



**This electronic thesis or dissertation has been
downloaded from Explore Bristol Research,
<http://research-information.bristol.ac.uk>**

Author:
Bi, Yu

Title:
Resource Allocation for Service Chaining in Multi-layer Edge-Cloud Networks

General rights

Access to the thesis is subject to the Creative Commons Attribution - NonCommercial-No Derivatives 4.0 International Public License. A copy of this may be found at <https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode>. This license sets out your rights and the restrictions that apply to your access to the thesis so it is important you read this before proceeding.

Take down policy

Some pages of this thesis may have been removed for copyright restrictions prior to having it been deposited in Explore Bristol Research. However, if you have discovered material within the thesis that you consider to be unlawful e.g. breaches of copyright (either yours or that of a third party) or any other law, including but not limited to those relating to patent, trademark, confidentiality, data protection, obscenity, defamation, libel, then please contact collections-metadata@bristol.ac.uk and include the following information in your message:

- Your contact details
- Bibliographic details for the item, including a URL
- An outline nature of the complaint

Your claim will be investigated and, where appropriate, the item in question will be removed from public view as soon as possible.

Resource Allocation for Service Chaining in Multi-layer Edge-Cloud Networks

By

YU BI



Department of Electrical and Electronic Engineering
UNIVERSITY OF BRISTOL

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of DOCTOR OF PHILOSOPHY in the Faculty of Engineering.

NOVEMBER 2021

Word count: forty-six thousand nine hundred and sixty

ABSTRACT

Architectural innovation is one of the leading development directions of the telecommunication network. It overcomes barriers left by the traditional network, such as inefficient resource usage, costly network growth, and ossification service levels. Network Function Virtualisation (NFV) plays a crucial role in promoting architecture innovation by disaggregating hardware and software. In NFV-enabled architecture, end-to-end (E2E) network services can be deployed flexibly as ordered Service Function Chains (SFCs). Another critical technology to stimulate architectural innovation is Multi-access Edge Computing (MEC), which brings computing resources to the edge to reduce latency, reduce load, and improve performance and user experience. To gain the advantage of architectural transformation, the resource allocation problem for SFCs has been extensively investigated. However, there is still a gap in serving a new class of advanced high capacity and ultra-low latency services. Motivated by the emergence of diversified and sometimes extreme service Quality-of-Service (QoS) requirements, the necessity for resource management on densely developed but geographically distributed MEC nodes, as well as the difficulty of dealing with complex and rapidly changing network and environment, this thesis focus on solving the resource allocation problem for QoS-aware SFCs in multi-layer edge-cloud networks.

This thesis creates a comprehensive road map for addressing the QoS-aware SFCs resource allocation problem under various scenarios. The developed Mixed Integer Linear Programming (MILP) model can find optimal solutions for small-scale networks. Meanwhile, the proposed heuristic and Deep Reinforcement Learning (DRL) approaches can deliver solutions in a reasonable time for large-scale networks. For centralised control, the designed meta-heuristic and multi-objective DRL algorithm can balance service performance and resource utilisation. For decentralised control, the adopted congestion game model and multi-agent DRL model assure privacy and scalability. For ultra-low latency services, adding an optical layer reduces the transmission and queueing latency on intermediate switches, and proposed algorithms achieve impressive service acceptance performance. In practical scenarios, on the one hand, real test-bed experiments bridge the gap between theoretical and practical algorithms' performance. On the other hand, provided online solutions cater to dynamic and unforeseeable service requests and network environments. The thesis contributions on the resource allocation for SFCs in multi-layer edge-cloud networks are of significant value to unleash the great potential of the evolved telecommunication network.

DEDICATION AND ACKNOWLEDGEMENTS

First and foremost, I would like to convey my gratitude to my supervisor, Prof. Reza Nejati, for all his guidance, support, and encouragement over the last four years. Without his financial support, I would not be able to pursue my PhD degree after completing my postgraduate studies and concentrate on research. Through his profound knowledge, I improved my research and expanded my vision of telecommunication. I would like to thank my second supervisor, Prof. Dimitra Simeonidou, for her professional working style and insights, which influenced me a lot. I would also like to thank my annual progress reviewer Dr. George Oikonomou, for his clear and valuable suggestions on every critical phase of my PhD.

In addition, I would like to thank my colleagues in High Performance Networks group for their generous and helpful support during my PhD. To Dr. Carlos Colman Meixner, for his help with the research problem analysis, mathematical modelling, and paper organisation. To Dr. Fanchao Meng, for his help with the methodologies, research direction, and paper writing skills. To Dr. Monchai Bunyakitanon, for his help on the experiment design and test-bed setup. To Dr. Xenofon Vasilakos, Dr. Rui Wang, Dr. Abubakar Muqaddas, Navdeep Uniyal, and Anderson Bravalheri for their help on the simulation, experiment, and paper writing. To Dr. Shuangyi Yan, Dr. Emilio Hugues Salas, Dr. Yanni Ou, and Dr. Bo Dai, for guiding me on the research and experiments.

Apart from the academics, firstly, I would like to thank my husband, Anthony Siming Chen, for his warm encouragement when I was having difficulties, his assistance with machine learning and proofreading, and his positive attitude towards extracurricular activities, which greatly enriched my daily life. Secondly, I would like to thank my other family members for their suggestions, support, and trust in my education, career, and interests. Thirdly, I would like to thank my friends Cathy, Danny, Becky, Marion, Yue Zong, Aloizio P. Silva, Kaho Chiu, Wenda Li, Zhengguang Gao, Xi Chen, Lida Liu, Hao Song, Xueqing Zhou, Ruizhi Yang, Kote Kondepu, Sam Gunner, Kalyani Rajkumar, Thierno Diallo, Ekin Arabul, etc. Every gathering at the important festivals makes me feel not alone, and they kindly help and support me in daily life.

Finally, I would like to express my sincere gratitude to the University of Bristol. From my master to my PhD, I have benefited much from the study resources, the networking opportunities through various activities, the training courses, and the career services provided by the university. My impressive growth is inseparable from this university, and I will never regret spending six years here.

AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

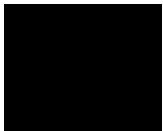
SIGNED:..... DATE:30/11/2021.....

TABLE OF CONTENTS

	Page
List of Tables	xi
List of Figures	xiii
1 Introduction	5
1.1 Background	5
1.1.1 Network Transformation	5
1.1.2 Technologies	8
1.2 Resource Allocation in NFV-enabled Edge-Cloud Networks	13
1.2.1 Motivation	13
1.2.2 Problem Statement and Challenges	15
1.2.3 Research Contribution	19
1.3 Thesis Structure	20
1.4 Publications	21
2 Literature Review	23
2.1 Literature Review on QoS-aware SFCs Placement	23
2.1.1 Single-Objective Optimisation for SFCs Placement	23
2.1.2 Multi-Objective Optimisation for SFCs Placement	25
2.1.3 Reinforcement Learning-assisted SFCs Placement	26
2.1.4 Distributed SFCs Placement	27
2.1.5 SFCs Placement in Optical Networks	29
2.2 Literature Review on QoS-aware SFCs Scheduling	30
2.2.1 Traditional Ways Solving SFCs Scheduling	30
2.2.2 Reinforcement Learning (RL)-assisted SFCs Scheduling	32
2.3 Literature Review on Multi-Objective RL	33
2.4 Literature Review on Multi-Agent RL for Scheduling	35
3 Methodology	41
3.1 Mixed Integer Linear Programming	41

TABLE OF CONTENTS

3.2	Deep Reinforcement Learning	42
3.3	Distributed Control and Congestion Game	44
3.4	Multi-Agent Reinforcement Learning	45
3.5	Management Tools	46
3.5.1	OpenStack	46
3.5.2	Open Source MANO	48
4	Single-Objective Optimisation for SFCs Placement	51
4.1	Introduction	51
4.2	MILP Formulation for Single-Objective SFCs Placement	52
4.2.1	Network Topology and Problem Statement	52
4.2.2	MILP Formulation	54
4.3	MILP Simulation Evaluation	60
4.3.1	Simulation Setup	60
4.3.2	Simulation Results and Analysis	61
4.4	Data Rate-based Heuristic Algorithm	63
4.4.1	Algorithm Design	63
4.4.2	Algorithm Performance Analysis	65
4.5	Summary	68
5	Multi-Objective Optimisation for SFCs Placement	69
5.1	Introduction	69
5.2	MILP Formulation for Multi-Objective SFCs Placement	71
5.2.1	SFCs placement in Edge-Cloud Networks	71
5.2.2	Multi-Objective MILP Formulation	73
5.3	Algorithms for Non-dominated Solutions	76
5.3.1	Heuristic-based GA & NSGA-II Algorithm	77
5.3.2	Constrained Two-Archive Evolutionary Algorithm	79
5.4	Simulation-based Evaluation	80
5.4.1	Simulation Setup	80
5.4.2	Simulation Results and Analysis	82
5.5	Summary	86
6	Multi-objective DRL-assisted SFCs Placement	89
6.1	Introduction	89
6.2	Multi-Objective SFCs Placement Problem Formulation	91
6.2.1	SFCs placement in Edge-Cloud Networks	91
6.2.2	DRL Model Formulation	92
6.3	Multi-Objective DRL Algorithm	96

6.4	Simulation-based Evaluation	98
6.4.1	Simulation Setup	98
6.4.2	Simulation Results and Analysis	99
6.5	Testbed Experimentation-based Evaluation	110
6.5.1	Experiment Description	110
6.5.2	Experiment Results and Analysis	112
6.6	Summary	114
7	Distributed Game Theory-based SFCs Placement	115
7.1	Introduction	115
7.2	Distributed SFCs Placement Problem Formulation	117
7.3	Game Model	118
7.3.1	Congestion Game Formulation	119
7.3.2	Proof of Potential Game	122
7.4	Distributed Algorithm	123
7.5	Simulation-based Evaluation	125
7.5.1	Simulation Setup	125
7.5.2	Simulation Results and Analysis	126
7.6	Experiment-based Evaluation	130
7.7	Summary	132
8	Multi-Agent DRL for SFCs Placement and Scheduling	133
8.1	Introduction	133
8.2	Multi-Agent DRL Formulation for SFCs Scheduling	135
8.2.1	SFCs Placement and Scheduling Formulation	135
8.2.2	DRL Formulation for SFCs Placement and Scheduling	143
8.3	Multi-Agent DRL Algorithm	144
8.4	Simulation-based Evaluation	145
8.4.1	Simulation Setup	145
8.4.2	Simulation Results and Analysis	147
8.5	Summary	150
9	Conclusion and Future Work	153
9.1	Summary and Achievements	153
9.2	Future Work	155
A	Appendix A	159
A.1	Hyperparameters and Training Performance for Multi-Objective DRL	159
A.2	Hyperparameters and Training Performance for Multi-Agent DRL	161

