challenge at hand. Ultimately, the chapter delves into the optimization of MEC infrastructure and resources, with an emphasis on performance capabilities.

Table 3.1 Different Performance Optimization Approaches

Ref.	Target Component	Optimization Objective
[93]	Infrastructure Provider	Minimize MEC environment power consumption
[94]	Infrastructure Provider	Minimize overall system energy consumption
[95]	Infrastructure Provider	Use Q-learning mechanism to reduce MEC system energy consumption
[96]	Infrastructure Provider	Reduce communication delay in MEC clusters
[97]	Infrastructure Provider	Minimize energy consumption and task execution time on MEC nodes
[98]	Infrastructure Provider	Enhance computing, storage, and bandwidth resources at the edge nodes
[8]	Infrastructure Provider	Minimize hardware deployment costs while maintaining the required communication delay
[99]	Service Provider	Reduce services' latency and increase their availability
[100]	Service Provider	Minimize end-to-end communication delay and reduce service deployment cost

[101]	Service Provider	Reduce end-to-end service time in a MEC network
[102]	Service Provider	Minimize service operating costs and provide high-availability services
[103]	Service Provider	Minimize service energy consumption
[104]	Service Provider	Reduce the cost of service migration and provide high-availability services
[105]	Service-Provider	Use ML to utilize computing resources efficiently
[106]	End-user	Minimize latency and support real-time and critical-latency applications
[107]	End-user	Achieve service delay required by users in a MEC-based 5G network while minimizing inter-MEC handover costs
[6]	End-user	Provide high-availability services to the end-users
[108]	End-user	Provide low-cost services to the end-users in a multi-tenancy architecture
[40]	End-user	Minimize energy consumption on the end device and reduce the task computing execution times
[109]	End-user	Improve energy efficiency and QoS for edge end-users
[110]	End-user	Support heavy computation tasks received from end-users while minimizing the energy
[9]	End-user	Provide tasks with low latency to the end-users

Chapter 4

NFV- BASED MEC AGILE FRAMEWORK FOR HETEROGENEOUS COMPUTING APPLICATIONS

4.1 Introduction

In this chapter, we present a detailed implementation process that is necessary for deploying a comprehensive, agile, and scalable NFV-based MEC infrastructure at a mobile operator. This process aims to ensure that the service provider can effectively handle and process the diverse requests from mobile users with high acceptance rates, real-time response, guaranteed quality of service, and minimal costs. To achieve this objective, we have identified three distinct stages for implementing an NFV-enabled MEC infrastructure.

First, assuming the absence of pre-existing MEC infrastructure adjacent to the BS at the MNO, the construction of an MEC infrastructure is reliant on maximum workload predictions for mobile traffic during a specific period, as received from end-users, such as mobile users, connected vehicles, or IoT devices, at each BS location. In this study, we use the maximum predicted values over a six-month period. The influx of mobile traffic in specific BS locations can lead to network congestion on the cloud, which can have an adverse impact on network performance. The deployment of localized MEC infrastructure in high-traffic BS areas can provide benefits to conventional cloud providers, mobile core operators, and Internet service providers. However, MNOs face significant obstacles when it comes to the cost of deploying edge servers, given the widespread and substantial number of mobile devices expected to be served with low latency [111].

Second, the MEC infrastructure serves as the intermediary between mobile end-users and computational resources, necessitating its efficient resource management. Virtualizing the MEC infrastructure, or NFV, presents a more viable option for operators to achieve this objective. In the NFV environment, MEC network functions are implemented as virtualized functions, and

user requests are deployed as SFCs that comprise a sequence of VNFs hosted on the edge servers and interconnected in a particular order. Predicting the number of SFCs, the number of VNF instances required to be hosted, and the computing resources necessary for an SFC to support time-varying traffic can be arduous, and it directly impacts the achieved response time [65].

Third, MEC infrastructure should be optimized to cater the performance requirements of mission critical MEC applications. The infrastructure must be utilized efficiently to accommodate numerous mobile devices that simultaneously rely on the computational capabilities of edge servers. Consequently, an optimized, efficient, and dynamic workload allocation to hosted VNF instances is essential.

This research aims to tackle the intricacies associated with MEC-NFV Infrastructure design and dimensioning, which is no easy feat. Specifically, we strive to provide an effective, cost-efficient, and scalable solution for the heterogeneous SFCs main problem, which can be broken down into three sub-problems.

1) *The MEC dimensioning sub-problem* decides the number and placement of MEC edge servers.
2) *The SFC autoscaling sub-problem* aims to proactively determine the number of SFCs required to be deployed on the edge servers within the MEC infrastructure to process different users' request types at a specific time. The number of the predicted SFCs represents the output of the ML model at a specific time.
3) *The SFC mapping sub-problem* aims to map received SFC requests into a set of VNFs and virtual links within the MEC system to satisfy different latency requests' requirements with a high acceptance rate.

In this study, we present a formulation of the three sub-problems as a one-shot Mixed Integer Linear Program (MILP) aiming to minimize the overall cost of MEC server deployment while ensuring that the latency and acceptance rates are within guaranteed service provider thresholds. By testing the MILP formulation under different parameters, we demonstrate that the proposed approach is efficient, scalable, and yields comparable results to other existing benchmarks.

It is worth nothing that the number and distribution of edge nodes in the network are considered, as well as their processing capabilities and resource availability when deploying

MEC infrastructure. In our simulation experiments, we have varied the number of edge nodes to evaluate the performance of our proposed design and dimensioning scheme under different network densities.

Furthermore, we have also considered the impact of varying the density of user devices in the network, as this can have a significant effect on the workload and resource requirements of the MEC infrastructure. Our simulation experiments have included scenarios with varying numbers of user requests to evaluate the scalability and adaptability of our proposed approach under different levels of network density. Overall, we have taken a comprehensive approach to considering the density of the MEC network in our simulation tests and have ensured that our proposed approach is effective and efficient under a wide range of network densities and user scenarios.

4.2 Agile MEC-NFV Architecture Framework

This section lists the main components required to deploy an agile comprehensive MEC-NFV infrastructure at the operator site. We rely on the ETSI MEC reference architecture in an NFV environment model Figure 4.1 to design the MEC-NFV architectural framework. In this model, ETSI proposed the following changes in the MEC architecture [31]: (i) MEC applications are deployed as VNFs and are managed and orchestrated by the NFV components(i.e., Virtual Infrastructure Manager “VIM,” Virtual Network Function Manager “VNFM,” and Network Function Virtualization Orchestrator “NFVO”); (ii) VNFs are managed using VNFM on the MEC platform; (iii) the VIM manages the virtualized computing and communication resources of MEC infrastructure; (iv) MEC platform manager is responsible for assigning the VNFs lifecycle to the VNFM; (v) MEC application Orchestrator is linked to the NFVO for resources and services orchestration.

In our proposed framework, we have streamlined the ETSI MEC-NFV reference architecture presented in Figure 4.2 by identifying and selecting the essential blocks and reference points necessary for deploying a proactive system capable of handling dynamic, heterogeneous, and heavy computational traffic generated by mobile end-users in various locations. The system must support real-time, low-latency applications and services with assured QoS at a minimal cost.

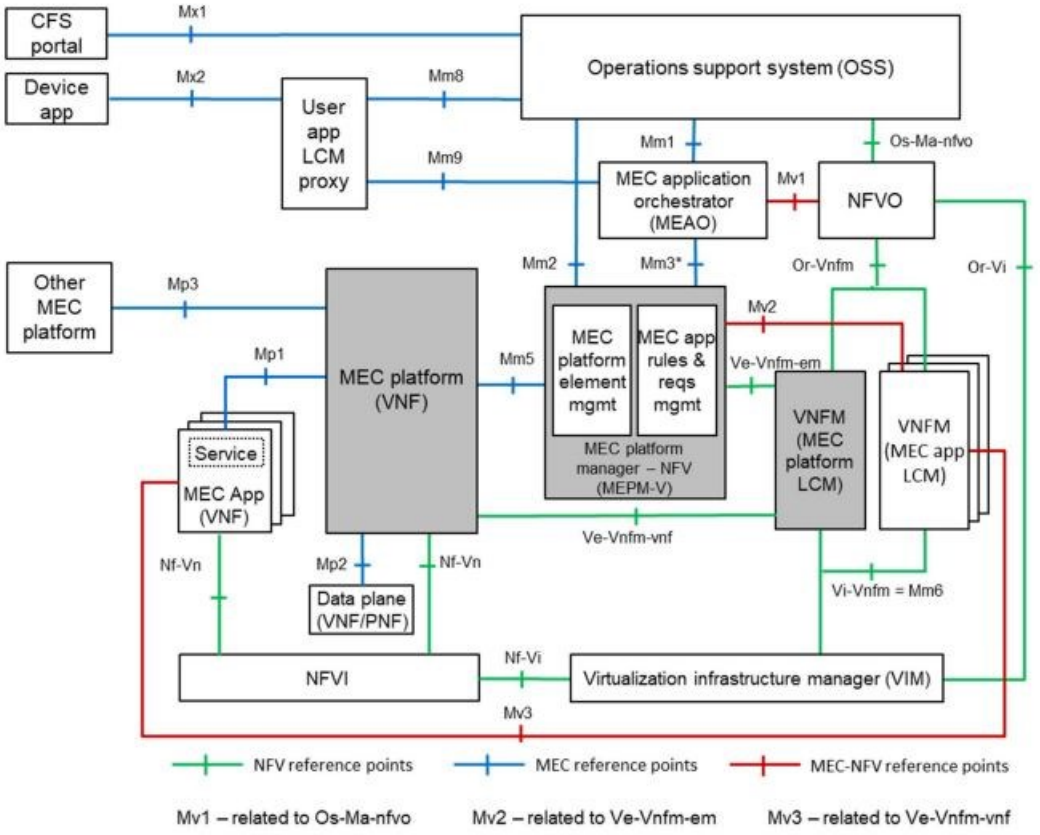


Figure 4.1 ETSI MEC-NFV Reference Architecture [31]

This framework assumes a mobile operator network without pre-existing MEC infrastructure and anticipates a surge in mobile traffic. We further posit that the MEC nodes are located within the radio access network of the base station. The MEC-NFV Management and Orchestration block plays a critical role in scaling the VNFs in unison with the varying network traffic dynamics and the projected SFCs. The VNFs are dynamically placed on the MEC nodes situated in various locations to provide low latency and high-quality services based on auto-scaling decisions.

4.3 Assumptions and System Model

The present study targeted a system model similar to the scenario illustrated in Figure 4.3, where a mobile network operator receives a variable amount of mobile traffic from diverse Internet-connected mobile devices that generate heterogeneous requests with various QoS requirements, including web services, video streaming, and online gaming. We assumed that the

MEC infrastructure deployed at the operator's site is virtualized using the concept of NFV. In an NFV-enabled MEC environment, MEC network functions utilized to process users' requests are deployed as service chains called SFCs and mapped into a collection of virtual nodes and links. These virtual nodes can run on a virtualized platform (i.e., VM or container) and can be effortlessly placed at the hosted edge nodes and proximate to the end-user to process their requests. Carefully managing the placement of VNFs on the deployed instances at the proper edge nodes and assigning them the appropriate amount of computing resources should be done by the operator, considering the edge node capacity limitations, service latency necessities, operational and setup costs, and time-varying traffic.

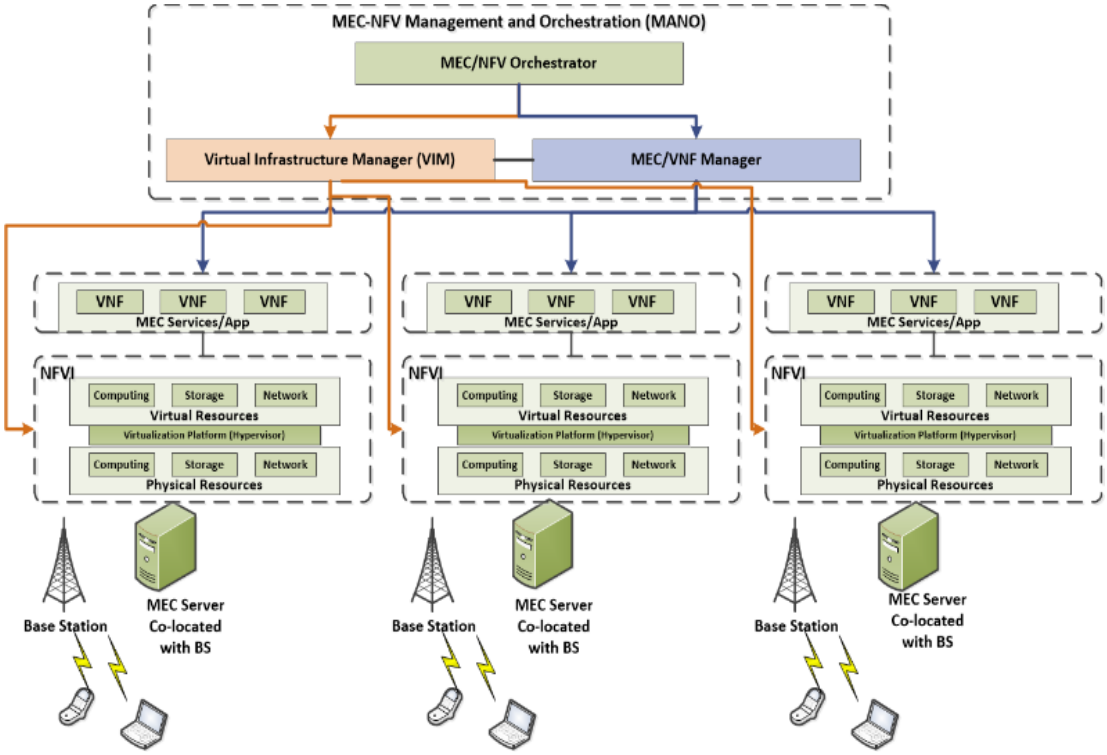


Figure 4.2 Proposed MEC-NFV Architectural Framework

We further assume the existence of a geographically distributed network of mobile devices that generate diverse computational workloads. Each workload is characterized by specific computational requirements, such as CPU, memory, and storage, and is managed through a service chain deployment on the edge infrastructure. However, the deployment of these service chains poses significant challenges, particularly in ensuring that real-time requirements, such as those for online gaming, are satisfied. To minimize latency and bandwidth consumption, we adopt a strategy that involves mapping an SFC request to a single server that can guarantee its

QoS requirements, while also reducing the need for data transfer through physical links. Nevertheless, this approach may result in excessive resource overhead, leading to increased resource consumption. Thus, we propose an efficient placement algorithm that can mitigate this overhead while optimizing resource utilization.

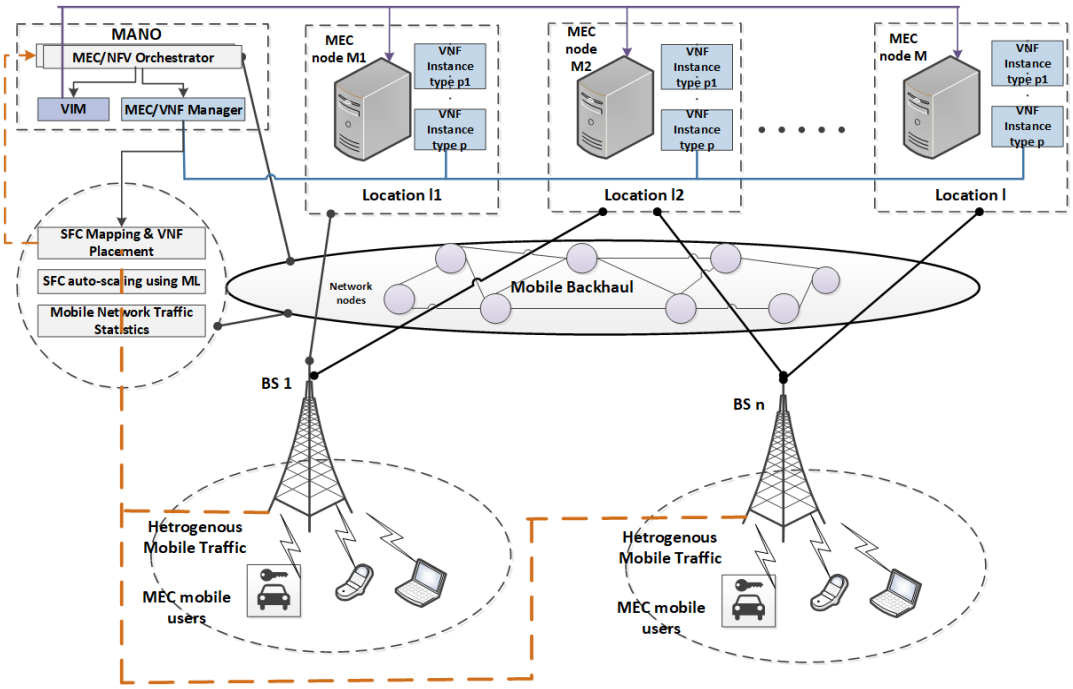


Figure 4.3 High-level MEC-NFV System Model

In the context of our proposed NFV-enabled MEC framework, the deployment of virtual instances to host VNFs introduces resource overheads that can negatively impact system performance. Specifically, each virtual instance requires independent operating system (OS), hypervisor, and related libraries, which incurs additional resource overheads. Moreover, VNFs that share the same virtualized instance on an edge node also share the resource overheads associated with that instance. To mitigate this issue, combining VNFs that require the same type of virtual instance can reduce the number of instances, as well as the associated resource overheads and resource consumption on the edge node. However, deploying fewer virtual instances can increase the number of flow paths for SFCs, leading to increased delay and bandwidth consumption. Consequently, a trade-off between delay constraints, instances' resource overheads, and bandwidth consumption is necessary.

The proposed system model considers that various edge servers can host different VNF instance types and should be accessible by mobile devices from different locations via the mobile

backhaul infrastructure. The underlying substrate network is abstracted as a direct graph $G = (M \cup E)$ containing a set of possible *M* network node locations, and *E* physical links interconnect the network nodes. Each mobile device connects to the closest BS and can offload the processing of the computing request to the MEC edge server located within the radio access network at the base station. Each $m \in M$ has a set R of computational resource types, including CPU, memory, and storage (i.e., $R = \{CPU, MEM, STR\}$) with a maximum capacity S_m^r for each computational resource type $r \in R$. The physical link $e_{(m,m`)} \in E$ that interconnects two physical nodes $(m, m`) \in M$ has a maximum bandwidth capacity $\beta_{e(m,m`)} > 0$, propagation delay d_{prop}^e and transmission delay d_{trans}^e.

Let F denote the predicted SFC requests with different types $p \in P$ received at a specific time. F metric represents the auto-scaling decision generated proactively by the ML classifier model based on the mobile traffic statistics received from the mobile user and the bandwidth requirement of a single VNF. Each SFC request $f \in F$ is divisible into a set of interdependent virtual functions modeled as a weighted directed virtual network graph and represented by a VNF Dependency Graph $G_f = (V_f, \mathrm{K}_f, \partial_{f_{max}})$; V_f denotes the set of VNFs in the SFC request, K_f denotes the set of directed virtual networking links between VNFs, $\partial_{f_{max}}$ indicates the maximum tolerable delay of the received SFC request. For an SFC request $f \in F$, each VNF $v \in V_f$ has a set of computing requirements: (a) CPU capacity S_v^{CPU}, (b) Memory capacity S_v^{mem}, (b) Storage capacity $S_v^{STR}$and (d) Transmission bandwidth requirement $\boldsymbol{b_v}$.

Considering the heterogeneity of the received requests and virtual instances overheads, let *T* represent the available VNF types that users can request. In case of requesting a VNF type in an SFC request, the mobile operator activates an instance of the VNF type by running the image of the VNF on a VM and installing the required licenses. Let $i \in I_t$ represents all available instances of type $t \in T$ that can be activated on demand. The resource overheads upon activating instances type $t \in T$ can be expressed as CPU, storage, and memory resources consumptions. Taking shareable VNF instances into account, VNFs with the same type could be assigned the same VNF instance they require, even if these VNFs are from different SFC requests. In this way, we need fewer VNF instances to process heterogenous users' requests at a given time, leading to fewer resource overheads.

Figure 4.4 shows the SFC mapping G_f to the NFV-based MEC infrastructure, *G*. A

feasible mapping of all VNF to the MEC physical nodes must adhere to the resources and bandwidth capacities of the MEC infrastructure. The mapping of each SFC request $f \in F$ can be divided into hosting and network mapping. Each VNF hosting node $v \in V_f$ belongs to SFC request f $\in F$ and is deployed on the instance I_t of type $t \in T$, is mapped to a distinguished MEC node m ϵ M. Moreover, we define the set of all possible paths between two physical nodes $(m, m`) \in M$ as $\Pi_{(m,m')}$. Each virtual link between two VNFs $k\ (v, v`), k\ \in K_f$ belong to SFC request $f \in F$ is mapped to a physical path $\pi \epsilon \Pi_{(m,m')}$.