TABLE OF CONTENTS

Bibliography	165
---------------------	------------

LIST OF TABLES

TABLE	Page
0.1 Acronyms	1
4.1 Input Parameters of Single-objective Optimisation Model	56
4.2 Output Variables of Single-objective Optimisation Model	57
4.3 Service Requests Setting [1][2]	61
4.4 VNF CPU Resource Requirements [1, 3]	61
5.1 Output Variables of Multi-objective Optimisation Model	73
5.2 Service Requests Setting [4–8]	82
5.3 VNF required Resources and Properties[1, 3]	82
5.4 Time Consumption for Small Scale Network	86
5.5 Time Consumption for Large Scale Network	86
6.1 Algorithm Performance Indicator	102
6.2 Time Consumption of Different Approach	104
6.3 Running Time of Weighted-sum Approaches	110
7.1 Network Parameters and SFC Parameters	119
7.2 Game Model Parameters	120
7.3 Node Resource Capacity and Price	125
7.4 Application Latency and Ping Latency Results	131
8.1 Input Parameters of Multi-agent DRL Model	139
8.2 Service Requests of Multi-agent DRL Model	140
8.3 Output Variables of Multi-agent DRL Model	141
A.1 Hyperparameter Configuration for Pointer Network	159
A.2 Hyperparameter Configuration for Multi-Agent DRL	162

LIST OF FIGURES

FIGURE	Page
1.1 5G Challenges, Potential Enablers, and Design Principles.	6
1.2 5G Architecture.	7
1.3 ETSI NFV Reference Architectural Framework.	9
1.4 Integrated MEC Deployment in 5G Network.	11
1.5 Examples of the Physical Deployment of MEC.	13
1.6 MEC/NFV Architecture.	14
1.7 SFC Placement Example and Main Flow.	16
3.1 Overview of Reinforcement Learning.	43
3.2 Overview of Deep Reinforcement Learning.	43
3.3 OpenStack Cloud Operating System.	47
3.4 Components of OpenStack.	48
3.5 OSM in Service Platform View.	49
3.6 Next-Generation 5G Architecture.	50
4.1 5G Network Topology and SFCs Placement.	53
4.2 5-Node Network Topology	60
4.3 Simulation Results on 5-Node Topology.	62
4.4 Simulation Results on 35-Node Topology.	67
5.1 SFC Placement Example in Multi-layer Edge-Cloud Network.	71
5.2 SFCs Coding in the Proposed NSGA-II.	78
5.3 Large-Scale Simulation Topology.	81
5.4 Total Service E2E Latency on 7-Node Topology.	82
5.5 Total Congestion on 7-Node Topology.	83
5.6 Non-dominated Solutions on Small-Scale Network.	85
5.7 Non-dominated Solutions on Large-Scale Network.	87
6.1 Encoding and decoding of SFC.	93
6.2 Non-dominated Solutions on Small-Scale Network.	101
6.3 Non-dominated Solutions on Large-Scale Network.	103

LIST OF FIGURES

6.4	Simulation Results on 29-Node Network Part I.	105
6.5	Statistical Simulation Results on 29-Node Network Part I.	106
6.6	Simulation Results on 29-Node Network Part II.	107
6.7	Statistical Simulation Results on 29-Node Network Part II.	109
6.8	Experiment Testbed.	111
6.9	Simulation and Experiment Results on Real Testbed.	113
7.1	5G Network Topology and Resource Competition.	118
7.2	Average CPU Utilisation in MEC Servers.	126
7.3	Average CPU Utilisation in Edge DC.	126
7.4	Average CPU Utilisation in Core DC.	127
7.5	Average OE Resource Utilisation in MEC Servers.	128
7.6	Average EO Resource Utilisation in MEC Servers.	128
7.7	Network Payoff.	129
7.8	Service Acceptance Ratio.	129
7.9	Experimental Testbed.	130
7.10	Potential Function on 5-Node Network.	131
8.1	SFCs Placement and Scheduling.	136
8.2	Resource Utilisation Ratio.	148
8.3	Rewards.	149
8.4	Service Acceptance Performance.	150
A.1	Training Performance for Multi-Objective DRL.	160
A.2	Training Performance for Multi-Agent DRL.	163

Table 0.1: Acronyms

A3C	Asynchronous Advantage Actor Critic
AO	Access Offices
API	Application Programming Interface
AR	Augmented Reality
BIP	Binary Integer Programming
BS	Base Station
BSS	Business Support System
CAPEX	Capital Expenses
CDC	Core Data Centre
CG	Cloud Gaming
CO	Central Office
COTS	Commercial Off the Shelf
C-RAN	Cloud-Radio Access Networks
CSPs	Communications Service Providers
C-TAEA	Constrained Two-Archive Evolutionary Algorithm
DC	Data Centre
DDPG	Deep Deterministic Policy Gradients
DDQN	Double Deep Q Network
DEMUX	Demultiplexer
DHCP	Dynamic Host Configuration Protocol
DL	Deep Learning
DNS	Domain Name Service
DQN	Deep Q Network
DRL	Deep Reinforcement Learning
DT	Digital Twin
EA	Evolutionary Algorithm
EDC	Edge Data Centre
eMBB	enhanced Mobile Broadband
EMS	Element Management System
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
E2E	End-to-End
FIFO	First In First Out
FIP	Finite Improvement Property
GA	Genetic Algorithm
GNN	Graph Neural Network

LIST OF FIGURES

IETF	Internet Engineering Task Force
ILP	Integer Linear Programming
IM	Information Model
IoT	Internet of Things
IP	Internet Protocol
ISG	Industry Specification Group
ISP	Internet Service Providers
JSSP	Job Shop Scheduling Problem
KPI	Key Performance Indicators
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
MANO	Management and Orchestration
MARL	Multi-Agent Reinforcement Learning
MASS	Multi-Agent Scheduling System
MBS	Macro-Base Stations
MEC	Multi-access Edge Computing
MEO	Mobile Edge Operator
MEPM	Mobile Edge Platform Management
MILP	Mixed Integer Linear Programming
MMTC	Massive Machine-Type Communications
MORL	Multi-Objective Reinforcement Learning
MPLS	Multi-Protocol Label Switching
MUX	Multiplexer
NE	Nash Equilibrium
NEF	Network Exposure Function
NFs	Network Functions
NFV	Network Function Virtualisation
NFVI	NFV Infrastructure
NFVO	NFV Orchestrator
NFV-RA	Network Function Virtualisation-Resource Allocation
NN	Neural Network
NP	Non-deterministic Polynomial time
NRF	Network Resource Function
NS	Network Service
NSGA-II	Non-dominated Sorting GA-II
NSSF	Network Slice Selection Function
OEO	Optical-to-Electrical-to-Optical
OPEX	Operating Expenses

OSM	Open Source MANO
OSS	Operations Support System
OXC	Optical Cross Connector
PCF	Policy Control Function
PF	Pareto Front
PNE	Pure strategy Nash Equilibrium
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RL	Reinforcement Learning
SAR	Service Acceptance Ratio
SDM	Space Division Multiplexing
SDN	Software Defined Networking
SFC	Service Function Chain
SFP	Small Form-factor Pluggable
SGD	Stochastic Gradient Descent
SLA	Service Level Agreement
SMF	Session Management Function
SPoF	Single Point of Failure
TD	Temporal Difference
TSP	Telecommunication Service Provider
UDM	Unified Data Management
UE	User Equipment
UPF	User Plane Function
URLLC	Ultra-Reliable and Low-Latency Communications
VM	Virtual Machine
VNFM	VNF Manager
VNF	Virtual Network Function
VIM	Virtualised Infrastructure Manager
V2V	Vehicle-to-Vehicle
WDM	Wavelength Division Multiplexing
WIM	Wide Area Network Infrastructure Managers
5G	The Fifth Generation of Mobile Technology

INTRODUCTION

This chapter begins with the introduction of the network transformation, which is the background of this study. Then, supporting technologies, such as Network Function Virtualisation (NFV) and Multi-access Edge Computing (MEC), are detailed. In the NFV-enabled edge-cloud network, on-demand computing and networking resources should be provided to satisfy diverse user requirements. Driven by this, the Quality of Service (QoS)-aware NFV-Resource Allocation (NFV-RA) problem has been widely studied. Then, the motivation, problem statement, challenges, and contributions of this study are discussed. Next, the thesis goals and structure are followed. Finally, at the end of this chapter, all the publications are listed.

1.1 Background

1.1.1 Network Transformation

Faced with the explosive growth of data traffic, diverse requirements of novel applications, and the popularity of smart devices and machine type communication, communications service providers (CSPs) require the transformation of the whole network [9, 10]. Such transformation should add agility and flexibility to the network, reduce architecture and management complexity, assist in faster service innovation, and reduce cost. The fifth generation of mobile technology (5G) is, therefore, proposed to transform traditional network infrastructure to support traffic growth and service complexity, satisfy users' higher Quality of Experience (QoE) requirements, and provide a more scalable, flexible, and intelligent network [11].

This transformation is not primarily about implementing the most cutting-edge technologies at each layer of the infrastructure. Rather, it brings automation, openness, and agility to service creation and delivery and allows intelligent insights into users and infrastructure behaviour.

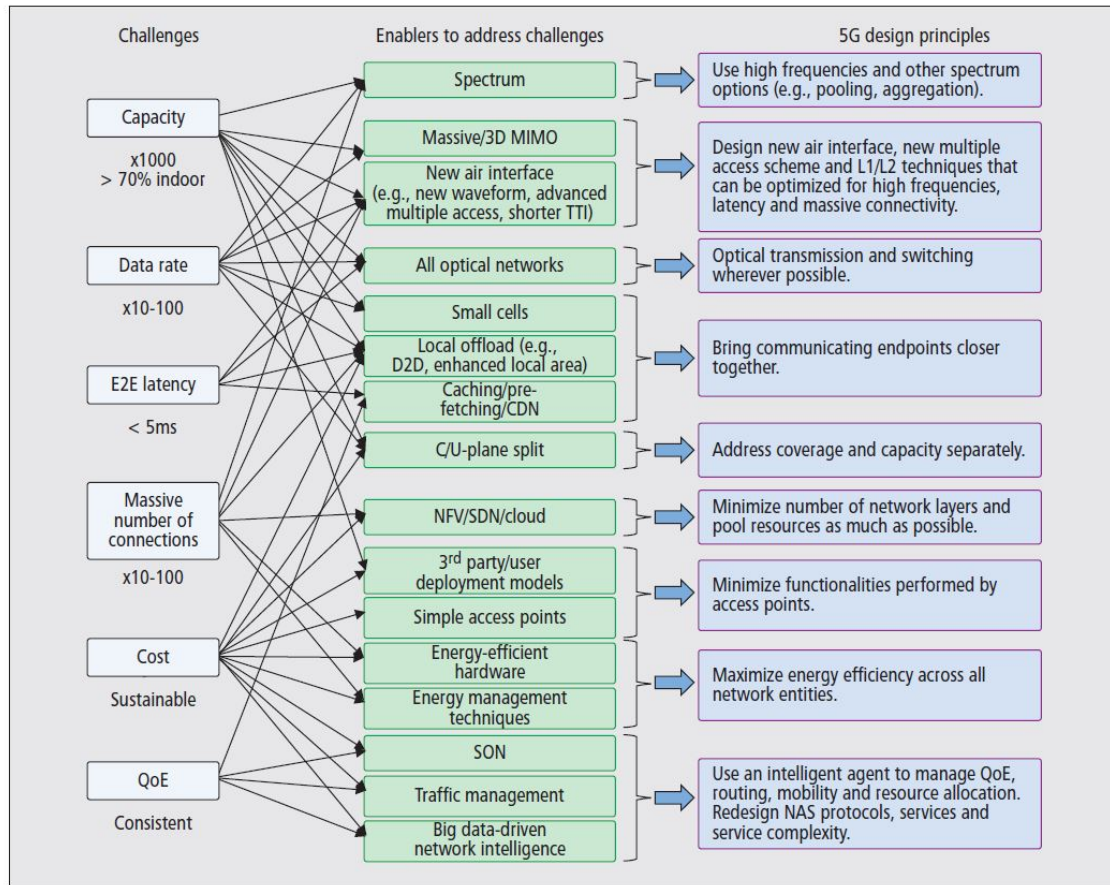


FIGURE 1.1. 5G Challenges, Potential Enablers, and Design Principles [13].

Furthermore, in the development of the value chain of services to end-users, this transformation will also support new business models and provide new opportunities [12].

5G is an end-to-end (E2E) ecosystem that supports a fully mobile and connected society and empowers value creation towards customers and partners through the present, and emerging use cases [11]. In general, 5G is going to address six challenges: 1) much higher capacity (1000 times improvement), 2) much higher data rate (10-100 times improvement), 3) lower E2E latency (less than 5ms), 4) higher number of connections (100 times improvement), 5) cost-efficient network solutions, 6) consistent QoE provisioning (shown in Figure 1.1) [11, 13]. For individual 5G use cases, requirements can be very diverse and sometimes extreme. There will be three major use cases in the 5G scenarios, including: 1) enhanced mobile broadband (eMBB); 2) ultra-reliable and low-latency communications (URLLC); and 3) massive machine-type communications (MMTC) [9].

It is anticipated that several services with different requirements will be active at the same time, and it is high cost and impractical to provide a specific service model for each use case. In addition, considering the nature of stochastic and unpredictable network services in the future,

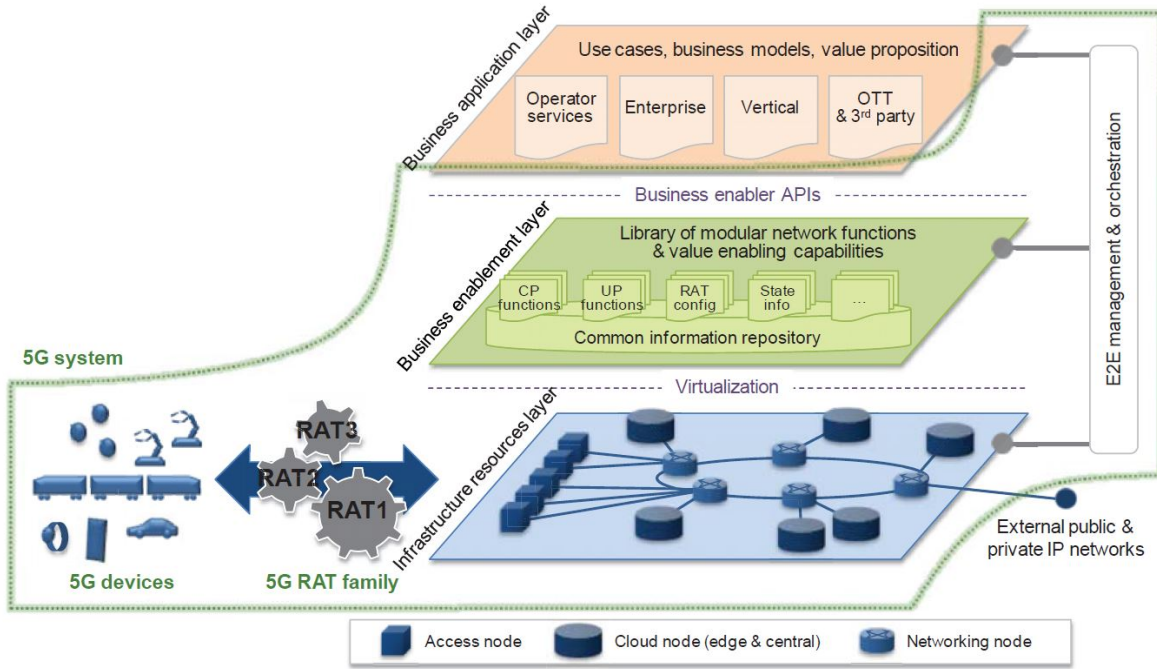


FIGURE 1.2. 5G Architecture [11].

it requires re-architect the network to efficiently support network services and fulfil network transformation.

Figure 1.2 shows the 5G architecture proposed by the Next Generation Mobile Network [11]. It consists of three layers and an E2E management and orchestration entity. The infrastructure resource layer includes computing nodes, network nodes, mobile devices, and physical links. All the physical resources are virtualised and utilised by the upper layer and monitored by the E2E management and orchestration entity. The business enablement layer is a library of all functions, and the business application layer contains specific network services. The E2E management and orchestration entity translates service requirements into actual modular network functions, chains these network functions, and maps all of them to the infrastructure layer [11].

Benefits from this 5G architecture, physical resources can be virtualised and network functions can be relocated to more general-purpose servers. Compared to previously fixed location resources and network functions, such transformation enables network resources and services to be offered in a more effective way. Hence, 5G will no longer provide strictly defined and single service models. Instead, it will support tailored and dynamic services for end-users, enterprises, verticals, and stakeholders, with diverse requirements, in terms of latency, bandwidth, reliability, privacy, etc [14].

Apart from diverse service requirements, there are still some other barriers to enable network transformation, such as strict E2E latency requirements (around 1ms) for real-time applications,

costly network growth, long and complex launch cycle for new applications, and organization development [9, 11]. To break through these barriers, innovations in the radio access network (RAN), the backbone, the fronthaul and backhaul, and the control and management are necessary [13]. The network needs to provide both high throughput and low latency, both flexibility and programmability combined with efficient use of available resources so as to meet all of the challenges [13, 15].

The new network architecture framework should satisfy the following requirements: 1) Complete separation of hardware and software, 2) Flexible automation and scalability of the network function deployment, and 3) Dynamic operations in the network function control through control and monitoring of the network state [16].

1.1.2 Technologies

Leveraging technologies, such as NFV and MEC, a traditional network can be evolved to respond to network transformation, support various 5G applications, and provide scalable and flexible network management [10, 13, 17, 18]. These two major technologies will be analysed in this article to demonstrate their importance in 5G and their benefits.

1.1.2.1 Network Function Virtualisation

The traditional service provision paradigm is inflexible and costly due to the complexity of middlebox implementation [19]. Network functions (NFs), such as gateways, proxies, firewalls, and transcoders, are generally placed on dedicated hardware, and service providers have to spend high Capital Expenses (CAPEX) and Operating Expenses (OPEX) on maintenance and management. NFV, defined by the industry specification group (ISG) under the European Telecommunications Standards Institute (ETSI), has drawn significant attention from both academy and industry in recent years [13, 20]. It shifts the service provision paradigm by decoupling NFs from dedicated hardware and implementing them in virtual machines (VMs) or containers on standard commercial-off-the-shelf (COTS) hardware [20, 21].

Such migration enables Virtual Network Functions (VNFs) to be instantiated and run on the cloud, programmed by software rather than physical hardware and implemented flexibly and dynamically in the network. The main benefits can be classified in the following aspects:

i) Add flexibility to service and resource provisioning. As NFs are no longer location-dependent, they can be placed in any location at any time according to service requirements and current resource status [9]. With shared hardware and software resources, these NFs can be connected in a more flexible way [22]. Such flexible deployment can satisfy diversified service requirements, such as latency, and efficiently use virtualised hardware resources.

ii) Auto-configuration and softwarisation to satisfy dynamic property and ease operation. Considering that network traffic is dynamic and stochastic, automated configuration and management allow network operators to instantiate, scale-up, scale down, and delete VNFs dynamically

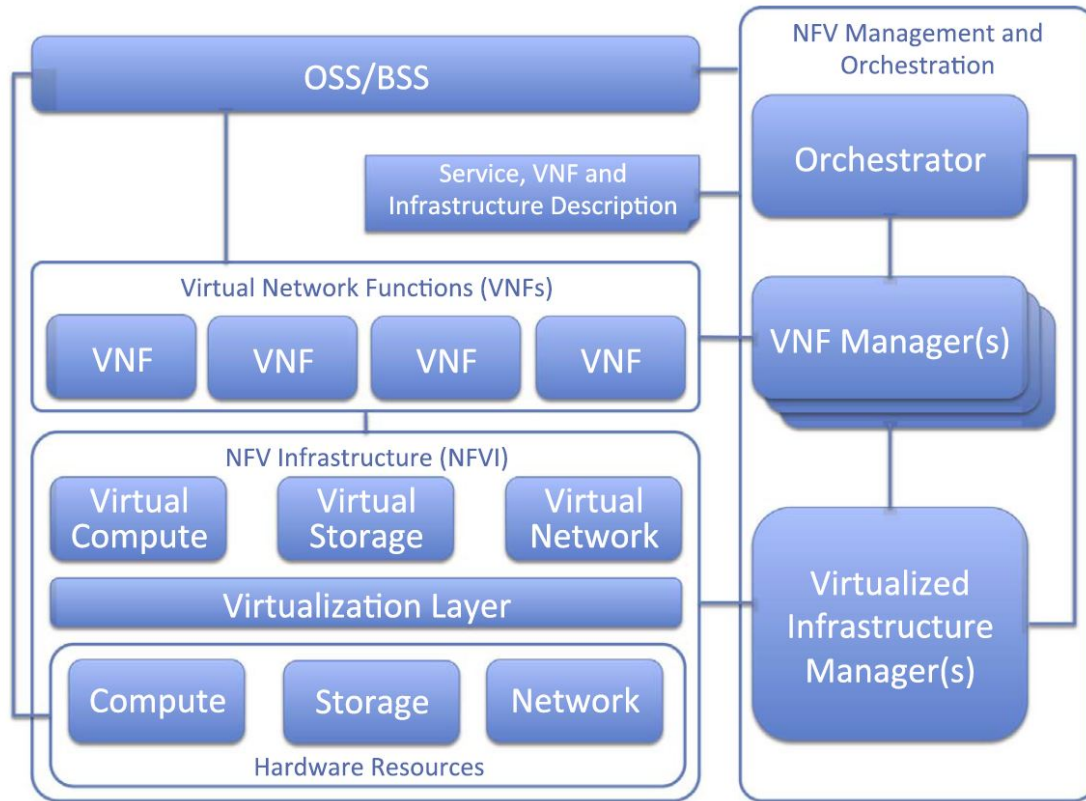


FIGURE 1.3. ETSI NFV Reference Architectural Framework [10].