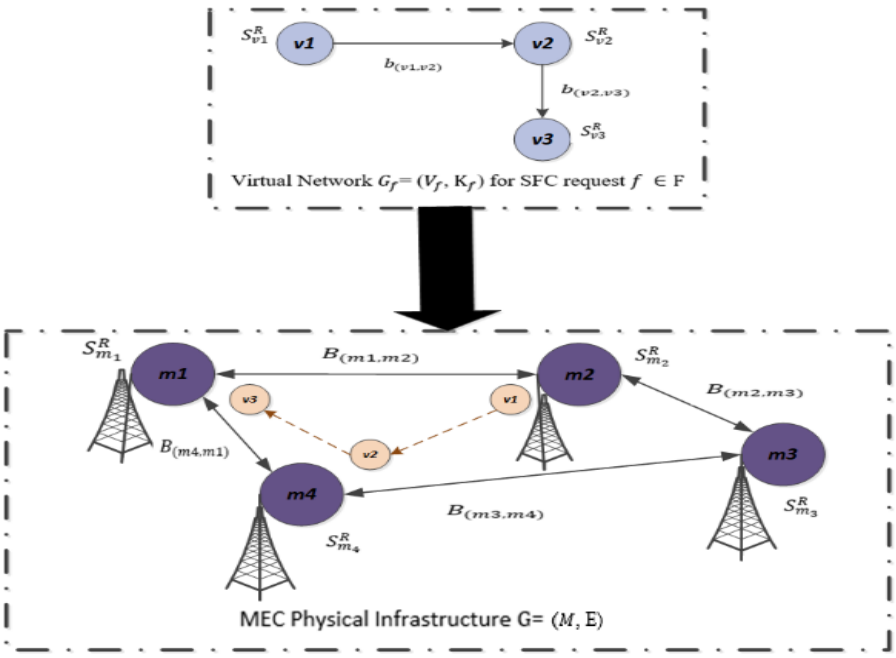


Figure 4.4 Mapping of SFC Request virtual network $\boldsymbol{G_f}$= ($\boldsymbol{V_f}$, $\mathbf{K_f}$) to MEC physical infrastructure **G**= (***M***, E)

This research study considers the average E2E delay for SFC requests as the QoS metric. The average E2E delay is a useful metric for measuring the overall performance of SFCs, it may not accurately reflect the performance of individual SFCs in latency-critical applications. To address this issue, we propose thresholds for maximum tolerable delay E2E delay of SFC $f \in F. \left(\text{i. e.}, \partial_{f_{max}}\right)$ and continuously monitor the performance of individual SFCs in real-time. The latency of every request can be decomposed into two primary components. Firstly, network delays encompassing transmission, propagation, and queueing delays that are a function of the link utilization. Secondly, queueing and processing delays that are incurred by the hypervisors

and VNF instances while processing the SFC requests at the edge nodes. These delays can cause additional overhead, which can negatively impact the overall system's performance. Thus, it is essential to minimize the overall latency of SFC requests by effectively managing the network and processing resources.

Our proposed model is aimed at minimizing the cost of deploying the MEC-NFV infrastructure while ensuring that the acceptance ratio for predicted SFCs from users at a specific time is above a certain threshold, and the average SFC latency is below another threshold. To achieve this, mobile operators must consider various costs, such as license fees for activating VNF instances, computing resource allocation costs, and bandwidth allocation costs for physical links. One approach to reducing the overall cost of implementing the MEC infrastructure is to share enabled VNF instances among multiple SFC requests, depending on the traffic processing capacity of the instance[112]. However, this introduces a tradeoff between cost and the QoS requirements of the SFCs. By setting thresholds for acceptance ratio and SFC latency, we can ensure that the tradeoff is balanced, and the model's objective function is to minimize the cost of deployment while meeting these constraints.

4.4 Problem Statement

Providing a comprehensive, efficient and scalable mapping of SFC requests requires upgrading the existing mobile operator infrastructure to an NFV-enabled MEC infrastructure. Based on the system model proposed in the previous section and given the stringent latency and varying QoS requirements of the predicted mobile workloads, we investigate and examine three nested challenges inherent in a successful, cost-effective, and machine-driven NFV-capable MEC infrastructure deployment in the cellular network. Therefore, we exploit the following subproblems:

1) *SFC Autoscaling sub-problem:* It aims to predict the number of SFC requests that will be received at any given time and to proactively determine the number and type of VNFs that need to be provisioned to process these requests. The decision is coupled with the requested time-varying traffic and VNF bandwidth requirement to handle specific services with varying latency requirements. The autoscaling method employs a machine learning (ML) model to proactively handle the workload dynamics and quantifies the VNFs required to process different requests at specific times while ensuring that QoS

requirement is not violated, and operational costs are minimized.

2) *SFC Mapping sub-problem:* It involves the allocation of resources to map SFC requests onto a set of interconnected VNFs, in a specific order. Multiple instances of VNFs of varying types can be flexibly assigned to different edge nodes within the mobile operator. To meet the predicted SFC requirements, a decision must be made on whether to choose the already allocated virtual instances or place new virtual instances to process incoming requests on the same or different edge node. The different placement options for VNFs can impact the utilization of resources on the physical links and edge servers. This sub-problem seeks to determine the optimal deployment of virtual instances for SFC requests, while minimizing overall resource consumption and ensuring SFC request latency.
3) *MEC dimensioning sub-problem:* It pertains to determining the optimal number and placement of MEC edge nodes, considering the varying SFC requests from different sites and their corresponding response times. The computing resource capacities of the MEC servers to be deployed at different locations also play a role in this decision, which is influenced by the number of predicted SFC requests. As MEC servers with different computing resource capacities may be available for deployment, selecting an appropriate amount of computing capacity for specific locations becomes crucial. The decision on computing capacity allocation directly affects the setup and deployment costs, which increase with the required computing resources of the edge nodes.

Based on the interdependencies between the above sub-problems, we define a common problem titled “An Agile and Scalable Design and Dimensioning of NEV-enabled MEC Infrastructure for Heterogeneous Latency-critical Applications” problem. The formulation of this approach is discussed in the following sections of this thesis.

4.5 SFC Autoscaling sub-problem

Auto-scaling in edge computing networks is a challenging task due to the dynamic and unpredictable nature of workload demand. Traditional methods like ARIMA have been widely used to forecast and predict the future workload in order to scale up or down the resources accordingly. However, these traditional methods are limited in their ability to handle complex and non-linear patterns of workload fluctuations, and they assume that the future workload behavior will be the same as the past behavior [113].

Machine learning, on the other hand, can provide a more accurate and flexible solution for auto-scaling in cloud computing systems. ML algorithms can learn from the historical workload data and identify the complex and non-linear patterns of workload behavior. Moreover, ML algorithms can adapt to the changing workload behavior and adjust the auto-scaling policies accordingly.

Several comparative studies have been conducted to compare the performance of traditional methods like ARIMA with ML-based methods in auto-scaling cloud computing systems. These studies have shown that ML-based methods, such as neural networks and decision trees, outperform traditional methods in terms of accuracy and efficiency. For instance, [113] compared the performance of ARIMA, neural networks, and decision trees in predicting the workload demand for a cloud computing system. The results showed that neural networks and decision trees outperformed ARIMA in terms of accuracy and efficiency.

In our study, the ML model predicts the number of SFC requests at a given time based on various factors such as the required resources (CPU, memory, storage) for each VNF in the SFC request and the required transmission bandwidth between two VNFs at any given time. The ML prediction model is part of the VNF orchestration and management engine implemented in the NFV-enabled MEC infrastructure. This ML model determines the required number of VNFs, and instance types deployed on edge servers without violating QoS requirements and avoiding network congestion.

The proposed ML classifier is identified with a set of features (inputs) associated with a proactive scaling decision (class). This study uses a real base station dataset from Shanghai Telecoms [114] for six months. The workload load measurements received at a given base station from different users for processing SFC request $f \in F$ of type $p \in P$ are denoted as λ_f and represented with the characteristics listed in Table 4.1. We considered 10 features containing momentary information on measured traffic loads and changes in such data from recent times. Features 1-3 capture the temporal properties of the data; the rest capture the measured traffic load and changes in traffic load over time. Using features 4-10, we can determine the expected compute resources required at each edge server site to process incoming SFC requests at any given time knowing BW requirement for different SFC request types (see Table 5.4 and Table 5.5 as an example for BW conversion).

Table 4.1 ML Features Set

ML Model Features
1. Year 2. Month 3. Day 4. Base station ID 5. Application type p 6. Timestamp of the decision τ 7. Number of new requests received for application p at time τ be $\boldsymbol{\lambda_f(\tau)}$ $\forall$ $\boldsymbol{f} \in \boldsymbol{F}$ 8. Number of new requests received for application p at time τ-*1* be $\boldsymbol{\lambda_f(\tau\text{-}1)}$ $\forall$ $\boldsymbol{f} \in \boldsymbol{F}$ 9. Request difference between $\boldsymbol{\lambda_f(\tau)}$ and $\boldsymbol{\lambda_f(\tau\text{-}1)}$ $\forall$ $\boldsymbol{f} \in \boldsymbol{F}$ 10. Number of running requests for application p at time τ that start running before $\boldsymbol{\lambda_f(\tau\text{-}1)}$, and it is still running

Next, the output of the ML classifier is defined. It represents the number of SFC requests $f \in F$ that should be deployed to handle different user requests of type p at any given time. In our study, the class represents the number of SFC requests to process different users' requests which is an integer between f_{max}^{p} and f_{min}^{p}. These limits are specified based on the bandwidth required to process each VNF and the maximum throughput capacity for each physical edge server. To generate a labeled training set (instances with known class labels), the scaling decision taken at time τ at this point allocates enough resources to process the incoming traffic until the next step τ+1.

To get the optimal class value $f \in F$ without violating the QoS, qos(.) function is proposed to take the workload $\lambda_f(\tau)$ at time τ as an input and output, the number of SFCs required to process measured load without violating QoS requirements by avoiding network congestion and assigning enough resources to deploy VNFs responsible for serving SFC requests.

The *qos(.)* function generates the class value at time τ as:

$$qos(\lambda_f(\tau)) = min\ \{f,\ f_{max}^{p}\};\ f \in F \qquad (4.1)$$

f_{max}^{p} is the maximum SFC request accepted to handle various user requests of type p without violating QoS requirements. The maximum limit depends on the VNF-BW requirement and the capacity of the physical connection at the edge server. It is worth noting that the *qos(.)* function is a useful tool for predicting the number of SFCs required to process measured load without violating QoS requirements, it cannot accurately predict the optimal class value without

the use of an ML model. The ML model can learn complex relationships between input features and the optimal class value and adapt to changing conditions over time, providing more accurate predictions.

This study assumes that the historic traffic load measurement data $H(f,\tau)$ is accessible at the edge nodes where the auto-scaling decision is made. Table 4.2 illustrates a sample of the training instances after preprocessing, which are employed for scaling decisions at time τ at base station BS1 to forecast future outcomes. The table considers the bandwidth requirement for VNF processing as 1 Gbps, maximum physical link capacity as 100 Gbps, and each SFC request is mapped to 5 VNFs.

Table 4.2 Sample for Processed Training Dataset

ID	Day	Month	Year	BS ID	App Type	Time Stamp	New_req	Old_req	Total_req	QoS(.)
32469	18	6	2014	4	1	00:10:26	1	1	2	2
735000	16	7	2014	5	3	00:01:52	4	3	7	5
878362	2	8	2014	0	4	00:07:43	1	1	2	2
1007818	8	9	2014	13	4	00:09:40	1	1	2	2
1773725	15	10	2014	6	2	00:07:37	2	6	8	5
1773724	10	10	2014	4	1	00:10:29	2	1	3	3
2260986	6	10	2014	9	2	00:18:10	1	1	2	2
3386052	16	10	2014	0	4	00:07:20	1	3	4	4
5304811	5	11	2014	2	3	00:30:53	2	3	5	5
6680206	5	11	2014	2	3	00:32:01	2	2	4	4

Choosing the proper ML algorithm is the next task for training the classifier. Two ML models are used in this work to formulate a prediction problem for SFC requirements: Deep Neural Network (DNN), a supervised classification model, and Long Short-Term Memory (LSTM), an artificial recurrent neural network architecture widely used in the deep learning performance of the trained classifier. These ML models are likely chosen due to their prominence and efficacy in modeling sequential data and time series analysis. LSTM networks are particularly adept at handling sequential data by capturing long-term dependencies. DNNs, on the other hand, are a versatile class of neural networks that can be applied to a broad range of problems. It is plausible that other topologies were not selected for comparison owing to their limited applicability to the specific problem at hand or a dearth of prior research substantiating

their effectiveness in modeling the data [115]. Moreover, the investigation may have focused on comparing the specific characteristics of LSTM and DNN architectures, rather than comparing a broader spectrum of network topologies. Ultimately, the selection of neural network architectures for comparison should be guided by the research question and the data characteristics.

We followed a systematic approach to choose the parameters of the DNN and LSTM models. Specifically, we used a combination of manual tuning and grid search to select the optimal hyperparameters for the models. In the manual tuning phase, we used our domain knowledge and expertise to select a range of values for the hyperparameters based on the characteristics of the dataset and the models. This involved adjusting the number of layers, the number of neurons per layer, the activation functions, and other model-specific hyperparameters.

Dataset in [114] was split into training and validation sets. The training set was used to train the machine learning model, the validation set was used to fine-tune the hyperparameters and avoid overfitting, and the testing set was used to evaluate the model's performance on unseen data. To ensure that the validation data was never used before, we used a technique called k-fold cross-validation. This involves randomly dividing the data into k subsets, using k-1 subsets for training, and the remaining subset for validation. This process is repeated k times, with each subset being used for validation exactly once. This ensures that each data point is used for validation exactly once, and that the model's performance is evaluated on a diverse set of validation data.

Given a trained classifier and a test set, the test output includes four main performance metrics:

True Positive (TP) and True Negative (TN) are when the model makes correct predictions for positive and negative actual samples, respectively. In contrast, False Positive (FP) and False Negative (FN) are when the model incorrectly predicts respective negative and positive actual samples. Thus, both ML models (DNN and LSTM) can be evaluated using performance metrics, with D the number of classes in the ML model:

1) *Accuracy*: is the most insightful performance metric when evaluating a given ML model. It gives the ratio of accurate predictions to total observed predictions:

$$Accuracy = \frac{1}{D}\sum_{j=1}^{D}\frac{TP_j+TN_j}{TP_j+TN_j+FP_j+FN_j} \qquad (4.2)$$

2) *Precision*: is the ratio of actual predicted observations to predicted positive observations.

$$Precision = \frac{1}{D}\sum_{j=1}^{D}\frac{TP_j}{TP_j+FP_j} \quad (4.3)$$

If the false positive indicator is high in the case of auto-scaling SFC requests, there will be over-provisioning of SFCs with an increase in operational costs and a reduction in the utilization of computing resources.

3) *Recall*: it indicates the actual positive observations captured in the model. So, if FN is high in the case of SFC autoscaling, it means there will be an under-provision of computing resources and degradation of QoS. The recall metric can be calculated as:

$$Recall = \frac{1}{D}\sum_{j=1}^{D}\frac{TP_j}{TP_j+FN_j} \quad (4.4)$$

4) *Fmeasure*: is the weighted average of precision and recall:

$$F_{measure} = 2.\frac{Precision\,.\,Recall}{Precision + Recall} \quad (4.5)$$

4.5.1 Approach 1: Deep Neural Network

A deep neural network (DNN) is a type of artificial neural network that is composed of multiple layers of interconnected nodes or neurons. DNNs are designed to automatically learn representations of data through a hierarchical approach, where each layer extracts increasingly complex features from the input data. This makes them particularly useful in complex applications such as image and speech recognition, natural language processing, and autonomous vehicles. DNNs have achieved remarkable performance in many fields, thanks to their ability to capture and represent high-level abstractions of complex patterns and relationships in large datasets [92]. In this work, the DNN model extracts diverse characteristics pertaining to a particular timestamp, as exemplified in Table 4.1, while also taking into account alterations in traffic patterns from preceding intervals. In this study, DNN model composed of numerous densely connected layers is utilized. The input layer, with an optimal input dimension, contains 12 neurons and acts as the input for the model. The output layer comprises six neurons, one for each class, and employs the softmax activation function, appropriate for multi-class classification tasks [92].

A hidden layer is sandwiched between the input and output layers, and an activation function is used to produce the output of this layer. This hidden layer has ten neurons, utilizing the ReLU activation function that assigns a probability range of feature vectors for the work. The

DNN model is trained with 1000 epochs, each batch containing 64 samples. The modified dataset comprises 55,311 traffic data sets, with 80% of the input dataset allocated for training and the remaining 20% reserved for testing.

4.5.2 Approach 2: Long Short-Term Memory Neural Network

Long Short-Term Memory Neural Network (LSTM) has been utilized as a distinctive variant of Recurrent Neural Network (RNN) for learning long-term dependencies [116]. LSTM has been employed to forecast the number of SFCs that are required to be deployed at any particular time [113]. A univariate time series dataset has been considered, where each record contains only the necessary number of SFCs for each timestamp. Compared to vanilla RNNs, LSTM provides several benefits in forecasting SFCs, including overcoming the vanishing gradient issue, modeling both short-term and long-term dependencies in the input data, and learning more intricate representations of the input-output relationship.

LSTM architecture used in this study includes a five-neuron input layer and a six-neuron output layer, employing the softmax activation function. The hidden layer has eight neurons with a relu activation function, and the model is trained for 1000 epochs with 64 samples per batch. The dataset is split into training and test sets, with 80% of the data used for training and 20% for testing. The data is transformed into a supervised learning dataset, where the (τ -n) data serves as input and the current time step τ as output. During training, 20% of the training data is reserved for validation. The evaluation section (Section 5.2) in the following chapter of this thesis assesses accuracy and loss for training and validation datasets for each training step, and also presents a performance comparison with the DNN model. It is demonstrated that the LSTM model performs better than the DNN model, showing the potential of the LSTM architecture to identify traffic patterns for both periodic and non-periodic time-series data [117].

The superiority of LSTM models over DNNs in predicting SFC requests in the mobile operator context is attributed to their ability to handle sequential data, capture both short-term and long-term dependencies, adapt to variable-length input sequences, and overcome the limitations of traditional RNNs. These characteristics can be supported by both theoretical and empirical evaluations of the models.

4.5.3 ML Model Implementation

Keras is an open-source python neural network library that runs on Theano [118] or TensorFlow[119]. It has a clean, unified, and streamlined high-level API that enables users to define quickly, train, and evaluate neural network models [120].

In Keras, a high-level neural network API, allows modular definition of the neural network model structure as a sequence of self-contained and fully configurable modules that can be flexibly interconnected. It offers an array of predefined neural layers, including dense, recurring, and convolutional layers, that can be easily customized to cater to the specific requirements of the problem at hand.

To implement DNN and LSTM models in Keras, a sequential model was constructed using the aforementioned predefined layers along with the corresponding activation functions. The model training process was configured by selecting an optimizer, such as the Adam optimizer's loss function, and evaluated by reporting a list of metrics, which are discussed in section 4.5. The training of the model was carried out using Google Collab, which provides a built-in graphics processing unit (GPU) to speed up the training process and enable efficient model evaluation.

It is worth noting that ML can be a powerful tool for optimization, but it has some limitations when used in isolation. One key limitation is that ML models are typically trained on historical data and may not be able to adapt to changing conditions or unforeseen scenarios. In contrast, optimization algorithms such as linear programming can explicitly model the constraints and objectives of a problem and can be designed to incorporate domain knowledge and heuristics. This can make them more robust and adaptable to changing conditions, as well as more interpretable and transparent.

Therefore, while ML can be a useful component of an optimization system, it is unlikely to be a complete replacement for traditional optimization techniques. Rather, ML and optimization can be used together in a complementary way, with ML used for prediction and optimization used for decision-making and resource allocation. This approach can leverage the strengths of both techniques and lead to more effective and efficient MEC network design and dimensioning. For this purpose, next section presents the MILP modeling for the design and dimensioning MEC sub problem.

4.6 Design and Dimensioning Sub-problem

Design and dimensioning are crucial factors in managing NFV-enabled MEC systems and are directly related to cost and heterogeneous QoS requirements. Effective design and dimensioning schemes have a positive impact on the profitability of mobile operators. The allocation of resources to predicted SFC requests will vary depending on the network topology and mobile traffic considered, to ensure a real-time response to the offered SFC requests.

An essential aspect of designing and dimensioning NFV-enabled MEC systems is the consideration of the bin packing concept when mapping SFC requests. The efficiency of bin packing requests can vary depending on whether homogeneous or heterogeneous bin packing is used. Homogeneous bin packing assumes that all requests have the same resource requirements, while heterogeneous bin packing considers requests with varying resource requirements. In terms of time processing, homogeneous bin packing is generally more efficient because it requires less computational overhead to assign requests to available resources. However, heterogeneous bin packing can provide better resource utilization and QoS guarantees for diverse workloads.

To manage the extra time needed for dynamically managing traffic in heterogeneous bin packing, various techniques can be used. For example, machine learning algorithms can be used to predict future workload patterns and optimize resource allocation accordingly. Additionally, load balancing algorithms can be used to distribute the workload among available resources based on their capacities and QoS requirements. The best trade-off between homogeneous and heterogeneous bin packing depends on the specific requirements of the MEC application and the available resources. In the short term, homogeneous bin packing may be more efficient due to its lower computational overhead. However, in the long term, heterogeneous bin packing may provide better resource utilization and QoS guarantees for diverse workloads.

The design and dimensioning problem of an NFV- MEC system in this study entails two main tasks: (i) identifying the optimal location and size of edge nodes, and (ii) minimizing the cost of resources, while simultaneously adhering to the desired QoS requirements for SFC requests. Such requirements encompass various aspects, such as latency (the duration taken to execute an SFC request and obtain a response), the amount of computational resources necessary to handle VNFs in the SFC request (where VNFs of the same type may share a virtual instance), the memory and storage capacity demanded by each VNF, and the processing sequence of the

constituent VNFs in the SFC request.

As discussed in Figure 4.4, the predicted SFC requests are mapped into a set of specific VNF types interconnected by virtual links in a particular order. A virtual link, $k \in K_f$ is defined by the pair $k\,(v, v`)$ such that $(v, v') \in V_f$, f $\in F$ where v' immediately follows v when processing f $\in F$. Each virtual link, $k \in K_f$ has a transmission bandwidth capacity b_v between a pair of VNFs $(v, v') \in V_f$. $\partial_{f_{max}}$ indicates the maximum tolerable delay of received SFC request f $\in F$ that is mapped to a set of virtual functions V_f interconnected by a set of virtual links K_f.

It is worth noting that the SFC auto-scaler module plays a pivotal role in predicting the number of SFC requests at a particular time τ, as well as determining the resources required for the VNFs and links within the chain. Subsequently, the auto-scaler's output serves as an input to the dimensioning and SFC mapping optimization problem. As such, the mobile operator must decide whether to accept or reject any anticipated SFCs based mainly on the QoS requirements of predicted SFC requests, the availability of MEC physical resources, and the economic cost of accepting such requests. Therefore, an efficient SFC mapping mechanism that hinges on the shareable VNF instances is essential to increase the acceptance ratio of anticipated SFC requests, minimize the operational cost required to deploy the necessary VNF types, and optimize the substrate resource utilization.