[23]. Softwarisation can reduce operation complexity, simplify operation infrastructure, and ease upgrade and maintenance [18].

iii) Reduce CAPEX and OPEX. CAPEX savings comes from the optimised utilisation and on-demand provision of network resources, the efficient management of shared and flatter network infrastructures, and the adoption of COTS hardware devices [9, 23]. VNFs can be instantiated without installing new equipment, which can reduce capital investment and energy consumption. OPEX saving comes from the simplified operation and management approaches. For example, most fault detection and restoration can be done remotely instead of working on the spot.

iv) Accelerate the product cycle. NFV has the potential to bring new services with increased agility and faster time-to-value by using COTS hardware [22]. Previously, it was costly and took a long time to launch new services because of the proprietary nature of dedicated devices and the lack of professional knowledge and experience to integrate and maintain these services [23].

A VNF is normally administered by an Element Management System (EMS), responsible for its creation, configuration, monitoring, performance, and security. In a Telecommunication Service Provider (TSP)'s environment, an EMS provides the essential information required by the Operations Support System (OSS). In conjunction with the Business Support System (BSS), the

OSS is a general management system that assists providers in deploying and managing a variety of E2E telecommunications services (e.g., ordering, billing, renewals, problem troubleshooting, etc). Therefore, the focus of NFV specifications is on integration with existing OSS/BSS solutions [21].

The development of NFV is based mainly on the adoption of industrial-standard hardware and cloud computing technologies [23]. Compared to dedicated hardware, industrial-standard hardware has larger volumes and interchangeable components inside to support different NFs. By replacing a great number of specific hardware devices in the network, general-purpose devices can simplify the complexity of hardware architectures in the network. Recent developments of cloud computing technologies, such as various hypervisors, OpenStack, and Open vSwitch, also make NFV achievable in reality [23].

According to ETSI ISG, the NFV high-level architecture framework (illustrated in Figure 1.3) consists three major building blocks: *i*) VNFs, *ii*) NFV Infrastructure (NFVI), *iii*) NFV Management and orchestration (MANO) [20]. They are identified and introduced in detail as follows:

i) VNFs are defined as a software implementation of network functions, which are capable of running over the NFVI [20].

ii) NFVI includes diverse hardware resources and how they can be virtualised, which can support the execution of VNFs [20].

NFVI consists of both physical and virtual resources. Physical resources refer to computing, storage, and network hardware resources of the core network, backhaul network, and edge network [24]. Virtual resources are abstractions of these hardware resources, which are achieved through the virtualisation layer [22]. Virtual resources include virtual nodes that support processing or routing functionality and virtual links that logically connect two virtual nodes directly. Such virtualisation hides the complexity of the physical layer and provides hardware resources as a pool and realises adaptable and efficient utilisation.

iii) NFV MANO covers all virtualisation-specific management and orchestration tasks for the NFV framework, such as the life-cycle management of physical and/or software resources and the life-cycle management of VNFs [20, 24].

The NFV Orchestrator (NFVO) is responsible for the life-cycle management and auto-deployment of network service (NS) over NFVI [25]. It achieves life-cycle management automation by realising closed-loop service assurance, service fulfilment, and service orchestration in the instantiation, configuration, activation and running phases [24]. The VNF Manager can be used for the VNF life-cycle management covering instantiation, updating, scaling, querying, and termination [20]. The Virtualised Infrastructure Manager (VIM) controls the NFVI and the interaction between NFVI and VNFs [23]. It can be used for resource allocation, resource management, performance analysis from the NFVI perspective, collection of infrastructure fault information, and monitoring for further optimisation [20].

As we can see, this framework also includes the coordination between NFV and traditional

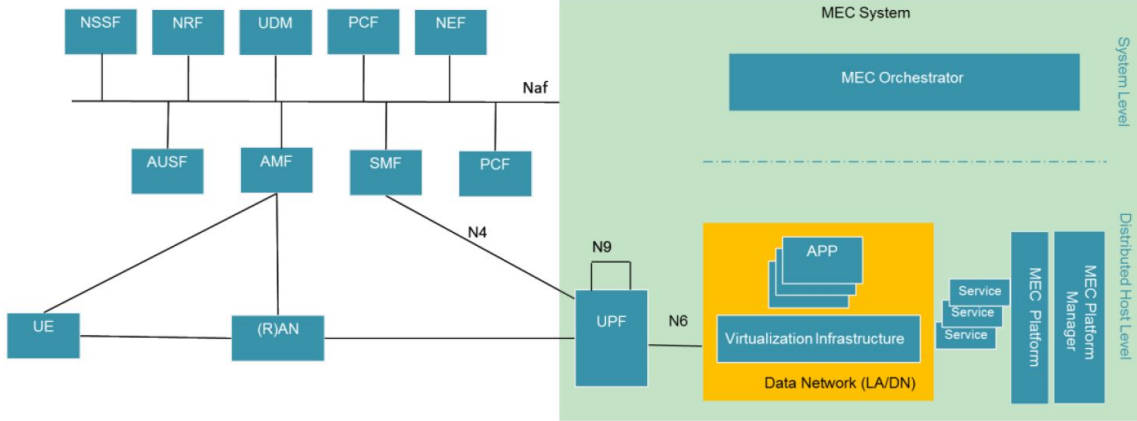


FIGURE 1.4. Integrated MEC Deployment in 5G Network [26].

network management systems such as OSS and BSS, which enables the management of VNFs running on legacy equipment [22]. However, the requirements of managing and orchestrating network services under this architecture bring new challenges as well, such as how to describe VNFs, how to connect them into services, and how and where to deploy them [24].

1.1.2.2 Multi-access Edge Computing

MEC can be seen as an evolution of clouding computing to meet the demanding 5G Key Performance Indicators (KPIs), especially on low latency and bandwidth efficiency [26]. By placing computing and storage resources at the edge of the network, processing and caching capabilities are brought from cloud data centres (DCs) to locations closer to end-users [27]. It can enable the network transformation, support ultra-low latency services, fulfil data privacy and context awareness requirements and improve edge intelligence [13, 18, 25, 27].

MEC is recognised by the European 5G Infrastructure Public Private Partnership (5G PPP) as one of the critical enablers for the 5G network [28]. It is currently being standardised in an ETSI ISG [26]. Figure 1.4 in the MEC white paper shows how to deploy the MEC system in the 5G network.

On the left side of this figure, NFs of the 5G Radio Access Network (RAN) network are included. Network Slice Selection Function (NSSF) can allocate users to proper network slices based on their requirements. MEC services are registered in the service registry on the MEC platform, while other services are registered on the Network Resource Function (NRF). Unified Data Management (UDM) handles user identification, manages access authorisation, supports service continuity, and performs subscription management procedures. Network Exposure Function (NEF) acts as a centralised point for service exposure and authorising the coming requests [26].

Session Management Function (SMF) provides functionality that includes session manage-

ment, IP address allocation and management, selection/re-selection and control of the User Plane Function (UPF), configuring the traffic rules for the UPF, and charging and support for roaming [26]. It plays a crucial role in MEC because it selects and controls the UPF and configures its traffic steering rules. In addition, the SMF exposes service operations to allow MEC to control the policy settings and traffic rules [26].

On the right side of this figure is the MEC system deployed in a data network external to the 5G system [26]. MEC orchestration and management is responsible for the MEC host and applications, traffic routing configuration, dynamic application deployment, and user mobility monitoring of MEC applications [26]. The distributed MEC host can accommodate MEC platform services. MEC platform can interact with VNFs and NFVI via service Application Programming Interfaces (APIs).

5G system offers both generic traffic rule-setting and specific traffic rule-setting for User Equipment (UE). It is the UPF that is responsible for steering the user plane traffic towards the targeted MEC application in the data network [26]. When a MEC application is instantiated, the MEC platform will interact with the Policy Control Function (PCF) to request traffic steering for the specific traffic. Then, PCF will transform the request into policies and provide the routing rules to the appropriate SMF. After receiving information, SMF will identify the target UPF and configure traffic rules accordingly.

UPF is critical in the MEC system deployment because its location can affect the location of the MEC system. Thanks to its flexibility in locating, network operators can place it based on user requirements, network load, and available physical resources. Hence, MEC can also be deployed flexibly in the network. According to the white paper [26], there are four MEC deployment scenarios (presented in Figure 1.5):

1. Both the MEC and the local UPF are built on the Base Station (BS).
2. The MEC is located within a transmission node, perhaps with a local UPF.
3. Both the MEC and the local UPF are built on a network aggregation point.
4. The MEC is located within the Core Network functions (i.e., in the same DC).

As we can see, MEC can be placed from the BS to the central DC, catering for different requirements of operation, performance, and security [26].

Benefits from the adoption of MEC technology, a wide variety of use cases can be supported for the new market, such as Augmented Reality (AR), Cloud Gaming (CG), vehicle-to-vehicle (V2V) communication, e-health, smart factoring, and Internet of Things (IoT) [27]. Take AR, for example, it requires a high data rate and low latency. Furthermore, it needs to be aware of the user's location so as to analyse the output from a device's camera and provide real-time information. In most cases, such information is position-relative and even related to the direction users face. Hosting AR at the MEC platform can significantly enhance the QoE because it caches the localised data, reduces traffic propagation latency to the cloud, and updates information at a fast rate [27]. It is the key application considered in this thesis to represent low latency traffic.