Prior to delving into the problem formulation, it is worth examining a standard example of an SFC mapping problem and its impact on resource utilization, bandwidth consumption, operational cost, and QoS requirements. Consider, for instance, an NFV-enabled MEC network comprising four edge servers and network nodes (e.g., switches, routers, or access points) in between, as depicted in Figure 4.5.

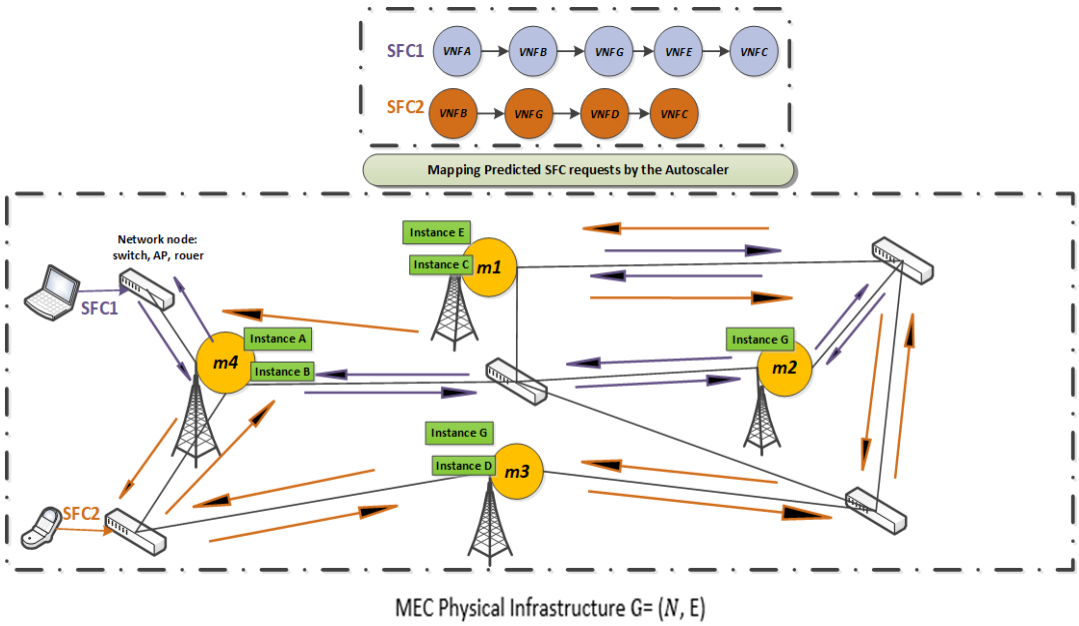


Figure 4.5 SFC Mapping in an NFV-enabled MEC network

In this scenario, we utilize multiple virtual instances, denoted as I_t, and various types of VNFs. The predicted SFCs 1 and 2 transmit data through the mobile backhaul, which comprises switches, routers, or access points. The purple and orange arrows depict the traffic flow of SFC 1 and SFC 2, respectively. The original mapping plan is as follows: VNF A, VNF B from SFC 1, and VNF B from SFC 2 are allocated to edge node 4; VNF G from SFC 1 is allocated to edge node 2; VNF G, VNF D from SFC 2 are allocated to edge node 3; VNF E, VNF C from SFC 1, and VNF C from SFC 2 are allocated to server 1. In light of the adoption of shareable VNF instances, the VNF placement scheme entails allocating two instances of type A and B to edge node 4, one instance of type G to edge node 2, two instances of type G and D to edge node 3, and two instances of type E and C to edge node 1.

Consequently, the traffic path of SFC 1 is $4 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 4$. The resource overhead on a server depends on the number of VNF instances installed on the edge server. The more VNF instances on a server, the higher the resources overheads on that edge server. Therefore, a feasible alternative is to map VNF G from SFC 1 on edge node 3 instead; VNF G from SFC 1 and SFC2 share the same instance type G on edge node 3. So, we no longer need to place an instance of type G on edge node 2, which reduces the resources overheads on edge node 2 and minimizes the operational cost as there is no need to pay activation licenses for VNF instances. However, this scheme changes the traffic path of SFC I to $4 \rightarrow 3 \rightarrow 1 \rightarrow 3 \rightarrow 4$, where the traffic

of SFC 1 has to pass through 2 more hops of link, resulting in more bandwidth consumption. The processing delay of an SFC relies on the resource utilization of the related edge servers where it is hosted. The higher the resource utilization of a server, the higher the processing delay of the relevant SFCs deployed on the server. Since there are more VNFs mapped edge node 3, the resource utilization increases, leading to a higher processing delay of SFC 1 and 2, affecting their SLA performance. This simple case demonstrates the tradeoff between server resource consumption, cost, bandwidth consumption, and delay requirements, and we need to achieve a balance among them. The following sections present the formulation for SFC mapping and the MEC dimensioning problem.

4.6.1 Design and Dimensioning Approach: MILP Model

4.6.1.1 Parameters

This subsection provides an abridged account of the input parameters that have been employed in this section to depict the physical MEC infrastructure and the virtual network of SFC requests. The relevant details are presented in Tables 4.3 and 4.4.

4.6.1.2 Decision Variables

1) *Design and Dimensioning Variables:* We recall that r is the type of MEC computing resource that takes values in the set R = {CPU; MEM; STR}.

We define a float decision variable $\boldsymbol{x_m^r}$ to measure the amount of computing resources of type $r \in R$ required to be set up in MEC node $m \in M$. An upper bound $\boldsymbol{S_m^r}$ will be set up to limit the available MEC computing resources of type $r \in R$ that can be set up in MEC node $m \in M$. We define a binary decision variable $\boldsymbol{y_m^r}$ that denotes whether MEC node $m \in M$ requires resources $r \in R$ to be setup

$$\boldsymbol{y_m^r} = \begin{cases} 1, if \ \boldsymbol{x_m^r} > 0 \\ 0, otherwise \end{cases} \quad (4.6)$$

2) *SFC Mapping & VNF resource Allocation Variables:* To decide on the acceptance of an SFC request $f \in$ F, we need the following decision variables for mapping VNF for the request:

$$\boldsymbol{\theta_f} = \begin{cases} 1, if \ \text{SFC request } f \in \text{F is accepted by the MEC system} \\ 0, otherwise \end{cases} \quad (4.7)$$

$$z_m^v = \begin{cases} 1, \text{if VNF v in SFC request f is mapped to instance } i_t \\ \text{that is running on edge node m} \\ 0, otherwise \end{cases} \quad (4.8)$$

$$y_m^i = \begin{cases} 1, \text{if instance i of type t is presented at node } m \in M \\ 0, otherwise \end{cases} \quad (4.9)$$

$$z_\pi^k = \begin{cases} 1, \text{if virtual link k} \in K \text{ is mapped to the physical path } \pi \epsilon \Pi \\ 0, otherwise \end{cases} \quad (4.10)$$

Table 4.3 MEC Physical Infrastructure **G**= (***M***, E)

Parameter	Description
$G=(M, E)$	Directed graph of the physical infrastructure composed of network nodes M locations and E links connecting them with latency requirement.
M	Set of potential MEC nodes' locations co-located with the base stations.
E	Set of physical links within the network between edge nodes co-located with base stations.
R	Type of computational resource that can be CPU, memory, or storage.
Y_{max}^r	Maximum network node locations that deploy resource $\boldsymbol{r} \in \boldsymbol{R}$.
S_m^r	The maximum capacity of computational resource $\boldsymbol{r} \in \boldsymbol{R}$ at node $\boldsymbol{m} \in \mathbf{M}$.
β_e	The bandwidth capacity of the physical link $\boldsymbol{e} \in \boldsymbol{E}$.
c_e	Bandwidth unit cost for using physical link $\boldsymbol{e} \in \boldsymbol{E}$.
d_p^e	Propagation delay for physical link $\boldsymbol{e} \in \boldsymbol{E}$.
d_t^e	Transmission delay for the physical link $\boldsymbol{e} \in \boldsymbol{E}$.
c_m^r	Cost unit for using resource type $\boldsymbol{r} \in \boldsymbol{R}$ at node $\boldsymbol{m} \in \mathbf{M}$.
u_m^r	Cost unit for setting up resource $\boldsymbol{r} \in \boldsymbol{R}$ at MEC node $\boldsymbol{m} \in \mathbf{M}$.
D_m^r	The capital cost of setting up resource $\boldsymbol{r} \in \boldsymbol{R}$ at MEC node $\boldsymbol{m} \in \mathbf{M}$.
Π	Set of all possible physical paths in MEC-NFV infrastructure.
$\Pi_{(m,m')}$	Set of possible paths between network nodes $\boldsymbol{m}$ and $\boldsymbol{m}$`.
μ_m	Processing rate for MEC node $\boldsymbol{m} \in \mathbf{M}$.

Table 4.4 Virtual Network $\boldsymbol{G_f}$= ($\boldsymbol{V_f}$, $\mathbf{K}_{\boldsymbol{f}}$) for SFC Request F

Parameter	Description
F	Set of the predicted SFCs.
G_f= (V_f, K_f)	Virtual network representing SFC request $\boldsymbol{f} \in \mathbf{F}$.
P	Set of SFC application request types.
V_f	Set of all VNFs in an SFC request $\boldsymbol{f} \in \mathbf{F}$ from all types.
K_f	Set of virtual links between network functions in an SFC request $\boldsymbol{f} \in \mathbf{F}$.
T	A set of all VNF types can be used in the received SFC requests.
I_t	Set of all available instances of type t ∈ T.
R	Type of computational resource that can be CPU, memory, or storage.
s_v^r	The required capacity of resource of type $\boldsymbol{r} \in \boldsymbol{R}$ for VNF $\mathbf{v} \in \boldsymbol{V_f}$ of type $\boldsymbol{t} \in \boldsymbol{T}$ at specific BW requirements.
b_v	Required transmission bandwidth between one VNF v and the next ordered VNF.
∂_f	The E2E delay threshold for completion SFC request $\boldsymbol{f} \in \mathbf{F}$ of type $\boldsymbol{p} \in \boldsymbol{P}$.
μ_t	Traffic capacity of VNF with type $\boldsymbol{t} \in \boldsymbol{T}$.
c_i	License cost to activate VNF instance $\boldsymbol{i} \in \boldsymbol{I_t}$.
λ_{vt}	The arrival rate of VNF $\boldsymbol{v} \in \boldsymbol{V_f}$ of type $\boldsymbol{t} \in \boldsymbol{T}$ for SFC request $\boldsymbol{f_p} \in \mathbf{F}$.
s_i^r	Resource of type $\boldsymbol{r} \in \boldsymbol{R}$ overhead when placing an instance $\boldsymbol{i} \in \boldsymbol{I_t}$ of type t ∈ $\boldsymbol{T}$.

4.6.1.3 Objective Function

The primary goal of this optimization problem is to minimize the overall cost of deploying the MEC-NFV infrastructure at the mobile operator while guaranteeing that the SFC acceptance ratio and the latency requirements adhere to the predefined thresholds. This can be achieved by formulating an objective function that considers three essential factors. Firstly, the installation cost of physical resources allocated to each MEC server, represented as $m \in M$. Secondly, there is the cost of allocating virtual resources, $r \in R$, to support the deployment of SFCs, denoted as $f \in \mathrm{F}$. This involves mapping various Virtual Network Functions (VNFs), represented as $v \in V_f$ to network nodes $m \in M$ while considering the cost of activation licenses for shareable instances, which includes the overhead cost of resources. Finally, the cost of mapping the path $\pi \in \Pi_{(m,m')}$ to a virtual link $\mathrm{k}(v, v') \in \mathrm{K}_f$, $f \in \mathrm{F}$ is determined by the

bandwidth cost, while ensuring that the total end-to-end (E2E) latency of all SFC requests from mobile users located in different areas is below the guaranteed QoS threshold..

The objective function is represented in eq (4.11) as follows:

$$COST = \sum_{\mathrm{m} \in \mathrm{M}} \sum_{r \in R} (D_m^r . y_m^r + u_m^r . x_m^r) + \sum_{m \in M} \sum_{r \in R} \sum_{f \in F} \sum_{v \in V_f} c_m^r s_v^r z_m^v + \sum_{m \in M} \sum_{i \in I} c_i \; y_m^i + \sum_{f \in F} \sum_{k \in K_f} \sum_{\pi \in \Pi_{(n,n')}} \sum_{e \in \pi} z_\pi^k \; c_e \; b_v \quad (4.11)$$

$$Z_{MILP} = Maximize \left(\sum_{f \in F} \alpha * \theta_f - COST \right) \quad (4.12)$$

α is an essential parameter that denotes the value function corresponding to the acceptance of an SFC request $f \in F$, and it is associated with the potential revenue that the mobile operator can generate. The gain objective function presented in (4.12) is used to compare the performance of our MILP model to the ABSA benchmark presented in section 4.9.

4.6.1.4 Model Constraints

1) Mapping VNFs *in the SFCs to the edge nodes and allocating resources for them:*

Constraint 1 expresses the acceptance of SFC request $f \in F$ by the proposed NFV-enabled MEC design:

$$\theta_f \leq \sum_{m \in M} z_m^v \qquad \forall v \in V_f \quad (4.13)$$

Constraint 2 expresses the selection of the physical nodes $m \in M$ to process VNF $v \in V_f$:

Binary variable z_m^v is defined as indicating whether VNF $v \in V_f$ of SFC request $f \in F$ is mapped to instance $i \in I_t$ of type $t \in T$ running on physical nodes $m \in M$ or not. If the virtual instances have the same types as the VNFs, this variable is obtained by the multiplication of the scalar u_t^v with the binary variable y_m^i :

$$z_m^v \leq u_t^v * y_m^i \quad \forall v \in V_f, f \in F, i \in I_t, m \in M \quad (4.14)$$

The scalar binary variable u_t^v is employed to indicate the mandatory instance type *t* that is

needed for the processing of the virtual network function VNF $v \in V_f$. Based on Equation (4.14), the selection of physical node $m \in M$ for VNF $v \in V_f$mapping is influenced by two crucial factors: Firstly, when an SFC is accepted, all VNFs must be allocated to the same type of instance. Secondly, it is imperative to establish whether the instance of type t ought to be allocated to physical node $m \in M$, given that at least one VNF is linked to $i \in I_t$.

Constraint 3 equation ensures that only one valid physical path is assigned for each virtual link between two VNFs $v, v' \in \mathrm{V}_f$ of service chain $f \in \mathrm{F}$

$$z_m^v \,.\, z_{m'}^{v'} = \sum_{\pi \in \Pi_{(m,m')}} z_\pi^\kappa \quad \forall \left(m, m`\right) \in M, k(v, v') \in K_f \qquad (4.15)$$

Constraint 4 ensures that the BW and the capacities of the resources (CPU, Memory, and Storage) assigned to deploy VNF with different types should not exceed the allowed limit on the physical edge node $m \in \mathrm{M}$:

$$\sum_{f \in F} \sum_{k(v,v') \in K_f} \sum_{(m,m`) \in M} \sum_{\pi \in \Pi^e_{(m,m`)}} b_v \,.\, z_\pi^\kappa \leq \beta_e \quad \forall e \in E \quad (4.16)$$

$\Pi^e_{(\mathrm{m},m`)}$ represents the set of paths from m to $m`$ that contains the link *e*.

$$\sum_{f \in F} \sum_{v \in V_f} z_m^v \,.\, s_v^r + \sum_{i \in I_{tv}} y_m^i \,.\, s_i^r \leq x_m^r \quad \forall r \in R, m \in M \qquad (4.17)$$

*E*q (4.17) shows that the physical resources required to be deployed on the edge node $m \in \mathrm{M}$ depend on the virtual resources needed to process VNF $v \in \mathrm{V}_f$ and the resource overheads introduced by the virtual instance $i \in I$.

Constraint 5 The concept of QoS requirements is being referred to in the following formulation presented in (4.18). The formulation considers the propagation delay, transmission delay, and queuing delay on the link to ensure that mapping SFC requests onto VNF chains does not violate end-to-end latency requirements. It also considers processing delays on the server. It is noteworthy that the approach used by the authors in [121] and [122] to calculate network delay is similar. To guarantee the delay requirements of each received SFC request of type $p \in$P, the total delay experienced by the SFC request is constrained to a specified E2E delay threshold in equation (4.18).

$$d_s^f + 2d_l^f \leq \boldsymbol{\partial}_f \qquad \forall f \in F \qquad (4.18)$$

where d_l^f represents the total network delays of the bidirectional links forwarding the traffic of the SFC f $\in F_p$ request among edge nodes and is given by:

$$d_l^f = \sum_{k(v,v')\in K_f} \sum_{(m,m`)\in M} \sum_{\pi \in \Pi_{(n,n')}} z_\pi^\kappa . (d_{prop}^\pi + \boldsymbol{d_{trans}^\pi} + \boldsymbol{d_{trans}^\pi} \frac{b_v}{\beta_e - b_v}) \quad (4.19)$$

$\frac{b_v}{\beta_e - b_v}$ represents the percentage of the traffic of the virtual link $\kappa \in$ K that is mapped to the physical link $e \in E$ for the requested service chain $f \in F$.

d_s^f represents the server delay (including the processing and queuing delays) experienced at the edge server when processing the received SFC requests of type *p*.

Our study takes into account the server delay, which comprises the processing and queuing delays, as the primary parameters of interest. In each edge node, the hypervisor that hosts the virtual functions and the virtual instances $v \in V_f$ running on the hypervisor induce these delays. Once an SFC request is received and accepted by the MEC network, it is queued in the hypervisor and processed by it, and a set of VNFs is assigned to it. Then, each VNF instance delivers the required service.

So d_s^f can be determined by eq (4.20), i.e., the sum of the delay induced by the hypervisor and the instance:

$$d_s^f = d_{hyper}^f + d_{instance}^f \quad \forall f \in F \quad (4.20)$$

where both delays can be modeled as M/M/1 systems:

$$d_{hyper}^f = \sum_{m\in M} \left(\frac{1}{\mu_m - \sum_{f\in F_p} \sum_{v \in V_f} z_m^v . \boldsymbol{\lambda_{v(t)}}} \right) \quad \forall f \in F \quad (4.21)$$

Eq (4.21) can be linearized by declaring a new decision variable:

$$\varsigma_m^f \geq \left(\frac{1}{\mu_m - \sum_{t\in T} \sum_{i\in I_t} \sum_{f\in F_p} \sum_{v\in V_f} z_m^v . \boldsymbol{\lambda_{vt}}} \right) \quad \forall f \in F, m \in M \quad (4.22)$$

So, equation 4.21 could be re-written as below.

$$\varsigma_m^f \mu_m - \sum_{f\in F_p} \sum_{v\in V_f} \varsigma_m^f . z_m^v . \boldsymbol{\lambda_{v(t)}} \geq 1 \qquad \forall f \in F, m \in M \quad (4.23)$$

and then declaring a new decision variable γ_m^{fv}

$$\gamma_m^{fv} = \varsigma_m^f . z_m^v \quad (4.24)$$

Hence, Eq (4.21) can be replaced by these equations:

H is an integer with a large value

$$\varsigma_m^f \mu_m - \sum_{f \in F_p} \sum_{v \in V_f} \gamma_m^{fv} . \boldsymbol{\lambda_{vt}} \geq 1 \quad \forall f \in F,\ m \in M \qquad (4.25)$$

$$\gamma_m^{fv} \leq H \qquad \forall f \in F,\ m \in M, v \in V_f \quad (4.26)$$

$$\gamma_m^{fv} \leq \varsigma_m^f \qquad \forall f \in F, m \in M, v \in V_f \quad (4.27)$$

$$\gamma_m^{fv} \geq 0 \qquad \forall f \in F, m \in M, v \in V_f \quad (4.28)$$

Regarding the delay induced by instance, it could be modeled using M/M/1 system model as below:

$$d_{instance}^f = \sum_{v \in V_f} \sum_{i \in I_t} u_t^v \left(\frac{1}{\mu_t - \sum_{f \in F_p} \sum_{v' \in V_f} u_t^{v'} . \boldsymbol{\lambda_{v'(t)}}} \right) \quad \forall f \in F \quad (4.29)$$

and it is linearized in the same way as we linearized the hypervisor delay.

2) MEC Design and Dimensioning Constraints:

Constraint 6 ensure that there are no resources $r \in R$ can be setup in a given mobile edge node m ∈ M if it is not selected as an optimal location to setup an edge node, and must not exceed the maximum resource amount for that edge server location denoted by S_m^r $m \in M$

$$x_m^r \leq S_m^r . y_m^r \qquad \forall r \in R, m \in M \quad (4.30)$$

Constraint 7 determines the maximum number of network node locations Y_{max}^r that can receive a resource $r \in R$.

$$\sum_{m \in M} y_m^r \leq Y_{max}^r \qquad (4.31)$$

4.7 Drawbacks of MEC-SFC-MILP Formulation

The previously discussed one-shot formulation of Mixed-Integer Linear Programming (MILP) permits the simultaneous mapping of both nodes and links, but it is subject to scalability issues. Due to the fact that the MILP formulation is built on an integer linear programming model, the scalability bottleneck emerges when handling a considerable quantity of SFC requests and a high-scale network of mobile operators. The mathematical model's management of numerous variables and constraints can pose a significant challenge for optimal solution

attainment within a reasonable computation time. The scalability challenge can be expressed as follows:

a. For a given predicted SFC request $\boldsymbol{G_f}=(\boldsymbol{V_f}, \mathbf{K_f})$ at a given time, a virtual link $k \in K_f$ can be mapped to up to: |M| x |M| x |Π| possible physical paths.

b. Hence, for |K| = $\max_{f \in F} |\mathbf{K_f}|$ being the maximal size of the virtual links, |K| virtual links number over |F| SFC requests can be mapped to up to: $(|E|x\ |\mathrm{M}|\mathrm{x}\ |\mathrm{M}|\mathrm{x}\ |\Pi|)^{|F|}$ possible physical paths.

In summary, node and link embedding is widely recognized as an NP-hard problem, which is tantamount to a multi-way separator problem [123]. To mitigate this complexity, we introduce a decomposition strategy based on the Benders Decomposition (BD) approach [124]. The BD technique utilizes a relaxation strategy for the feasibility and optimality cuts, leading to a Master Problem (MP) and a subproblem that are solved iteratively to direct the search process and create the violated cuts. We present a detailed explanation of the BD approach formulation in the following section.