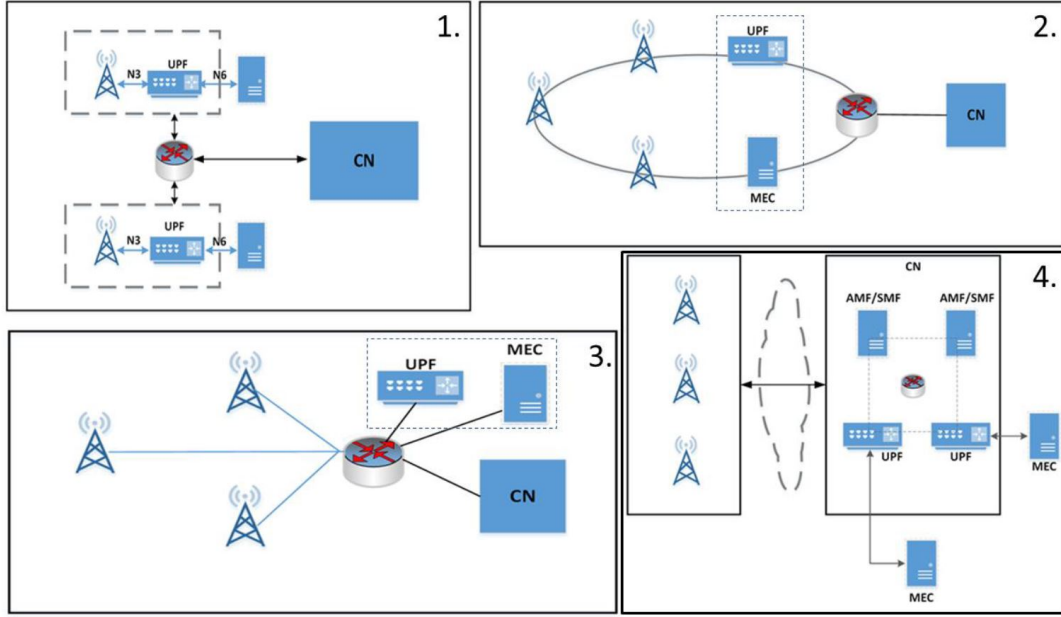


FIGURE 1.5. Examples of the Physical Deployment of MEC [26].

1.2 Resource Allocation in NFV-enabled Edge-Cloud Networks

1.2.1 Motivation

Previously, NFV was adopted only in the DC networks. When the network edge has been equipped with computing and storage resources, it is advisable to consider NFV and MEC jointly because it enables the flexible deployment of MEC applications and brings common management and orchestration system to the entire network [18]. Furthermore, since the MEC MANO and the NFV MANO have similar operations, merging the MEC and NFV architecture is feasible.

Figure 1.6 shows the proposed MEC/NFV architecture [29]. The NFV domain, comprised of NFVO, VNF Manager (VNFM), VIM, VNF, and NFVI, is responsible for supporting underlying resources. While, the MEC domain, comprised of Mobile Edge Operator (MEO), ME Platform Management (MEPM), and ME Platform, is responsible for application demand information and application management. The NFVI implements virtualised infrastructures for MEC applications. The ME platform, which can be implemented by VNF, supports application-level traffic forwarding and processes application content data. The MEPM provides management of communication interfaces, application-level traffic forwarding rules, and Domain Name Service (DNS) configuration. The VNFM controls the creation, release, and scaling of MEC service instances. It also works with the VIM to apply resources according to MEC application requirements and network status [29].

The orchestration and management layers span over the NFV and MEC domains, ensuring the interaction between the two domains to meet the low delay and high bandwidth requirements

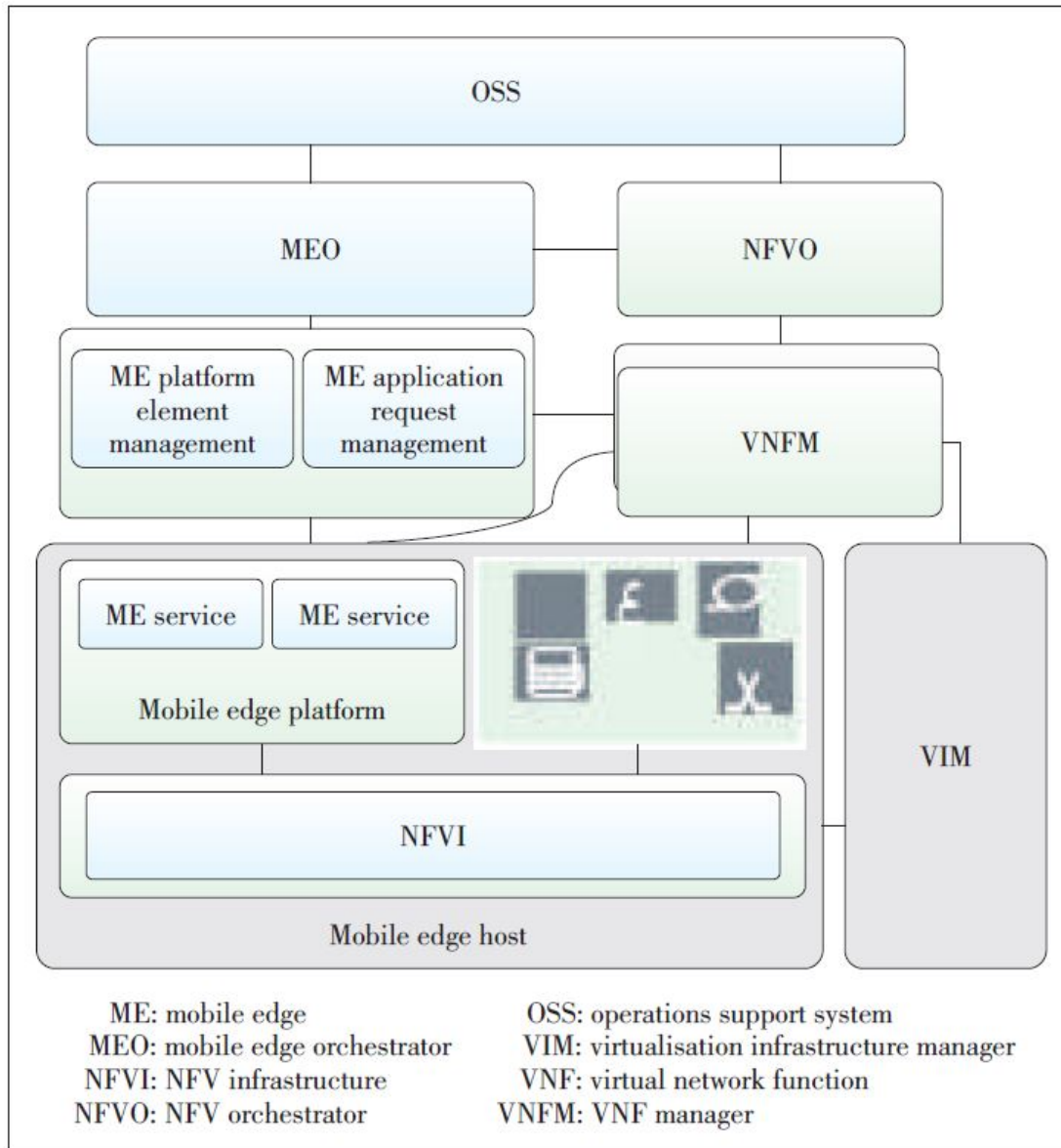


FIGURE 1.6. MEC/NFV Architecture [29].

jointly [29]. For the orchestration layer, the NFVO will update the system orchestration according to the information such as MEC application changes and edge network system status provided by the MEO. For the management layer, the VNFM manages the service instance life-cycle according to the messages, like conflicts and location changes of MEC services, provided by the MEPM [29].

It allows the whole network to take advantage of the flexibility by integrating MEC and NFV. VNFs can be deployed at the edge as well as in the central cloud, depending on the needs of use cases and business models. NFVI should support VNFs to be instantiated in suitable locations at the right time. Nevertheless, not all the resources should be provided to the same use case at the

same time. Therefore, it requires flexible and scalable schemes to provide and scale both network and computing resources in edge and cloud networks automatically, dynamically, and on-demand to these VNFs [11, 23]. Taking into account that edge resources are limited in comparison to central resources, and MEC services have stricter requirements on latency compared to other services, the resource provisioning schemes should be both efficient and tailored to ultra-low latency network services [11].

In the NFV ecosystem, a network service (NS) is a collection of chained VNFs, and it is created and deployed by defining the number of VNFs, their order in the chain, and the chain's allocation in the NFVI [21]. One of the most difficult aspects of NFV deployment is achieving fast, scalable, and dynamic composition and allocation of NFs to perform an NS. However, because an NS necessitates a set of VNFs, attaining efficient service coordination and management in NFV raises two issues: 1) how to compose VNFs for a determined NS, and 2) how to efficiently distribute and schedule the chained VNFs onto the network [21].

Hence, a resource allocation scheme is essential for NFV-based networks to fulfil the requirements mentioned above [21]. Efficient algorithms running at the control and management layer can make decisions in a holistic view for physical resources utilisation to achieve economics on a large-scale provided by NFV. These algorithms should decide [24]: 1) Placement: which VNF instance of the service requests should run on which computing nodes and should be supported by how many resources; 2) Routing: which link should be chosen to connect the placed VNF instances in the correct order, and it is a flow level decision; 3) Scheduling: which execution time slot is selected for the VNFs processing and transmission in the NS [30]; 4) Monitoring: which kind of data should be monitored, and where and how often should they be monitored.

The placement, routing, scheduling, and monitoring decisions in edge and central cloud networks can be formulated as an optimisation problem to satisfy objectives such as service acceptance maximisation, resource utilisation maximisation, congestion minimisation, etc [22]. Such a problem is called as NFV-Resource Allocation (NFV-RA) problem [21] consisting of three stages below: 1) VNFs-Chain Composition: decides which type of VNFs should be chosen for the network services. 2) VNF-Forwarding Graph Embedding: includes the placement (mapped to substrate nodes) and routing (mapped to substrate paths) problem. 3) VNFs-Scheduling: decides how to execute each VNF in order to minimise the execution time. In this thesis, this problem's second and third stage has been studied especially for service requests with different QoS requirements in 5G edge-cloud networks.