4.8 Benders Decomposition Approach

As described in the preceding section, mathematical optimization methods are frequently utilized to optimize a particular criterion under specific system constraints. As the size of the network grows, so does the number of decision variables and constraints, and the computational complexity of the decision-making algorithm becomes a major bottleneck. Leveraging any mathematical structure in the problem formulation is crucial to implementing practical algorithms for large-scale problems. Decomposition methods provide a rational solution in such situations. In these cases, the variables and constraints are partitioned into subsets, with the constraints in one subset depending heavily on the variables in the other. Decomposition methods operate by temporarily fixing the linking variables or dropping the complex constraints, effectively decoupling the blocks and breaking down the problem into multiple smaller sub-problems.

Benders Decomposition (BD) is a widely used decomposition model [124] that leverages the problem's structure to distribute the computational complexity. The fundamental concept behind BD is to divide the problem to be solved into two simpler problems: the master problem

(MP) and the auxiliary problem (AP) (or sub-problem). The MP is a simplified version of the original problem that contains only a subset of the original integer variables and constraints. Solving the MP yields a lower bound on the problem's objective function. The auxiliary problem is equivalent to the original problem, but with the dual variables derived from the MP fixed. Its solution generates an upper bound on the problem's objective function, which is used to create cuts for the MP. The MP and AP are iteratively solved until the upper and lower bounds converge to an acceptable level. Figure 4.6 provides an overview of the BD variants and their workflow.

Let us consider an optimization problem defined as follows ([125]):

$$\begin{aligned} &\min_{x,y} c^T x + d^T y \\ &A_1 x_1 + E_1 y \geq b_1 \\ &A_2 x_2 + E_2 y \geq b_2 \end{aligned} \tag{4.32}$$

The preceding model comprises variable y, a complex variable that hinders the partitioning of the model into sub-problems. However, we can divide this problem into two sub-problems by setting this variable to a constant value ($\bar{y}$ is a fixed value for y):

$$\min_{x_1} c^T x_1$$

$$A_1 x_1 \geq b_1 - E_1 \bar{y} \tag{4.33}$$

$$\min_{x_2} c^T x_2$$

$$A_2 x_2 \geq b_2 - E_2 \bar{y} \tag{4.34}$$

In accordance with the BD methodology presented in Figure 4.6, the master problem represents a relaxed version of the initial problem and includes sets of integer variables (referred to as complicated variables) and their associated constraints. The sub-problems represent dual versions of the original problem, with the integer variables temporarily fixed to their values from the master problem. This results in an LP model for the sub-problems instead of the original MILP problem. The BD algorithm solves the master problem iteratively, followed by the corresponding sub-problems. During each iteration, a new valid inequality (known as a Benders cut) derived from the sub-problems is included in the master program. If such inequality exists, the algorithm converges to an optimal solution for the initial mixed-integer program.

Now, we explain the above procedure in a MILP model defined as follows:

$$\min_{x,y} c^T x + f^T y$$

Subject to:

$$\begin{aligned} Ay &\geq b \\ Bx + Dy &\geq d \\ y &\in Z^n \\ x \geq 0 \quad , y &\geq 0 \end{aligned} \tag{4.35}$$

If y is fixed to a feasible integer configuration, the resulting model to solve is:

$$\min_{x} c^T x$$

subject to:

$$\begin{aligned} Bx &\geq d - D\bar{y} \\ x &\geq 0 \end{aligned} \tag{4.36}$$

Therefore, the complete minimization problem can be written as:

$$\min_{y \in Y}[f^T y + \min\{c^T x | Bx \geq d - D\bar{y}\}] \tag{4.37}$$

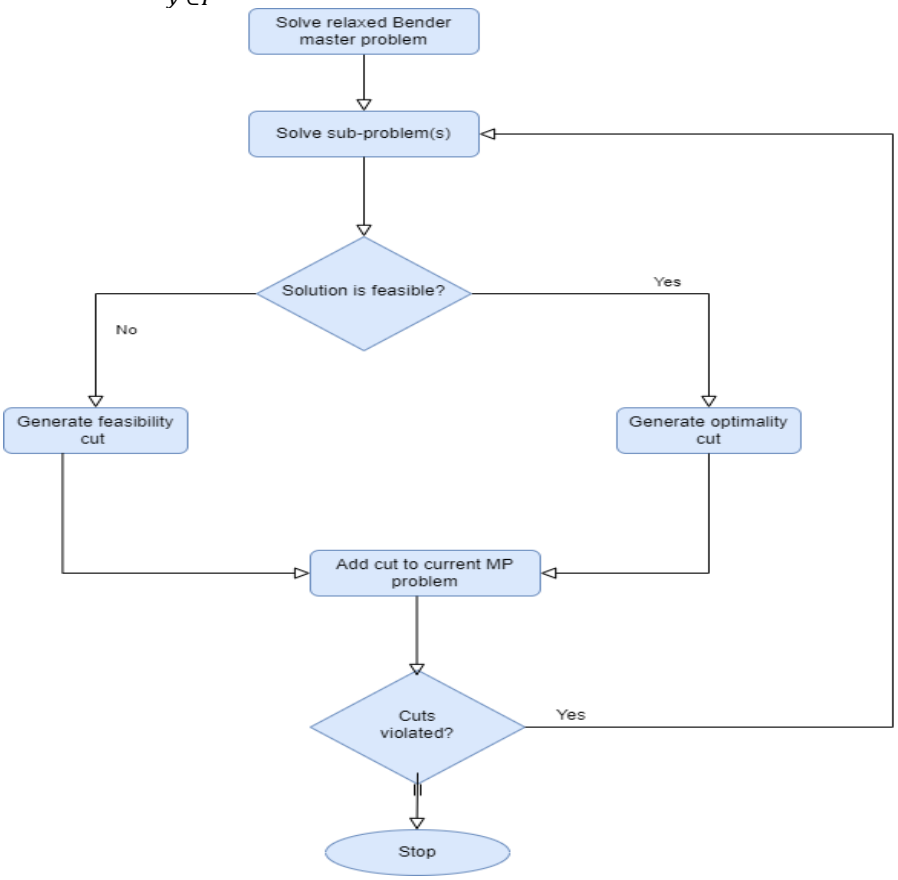


Figure 4.6 BD Flowchart

The dual of the inner LP problem is:

$$q(y) = \max_{u}(d - D\bar{y})^T u + f^T \bar{y}$$

subject to:

$$B^T u \leq c$$
$$u \geq 0 \qquad (4.38)$$

where the constraints of the dual LP sub-problem do not contain any y value. Hence, the feasibility of the MILP model could be analyzed without considering the binary variable. According to [125], we can formulate the dual LP problem in terms of extreme points for vector u as u^p and extreme rays for vector u as u^r. Suppose that we enumerate all the extreme points and rays for vector *u,* then:

$$u = \sum_i \lambda_i^p \; u_i^p + \sum_j \lambda_j^r \; u_j^r \qquad (4.39)$$
$$\sum_i \lambda_i^p \; u_i^p = 1$$
$$\lambda_i^p, \; \lambda_j^r, \; u \geq 0$$

Then the dual LP sub-problem can be expressed after substitute the value of vector *u* as:

$$q(y) = \max_{u} \; (d - D\bar{y})^T \, \mathrm{x} \sum_i \lambda_i^p \; u_i^p + (d - D\bar{y})^T \mathrm{x} \; \sum_j \lambda_j^r \; u_j^r \; + f^T \bar{y}$$

subject to:

$$\sum_i \lambda_i^p \; u_i^p = 1 \qquad (4.40)$$
$$\lambda_i^p, \; \lambda_j^r \geq 0$$

If we choose a value for y such that just one term $(d - D\bar{y})^T \, u_j^r$ becomes positive, then the LP sub problem is unbounded because there are no limits on the λ_j^r. To get an unbounded dual $q(y)$ function corresponding to not getting an infeasible $q(y)$ we have to ensure that this does not happen for any extreme ray.

If an optimal solution exists (i.e., the generated cut is not un-bounded or infeasible), it means at least one optimal solution to the dual $q(y)$ function is a corner point u_i^p. Hence the dual $q(y)$ function should have the value of the maximal point denoted as $\boldsymbol{\varpi}$ to ensure the feasibility of the solution. So, let's add the value of the maximal point:

$$min \, \boldsymbol{\varpi} \qquad (4.41)$$

subject to:

$$(d - Dy).\overline{u_j^r} \leq 0 \qquad \forall j$$

$$(d - Dy).\overline{u_i^p} \leq \boldsymbol{\varpi} \qquad \forall i$$

$$y \in Y$$

$$\boldsymbol{\varpi} \in Z^n$$

The above problem becomes the bender master problem:

$$min\, \boldsymbol{\varpi} + f^T y \tag{4.42}$$

subject to:

$$(d - Dy).\overline{u_j^r} \leq 0 \qquad \forall j$$

$$(d - Dy).\overline{u_i^p} \leq \boldsymbol{\varpi} \qquad \forall i$$

$$y \in Y$$

$$\boldsymbol{\varpi} \in Z^n$$

By solving the bender LP sub problem in equation (4.38). we can find the extreme points $\overline{u_i^p}$. These we use to generate the *optimality cuts.* While, to generate the *feasibility cuts*, we need to be able to find extreme rays when the LP sub problem is un-bounded as we are trying to solve it.

The present study concerns the MILP one-shot formulation for the design and dimensioning of NFV-enabled MEC. To address this issue, we put forward a novel approach, MEC-SFC-BD, utilizing Bender decomposition. In this approach, the MILP decision variables and constraints are relaxed to construct a simplified master problem and subproblem. The relaxation process ensures that the problem can be decomposed into simpler subproblems. Specifically, the subproblem is a relaxed version of the original MILP problem where certain constraints are relaxed and variables are fixed based on the master problem's solutions. The proposed method's success is dependent on the quality of the relaxation applied to the MILP model, and the effectiveness of the decomposition in guiding the algorithm to a high-quality solution. These relaxations are:

Relaxation 1: To simplify the QoS constraint (eq 4.18) in the MILP problem, we assume that the total SFC request delay ($\boldsymbol{\partial}_f$) is divided equally between the hypervisor delay (d_{hyper}^f)

and the networking delay (d_l^f). Then we bound the denominator in the hypervisor delay by (L/$\boldsymbol{\partial}_f$) where L is a fixed value. The impact of the virtual instance delay ($d_{instance}^f$) is ignored here because the MILP one-shot results show that instance delay has a low impact compared to the hypervisor delay on the total delay of the SFC. Applying these assumptions to the total delay constraints, we get the following from MILP QoS constraint:

$$d_s^f + 2d_l^f \leq \boldsymbol{\partial}_f \qquad \forall f \in F \tag{4.43}$$

$$d_s^f = d_{hyper}^f = \sum_{m \in M} \left(\frac{1}{\mu_m - \sum_{f \in F_p} \sum_{v \in V_f} z_m^v . \boldsymbol{\lambda}_{v(t)}} \right) \leq \frac{\boldsymbol{\partial}_f}{2} \tag{4.44}$$

Assume that:

$$\mu_m - \sum_{f \in F_p} \sum_{v \in V_f} z_m^v . \boldsymbol{\lambda}_{v(t)} \leq \frac{L}{\boldsymbol{\partial}_f^{min}} \tag{4.45}$$

where $\boldsymbol{\partial}_f^{min}$ is the worst case or most strict request delay threshold considered instead of thresholding each MF delay and processing time for each edge node.

Similarly, for the network delay (d_l^f):

$$2d_l^f \leq \frac{\boldsymbol{\partial}_f}{2} \qquad \forall f \in F \tag{4.46}$$

Relaxation 2: To simplify physical resources constraints in the MILP approach (eq 4.17), we assume function *i(v)* as the type *i* for VNF *v* where each VNF has a type of *i*:

So, when $z_m^v = 1$ then $z_m^{i(v)} = y_m^i = 1$. We can rewrite the constraint in (4.17) as below:

$$\sum_{f \in F} \sum_{v \in V_f} z_m^v . s_v^r + \sum_{v \in V_f} z_m^{i(v)} . s_{i(v)}^r \leq x_m^r \quad \forall r \in R, m \in M \tag{4.47}$$

Relaxation 3: To simplify mapping virtual links into physical links constraint (eq 4.15), we can get rid of multiplication and replace is it with an addition operation:

$$z_m^v + z_{m'}^{v'} - 1 = \sum_{\pi \in \Pi_{(m,m')}} z_\pi^\kappa \quad \forall \left(m, m`\right) \in M, k(v, v') \in K_f \tag{4.44}$$

Relaxation 4: Assume the following decision variables are continuous: z_m^v , $y_m^{i(v)}$, $z_{m'}^{v'}$, z_π^κ

By means of the relaxations mentioned earlier, we can decompose the original MILP problem into a master problem and a sub-problem utilizing the Benders Decomposition technique, resulting in two distinct objectives:

1) *Master Problem:* the problem of determining the best subset of previously generated bender cuts to minimize dimensioning costs.
2) *Sub Problem:* the problem of adding a new feasible bender cut to the Master Problem.

4.8.1 Auxiliary Problem (Sub Problem)

1) *Objective Function:*

The *objective function* for the dual inner minimization sub-problem is:

$$COST = \sum_{m \in M} \sum_{r \in R} (u_m^r \,.\, x_m^r) + \sum_{m \in M} \sum_{f \in F} \sum_{\boldsymbol{v_t} \in \boldsymbol{V_f}} \sum_{r \in R} s_v^r z_m^v \, c_m^r + \sum_{m \in M} \sum_{v \in V} c_{i(v)} \; z_m^{i(v)} + \sum_{f \in F} \sum_{k \in K_f} \sum_{m \in M} \sum_{\pi \in \Pi_{(n,n')}} \sum_{e \in \pi} z_\pi^\kappa \; c_e \; b_v \qquad (4.48)$$

Here, (u_m^r, c_m^r, $c_{i(v)}$) are the same unit resources costs as in section 4.6.1.

2) *SFC mapping and VNF resource allocation constraints*:

$$\bar{\theta}_f \;\leq \sum_{m \in M} z_m^v \;:\quad \bar{\theta}_f \; is\; fixed \qquad (\alpha^v) \qquad (4.49)$$

$$z_m^v + z_{m'}^{v'} - 1 - \textstyle\sum_{\pi \in \Pi_{(m,m')}} z_\pi^\kappa \leq 0 \qquad (-\vartheta^k) \qquad (4.50)$$

$$\textstyle\sum_{f \in F} \sum_{k(v,v') \in K_f} \sum_{(m,m`) \in M} \sum_{\pi \in \Pi^e_{(m,m`)}} b_v \,.\, z_\pi^\kappa \;\leq\; \mathrm{B}_e \qquad (-\beta_e) \qquad (4.51)$$

$$\textstyle\sum_{v \in V_f} z_m^v \,.\, s_v^r + \sum_{v \in V_f} z_m^{i(v)} \,.\, s_{i(v)}^r - x_m^r \leq 0 \qquad (-\eta_m^r) \qquad (4.52)$$

$$2d_l^f \;\leq\; \frac{\boldsymbol{\partial}_f}{2} \qquad \left(-\varepsilon_f^l\right) \qquad (4.53)$$

$$\frac{1}{d_s^f} \leq \frac{L}{\boldsymbol{\partial}} : \quad \boldsymbol{\partial} = \min_{f \in F} [\boldsymbol{\partial}_f \; | L \geq 2\}] \qquad \left(\varepsilon_f^s\right) \qquad (4.54)$$

Equation (4.49) grants the acceptance of the maximum number of SFC requests by the optimized MEC design. Equation (4.50) validates that only one physical path is assigned for each virtual link between two VNFs $v, v' \in \mathrm{V}_f$ of service chain $f \in \mathrm{F}$. Equation (4.51) expresses the bandwidth capacity of networking link $e \in E$. Equation (4.52) expresses the available capacity of resource $r \in R$ in MEC node $m \in M$. Equation (4.53) expresses the network latency threshold that must be satisfied for each SFC request. Equation (4.54) expresses the server latency threshold that must be satisfied for each SFC request, and it is bounded by a fixed value L.

3) *MEC Design and Dimensioning constraint*:

$$x_m^r \leq S_m^r \,.\, \bar{y}_m^r \quad : \quad \bar{y}_m^r \; is \; fixed \qquad (-\delta_m^r) \qquad (4.55)$$

Equation (4.55) ensures that the resource $r \in R$ that is deployed in a given mobile edge node $\mathrm{m} \in \mathrm{M}$ does not exceed the maximum resource amount for that edge server location denoted by $S_m^r \; m \in M$ when the integer decision variable $\bar{y}_m^r$ is fixed.

Given the constraints above, then the possible cuts value for vector *u* can be as:

$$u = [\alpha^v \,, -\vartheta^k, -\beta_e \,, \; -\eta_m^r \,, -\varepsilon_f^l \,, \varepsilon_f^s \,, -\delta_m^r]$$

And the lower boundaries can be represented as:

$$d - Dy = [\; \bar{\theta}_f \;, -\mathrm{B}_e \,, 0 \,, 0, -\frac{\boldsymbol{\partial}_f}{2} \,, \frac{L}{\boldsymbol{\partial}}, \; -S_m^r \,.\, \bar{y}_m^r]$$

Therefore,

$$(d - Dy).u^T = \; \sum_v \alpha^v .\; \bar{\theta}_f \; + \sum_e \beta_e \,.\, \mathrm{B}_e + \sum_f \varepsilon_f^s \;.\frac{L}{\boldsymbol{\partial}} \qquad (4.56)$$
$$+ \sum_f \varepsilon_f^l \,.\frac{\boldsymbol{\partial}_f}{2} + \sum_m \sum_r \delta_m^r \,.\, S_m^r \,.\, \bar{y}_m^r$$

4.8.2 Master Problem

1) Objective Function:

As mentioned previously, the sub-problem is a dual LP problem, and the master problem is a pure IP problem (no continuous variables are involved). The master problem, denoted by MEC-SFC-BD-M, is defined as follows:

$$Minimize \sum_{\mathrm{m} \in \mathrm{M}} \sum_{r \in R} (D_m^r \,.\, y_m^r) \; + \; \boldsymbol{\varpi} \qquad (4.57)$$

Here y_m^r and θ_f are integer binary values.

2) Constraints:

Subject to:

$$\theta_f \; \geq \; \alpha \,.\, |F| \qquad (4.58)$$

$$\boldsymbol{\varpi} \geq \; u_p^T \; (d - Dy) \qquad (4.59)$$

$$\mathbf{0} \geq u_r^T \; (\mathrm{d} - \mathrm{Dy}) \qquad (4.60)$$

where u_p^T and u_r^T values present the benders cuts generated from the subproblem and added to the master problem. u_r^T represents the extreme ray vectors, and u_p^T represents the extreme point vector. While $\boldsymbol{\varpi}$ represents the maximal point based on equation (4.42).

4.8.3 Solving the MEC-SFC-BD Model

The steps below are followed in solving the MEC-SFC-BD model formulated in this section to support high-scale system demand:

1) Reformulating the problem into two nested minimization problems, where the outer minimization (i.e., the master problem) is over discrete variables (θ_f , y_m^r), while the inner minimization (i.e., the sub-problem) is only over continuous variables (z_m^v , y_m^i, $z_{m'}^{v'}$, z_π^κ).
2) Assign initial bounds UB=+∞, LB=−∞ and $\epsilon = a$ small number.
3) Assign initial random dummy values for y.
4) Set in the fixed $\overline{y}$ into the Benders sub-problem and solve the problem to get the extreme point $\overline{u_\iota^p}$ and objective function value for the sub-problem.
5) Find the upper-bound for the objective function value of the sub-problem
6) Add new feasible cut $\sum_i \left(d_i - \sum_k D_k^i \,.\, y_k\right) \overline{u_\iota} \leq \boldsymbol{\varpi}$ to the constraints of the master problem.
7) Solve the Benders master-problem to get new *y* value.
8) Calculate the lower-bound for the objective function of the master problem.
9) Terminate if UB − LB ≤ ϵ
10) Otherwise, go to step 4 with the new value for *y*.

4.9 Benchmarks

To conduct a comprehensive evaluation of the efficiency of the dynamic design and dimensioning methodology, we have employed a rigorous benchmarking process. Specifically, five distinct benchmarks have been selected to gauge the model's performance and assess its efficacy. These benchmarks have been carefully chosen to represent a range of different scenarios and challenges that may arise in practical applications of the methodology. The

resulting assessment provides a thorough and multifaceted evaluation of the methodology's effectiveness and robustness in a variety of contexts:

1) ***Cloud-only scenario*** which entails deploying an NFV-enabled MEC system within a remote cloud data center that shares the same simulation infrastructure as the edge nodes. Such an approach involves a centralized architecture that offers a range of benefits, such as scalability and ease of management, while eliminating the need for distributed resources. Nonetheless, the latency introduced by long-haul data transmissions between the centralized cloud infrastructure and mobile devices poses significant challenges in meeting the stringent QoS requirements of modern mobile applications. Furthermore, the limited bandwidth capacity and unpredictable network connectivity of mobile devices can result in a poor user experience and degraded network performance [126].
2) ***No-scaling scenario***, where we are not introducing the ML model at the first stage to predict the incoming SFC requests and take the autoscaling decision. ML allows to allocate the resources needed, resulting in lower blocking rates for incoming requests and better resource utilization [127].
3) ***No v-instance sharing scenario***, In the absence of v-instance sharing in the system model, the edge nodes' v-instances types are installed and activated without any sharing. This scenario assumes that each candidate edge node has all types of virtual instances installed and is available for immediate use without any resource constraints [128]. This approach is considered suboptimal in terms of resource allocation and utilization, and it may lead to under-utilization of resources, thereby leading to higher costs and lower performance levels.
4) ***ABSA approach*** in [129] is an affinity-based simulated annealing heuristic approach that optimizes the cost and delay associated with VNF placement. This approach is based on a set of greedy algorithms and is the most like to our work. However, there are several differences between our work and ABSA. Firstly, [129] does not consider v-instance sharing, assuming that all v-instances types should be installed on the edge nodes. Secondly, the model in [129] assumes constant delay for nodes and links and does not consider queueing and processing delays, which are considered in our work.
5) ***MEC-SFC-Greedy***

   For a set of ordered virtual functions $\{v_1 \ldots . v_n\}$ from a request $f \in$ F, we wish to

allocate task resources to MEC nodes by greedily choosing the node $m \in$ M with available resources S_m^r and with minimal server and network latency [130]. Once a servicing MEC node is chosen, we find a path with the minimal routing cost between m and m' that satisfies the bandwidth constraints per link. This benchmark assumes that all virtual instance types are installed on each physical node. Aside from reducing design and dimensioning costs, our main objective is to satisfy the latency requirements of tasks with a high acceptance rate. For this reason, our greedy metric is, first and foremost latency.

MEC-SFC-Greedy Algorithm*: Greedy algorithm by MEC node latency.*

Result: *Greedy Fog-DC Design and Dimensioning.*

Enumerate SFC requests such that F = {1, 2 …|F|}.

For request $f \in F$, *enumerate virtual functions* $\{v_1 \ldots . v_n\}$

Initialize $S_{m,USED}^r = 0$, $\beta_{e,USED} = 0$

for *f = 1 to |F|* **do**

  for *v = 1 to |V|* **do**

    $M` \leftarrow M$;

    // For VNF v, select a node $m \in$ M *with minimal server and network latency.*

    while *True* **do**

      if *v = 1* **then**

        $\Delta\, t_1 \leftarrow min\{(d_s^f + 2d_l^f) |\ v = v_1\}$

        $m_1 \leftarrow arg\ min\{(d_s^f + 2d_l^f) |\ v = v_1, m \in \text{M}\ \}$

      end

      else

        $\Delta\, t_v \leftarrow \min\{(d_s^f + 2d_l^f) |\ v = v_v\}$

        $m_v \leftarrow \arg\min\{(d_s^f + 2d_l^f) |\ v = v_v, m \in \text{M}\ \}$

      end

      if $s_v^r \leq S_m^r - S_{m,USED}^r$ **then**

        $S_{m,USED}^r \leftarrow s_v^r + S_{m,USED}^r$

        // Greedily route backward to m_{v-1}

        while *j > 1* **do**

          $c_\pi \leftarrow min_{\pi \in \Pi_{(m,m')}} \sum_{e \in E} b_v\, c_e$

          $\pi \leftarrow argmin_{\pi \in \Pi_{(m,m')}} \sum_{e \in E} b_v\, c_e$

```
                    if b_v ≤ β_e − β_e,USED then
                        β_e,USED ← b_v + β_e,USED ;
                        break;
                    end
                    else
                        Π_(m,m′) ← Π_(m,m′) \ π;
                    end
                end
            end
            else
                M` ← M` \ m_v
            end
        end
    end
end
```

4.10 Conclusion

In this chapter, we have put forth an agile design and dimensioning strategy that allows for the deployment of a comprehensive NFV-based MEC that can cater to a diverse set of SFC requests with varying requirements, assuming no pre-existing infrastructure. To begin with, we laid out the underlying assumptions and provided a detailed system model for an agile NFV-enabled MEC system. Subsequently, we highlighted the core problem and presented an illustrative example. We then proposed a solution named the Cost-Efficient Auto-Scalable Design and Dimensioning of MEC-NFV Infrastructure for Heterogeneous SFCs problem. We discussed the optimal design and dimensioning approach using MILP to reduce dimensioning cost. To tackle scalability issues while keeping cost-effectiveness in mind, we presented a near-optimal Bender decomposition formulation. In the following chapter, we will leverage simulation to substantiate that the BD approach is an effective means to arrive at optimal dimensions for high-scale systems.

Chapter 5

NUMERICAL EVALUATION AND DISCUSSION

5.1 Introduction

This chapter presents an analytical study that was conducted to evaluate the performance of two subproblems in the agile design and dimensioning approach, namely the Proactive SFC Autoscaling sub-problem and the Design and Dimensioning sub-problem. The study also investigates the efficiency of the approach with respect to varying parameters, such as the accuracy of the ML model used in the autoscaling mechanism, dimensioning cost, and acceptance ratio in comparison to benchmarks presented in section 4.9. Furthermore, we carried out a time and cost analysis between the MILP model and the Benders decomposition approach, and found that the latter yields near-optimal outcomes at a significantly lower design and dimensioning cost. This chapter provides a detailed account of the performance simulation we performed, including its settings and the results we obtained.

5.2 SFC Autoscaling Sub-problem Accuracy Evaluation

The present study involves an evaluation and comparison of the performance of DNN and LSTM algorithms using a real dataset [114]. The aim of the proposed system is to predict the number of SFC requests at any given time, and accordingly allocate the resources required to process these requests on different edge servers. The dataset used in the experiment was obtained from Shanghai Telecom [114] and comprises Internet access logs of users along with their base station locations. The dataset contains records of 3233 base stations, and a subset of these base stations were randomly selected for the training and validation of the ML model. It is assumed that each user request corresponds to an SFC request, which is mapped to a set of VNFs deployed on various edge nodes co-located with the base station. Furthermore, it is assumed that the received SFC request types can be categorized into four major types. The maximum number

of SFCs that can be served for each request type f_{max}^{p}on each edge server depends on the bandwidth requirement for each VNF within the chain request and the processing capacity for each server. For example, if we assume that the BW requirement for each VNF is 1 Gbps, the processing capacity for each server is 100 Gbps, and each SFC request is mapped into 5 VNFs. Then,

$$f_{max}^{p} = \frac{100Gbps}{1Gbps*5VNFs*4\ sfc\ types} = 5\ \text{SFCs} \quad (5.1)$$

The number of new SFC requests and running SFC requests at a specific time τ were determined using start and end times. At time zero, we assumed that we had a set of $\boldsymbol{\lambda_f}(0)$ requests waiting to be processed. At the beginning of each new period, which was based on an hourly basis, the SFC requests were updated as follows:

$$\boldsymbol{\lambda_f}(\tau) = \lambda_f(\tau - 1) - drop\ (\tau) + add(\tau) \quad (5.2)$$

with:

- $add(\tau)$: SFC requests are received between period τ-1 and period τ (i.e., its starting time is between τ-1 and τ).
- $drop\ (\tau)$: SFC requests that are finished at period τ.
- $\lambda_f(\tau - 1) - drop\ (\tau)$: represents the SFC requests still running in the system at time τ and their start time is less than or equal to time τ-1.

The dataset contains the operational status of all base stations in Shanghai for a period of six months. Each user ID in the dataset is associated with a particular SFC request type. The machine learning models used in the research were trained on a computer system with an i7 Intel(R) Core 2.5 GHz CPU and 16 GB RAM, running on the Windows 10 Professional operating system. The experiments and model training were conducted using online libraries available on Google Collab.

Figure 5.1 shows the performance of the LSTM model using validation and training datasets for each training epoch in terms of accuracy and loss. In these graphs, the models were generated with one hidden layer and two activation units for the whole model. Table 5.1 shows a performance comparison between DNN and LSTM models implemented in Keras. For the purpose of model training and validation, we used a dataset comprising 55,311 traffic data points. The results show that the LSTM model outperforms DNN in all measurements, with

95.6% accuracy, 95.0 % precision, 95.6% recall, and 95.6% f-measure.

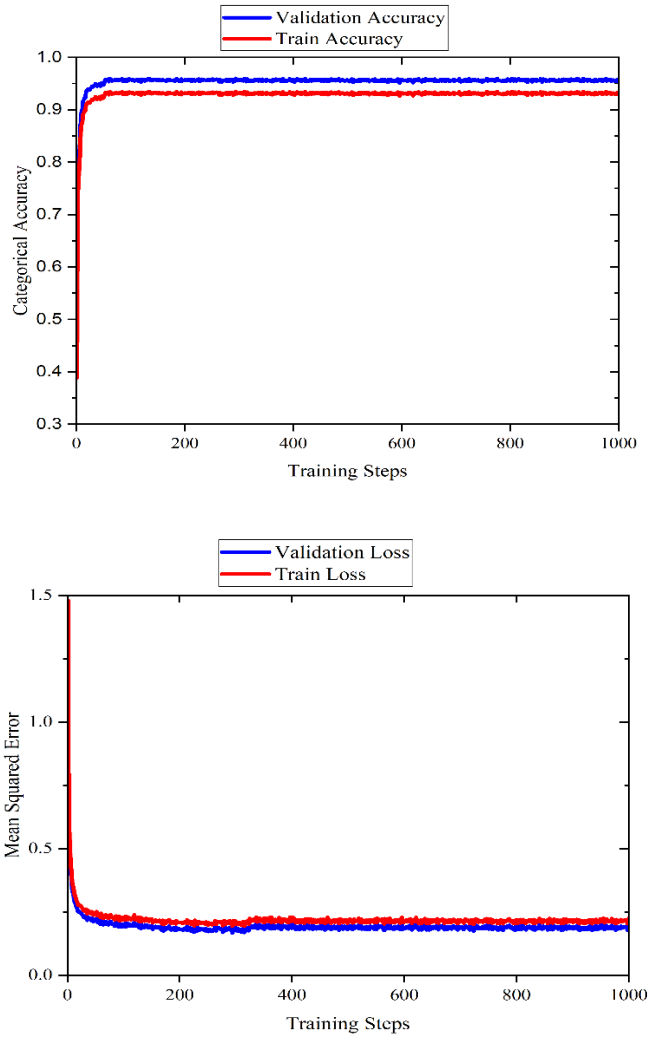


Figure 5.1 LSTM Model Prediction Accuracy

Table 5.1 Performance Comparison for LSTM and DNN Models

Performance Metric	DNN	LSTM
Accuracy	93.7%	95.6%
Recall	93.7%	95.6%
Precision	92.9 %	95%
F-measure	93.7%	95.6%

Table 5.2 shows the confusion matrix for the LSTM classifier model for the test data samples. For example, it can be seen that Class 2 has 8 and 9 misclassifications as Class 1 and Class 4, respectively.

Table 5.2 Confusion Matrix

Class	1	2	3	4	5
1	1245	1	0	0	0
2	8	728	0	9	0
3	0	2	481	0	0
4	0	0	5	961	0
5	0	0	0	4	989

The graphs presented in Figure 5.2 illustrate the performance of the LSTM classifier model in predicting SFC autoscaling for five different base stations (a-e) over the course of a day. The actual output generated from the dataset is depicted in blue, while the predicted SFC scaling determination is represented by the red line.

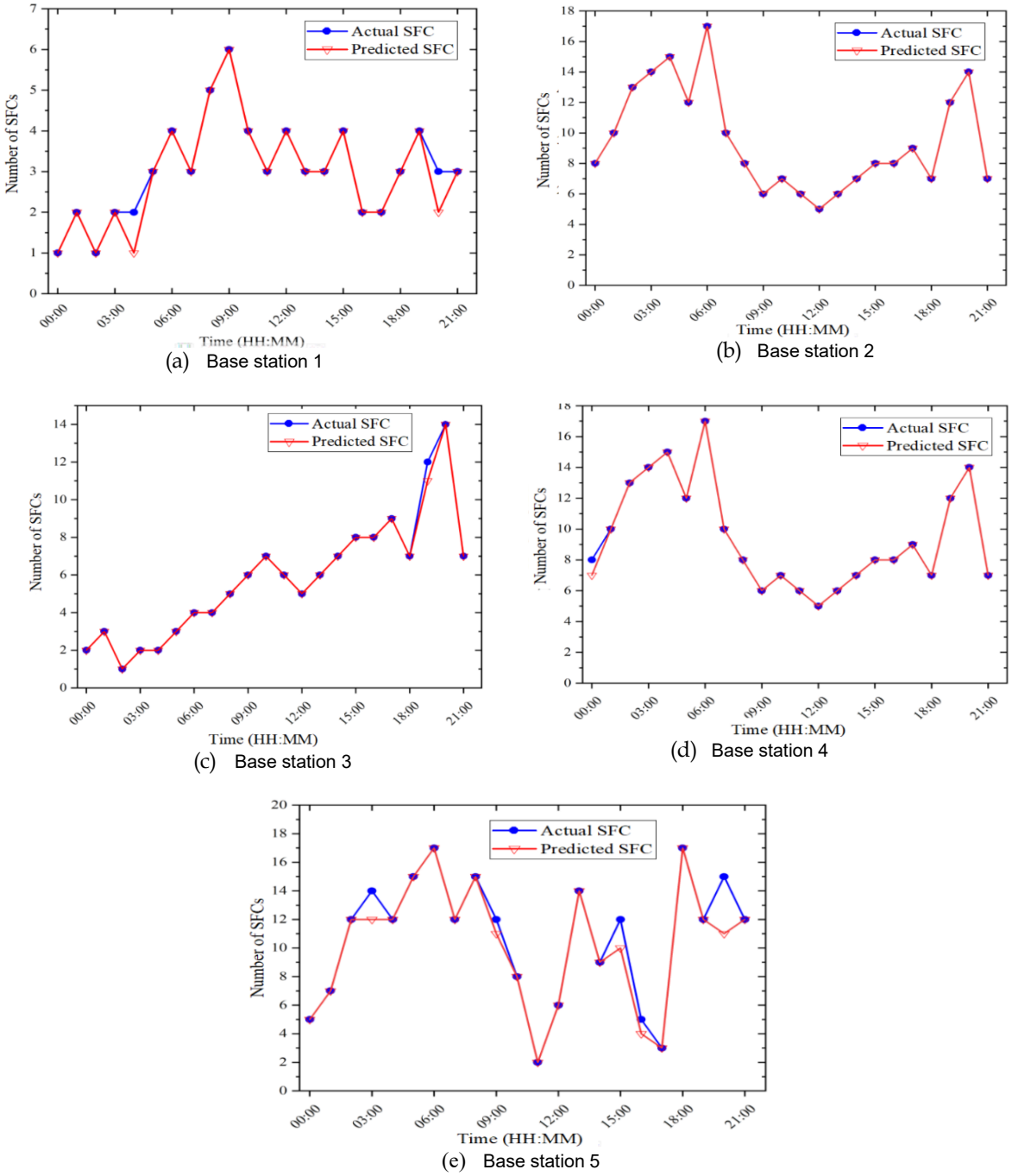
The zig-zag shape observed in the predicted output is a reflection of the real-world traffic patterns, which are often characterized by periodic fluctuations in demand. The model is able to capture these fluctuations and make accurate predictions based on the available historical traffic data. The robust predictive capability of the LSTM classifier model is evident in its ability to closely follow the actual traffic patterns, as shown by the high degree of overlap between the predicted and actual output lines.

It is worth noting that the predicted SFCs generated by the LSTM model are used as input to evaluate the SFC mapping and MEC server dimensioning subproblems presented in the subsequent sections of this thesis.

5.3 Design and Dimensioning MILP Approach Evaluation

To assess the effectiveness of the proposed design and dimensioning approach, an empirical study was undertaken to investigate its performance under varying parameters and analyze the trade-off between the deployment cost of mobile edge servers and the percentage of workloads that can be accommodated. The numerical evaluation was conducted using IBM ILOG CPLEX Solver on a computer equipped with an i7 Intel(R) Core 2.5 GHz CPU and 16GB of RAM. The model was assessed based on the predicted SFCs generated from the ML model, and the outcomes were compared in terms of cost and accepted load when deploying the same

infrastructure at the cloud center.



(a) Base station 1

(b) Base station 2

(c) Base station 3

(d) Base station 4

(e) Base station 5

Figure 5.2 Actual vs. Predicted SFCs at edge nodes using LSTM model for different BS

Efficient allocation of resources is paramount to process diverse SFC requests received from mobile users at varying locations. In this regard, we have developed a LSTM model that predicts the number of SFC requests, considering different bandwidth requirements, for seven

base stations in the mobile operator network. The mobile network operator allocates necessary resources at edge servers to process the VNFs within the SFC requests. Our proposed approach aims to optimize resource utilization while maintaining service quality for end-users.

To address this problem, we considered four types of SFCs: Web Services (Search, etc.), Voice over IP (VoIP) (audio), Video Streaming, and Online Gaming, and six types of VNFs: Network Address Translator (NAT), Firewall (FW), Traffic Shaper (TS), WAN Optimization Controller (WOC), Intrusion Detection and Prevention System (IDPS), and Video Optimization Controller (VOC). The SFCs with VNF chaining and their latency requirements "$\boldsymbol{\partial_f}$" [122] are listed in Table 5.3. For instance, processing video streaming services require deploying the following VNFs on the MEC servers, in this order: NAT, FW, TS, VOC, and IDPS. The transmission bandwidth requirement between two consecutive VNFs b_v is 100 Mbps. We set the license activation cost for all VNF instance types c_i is the same. For scalability, we assumed the arrival rate of the SFC requests ranges between 1 and 250 requests/sec for all types $p \in P$.

Over the VNF resource requirements, we used tables (5.4 & 5.5) to convert from the network bandwidth requirement to compute (CPU) and memory resources requirement that should be assigned to VNFs when deployed on the edge servers without QoS degradation. These values were generated by extrapolating information from prior studies [131] and industry data.

The storage resource requirement for different VNF types was selected from a uniform distribution with low and high values of S_{vmin}^{STR} = 2 GB and S_{vmax}^{STR} = 64 GB, respectively. The mobile devices were assumed to be scattered at different locations within the considered network. Each mobile device was connected to BS available at its location and send one type of the SFC requests listed in table 5.3.

Table 5.3 Service Chain Types

Service Chain Type (P)	**VNF Chain (T)**	**Latency Requirement**
Web Service	NAT-FW-TS-WOC-IDPS	<100 ms
VOIP	NAT-FW-TS-FW-NAT	<20 ms
Video Streaming	NAT-FW-TS-VOC-IDPS	<100 ms
Online Gaming	NAT-FW-VOC-WOC-IDPS	<100 ms

Table 5.4 Conversion to Computation Resource CPU

VNF Type	BW requirement (Mbps)			
	100	**200**	**500**	**1000**
NAT	0.1	0.1	0.1	0.2
FW	0.1	0.1	0.4	1.6
TS	0.1	0.2	0.8	1.6
WOC	0.1	0.1	0.2	0.4
DPS	0.1	0.1	0.4	1.6
VOC	0.2	0.2	0.8	1.6

Table 5.5 Conversion to Memory Resource GB

VNF Type	BW requirement (Mbps)			
	100	**200**	**500**	**1000**
NAT	0.1	0.1	0.1	0.2
FW	0.2	0.2	0.8	3.2
TS	0.1	0.2	0.8	1.6
WOC	0.1	0.1	0.2	0.4
DPS	0.2	0.2	0.8	3.2
VOC	0.4	0.4	1.6	6.4

We designed a MEC-NFV infrastructure with (7) candidate locations, as shown in Figure 5.3. Each edge node has an upper and lower limit for its physical computing resources: $S_{m\_min}^{CPU}$ = 0 and $S_{\mathrm{m}\,max}^{CPU}$ =30 GHz; $S_{m\_min}^{MEM}$ = 0 and $S_{m\_max}^{MEM}$ =125 GB; and $S_{m\_min}^{STR}$ = 0 and $S_{m\_max}^{STR}$ =1000 GB. Each MEC has a processing rate $\mu_m$range of 200 Mbps and can host a limited number of VNFs without degrading the QoS depending on the BW requirement for VNFs. ETSI MEC recommends deploying edge nodes to existing base stations since it is cost-efficient and vendor-agnostic. The maximum predicted SFCs number over the six months for different BW requirements is selected to dimension the physical network. The maximum capacity for each candidate edge node is 200 Gbps. All substrate links have the same transmission delay of 1 ms and a propagation delay of $1\mu sec$.

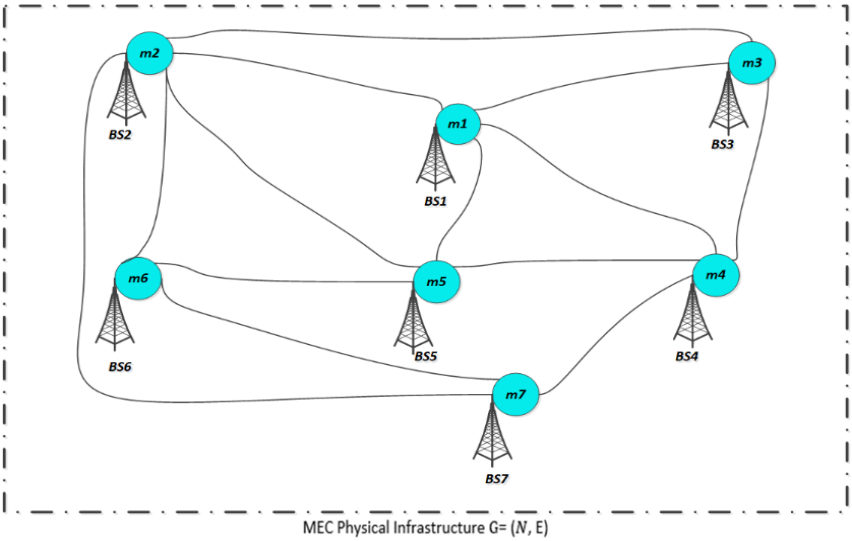


Figure 5.3 Mobile Network Architecture of Candidate MEC Nodes

We assume that edge nodes in proximity to base stations have the capability to host VNFs of various SFC requests on their NFV infrastructure. This deployment model is analogous to containerized applications in data center environments. The maximum bandwidth capacity of each edge node is set to 200 Gbps, and each VNF is assumed to handle a maximum of 1 Gbps traffic without compromising quality of service (QoS). We address the potential scalability issues with respect to embedded paths by providing a matrix that enumerates the feasible physical paths between edge nodes. The MILP problem formulated in our study is solved using IBM ILOG CPLEX Solver, and the resulting solution is compared with other relevant benchmarks.

The results of the NFV-enabled MEC MILP model are presented in Figure 5.4, demonstrating the optimal design and dimensioning solutions. The model illustrates the dimensioned nodes that should be deployed, the resource utilization, and the shared virtual instances installed on the deployed MEC nodes. The main goal of this model is to provide services to mobile devices while ensuring that delay requirements for different SFC requests are met with a 100% acceptance ratio. It is worth noting that the predicted SFC requests at different base stations represent the maximum request number over a period of six months, based on the dataset utilized. The MILP approach proposed in this study serves long-term planning purposes for the mobile operator.