1.2.2 Problem Statement and Challenges

The E2E services can be flexibly deployed in the NFV-enabled 5G network to satisfy user requirements and improve resource utilisation efficiency. These services can be represented by chaining different types of VNFs in order as Service Function Chains (SFCs) [21]. According to Internet Engineering Task Force (IETF), SFC is standardised as a set of ordered or partially

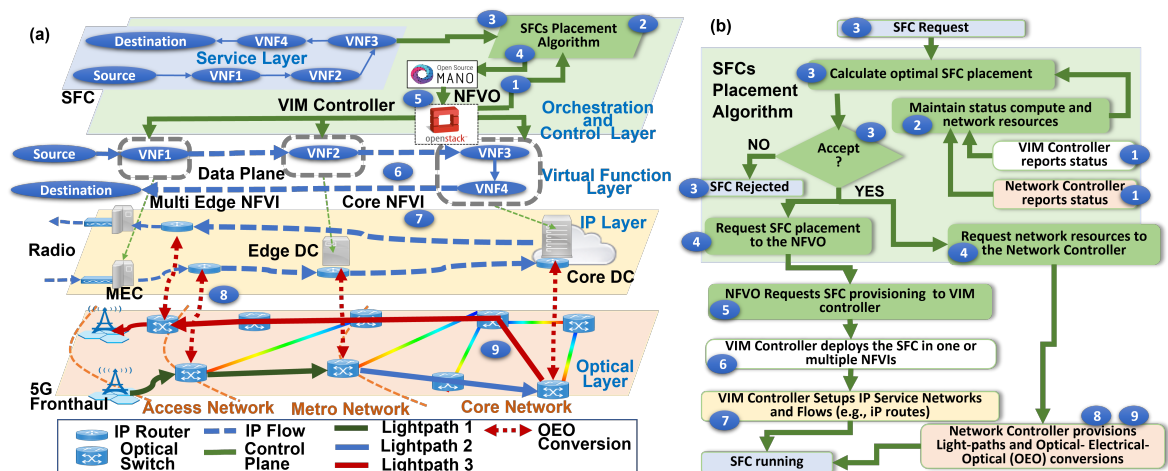


FIGURE 1.7. SFC Placement Example and Main Flow.

ordered VNFIs, and packets/frames must follow the path of strictly ordering VNFs [19]. SFCs provide architectural building blocks for network operators to instantiate and connect network functions across the network and realise diagnostics, security, and management for network services [31]. In the context of SFCs, the second stage of the NFV-RA problem is called the SFCs placement problem [32, 33], while the third stage is called the SFCs scheduling problem [34].

The SFCs placement problem contains two sub-problems. The first refers to determining the nodes where VNFs should be placed (i.e., VNF Placement), and the second consists of the link selection between nodes where VNFs run (i.e., routing) [35]. This problem has been widely studied in the core network [36–39]. However, to cope with new requirements in the 5G network, it needs to be further studied in combined MEC and core network scenarios considering QoS requirements, especially latency requirements. As we know, many novel 5G network services have strict E2E latency requirements, such as AR, V2V communication, and smart manufacturing. To make these ultra-low latency services in reality, they must be properly chained, placed, and routed to meet the stringent QoS and service-level agreement requirements of users/tenants [16].

In this thesis, the QoS-aware SFC placement problem has been well studied in the multi-layer 5G network, including access, metro, and core networks. MEC nodes, Edge Data Centres (EDCs), and Core Data Centres (CDCs) are equipped with different amounts and different types of resources. These nodes and switching nodes are connected via IP links or optical links. In this work, not only traditional ordering and resource constraints are taken into consideration, but also E2E latency constraints ranging from 1ms for ultra-low latency services to 500ms for non-real-time services. The E2E latency accounts for the time needed for the data packet from source to the destination across all the nodes and servers [11].

Figure 1.7 illustrates an example of the SFC placement. As illustrated, the virtual infras-

structure is deployed with OpenStack. Two MEC nodes and one edge DC are edge NFVIs, while the core DC is core NFVI. In this example, a single SFC contains 4 VNFs. According to the placement solutions, VNF1 is mapped to one of the MEC nodes, VNF2 is mapped to EDC, and the other two VNFs are mapped to CDC. The source and destination are also shown in this figure. Electrical signals to be transmitted through the network will be first transformed into optical signals and then transmitted via the network using optical fibres. After the transmission, optical traffic will be converted to electrical signals at the receiver and processed at the VNF instance. Optical-to-Electrical-to-Optical (OEO) conversions across the optical layer and IP layer are shown by the red dotted arrow line. In the optical layer, Wavelength Division Multiplexing (WDM) is adopted to give flexibility [11]. The optical signals transmitted between the transmitter and receiver via lightpath will use the same wavelength (demonstrated by the thick arrow lines in green, blue, and red).

In the NFV architecture, VIM manages the node resources and link resources to create VNF instances and orchestrate VMs based on OpenStack [40]. The NFVO (e.g., Open Source MANO) monitors the network status and interacts with the planning tool (e.g., SFCs placement algorithm) [41]. The right side of Figure 1.7 shows the main flow of SFC placement. After receiving the service request in NFVO, the SFC placement algorithm will make decisions based on monitored network status [40]. Such decisions refer to the VNFs' mapping on physical nodes and the virtual links' mapping to physical links without breaking corresponding constraints. Then, NFVO interconnects VNFs in numerous MEC nodes and DCs to meet all end-user requirements while improving resource utilisation performance [41]. With such an SFC placement algorithm and NFV MANO system, 5G infrastructure can support flexible and intelligent control and management of network resources and network services.

There are many challenges to be addressed to solve this QoS-aware SFC placement problem in multi-layer edge-cloud networks:

1) The SFC placement problem is proved to be NP-hard as it is a joint problem of two NP-hard problems: NF placement and flow routing [42, 43]. The optimal solutions for NP-hard problems can not be achieved in polynomial time and heuristic and meta-heuristic algorithms are usually proposed to approximate optimal solutions. For this problem, it is harder to consider node mapping and flow routing holistically since their results can affect each other [39].

2) The SFC placement is more challenging under the MEC environment compared to the core network. Firstly, many services in the MEC network are ultra-low latency services, and latency requirements should be considered with great emphasis [10, 44]. Furthermore, latency requirements can vary a lot among different types of services, which means the proposed solutions should be capable of dealing with different QoS requirements. Secondly, the computing resources of MEC nodes, as well as the bandwidth between MEC nodes, are more limited compared to those in core networks [44]. These limited resources need to be utilised very well to serve as many services as they can.

3) As there is a large number of MEC nodes with computing capability, it requires scalable approaches for SFC placement, control, and management [15]. Approaches used for SFC placement in the core network, such as Mixed Integer Linear Programming (MILP) and evolutionary algorithms do not scale well. For the intelligent approaches, considering a large number of hosting nodes, which leads to large action space, reinforcement learning (RL) is not suitable for the SFCs placement problem in edge-cloud networks [19]. In addition, centralised control is faced with a single-node failure problem and also does not scale well. Thus, novel and scalable approaches are expected for problem-solving, network control and management.

4) The SFCs placement problem can be multi-objective in reality. For example, reducing latency can result in congestion in the MEC nodes and degrading load balance performance. These objectives are usually conflicting with each other and it is hard to balance different objectives at the same time. In addition, NFV providers and customers may require different objectives. For example, NFV providers care more about cost-saving, but customers care more about QoS performance. Different objectives can not be optimized at the same time, thus it is difficult to get a win-win situation [33].

5) In real network scenarios, network status and traffic vary significantly due to the dynamic and stochastic arrival of different network service requests [33]. These service requests are not known in advance, and their requirements can change during the process. Therefore, this problem can not be solved just offline. Online approaches are also required to place SFCs dynamically to adapt to the changing network scenarios. It is also challenging to achieve dynamic scaling of VNF instances, whether horizontal (i.e., instantiate/remove VNF instances) or vertical (i.e., add/release resource to/from VNF instances) or both [45].

6) Studying SFCs placement problems in the multi-layer network composed of virtual layer, IP layer, and optical layer, offers significant merits in terms of high-capacity and low-latency transmission, especially for the increasingly bandwidth-hungry and low-latency applications in the 5G scenario. In order to leverage such benefits, the fundamental complexity challenge caused by adding the optical layer has to be resolved. Firstly, resource availability in three-layer networks needs to be considered [46]. In addition to computing and buffering resources, OEO conversion resources and wavelengths in the optical layer should be provided to SFCs. Secondly, adding an optical layer means adding complexity to the model and algorithm design. In the MILP model, constraints such as the mapping from virtual link to optical link, the wavelength continuity, wavelength resource capacity, and optical transmission latency must be satisfied. In the algorithm design, wavelength continuity and availability will affect the routing path determination. Thirdly, the scalability problem is even more severe. Due to a large number of wavelengths in different optical links, the computational complexity grows dramatically as the problem size increases for this kind of NP-hard problem [47].

Compared to the SFCs placement problem, few papers are studying the SFCs scheduling problem (third stage of NFV-RA problem), which determines the execution time slot for the

traffic of each SFC traversing the VNF that placed on the corresponding server [30, 48]. The SFC scheduling algorithm can also run as the SFC planning tool in Figure 1.7 and decide the VNFs' mapping, virtual links' mapping, VNF's processing time slot, and traffic transmission time slot. It can be formulated as a job-shop scheduling problem, which is proved to be NP-hard [49, 50]. Apart from the aforementioned five challenges, some additional challenges exist for this problem, such as the model complexity in terms of time sequence constraints and the algorithm design when considering E2E service latency, which is a challenging delay-aware scheduling problem [50].

1.2.3 Research Contribution

The main purpose of this thesis is to provide a comprehensive way for supporting QoS-aware network services, especially ultra-low latency services, in edge-cloud networks, from single-objective to multi-objective, from centralised control to distributed control, from mathematical optimisation to reinforcement learning, from offline to online.

The main contributions can be summarised as follows:

1) Formulate the QoS-aware SFCs placement problem and SFCs scheduling problem in multi-layer networks consisting of virtual layer, IP layer, and optical layer. Firstly, single-objective and multi-objective MILP models are designed to get the optimal solutions in the small-scale networks, which are taken as the benchmark for the algorithms' performance analysis. Secondly, the optical layer is included in these models to reduce the transmission latency on intermediate switches, increase the flexibility of ultra-low latency services placement, and increase the service acceptance performance.

2) Study the QoS-aware SFCs placement problem and SFCs scheduling problem in edge-cloud networks. Firstly, different models are applied to MEC nodes and DCs to capture their different resource capacity features (e.g., M/M/1 queueing model at MEC nodes but no queueing latency at DCs). Secondly, different objectives are designed to support services demanding different E2E latency (e.g., MEC nodes maximise the ultra-low latency service acceptance and DCs maximise all the service acceptance). Thirdly, a wide range of algorithms (e.g. heuristic, meta-heuristic, game theory-based, DRL algorithms) are designed for different scenarios to support various QoS services in the same network, reduce MEC nodes congestion during peak hours, increase service acceptance performance, especially for ultra-low latency services.