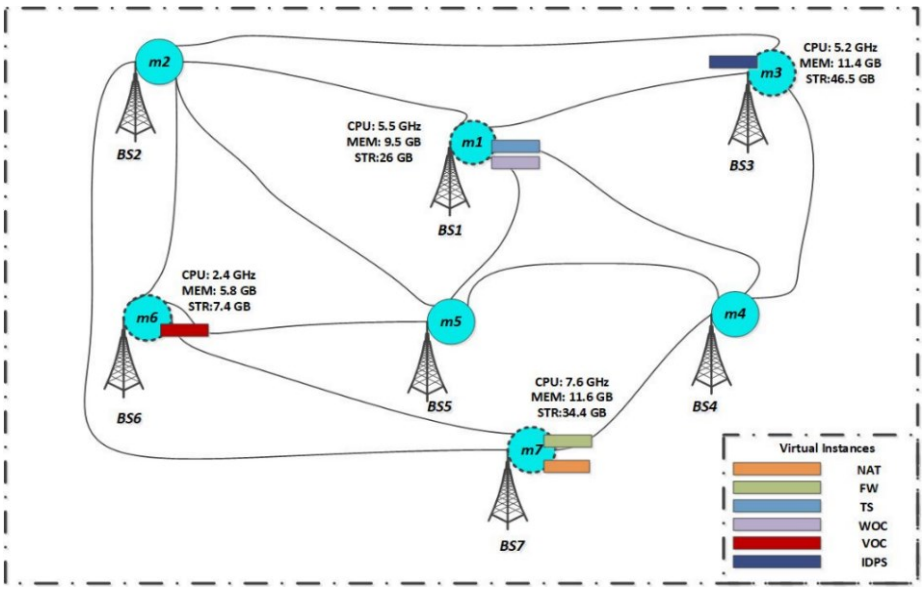


Figure 5.4 NFV-enabled MEC Model

5.3.1 MILP Approach Evaluation Results

The present section entails a comparative performance analysis of the one-shot MILP approach with various benchmarks enlisted in Section 4.9. Figure 5.5 displays the comparative assessment of the MILP model versus the cloud-only benchmark approach in two variants. The first variant pertains to measuring the acceptance rate while keeping the same resource limits, whereas the second variant entails measuring the dimensioning costs while ensuring a 100% acceptance rate and increasing the resource limits. The findings of this evaluation serve as a yardstick to gauge the effectiveness and efficiency of the proposed approach and its superiority over existing benchmarks.

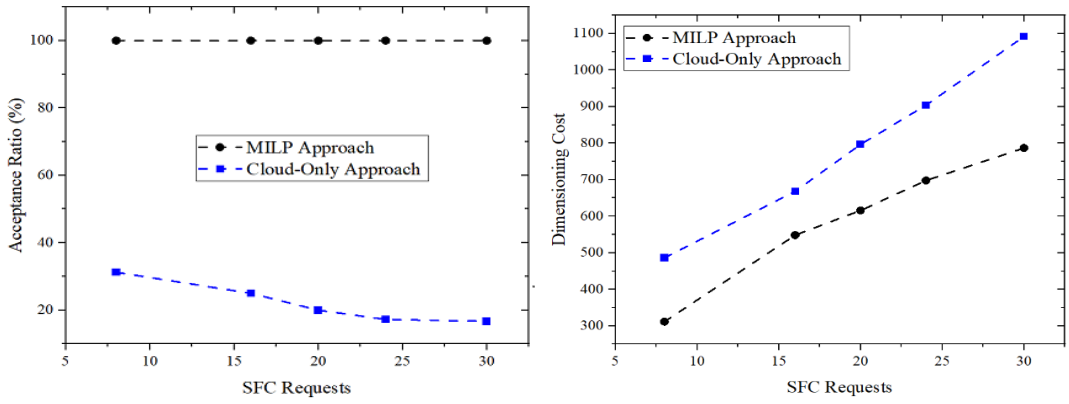


Figure 5.5 MILP vs. Cloud-only Approach Comparison: Acceptance ratio and Dimensioning Cost

The figure on the left shows that when small edge nodes are deployed in the cloud data center, the acceptance rate cannot exceed 31.25% due to the extended network delay incurred in the processing request time hampers the ability to meet the desired QoS standards. To boost the acceptance rate, it is necessary to enhance the physical link bandwidth and deploy high-end servers in the cloud data center to ensure that real-time response requests are fulfilled within the stipulated delay threshold, thereby maintaining high QoS levels. In the scenario illustrated in the right panel of the figure, an increase in dimensioning costs is observed when deploying edge nodes in the cloud with high specifications and bandwidth capacity.

Figure 5.6 demonstrates the benefits of hosting SFCs at the mobile edge nodes instead of the remote clouds in terms of end-to-end latency. In the cloud-only scenario, all the SFCs are assigned to the cloud data center, while in the machine-driven NFV-enabled MEC scenario, SFCs are assigned to the edge nodes. Using the same network topology proposed in Figure 5.4, we measured the E2E latency for different SFC types received by the system at different time slots for both scenarios. We used the ML model output to predict the number of SFCs received at a given time. The cloud-only scenario results in an average latency of 20 ms per VNF and 100 ms per SFC, assuming each SFC has 5 VNFs (20 ms * 5 = 100 ms). From the graph, we could see the reduction in E2E latency generated by the optimal placement of SFCs on the edge network for different service chains while guaranteeing the latency threshold.

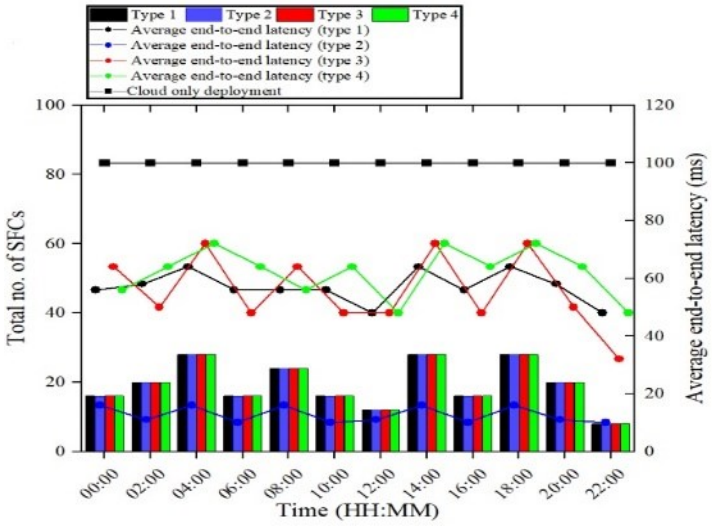


Figure 5.6 E2E Latency of MILP Model for different SFC Types

Figure 5.7 portrays the advantageous impact of implementing the auto-scaling

mechanism into the dimensioning process. The graph illustrates the augmentation in the dimensioning of the deployed NFV-enabled MEC system for the no-scale benchmark scenario. This is attributed to the fact that virtual instances remain installed on the edge server, even when idle, causing an increase in cost. By contrast, the auto-scaling approach dynamically manages resource allocation and virtual instances, leading to a more efficient and cost-effective solution.

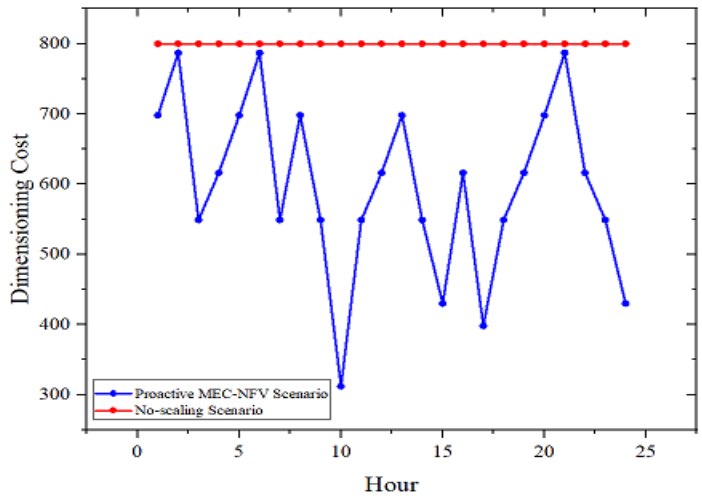


Figure 5.7 Dimensioning Cost: Comparison of Proactive MEC-NFV vs. No-Scaling Approach during the Whole Day

An evaluation of the performance comparison between the proposed MILP model and the no v-instance sharing benchmark approach in terms of dimensioning cost is also conducted. The dimensioning cost is measured for both approaches under different network topologies with different configurations, including the number of edge nodes and possible embedding path matrices. The evaluation also examines the impact of changing the number of predicted SFCs on the dimensioning cost for three candidate edge node numbers (M=2, M=4, and M=7), while guaranteeing a 100% acceptance rate for different scenarios.

Figure 5.8 demonstrates the performance of the MILP model and the no v-instance sharing benchmark approach in terms of dimensioning cost. The results show that the dimensioning cost is significantly lower for the proposed MILP model than for the no v-instance sharing benchmark approach, indicating the benefit of introducing the auto-scaling mechanism into the dimensioning process. The graphs in Figure 5.8 further illustrate how the dimensioning cost changes for different edge node numbers and predicted SFCs. Notably, the dimensioning cost increases for the no v-instance sharing scenario as both the virtual instance activation license

and the virtual instances overheads increase. It is worth mentioning that the virtual instances are installed on all edge nodes that process SFC requests, with all instances $i \in I_t$ types considered.

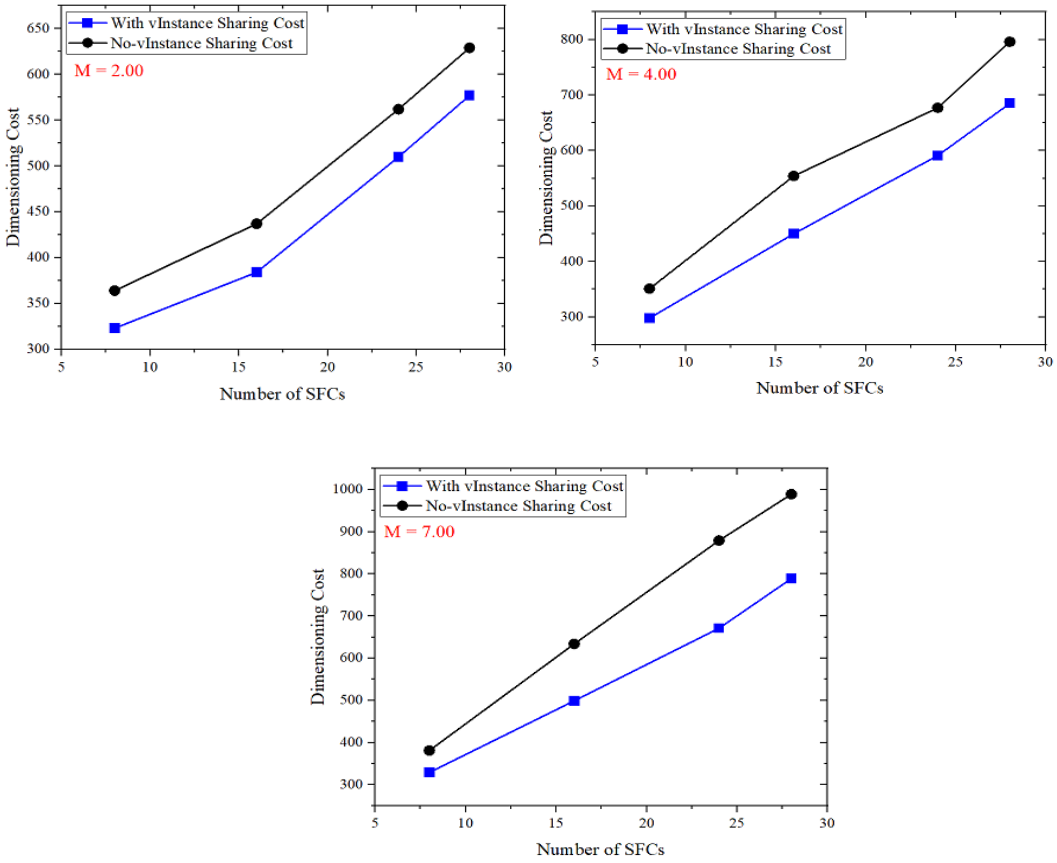


Figure 5.8 Dimensioning Cost: Comparison of MILP vs. No v-Instance Sharing Approach at different SFC requests number

Next, the Affinity-based simulated annealing (ABSA) heuristic approach [129] is compared to our MILP one-shot approach. To our knowledge, the problem investigated in [129] is the most similar to this work. However, there are some differences in the proposed model and solution between both works: (1) the objective function of paper [129] does not take into account the setup and installation cost for the physical resources while our work does ;(2) their model does not consider v-instance sharing and assumes that all v-instances types should be installed on the edge nodes; (3) in [129], the authors considered a constant delay for the nodes and links and did not take into account the queueing delay. In [129], the authors used a heuristic approach for cost-optimized delay-aware placement of VNFs along with a set of greedy approaches. The small network topology shown in Figure 5.9 was then used for the simulation

experiment to compare the proposed solution performance against the ABSA approach. In this topology, all network links' bandwidth equals 1000 MB. The CPU capacity of all the nodes is randomly chosen in [220, 260] cores. The storage capacity of all nodes is equal to 4000 GB, and the memory capacity is 1000 GB.

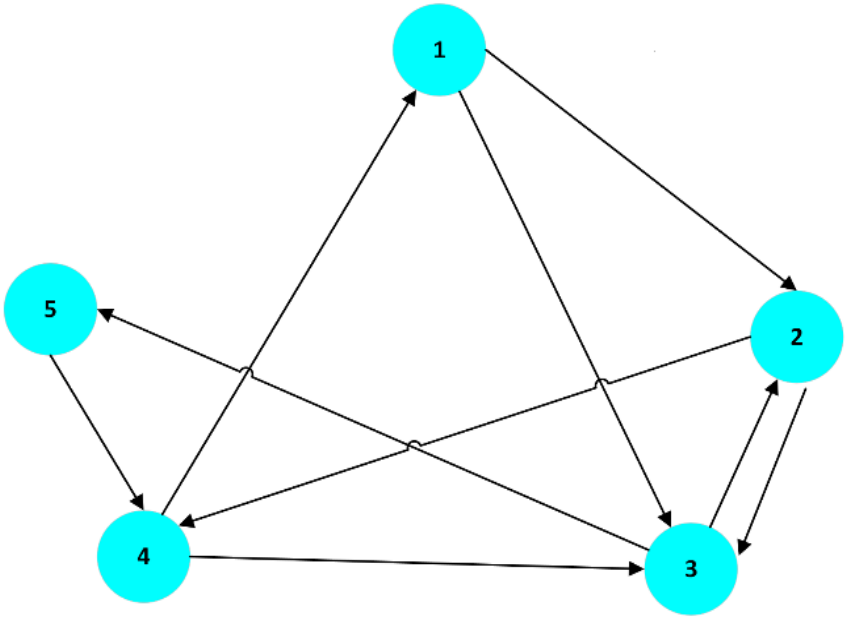


Figure 5.9 Proposed network topology used in ABSA Comparison[129]

Tables 5.6 and 5.7 show the details of input SFCs and VNF types used in the topology shown in Figure 5.9. In these simulations, in addition to the gain, which is the main objective of our problem, we also evaluated the results in terms of acceptance rate. Figure 5.10 and Figure 5.11 show the average gain and acceptance of MILP one-shot and ABSA approaches with respect to the number of predicted SFCs. Although the acceptance rate of our proposed solution is a bit lower than ABSA at the beginning, it perfectly accepts a subset demands of SFCs that make more gain, which is justified by the considerable gap between our approach and ABSA in Figure 5.10.

As the number of the predicted SFCs increases, the average gain of both algorithms increases as well but at different rates. As the number of SFCs increases, the required virtual instances capacity increases, leading to more instances sharing possibilities. In the MILP one-shot approach, the instance cost breaks down among multiple chains by exploiting this possibility. Consequently, the average cost of each SFC decreases, and as a result, the average gain increases. On the other hand, ABSA does not share the instances that lead to a high cost and

a lower gain.

Table 5.6 The Setting of the SFCs Used for Comparison with ABSA

Parameter	Value
VNF Types	Randomly in [1, 2, 3, 4]
VNF Number per SFC	3 VNFs
Virtual Link BW	100 MB
E2E delay threshold /SFC	3 μs

Table 5.7 The Setting of the VNFs Used for Comparison with ABSA

Parameter	Value per Type			
	1	2	3	4
CPU (cores)	4	7	6	5
Storage (GB)	90	90	120	100
Memory (GB)	16	24	32	24
Arrival Rate	150	175	125	150
v-Instance Activation fee	500	600	700	500

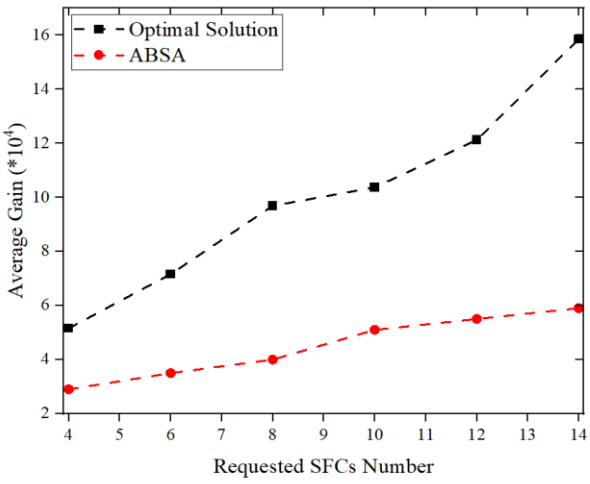


Figure 5.10 The Average Gain of Optimal Solution and ABSA

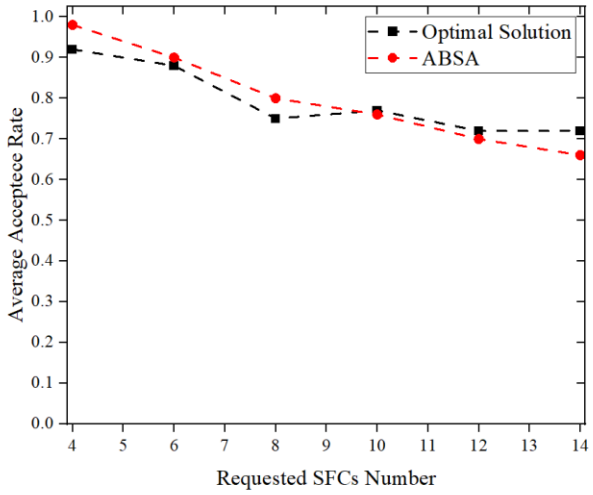


Figure 5.11 The Average Acceptance Ratio of Optimal Solution and ABSA

5.4 Benders Decomposition Approach Evaluation

In this section, we conducted a time and cost comparison between the NFV-based MEC MILP model (MEC-SFC-MILP) and the benders decomposition model (MEC-SFC-BD). Both models were solved using IBM CPLEX CONCRETE Solver on a machine with i7 Intel(R) Core 2.5 GHz CPU and 16GB of RAM. Moreover, we compared the results with the heuristics MEC-SFC-Greedy approach discussed in section 4.9 using the same network configuration.

We designed a MEC network infrastructure with 15 candidate edge node locations to check the system scalability as the number of predicted SFC requests changes. Our network topology has the same edge nodes computation capacities and virtual functions requirements as mentioned in the previous section (section 5.3). The only difference is that we added the virtual path complexity factor where a virtual link $k \in K_f$ can be mapped to up to: |M| x |M| x |Π| possible physical paths and apply relaxations listed in section 4.8.1 when solving both the MILP and BD problems.

First, we are most interested in whether MEC-SFC-BD can approximate the optimal MILP model in cost with a substantial decrease in time. Table 5.8 shows that MEC-SFC-BD attains a near-optimal design and dimensioning cost of the MEC Infrastructure, with a cost difference below 1% for up to 100 SFC requests.

Table 5.8: Cost Comparison in percentage between MEC-SFC-MILP and MEC-SFC-BD

Number of SFC requests	MEC-SFC-MILP Cost	MEC-SFC-BD Cost	Gap%	Gap
10	1818.211	1833.977	0.87%	15.766
20	3466.456	3482.961	0.48%	16.505
30	5029.476	5047.857	0.37%	18.381
40	6677.601	6697.260	0.29%	19.659
50	8240.603	8263.641	0.28%	23.038
60	9888.841	9912.430	0.24%	23.589
70	11451.812	11476.69	0.22%	24.878
80	13100.163	13125.002	0.19%	24.839
90	14663.095	14689.768	0.18%	26.673
100	16311.209	16339.655	0.17%	28.446

The absolute difference varies between 15.766 and 28.446, and it does not increase monotonically with increasing the predicted SFC requests, leading us to hypothesize that the percentage difference gradually decreases with the increasing number of requests.

Figure 5.12 shows the significantly reduced cost of MEC-SFC-BD compared to the heuristics MEC-SFC-Greedy. And it provides a near-optimal solution when compared with the MILP one-shot approach. We can notice in this figure that the MILP approach cannot be used to find the dimensioning cost for the highly intensive systems that receive a large number of requests, as it turns into an infeasible solution when trying to find the optimal solution due to the NP-hard problem scalability issues. Benders decomposition approach is suitable for finding the near-optimal solution for highly intensive systems.

The graphs in Figure 5.13 shows another performance comparison related to the solution time among the three approaches: MEC-SFC-MILP, MEC-SFC-BD, and MEC-SFC-Greedy. The benders decomposition approach calculates a dimensioning solution in a significantly reduced time compared to the MILP solution. In Figure 5.13, we executed MEC-SFC-MILP for at most 160 requests as a higher number of SFC requests proved computationally infeasible. We managed MEC-SFC-BD for up to 500 SFC requests within a practical and scalable amount of time with a near-linear time growth. Though MEC-SFC-Greedy is extremely fast, Figure 5.12 shows that the greedy approach is the worst performing in terms of dimensioning cost.