3) Evaluate the proposed models and algorithms through simulation and real test-bed experiments. Simulation works are carried out for all the considered network scenarios. For offline algorithms, the performances of the proposed algorithms are compared with optimal solutions and other benchmark solutions in small-scale networks. While large-scale simulation tests their scalability and complexity performance. For online algorithms, simulation results prove their capabilities of serving service requests in the online environment. For the experiment, SFCs with different latency and traffic sizes are implemented in VNF instances hosted by servers connected

through optical fibres. Real test-bed experiment results validate the performance of the proposed offline algorithms by providing real E2E latency, CPU resource usage, and service acceptance ratio (SAR).

1.3 Thesis Structure

The following is a thesis outline describing the major parts of the thesis.

In Chapter 1, first, the motivation for adopting NFV and MEC technologies and for studying the resource allocation problem in NFV-enabled edge-cloud networks are discussed. Next, the QoS-aware SFC placement and scheduling problem and its challenges are introduced followed by a description of our research contributions. Publications during the last four years are included in the end. In Chapter 2, the literature review of QoS-aware SFCs placement and scheduling problem is provided combined with the related work of multi-objective reinforcement learning (MORL) and multi-agent RL (MARL). In Chapter 3, methodologies for resource allocation are introduced such as MILP, deep reinforcement learning, game theory, and MARL, followed by the management tools for the virtualised infrastructure and network services.

To solve the QoS-aware SFCs placement and scheduling problem in NFV-enabled edge-cloud networks, different methods are proposed to address different challenges. Mathematical models, algorithm design, simulation settings, experiment settings, and results of these proposed methods are detailed explained and analysed in Chapters 4-8.

It starts from a single-objective optimisation to increase the SAR of ultra-low latency services (Chapter 4). A MILP model is designed for small-scale networks and a meta-heuristic algorithm is presented for large-scale networks. Then, a more realistic multi-objective optimisation problem is studied with the assistance of evolutionary algorithms (EAs) (Chapter 5). In the next step, to further improve the network performance in terms of QoS, resource congestion, and SAR, a scalable DRL approach is designed for this multi-objective SFCs placement problem. A Pointer Network-based DRL model is at first designed with the MILP model as environment constraints. Then, a Chebyshev-assisted Actor-Critic algorithm is designed to approximate the Pareto Fronts (PFs) by running it with a series of weights and to improve ultra-low latency service acceptance performance under different workloads by running it with the selected weights (Chapter 6). These methods above show great performance in simulation and experiment as offline solutions with centralised control.

To adapt to dynamic network scenarios, firstly, a distributed online algorithm is designed based on game theory (Chapter 7). Network services are players in the congestion game competing for resources to maximise their own payoff. By assigning different weights of resource consumption for services with different latency requirements, the ultra-low latency service acceptance performance can be improved. Secondly, an online MARL method is investigated to solve the joint SFCs placement and scheduling problem in edge-cloud networks (Chapter 8). Both cooperation

and self-interested scenarios are studied by the centralised training decentralised execution DRL algorithm, with different objectives for edge and cloud agents. It is still ongoing work, and more algorithms will be proposed in the future.

Finally, Chapter 9 concludes achievements and presents the outline of future work.

1.4 Publications

1. **Bi, Y**, Meixner, CC, Bunyakitanon, M, Vasilakos, X, Nejabati, R and Simeonidou, D, 'Multi-Objective Deep Reinforcement Learning Assisted Service Function Chains Placement'. in: 2021 IEEE Transactions on Network and Service Management Special Issue on Embracing AI for Network and Systems.

2. **Bi, Y**, Bunyakitanon, M, Uniyal, N, Bravalheri, AC, Muqaddas, AS, Nejabati, R and Simeonidou, D, 2019, 'Distributed Online Resource Allocation Using Congestion Game for 5G Virtual Network Services'. in: 2019 IEEE Global Communications Conference (GLOBECOM) - Proceedings. Institute of Electrical and Electronics Engineers (IEEE), pp. 1-7

3. **Bi, Y**, Meixner, CC, Wang, R, Meng, F, Nejabati, R and Simeonidou, D, 2019, 'Resource Allocation for Ultra-low Latency Virtual Network Services in Hierarchical 5G Network'. in: 2019 IEEE International Conference on Communications, ICC 2019 - Proceedings. Institute of Electrical and Electronics Engineers (IEEE), pp. 1-7

4. Meng, F, Mavromatis, A, **Bi, Y**, Wang, R, Yan, S, Nejabati, R and Simeonidou, D, 2019, 'Self-Learning Monitoring On-Demand Strategy for Optical Networks'. IEEE/OSA Journal of Optical Communications and Networking, vol 11., pp. A144-A154

5. Ou, Y, Salas, EH, Ntavou, F, Wang, R, **Bi, Y**, Yan, S, Kanellos, G, Nejabati, R and Simeonidou, D, 2018, 'Field-Trial of Machine Learning-Assisted Quantum Key Distribution (QKD) Networking with SDN'. in: 44th European Conference on Optical Communication-ECOC .

6. Wang, R, **Bi, Y**, Ou, Y, Salas, EH, Meng, F, Yan, S, Nejabati, R and Simeonidou, D, 2018, 'Coordinated Fibre Span Power Optimisation and ROADM Input Power Management Strategy for Optical Networks'. in: 2018 European Conference on Optical Communication (ECOC 2018): Proceedings of a meeting held 23-27 September 2018, Rome, Italy. Institute of Electrical and Electronics Engineers (IEEE), pp. 554-556

7. Meng, F, Yan, S, Nikolovgenis, K, **Bi, Y**, Ou, Y, Wang, R, Nejabati, R and Simeonidou, D, 2018, 'Field Trial of Gaussian Process Learning of Function-Agnostic Channel Performance Under Uncertainty'. in: 2018 Optical Fiber Communications Conference and Exposition (OFC).

8. Meng, F, Mavromatis, A, **Bi, Y**, Yan, S, Wang, R, Ou, Y, Nejabati, R and Simeonidou, D, 2018, 'Field Trial of Monitoring On-Demand at Intermediate-Nodes Through Bayesian Optimization'. in: 2018 Optical Fiber Communications Conference and Exposition (OFC).

LITERATURE REVIEW

This chapter provides the literature review of related works that study the QoS-aware SFCs placement problem at first. Specifically, these works are grouped into five classes: single-objective optimisation for SFCs placement, multi-objective optimisation for SFCs placement, RL-assisted SFCs placement, distributed SFCs placement, and SFCs Placement in Optical Networks. The first two classes focus on works utilising traditional methods such as ILP, heuristic, and meta-heuristic, to solve this problem. The third class focuses on RL methods. In the next, a literature study of QoS-aware SFCs scheduling problem is detailed, comprising both traditional and RL methods. Last but not least, the literature review of multi-objective RL and multi-agent RL-related works is discussed. By providing the background information, the motivation for adopting these two methods can be strengthened.

2.1 Literature Review on QoS-aware SFCs Placement

2.1.1 Single-Objective Optimisation for SFCs Placement

In this subsection, the related works, which use traditional methods such as ILP, heuristic, meta-heuristic, etc, to solve the single-objective QoS-aware SFCs placement in a centralised manner, are summarised and analysed.

Some works take the cost minimisation as their objective without impacting QoS requirements since cost saving is one of the main goals of NFV technologies [51]. Authors in [51] design a mixed integer programming model for SFCs placement in Evolved Packet Core (EPC) with the VNF deployment and resource utilisation cost minimisation as the objective function. The E2E latency requirements are modelled as constraints including processing, propagation, queueing, and virtualisation latency. In [8], authors design a MILP model with latency limitations for the

cost minimisation in metro core networks for SFCs placement. The VNF placement causes the cost on a physical node, cost per utilised resource on a physical node, and cost per utilised capacity on the physical link. The E2E latency is caused by processing, packet queueing, and propagation. However, both transmission and OEO conversion latency are not included in these two works.

Although some works have different goals, they are still latency-aware. A mixed integer quadratically constrained programme model is provided to minimise resource consumption for SFCs placement while delivering particular latency in [52]. It investigates the linear relationship between a VNF's processing time and the number of resources allocated to it. Authors in [53], aim at maximising the total number of requests that each SFC can send to the cloud. Their approach combines service chain consolidation, pod assignment, and machine assignment per pod problems using a linear programming model and efficient heuristics. Delay for transmitting a packet between any two machines is taken into account in a fat-tree DC network.

Others take latency minimisation as their objective directly. The authors of [43] use latency minimisation as their objective since it allows them to maximise average resource utilisation while reducing average response latency at the same time. For joint optimisation of SFCs placement and scheduling, a two-phase priority-driven weighted algorithm is presented to reduce response latency and increase resource utilisation in DCs networks. Similar to the algorithms proposed in this thesis, their devised algorithm can lower the job rejection rate. In [54], a VNF low latency placement technique is proposed to minimise network latency in DC networks. A MILP cost minimisation model and an online heuristic algorithm are developed to jointly optimise three steps in NFV-RA, including VNFs-Chain Composition, VNFs-Forward Graph Embedding, and VNFs-Scheduling in [38]. In this approach, the delay is considered in the cost objective. However, their link delays are generated by the uniform distribution.

However, the hierarchical network resources have not yet been considered in the latency-aware SFCs placement works stated above. As a result, their solutions are unsuitable for a hierarchical 5G network with MEC nodes and DCs. There are only two studies that examine hierarchical network resources for the single-objective optimisation [1, 55]. In [55], a shortest path decision mechanism is addressed for each SFC to minimise the E2E latency composed of processing and network latency in DC networks. They consider three DCs with different processing capacities: small DCs, medium DCs and large DCs. However, no optical layer is involved in their models. An algorithm is designed in [1] to minimise the average number of nodes required to host VNF instances as well as the blocking probability. The E2E latency caused by propagation, Forward Error Correction (FEC), and OEO conversion are taken as constraints. Although the work proposed in [1] studies SFCs placement problem in the hierarchical network including access, main and core central office connected via optical links, it neither takes queueing and transmission latency into account nor proposes a MILP model to get the optimum solution like the one proposed in Chapter 4.

2.1.2 Multi-Objective Optimisation for SFCs Placement

In this subsection, the related works, which use traditional methods such as heuristic, meta-heuristic, etc, to solve the multi-objective QoS-aware SFCs placement in a centralised manner, are summarised and analysed.