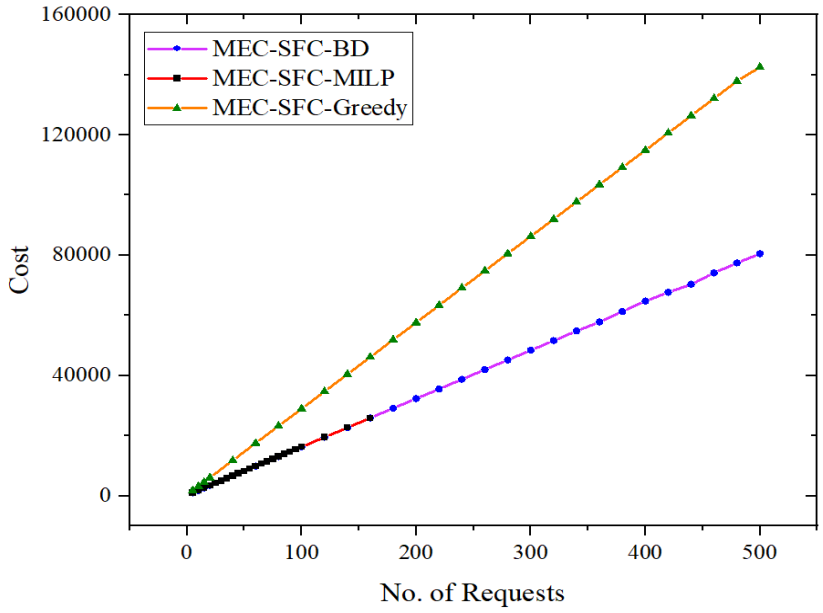
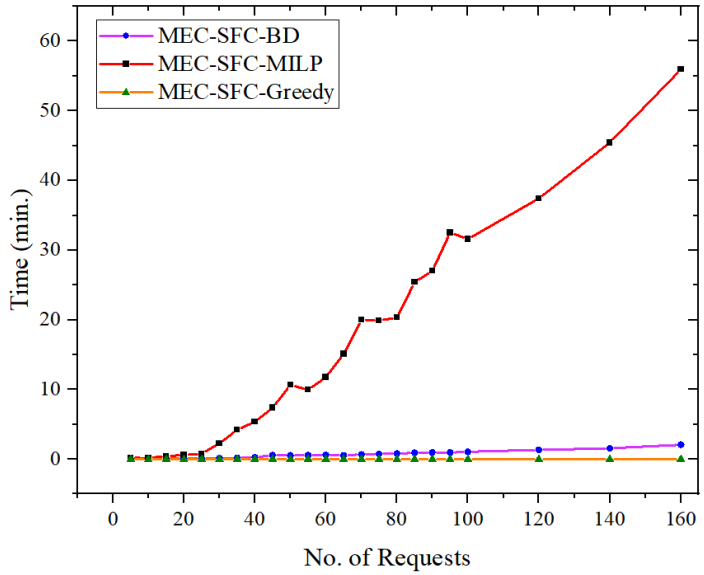
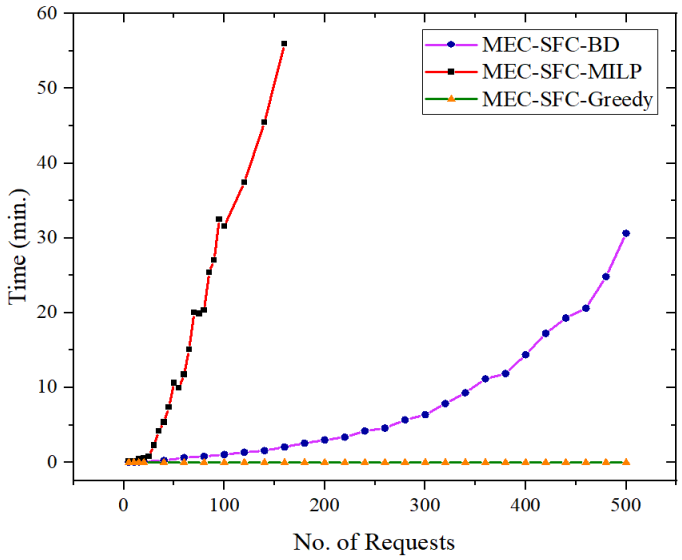


Figure 5.12 Dimensioning cost comparison of MEC-SFC-MILP, MEC-SFC-BD, and MEC-SFC-Greedy

Figure 5.14a is identical to Figure 5.14b, with the x-axis restricted to [0; 160]. This allows us to see the similar time performances between MEC-SFC-BD and MEC-SFC-Greedy for small-scale systems with a low number of requests. We also know the point at which the three models begin to deviate from each other in performance. In both cases, MEC-SFC-MILP has been growing at an alarming rate.



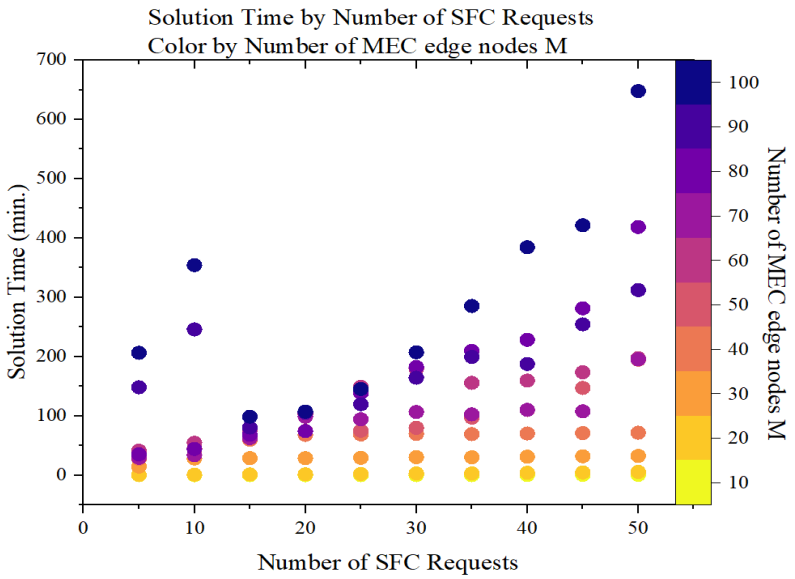
(a) Small Scale



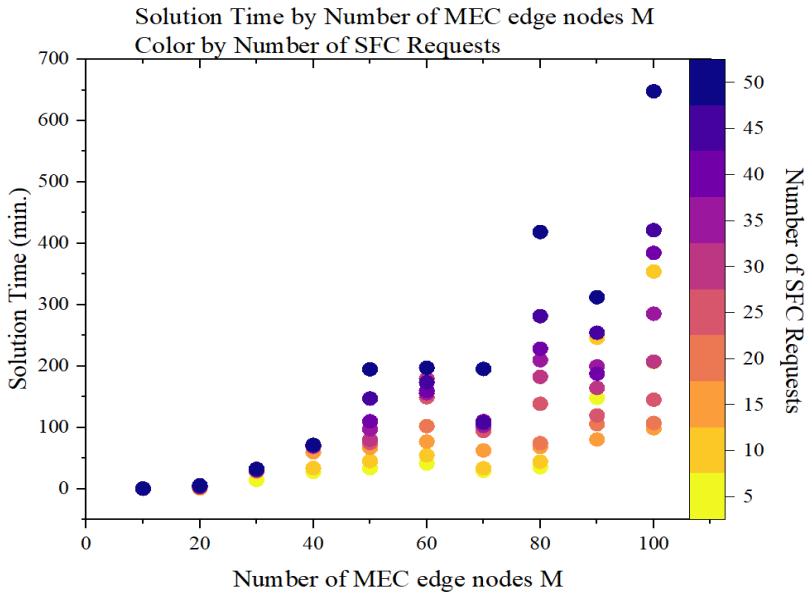
(b) Full Scale

Figure 5.13 Solution time comparison of MEC-SFC-MILP, MEC-SFC-BD, and MEC-SFC-Greedy

To better present the performance evaluation of the proposed MEC-SFC-BD approach, a simulation study was conducted on independent MEC system topologies with varying numbers of MEC candidate nodes and SFC requests, ranging from 10 to 100 and 10 to 50, respectively, with 5 VNFs per request. We assumed that each candidate node has a physical path for all other edge nodes in a specific topology and is co-located with the base station. The solution time for each MEC topology was recorded and presented in Figures 5.14a and 5.14b, with the number of SFC requests or MEC candidates on the x-axis and the other metric as a color map. The results showed that the number of MEC candidates had a significant impact on the solution time, as demonstrated in Figure 5.14b. However, Figure 5.14a revealed that the number of SFC requests had little influence on the solution time, up to a maximum of 50 requests. These findings provide useful insights into the performance efficiency of the MEC-SFC-BD approach and highlight the scalability benefits it offers.



(a) SFC Requests with MEC Nodes colormap



(b) MEC Nodes with SFC Requests colormap

Figure 5.14 Solution time of MEC-SFC-BD by number of SFC Requests and MEC Edge Candidates

5.5 Conclusion

This chapter presents the simulation setup and numerical evaluation results of the proposed agile design and dimensioning approach for NFV-based MEC systems. The

performance of the Proactive SFC Autoscaling sub-problem and the Design and Dimensioning sub-problem is assessed, and a comparison is made between the optimal, benders decomposition, and the greedy benchmark. In the evaluation of the proactive autoscaling sub-problem, the positive impact of using the ML model for predicting SFC requests is demonstrated. The evaluation of the MILP approach highlights its efficiency in deploying a cost-effective and high-acceptance rate NFV-enabled MEC system. The evaluation of the bender decomposition approach shows its performance efficiency for high-scale and high-intensive systems with different MEC topologies. The last chapter concludes the thesis and outlines potential areas for future research.

Chapter 6

CONCLUSION AND FUTURE RESEARCH

6.1 Conclusion

Mobile Edge Computing has emerged as a viable alternative to traditional Cloud processing for mobile applications due to its low latency and improved users' experience. With the increasing demand for critical-mission and computation-intensive applications, it has become necessary to implement MEC infrastructure. In this work, we propose an agile approach for deploying comprehensive MEC infrastructure and suggest high-level steps to deploy a combined MEC-NFV infrastructure for dynamic mobile traffic support. Our approach, called Cost-Efficient Agile Design and Dimensioning of Comprehensive MEC-NFV Infrastructure for Heterogeneous SFCs, aims to minimize dimensioning costs, support heterogeneous compute applications, and ensure that the delay constraints of SFC requests are met. We have considered the virtual resources consumption cost and shareable virtual instances to enhance network resource utilization. To overcome scalability issues in large-scale and compute-intensive networks while maintaining cost-effectiveness, we propose a near-optimal Bender decomposition approach. The simulation results indicate that using machine learning models to predict SFC requests from users and map them into VNFs at a specific time minimizes deployment costs while adhering to the response times for different application types.

Moreover, the evaluation results of our near-optimal Bender decomposition approach demonstrated that the MEC design and dimensioning solutions are within 1% of the optimal solution with significantly reduced computation time for high-scale systems. Furthermore, the Bender Decomposition approach was shown to be more cost-effective than the Greedy heuristics algorithm. The simulation and analysis of various NFV-enabled MEC system topologies revealed that the computation time is strongly correlated with the number of MEC node candidates and moderately correlated with the number of SFC requests. These findings provide insights into the scalability and efficiency of our proposed and comprehensive approach for

MEC infrastructure design and dimensioning.

6.2 Future Work

In our future work, we propose a reinforcement learning (RL) based approach for joint SFC autoscaling and placement to achieve short-term MEC designing purposes. ML and RL are two distinct approaches to building intelligent systems. ML is typically used for prediction tasks, where the goal is to learn a mapping from input data to output labels. RL is used for decision-making tasks, where the goal is to learn a policy that maximizes a reward signal. While both approaches have their strengths and weaknesses, they are not interchangeable. ML is well-suited for tasks where the goal is to predict an outcome based on historical data, such as traffic prediction or workload forecasting in MEC networks. RL, on the other hand, is better suited for tasks where the agent needs to interact with its environment to learn the optimal decision-making policy, such as resource allocation and placement in MEC networks.

Our current work mainly focused on long-term planning and dimensioning for MEC systems by considering the maximum received SFCs at various base station locations over six months when solving the dimensioning problem. The proposed RL approach aims to predict the SFC computation requirements at a specific time, determine the optimal solution for SFC placement, and place/release VNF instances at the edge node according to a threshold-based policy related to physical resource utilization and power consumption at the edge nodes. The performance of the RL module can be evaluated on a real-world mobile network topology.

In summary, the RL approach for MEC designing and planning could be considered by mobile operators to achieve various goals, including:

1) Enhancing the overall system performance and quality of experience (QoE) for end-users by optimizing the placement of SFCs and minimizing the processing delay.
2) Efficient resource utilization and cost reduction by dynamically allocating resources to match the varying workload demand.
3) Improving the system's scalability and flexibility to adapt to changes in network conditions and traffic patterns.
4) Providing better energy efficiency by managing the power consumption of edge nodes based on the workload demand and resource availability.

5) Enhancing network security and reliability by ensuring the availability of resources and avoiding resource depletion.
6) Forecasting the workload on the mobile network elements and entire subnetworks in the short and long term.
7) Meeting the communication networks requirements with ultra-low delays, where a concise identification of traffic in the mobile network without introducing additional delays in the flow is achieved.
8) Efficient allocation of the radio coverage at the mobile operator based on cell load prediction.

In summary, by utilizing RL, we aim to optimize resource allocation and placement decisions in MEC environments, which could lead to more efficient and effective scaling of SFCs. This approach is unique and has the potential to address a practical problem faced by MEC networks (i.e., the need for short-term design goals). By leveraging RL for SFC scaling and placement decisions, the proposed approach could enable MEC networks to adapt to changing demands in real-time, which is essential for meeting short-term goals. Moreover, the proposed approach has the potential to scale to larger and more complex MEC networks. It can adapt to different network configurations and enable more sophisticated decision-making in MEC environments.

REFERENCES

[1] A. Haibeh, Lina A. and Yagoub, Mustapha C. E. and Jarray, "A Survey on Mobile Edge Computing Infrastructure: Design, Resource Management, and Optimization Approaches," *IEEE Access*, vol. 10, pp. 27591–27610, 2022, doi: 10.1109/ACCESS.2022.3152787.

[2] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018, doi: 10.1109/JIOT.2017.2750180.

[3] S. Singh, "Optimize cloud computations using edge computing," in *2017 International Conference on Big Data, IoT and Data Science (BID)*, Dec. 2017, pp. 49–53, doi: 10.1109/BID.2017.8336572.

[4] P. Demestichas *et al.*, "5G on the Horizon: Key Challenges for the Radio-Access Network," *IEEE Veh. Technol. Mag.*, vol. 8, no. 3, pp. 47–53, Sep. 2013, doi: 10.1109/MVT.2013.2269187.

[5] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, Jan. 2016, pp. 1–8, doi: 10.1109/ISCO.2016.7727082.

[6] N.-T. Dinh and Y. Kim, "An Efficient Availability Guaranteed Deployment Scheme for IoT Service Chains over Fog-Core Cloud Networks," *Sensors*, vol. 18, no. 11, p. 3970, Nov. 2018, doi: 10.3390/s18113970.

[7] "Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges & Call for Action," *ETSI, Sophia Antipolis, France*, 2012.

[8] A. Leivadeas, G. Kesidis, M. Ibnkahla, and I. Lambadaris, "VNF Placement Optimization at the Edge and Cloud †," *Futur. Internet*, vol. 11, no. 3, p. 69, Mar. 2019, doi: 10.3390/fi11030069.

[9] S. Wang, M. Chen, X. Liu, C. Yin, S. Cui, and H. Vincent Poor, "A Machine Learning Approach for Task and Resource Allocation in Mobile-Edge Computing-Based Networks,"

IEEE Internet Things J., vol. 8, no. 3, pp. 1358–1372, Feb. 2021, doi: 10.1109/JIOT.2020.3011286.

[10] N. Hassan, K.-L. A. Yau, and C. Wu, "Edge Computing in 5G: A Review," *IEEE Access*, vol. 7, pp. 127276–127289, 2019, doi: 10.1109/ACCESS.2019.2938534.

[11] M. P. et Al., "Mobile-Edge Computing Introductory Technical White Paper," 2014.

[12] L. Bittencourt *et al.*, "The Internet of Things, Fog and Cloud continuum: Integration and challenges," *Internet of Things*, vol. 3–4, pp. 134–155, Oct. 2018, doi: 10.1016/j.iot.2018.09.005.

[13] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017, doi: 10.1109/COMST.2017.2682318.

[14] N. Takahashi, H. Tanaka, and R. Kawamura, "Analysis of Process Assignment in Multi-tier mobile Cloud Computing and Application to Edge Accelerated Web Browsing," in *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, Mar. 2015, pp. 233–234, doi: 10.1109/MobileCloud.2015.23.

[15] J. Dolezal, Z. Becvar, and T. Zeman, "Performance evaluation of computation offloading from mobile device to the edge of mobile network," in *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*, Oct. 2016, pp. 1–7, doi: 10.1109/CSCN.2016.7785153.

[16] O. Salman, I. Elhajj, A. Kayssi, and A. Chehab, "Edge computing enabling the Internet of Things," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Dec. 2015, pp. 603–608, doi: 10.1109/WF-IoT.2015.7389122.

[17] T. M. Ayenew, D. Xenakis, N. Passas, and L. Merakos, "Cooperative Content Caching in MEC-Enabled Heterogeneous Cellular Networks," *IEEE Access*, vol. 9, pp. 98883–98903, 2021, doi: 10.1109/ACCESS.2021.3095356.

[18] Z. Zhang *et al.*, "Foresight Tackling Obesities: Future Choices – Project report," *Int. J.*

Obes. Relat. Metab. Disord., 2010.

[19] F. Lobillo *et al.*, "An architecture for mobile computation offloading on cloud-enabled LTE small cells," in *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Apr. 2014, pp. 1–6, doi: 10.1109/WCNCW.2014.6934851.

[20] S. Wang et al, "Mobile micro-cloud: Application classification, mapping, and deployment," *Annu. Fall Meet. ITA*, pp. 1–7, 2014.

[21] K. Wang, M. Shen, J. Cho, A. Banerjee, J. Van der Merwe, and K. Webb, "MobiScud," in *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, Aug. 2015, pp. 19–24, doi: 10.1145/2785971.2785979.

[22] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Netw.*, vol. 27, no. 5, pp. 12–19, Sep. 2013, doi: 10.1109/MNET.2013.6616110.

[23] Jingchu Liu, Tao Zhao, Sheng Zhou, Yu Cheng, and Zhisheng Niu, "CONCERT: a cloud-based architecture for next-generation cellular systems," *IEEE Wirel. Commun.*, vol. 21, no. 6, pp. 14–22, Dec. 2014, doi: 10.1109/MWC.2014.7000967.

[24] A. Takeda, T. Kimura, and K. Hirata, "Evaluation of edge cloud server placement for edge computing environments," in *2019 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*, May 2019, pp. 1–2, doi: 10.1109/ICCE-TW46550.2019.8991970.

[25] G. Fodor *et al.*, "5G New Radio for Automotive, Rail, and Air Transport," *IEEE Commun. Mag.*, vol. 59, no. 7, pp. 22–28, Jul. 2021, doi: 10.1109/MCOM.001.2001106.

[26] H. Haile, K.-J. Grinnemo, S. Ferlin, P. Hurtig, and A. Brunstrom, "End-to-end congestion control approaches for high throughput and low delay in 4G/5G cellular networks," *Comput. Networks*, vol. 186, p. 107692, Feb. 2021, doi: 10.1016/j.comnet.2020.107692.

[27] S. Gangakhedkar, H. Cao, A. R. Ali, K. Ganesan, M. Gharba, and J. Eichinger, "Use Cases, Requirements and Challenges of 5G Communication for Industrial Automation," in *2018*

IEEE International Conference on Communications Workshops (ICC Workshops), May 2018, pp. 1–6, doi: 10.1109/ICCW.2018.8403588.

[28] F.-C. Kuo, F. A. Zdarsky, J. Lessmann, and S. Schmid, “Cost-Efficient Wireless Mobile Backhaul Topologies: An Analytical Study,” in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, Dec. 2010, pp. 1–5, doi: 10.1109/GLOCOM.2010.5683870.

[29] “SDN helps velocity in Big Data,” in *Big Data and Software Defined Networks*, Institution of Engineering and Technology, 2018, pp. 207–227.

[30] K. Antevski, C. J. Bernardos, L. Cominardi, A. de la Oliva, and A. Mourad, “On the integration of NFV and MEC technologies: architecture analysis and benefits for edge robotics,” *Comput. Networks*, vol. 175, p. 107274, Jul. 2020, doi: 10.1016/j.comnet.2020.107274.

[31] S. K. et Al., “MEC in 5G Networks,” *ETSI White Paper*, 2018.

[32] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A Survey on Mobile Edge Computing: The Communication Perspective,” *IEEE Commun. Surv. Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017, doi: 10.1109/COMST.2017.2745201.

[33] B. Varghese and R. Buyya, “Next generation cloud computing: New trends and research directions,” *Futur. Gener. Comput. Syst.*, vol. 79, pp. 849–861, Feb. 2018, doi: 10.1016/j.future.2017.09.020.

[34] L. F. Bittencourt, M. M. Lopes, I. Petri, and O. F. Rana, “Towards Virtual Machine Migration in Fog Computing,” in *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, Nov. 2015, pp. 1–8, doi: 10.1109/3PGCIC.2015.85.

[35] Q.-V. Pham, T. Leanh, N. H. Tran, B. J. Park, and C. S. Hong, “Decentralized Computation Offloading and Resource Allocation for Mobile-Edge Computing: A Matching Game Approach,” *IEEE Access*, vol. 6, pp. 75868–75885, 2018, doi: 10.1109/ACCESS.2018.2882800.

[36] Q.-V. Pham *et al.*, “A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art,” *IEEE Access*, vol. 8, pp. 116974–117017, 2020, doi: 10.1109/ACCESS.2020.3001277.

[37] Nak Woon Sung, Ngoc-Thai Pham, T. Huynh, and Won-Joo Hwang, “Predictive Association Control for Frequent Handover Avoidance in Femtocell Networks,” *IEEE Commun. Lett.*, vol. 17, no. 5, pp. 924–927, May 2013, doi: 10.1109/LCOMM.2013.031913.130127.

[38] Y. Dong, Z. Chen, P. Fan, and K. Ben Letaief, “Mobility-Aware Uplink Interference Model for 5G Heterogeneous Networks,” *IEEE Trans. Wirel. Commun.*, vol. 15, no. 3, pp. 2231–2244, Mar. 2016, doi: 10.1109/TWC.2015.2500566.

[39] S. K. et Al., “MEC in 5G Networks,” 2018.

[40] L. N. T. Huynh, Q.-V. Pham, X.-Q. Pham, T. D. T. Nguyen, M. D. Hossain, and E.-N. Huh, “Efficient Computation Offloading in Multi-Tier Multi-Access Edge Computing Systems: A Particle Swarm Optimization Approach,” *Appl. Sci.*, vol. 10, no. 1, p. 203, Dec. 2019, doi: 10.3390/app10010203.

[41] E. Ahmed and M. H. Rehmani, “Mobile Edge Computing: Opportunities, solutions, and challenges,” *Futur. Gener. Comput. Syst.*, vol. 70, pp. 59–63, May 2017, doi: 10.1016/j.future.2016.09.015.

[42] M. Jia, J. Cao, and W. Liang, “Optimal Cloudlet Placement and User to Cloudlet Allocation in Wireless Metropolitan Area Networks,” *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Oct. 2017, doi: 10.1109/TCC.2015.2449834.

[43] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, “Efficient Algorithms for Capacitated Cloudlet Placements,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 2866–2880, Oct. 2016, doi: 10.1109/TPDS.2015.2510638.

[44] H. Xiang *et al.*, “An Adaptive Cloudlet Placement Method for Mobile Applications over GPS Big Data,” in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2016,

pp. 1–6, doi: 10.1109/GLOCOM.2016.7841576.

[45] X. Wang, M. Razo, M. Tacca, and A. Fumagalli, “Planning and online resource allocation for the multi-resource cloud infrastructure,” in *2014 IEEE International Conference on Communications (ICC)*, Jun. 2014, pp. 2938–2943, doi: 10.1109/ICC.2014.6883771.

[46] Y. Li and S. Wang, “An Energy-Aware Edge Server Placement Algorithm in Mobile Edge Computing,” in *2018 IEEE International Conference on Edge Computing (EDGE)*, Jul. 2018, pp. 66–73, doi: 10.1109/EDGE.2018.00016.

[47] F. Zeng, Y. Ren, X. Deng, and W. Li, “Cost-Effective Edge Server Placement in Wireless Metropolitan Area Networks,” *Sensors*, vol. 19, no. 1, p. 32, Dec. 2018, doi: 10.3390/s19010032.

[48] P. Zhao and G. Dan, “Joint Resource Dimensioning and Placement for Dependable Virtualized Services in Mobile Edge Clouds,” *IEEE Trans. Mob. Comput.*, vol. 21, no. 10, pp. 3656–3669, Oct. 2022, doi: 10.1109/TMC.2021.3060118.

[49] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. Shen, “Cost-Efficient Resource Provisioning for Dynamic Requests in Cloud Assisted Mobile Edge Computing,” *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 968–980, Jul. 2021, doi: 10.1109/TCC.2019.2903240.

[50] D. Ta, S. Zhou, W. Cai, X. Tang, and R. Ayani, “Network-Aware Server Placement for Highly Interactive Distributed Virtual Environments,” in *2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, Oct. 2008, pp. 95–102, doi: 10.1109/DS-RT.2008.31.

[51] S. K. Kasi *et al.*, “Heuristic Edge Server Placement in Industrial Internet of Things and Cellular Networks,” *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10308–10317, Jul. 2021, doi: 10.1109/JIOT.2020.3041805.

[52] Z. Gao, L. Yang, and Y. Dai, “Large-Scale Computation Offloading Using a Multi-Agent Reinforcement Learning in Heterogeneous Multi-access Edge Computing,” *IEEE Trans.*

Mob. Comput., pp. 1–1, 2022, doi: 10.1109/TMC.2022.3141080.

[53] K. Cao, L. Li, Y. Cui, T. Wei, and S. Hu, “Exploring Placement of Heterogeneous Edge Servers for Response Time Minimization in Mobile Edge-Cloud Computing,” *IEEE Trans. Ind. Informatics*, vol. 17, no. 1, pp. 494–503, Jan. 2021, doi: 10.1109/TII.2020.2975897.

[54] A. Ceselli, M. Premoli, and S. Secci, “Mobile Edge Cloud Network Design Optimization,” *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1818–1831, Jun. 2017, doi: 10.1109/TNET.2017.2652850.

[55] K. T. Selvi and R. Thamilselvan, “Dynamic Resource Allocation for SDN and Edge Computing based 5G Network,” in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, Feb. 2021, pp. 19–22, doi: 10.1109/ICICV50876.2021.9388468.

[56] S. Josilo and G. Dan, “Joint Management of Wireless and Computing Resources for Computation Offloading in Mobile Edge Clouds,” *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1507–1520, Oct. 2021, doi: 10.1109/TCC.2019.2923768.

[57] B. Addis, D. Belabed, M. Bouet, and S. Secci, “Virtual network functions placement and routing optimization,” in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, Oct. 2015, pp. 171–177, doi: 10.1109/CloudNet.2015.7335301.

[58] S. Pasteris, S. Wang, M. Herbster, and T. He, “Service Placement with Provable Guarantees in Heterogeneous Edge Computing Systems,” in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, Apr. 2019, pp. 514–522, doi: 10.1109/INFOCOM.2019.8737449.

[59] N. Kiran, X. Liu, S. Wang, and C. Yin, “VNF Placement and Resource Allocation in SDN/NFV-Enabled MEC Networks,” in *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Apr. 2020, pp. 1–6, doi: 10.1109/WCNCW48565.2020.9124910.

[60] A. Garcia-Saavedra, G. Iosifidis, X. Costa-Perez, and D. J. Leith, “Joint Optimization of

Edge Computing Architectures and Radio Access Networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2433–2443, Nov. 2018, doi: 10.1109/JSAC.2018.2874142.

[61] Z. Cao, H. Zhang, and B. Liu, "Performance and Stability of Application Placement in Mobile Edge Computing System," in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, Nov. 2018, pp. 1–8, doi: 10.1109/PCCC.2018.8710802.

[62] S. Josilo and G. Dan, "Selfish Decentralized Computation Offloading for Mobile Cloud Computing in Dense Wireless Networks," *IEEE Trans. Mob. Comput.*, vol. 18, no. 1, pp. 207–220, Jan. 2019, doi: 10.1109/TMC.2018.2829874.

[63] S. Josilo and G. Dan, "Computation Offloading Scheduling for Periodic Tasks in Mobile Edge Computing," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 667–680, Apr. 2020, doi: 10.1109/TNET.2020.2968209.

[64] L. Zhao and J. Liu, "Optimal Placement of Virtual Machines for Supporting Multiple Applications in Mobile Edge Networks," *IEEE Trans. Veh. Technol.*, pp. 1–1, 2018, doi: 10.1109/TVT.2018.2808171.

[65] M. F. Maia, Adyson M. and Ghamri-Doudane, Yacine and Vieira, Dario and de Castro, "Optimized Placement of Scalable IoT Services in Edge Computing," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 189–197.

[66] B. Brik, P. A. Frangoudis, and A. Ksentini, "Service-Oriented MEC Applications Placement in a Federated Edge Cloud Architecture," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, Jun. 2020, pp. 1–6, doi: 10.1109/ICC40277.2020.9148814.

[67] D. Harris, J. Naor, and D. Raz, "Latency Aware Placement in Multi-access Edge Computing," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, Jun. 2018, pp. 132–140, doi: 10.1109/NETSOFT.2018.8459997.

[68] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service

migration and workload scheduling in edge-clouds," *Perform. Eval.*, vol. 91, pp. 205–228, Sep. 2015, doi: 10.1016/j.peva.2015.06.013.

[69] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the Starting Line: Joint Network Selection and Service Placement for Mobile Edge Computing," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, Apr. 2019, pp. 1459–1467, doi: 10.1109/INFOCOM.2019.8737543.

[70] Y. Zhang, L. Jiao, J. Yan, and X. Lin, "Dynamic Service Placement for Virtual Reality Group Gaming on Mobile Edge Cloudlets," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1881–1897, Aug. 2019, doi: 10.1109/JSAC.2019.2927071.

[71] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "Energy-Aware Application Placement in Mobile Edge Computing: A Stochastic Optimization Approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 909–922, Apr. 2020, doi: 10.1109/TPDS.2019.2950937.

[72] T. Bahreini and D. Grosu, "Efficient Algorithms for Multi-Component Application Placement in Mobile Edge Computing," *IEEE Trans. Cloud Comput.*, pp. 1–1, 2020, doi: 10.1109/TCC.2020.3038626.

[73] S. Wang, M. Zafer, and K. K. Leung, "Online Placement of Multi-Component Applications in Edge Computing Environments," *IEEE Access*, vol. 5, pp. 2514–2533, 2017, doi: 10.1109/ACCESS.2017.2665971.

[74] M. Wang, B. Cheng, W. Feng, and J. Chen, "An Efficient Service Function Chain Placement Algorithm in a MEC-NFV Environment," in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2019, pp. 1–6, doi: 10.1109/GLOBECOM38437.2019.9013235.

[75] M. A. Khoshkholghi, J. Taheri, D. Bhamare, and A. Kassler, "Optimized Service Chain Placement Using Genetic Algorithm," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, Jun. 2019, pp. 472–479, doi: 10.1109/NETSOFT.2019.8806644.

[76] I. Jang, D. Suh, S. Pack, and G. Dan, "Joint Optimization of Service Function Placement and Flow Distribution for Service Function Chaining," *IEEE J. Sel. Areas Commun.*, vol. 35,

no. 11, pp. 2532–2541, Nov. 2017, doi: 10.1109/JSAC.2017.2760162.

[77] S. Dutta, T. Taleb, and A. Ksentini, “QoE-aware elasticity support in cloud-native 5G systems,” in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6, doi: 10.1109/ICC.2016.7511377.

[78] G. A. Carella, M. Pauls, L. Grebe, and T. Magedanz, “An extensible Autoscaling Engine (AE) for Software-based Network Functions,” in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov. 2016, pp. 219–225, doi: 10.1109/NFV-SDN.2016.7919501.

[79] M. K. Mohan Murthy, H. A. Sanjay, and J. Anand, “Threshold Based Auto Scaling of Virtual Machines in Cloud Environment,” 2014, pp. 247–256.

[80] K.-C. Hung, Che-Lun and Hu, Yu-Chen and Li, “Auto-Scaling Model for Cloud Computing System,” *Int. J. Hybrid Inf. Technol.*, vol. 5, pp. 181–186, 2012.

[81] C. H. T. Arteaga, F. Risso, and O. M. C. Rendon, “An adaptive scaling mechanism for managing performance variations in network functions virtualization: A case study in an NFV-based EPC,” in *2017 13th International Conference on Network and Service Management (CNSM)*, Nov. 2017, pp. 1–7, doi: 10.23919/CNSM.2017.8255982.

[82] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, “A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments,” *J. Grid Comput.*, vol. 12, no. 4, pp. 559–592, Dec. 2014, doi: 10.1007/s10723-014-9314-7.

[83] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, “Machine Learning Meets Computation and Communication Control in Evolving Edge and Cloud: Challenges and Future Perspective,” *IEEE Commun. Surv. Tutorials*, vol. 22, no. 1, pp. 38–67, 2020, doi: 10.1109/COMST.2019.2943405.

[84] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, “Wireless Network Intelligence at the Edge,” *Proc. IEEE*, vol. 107, no. 11, pp. 2204–2239, Nov. 2019, doi: 10.1109/JPROC.2019.2941458.

[85] S. Waidande, “A Literature Survey on Scaling Approaches for VNF in NFV Monitoring,” *Int. Res. J. Eng. Technol.*, vol. 5, 2018.

[86] A. Bilal, T. Tarik, A. Vajda, and B. Miloud, “Dynamic Cloud Resource Scheduling in Virtualized 5G Mobile Systems,” in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2016, pp. 1–6, doi: 10.1109/GLOCOM.2016.7841760.

[87] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba, “Topology-Aware Prediction of Virtual Network Function Resource Requirements,” *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 1, pp. 106–120, Mar. 2017, doi: 10.1109/TNSM.2017.2666781.

[88] S. Ashtari, I. Zhou, M. Abolhasan, N. Shariati, J. Lipman, and W. Ni, “Knowledge-defined networking: Applications, challenges and future work,” *Array*, vol. 14, p. 100136, Jul. 2022, doi: 10.1016/j.array.2022.100136.

[89] C.-W. Huang, C.-T. Chiang, and Q. Li, “A study of deep learning networks on mobile traffic forecasting,” in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Oct. 2017, pp. 1–6, doi: 10.1109/PIMRC.2017.8292737.

[90] L. Fang, X. Cheng, H. Wang, and L. Yang, “Mobile Demand Forecasting via Deep Graph-Sequence Spatiotemporal Modeling in Cellular Networks,” *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3091–3101, Aug. 2018, doi: 10.1109/JIOT.2018.2832071.

[91] C. Qiu, Y. Zhang, Z. Feng, P. Zhang, and S. Cui, “Spatio-Temporal Wireless Traffic Prediction With Recurrent Neural Network,” *IEEE Wirel. Commun. Lett.*, vol. 7, no. 4, pp. 554–557, Aug. 2018, doi: 10.1109/LWC.2018.2795605.

[92] C. Gijón, M. Toril, S. Luna-Ramírez, M. L. Marí-Altozano, and J. M. Ruiz-Avilés, “Long-Term Data Traffic Forecasting for Network Dimensioning in LTE with Short Time Series,” *Electronics*, vol. 10, no. 10, p. 1151, May 2021, doi: 10.3390/electronics10101151.

[93] J. Ahn, J. Lee, S. Park, and H.-S. Park, “Power Efficient Clustering Scheme for 5G Mobile Edge Computing Environment,” *Mob. Networks Appl.*, vol. 24, no. 2, pp. 643–652, Apr.

2019, doi: 10.1007/s11036-018-1164-2.

[94] B. Blanco, I. Taboada, J. O. Fajardo, and F. Liberal, “A Robust Optimization Based Energy-Aware Virtual Network Function Placement Proposal for Small Cell 5G Networks with Mobile Edge Computing Capabilities,” *Mob. Inf. Syst.*, vol. 2017, pp. 1–14, 2017, doi: 10.1155/2017/2603410.

[95] J. Xu, L. Chen, and S. Ren, “Online Learning for Offloading and Autoscaling in Energy Harvesting Mobile Edge Computing,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 3, pp. 361–373, Sep. 2017, doi: 10.1109/TCCN.2017.2725277.

[96] B. Yang, W. K. Chai, G. Pavlou, and K. V. Katsaros, “Seamless Support of Low Latency Mobile Applications with NFV-Enabled Mobile Edge-Cloud,” in *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*, Oct. 2016, pp. 136–141, doi: 10.1109/CloudNet.2016.21.

[97] S. Pan, Z. Zhang, Z. Zhang, and D. Zeng, “Dependency-Aware Computation Offloading in Mobile Edge Computing: A Reinforcement Learning Approach,” *IEEE Access*, vol. 7, pp. 134742–134753, 2019, doi: 10.1109/ACCESS.2019.2942052.

[98] Z. Chen *et al.*, “A Novel Algorithm for NFV Chain Placement in Edge Computing Environments,” in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2018, pp. 1–6, doi: 10.1109/GLOCOM.2018.8647371.

[99] L. Yala, P. A. Frangoudis, and A. Ksentini, “Latency and Availability Driven VNF Placement in a MEC-NFV Environment,” in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2018, pp. 1–7, doi: 10.1109/GLOCOM.2018.8647858.

[100] A. Leivadeas, M. Falkner, I. Lambadaris, M. Ibnkahla, and G. Kesidis, “Balancing Delay and Cost in Virtual Network Function Placement and Chaining,” in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, Jun. 2018, pp. 433–440, doi: 10.1109/NETSOFT.2018.8459956.

[101] Y. Nam, S. Song, and J.-M. Chung, “Clustered NFV Service Chaining Optimization in

Mobile Edge Clouds," *IEEE Commun. Lett.*, vol. 21, no. 2, pp. 350–353, Feb. 2017, doi: 10.1109/LCOMM.2016.2618788.

[102] H. Zhu and C. Huang, "EdgePlace: Availability-aware placement for chained mobile edge applications," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, p. e3504, Nov. 2018, doi: 10.1002/ett.3504.

[103] C. Liang, Y. He, F. R. Yu, and N. Zhao, "Energy-efficient resource allocation in software-defined mobile networks with mobile edge computing and caching," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, May 2017, pp. 121–126, doi: 10.1109/INFCOMW.2017.8116363.

[104] R. Ford, A. Sridharan, R. Margolies, R. Jana, and S. Rangan, "Provisioning low latency, resilient mobile edge clouds for 5G," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, May 2017, pp. 169–174, doi: 10.1109/INFCOMW.2017.8116371.

[105] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate Edge and Cloud Computing With Distributed Deep Learning for Smart City Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8099–8110, Sep. 2020, doi: 10.1109/JIOT.2020.2996784.

[106] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, Latency-Optimal vNF Placement at the Network Edge," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, Apr. 2018, pp. 693–701, doi: 10.1109/INFOCOM.2018.8486021.

[107] Y.-T. Chen and W. Liao, "Mobility-Aware Service Function Chaining in 5G Wireless Networks with Mobile Edge Computing," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–6, doi: 10.1109/ICC.2019.8761306.

[108] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement and resource optimization in NFV and edge computing enabled networks," *Comput. Networks*, vol. 152, pp. 12–24, Apr. 2019, doi: 10.1016/j.comnet.2019.01.036.

[109] J. Wen, C. Ren, and A. K. Sangaiah, “Energy-Efficient Device-to-Device Edge Computing Network: An Approach Offloading Both Traffic and Computation,” *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 96–102, Sep. 2018, doi: 10.1109/MCOM.2018.1701054.

[110] P. Zhao, H. Tian, C. Qin, and G. Nie, “Energy-Saving Offloading by Jointly Allocating Radio and Computational Resources for Mobile Edge Computing,” *IEEE Access*, vol. 5, pp. 11255–11268, 2017, doi: 10.1109/ACCESS.2017.2710056.

[111] Y. Yu, “Mobile edge computing towards 5G: Vision, recent progress, and open challenges,” *China Commun.*, vol. 13, no. Supplement2, pp. 89–99, 2016, doi: 10.1109/CC.2016.7833463.

[112] A. Gupta, B. Jaumard, M. Tornatore, and B. Mukherjee, “A Scalable Approach for Service Chain Mapping With Multiple SC Instances in a Wide-Area Network,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 529–541, Mar. 2018, doi: 10.1109/JSAC.2018.2815298.

[113] S. Bhagavathiperumal, “Auto Scaling of Cloud Resources using Time Series and Machine Learning Prediction,” 2020.

[114] Sguangwang.com, “The Telecom Dataset (Shanghai Telecom),” *The Telecom Dataset*, 2018. http://sguangwang.com/TelecomDataset.html.

[115] T. P. da Silva, A. R. Neto, T. V. Batista, F. C. Delicato, P. F. Pires, and F. Lopes, “Online machine learning for auto-scaling in the edge computing,” *Pervasive Mob. Comput.*, vol. 87, p. 101722, Dec. 2022, doi: 10.1016/j.pmcj.2022.101722.

[116] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[117] F. A. Gers, D. Eck, and J. Schmidhuber, “Applying LSTM to Time Series Predictable through Time-Window Approaches,” 2001, pp. 669–676.

[118] “Theano,” *https://machinelearningmastery.com/introduction-python-deep-learning-library-theano/*. .

[119] “Tensorflow,” *https://www.tensorflow.org/*. .

[120] “Keras 2018,” *https://keras.io/*. .

[121] A. Dwaraki and T. Wolf, “Adaptive Service-Chain Routing for Virtual Network Functions in Software-Defined Networks,” in *Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization*, Aug. 2016, pp. 32–37, doi: 10.1145/2940147.2940148.

[122] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, “Optimal VNF Placement via Deep Reinforcement Learning in SDN/NFV-Enabled Networks,” *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 263–278, Feb. 2020, doi: 10.1109/JSAC.2019.2959181.

[123] D. G. Andersen, *Theoretical approaches to node assignment*. 2014.

[124] K. Aardal and T. Larsson, “A benders decomposition based heuristic for the hierarchical production planning problem,” *Eur. J. Oper. Res.*, vol. 45, no. 1, pp. 4–14, Mar. 1990, doi: 10.1016/0377-2217(90)90151-Z.

[125] E. Kalvelagen, “Benders decomposition with GAMS,” 2002.

[126] J. Repschlaeger, S. Wind, R. Zarnekow, and K. Turowski, “A Reference Guide to Cloud Computing Dimensions: Infrastructure as a Service Classification Framework,” in *2012 45th Hawaii International Conference on System Sciences*, Jan. 2012, pp. 2178–2188, doi: 10.1109/HICSS.2012.76.

[127] S. Jiang and J. Wu, “Multi-resource allocation in cloud data centers: A trade-off on fairness and efficiency,” *Concurr. Comput. Pract. Exp.*, vol. 33, no. 6, Mar. 2021, doi: 10.1002/cpe.6061.

[128] Z. Ye, X. Cao, J. Wang, H. Yu, and C. Qiao, “Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization,” *IEEE Netw.*, vol. 30, no. 3, pp. 81–87, May 2016, doi: 10.1109/MNET.2016.7474348.

[129] A. Fischer, D. Bhamare, and A. Kassler, “On the Construction of Optimal Embedding Problems for Delay-Sensitive Service Function Chains,” in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, Jul. 2019, pp. 1–10, doi:

10.1109/ICCCN.2019.8847151.

[130] A. F. and H. Groenevelt, “The greedy procedure for resource allocation problems: Necessary and sufficient conditions for optimality,” in *Operations research*, 1986, pp. 909–918.

[131] R. Subramanya, Tejas and Baggio, Giovanni and Riggio, “lightMEC: A Vendor-Agnostic Platform for Multi-access Edge Computing,” in *2018 14th International Conference on Network and Service Management (CNSM)*, 2018, pp. 198–204.