Considering the network domains, some works focus on the SFCs placement in cloud networks [36, 40, 45, 56, 57]. Genetic algorithms (GAs) are employed by the Authors of [45] to optimise the overall number of used servers, links, link resources, and the number of updated servers and links during scaling. However, their objective function is weighted-sum and the weights are chosen to be equal or zero, which is not reflected by the Pareto Fronts (PFs). The rest papers approach this problem in a Pareto-optimal fashion. Authors in [56] tackle this issue by providing a heuristic-based non-dominated sorting GA-II (NSGA-II) to minimise the number of employed servers and the link resource utilisation. In [40], two goals are involved, one is the minimisation of total bandwidth consumption, and the other is the minimisation of the maximum link utilisation. Authors in [57] use a multi-objective Evolutionary Algorithm (EA) based on decomposition to find the optimal placement for SFCs. They maximise the total cost and the incoming service acceptance ratio (SAR). However, none of these algorithms considers the latency requirements, let alone the latency objective, which are very important for 5G services with variant QoS requirements. There is only one paper considering latency. Authors in [36] apply a multi-objective modified simulated annealing algorithm (MOSA) to generate PFs that depict various trade-offs between the latency minimisation and the cost minimisation. The overall delay of all demands, the total number of placed VNF instances, and used CPU resources are minimised simultaneously in their model. However, the MOSA method highly depends on the initial solutions and cannot effectively find feasible solutions.

There are several studies of this problem in a mixed environment including edge and cloud [3, 39, 58–60]. Authors in [39] present a scalable solution based on the flow clustering module to balance multi-objectives such as path stretch minimisation, load balancing, and total network usage maximisation. The above three objectives are weighted equally and combined into a single-objective function. The authors of [58] minimise the combination of the edge cloudlet's maximum utilisation, QoS violation, and allocated cloud computing resources. They model the edge-central cloud architecture as MILP without breaching the delay constraint. Similar to the model proposed in Chapter 4 and Chapter 5, they use different queueing models for edge nodes and clouds, with M/M/1 queue for the edge node, and the M/G1/ ∞ queue for the clouds. Authors in [3] propose a GA to minimise both the CPU usage and the service blocking rate in Cloud/MEC architecture that include both optical network resources and computing resources. In [59], a fraction of SFCs mapping cost and load balancing is represented as one objective for the SFCs placement in edge servers and cloud. They consider the latency requirement for each service request. In a multi-domain scenario, authors in [60] design a heuristic algorithm to minimise E2E delay, service cost and operational cost together. However, none of the above papers includes

PFs' analysis and there is a lack of Pareto optimal solutions in a mixed environment.

Some of the aforementioned works use EAs to solve this problem [3, 36, 45, 56, 57]. Among them, the PF is studied in [36, 56] to select solutions that meet the providers' numerous objectives and performance trade-offs. Others convert multi-objective problems to a single-objective by introducing weighting factors [39, 58], or combining different objectives directly [59], or solving each objective sequentially [60].

There is a work [25] studying a GA framework tailored to ultra-low latency services in the MEC environment to minimise access latency and maximise service availability. The comparison of the objective values with different weighting values is carried out, which is very similar to the work in Chapter 5, however, this work is about the VMs placement.

2.1.3 Reinforcement Learning-assisted SFCs Placement

The approaches adopted to the SFCs placement problem are not limited to the optimisation methods listed in the above two subsections. In recent years, RL has also been applied to this problem. However, it is challenging to solve this problem with traditional RL algorithms because they are tabular methods and require a high degree of memory for large dimension problems, which is inefficient and impractical when the state space of the environment is huge [19]. The SFCs placement problem has both large state space (the network has plenty of hardware devices and different types of resources) and action space (SFCs can be flexibly placed in various network locations), which are hard to be recorded in finite Q tables [19]. Thanks to the development of Deep Reinforcement Learning (DRL), this problem can be solved because a finite Q table is substituted by a deep neural network (NN), and memory is only required to store the NN or experience replay [61]. High-dimensional network resource states and VNF placement actions for the optimal placement can be handled by DRL based on network feedback rewards.

Some works are dealing with SFCs placement problems via the DRL approach. Among them, most use value-based approaches [19, 49, 62–64]. In [62], the authors use a Graph Neural Network (GNN) to process the topology information, including the resource type and resource capacity and use a DRL architecture to train the parameters of the GNN by interacting with the network environment. The GNN-based scheme has a better relationship inferring ability among the network's nodes than other deep learning methods. This work is QoS-aware, and the routing nodes' processing delay, the VNF instances' processing delay, and the transmission delay are all considered. In [19], authors formulate the SFC placement problem as a Binary Integer Programming (BIP) problem and use the Double Deep Q Network (DDQN) method to design the VNF placement algorithm. DDQN is proposed to solve the overestimation and instability of the traditional DQN approach. It includes an online neural network taking charge of sampling and action selection and a target neural network evaluating the Q-values. Their algorithm, which includes the offline training and online running process, has been proved to improve the reject number and reject ratio of SFC requests, throughput, E2E delay, VNFI running time, and

load balancing. In reference [63], authors suggest an offline parallel deployment scheme based on DRL to satisfy all demands and deploy all SFCs simultaneously. They begin by calculating the number of VMs and locating servers to host VNFs using DRL. Then, they proportionally distribute resources. In comparison to selected benchmarks, their solution serves demands with the minimum cost.

Indeed, it is difficult to calculate all the value functions with the value-based approach for problems with large action space and continuous control [33]. Hence, some works use policy-based approaches to solve the SFCs placement problem [33, 46, 65, 66]. In reference [33], authors propose an adaptive, online, DRL approach NFVdeep to automatically deploy SFCs to minimise the operation cost and maximise the overall throughput of requests. A policy gradient-based method is used to improve the training efficiency and convergence to optimality. The NFVdeep architecture covers the NFV environment (servers and network links) and agent. In particular, the agent collects state information from the NFV environment and operates automatically. Once the agent has taken action, the NFV environment distributes the reward to it. Finally, the agent updates the relevant policy according to the reward. Such a procedure repeats until the reward converges. Authors in [46], consider this problem in metro-core optical networks. They design a multi-layer optimisation model based on MILP to maximise the number of successfully routed chains and minimise the reconfiguration penalty as a weighted-sum objective. Reference [65] studies the intelligent SFC routing in dynamic network settings. They introduce a DRL scheme to achieve overall load balancing and minimise maximum flow path delay in multi-service networks with the coexistence of background traffic and SFC flows. Compared with MILP, their DRL approach can obtain near-optimal network performance with only lightweight training before being applied to different traffic matrices. But there is only one work modelling this problem as the constrained combinatorial optimisation problem and using DRL to solve it [66]. The designed agent is able to learn placement decisions to minimise the overall power consumption. This work is going to be detailed analysed in subsection 2.2.

In addition, there is work using the actor-critic method. Authors in [67], design an adaptive DRL resource allocation method to minimise E2E delay (processing and queueing delay). According to the problem's continuity, they improve the Asynchronous Advantage Actor-Critic (A3C) algorithm with unsupervised reinforcement and auxiliary learning to obtain the optimal allocation policy.

Although plenty of research works have been using DRL for the SFCs placement problem, none are in a multi-agent environment. For large-scale problems, it is necessary to develop distributed methods in a multi-agent scenario, which is studied in Chapter 8.

2.1.4 Distributed SFCs Placement

This subsection summarises the works utilising distributed approaches to solve the SFCs placement problem. Compared to the aforementioned works that rely on a centralised approach, the

distributed methods bring advantages such as 1) scalability; 2) robustness; 3) privacy; 4) practical feasibility, which will be detailed explained in Chapter 3. Based on these benefits, it is believed that distributed approach is a better approach for the NFV-RA problem [68]. Several works are studying the SFCs placement problem with game theory, however, they are all in the cloud environment [68–70].

In [69], a game theory-based heuristic method is developed for load balancing, with the SFCs placement problem modelled as an ILP. Various costs are included: 1) opening cost: the licensing and energy cost of having an idle VNF; 2) processing cost: a piece-wise linear function of the load of each server; 3) link cost: linear with respect to the bandwidth used. The operator determines the cost of each SFC by adjusting the price of each VNF and each link to better fit for dynamic network status. Each SFC request is a greedy player trying to minimise its own costs, and each player is considered successful if its cost is lower than the cost in its previous position. However, the problem they studied is static and no latency involved.

In [68], authors formulate the resource allocation problem as a convex optimisation problem, maximising the overall system utility function for 5G service slices requiring different resource demands. A resource allocation game model is designed for the resource auction between the slices and the DCs based on the notion of dominant resource fairness. It is proven that the proposed game holding an NE is the same as the solution of the centralised scheme. Moreover, they design a fully distributed algorithm to solve the game and show how the selfish behaviour of non-collaborative slices affects the fair performance of the system. Both simulation and numerical analysis validate the results, showing the convergence of the proposed solutions and the optimal solutions. Resources, such as CPU, RAM, storage, and bandwidth are included. However, they do not consider latency, which is a crucial KPI for 5G services.

The following two works consider latency. The authors of [70] create a distributed approach for deploying SFCs in DCs by formulating a graph partitioning game and designing a distributed algorithm. With the latency constraints, the computation and communication costs are minimised. By partitioning a graph's vertices into a set of disjoint subsets, the number of nodes in each subset is fewer than a certain threshold and the number of cut edges is kept to a minimum. A cost function representing constraints is introduced and the Nash equilibrium (NE) is mathematically proven to exist. To find the NE and thus the optimal solution, a simplified iterative distribution algorithm is provided, which involves three steps: the initial solution, the refinement algorithm, and the termination criterion. A random walk in the search space is adopted to avoid being trapped in a local optimum and to converge towards the ideal solution. The termination criterion is used to make a trade-off between cost and execution time.

Authors in [71] minimise the congestion, latency, and cost collaboratively. They investigate the SFC composition problem (i.e., composing an ordered chain of VNFs) as an atomic weighted congestion game with unsplittable flows and player-specific cost functions. Each network user takes on a player's role, searching for the best service chain of VNF instances that meets their

individual requirements. Then, with a privacy-preserving distributed algorithm, this model is solved in a distributed manner. They show that the proposed game model possesses a weighted potential function and admits a NE, and the designed distributed algorithm converges toward an NE of the game in polynomial time. Through extensive numerical results, the performance of the proposed distributed solution is assessed. They consider NFV-specific network requirements and features, such as VNF server congestion, traffic flow latency, and the price charged by VNF servers. Specifically, the latency combines all inter-server, ingress-to-server, and server-to-egress latency. Inspired by this work, we will implement a congestion game for the SFCs placement problem, considering not just node mapping but also traffic routing in multi-layer networks.

There is one work studying the dynamic SFCs placement problem at the network edge using game theory [72]. Authors formulate the problem as a mean-field game where VNFs are modelled as entities contending over MEC nodes with the goal of reducing the resource consumption of MEC nodes and reducing service latency. The game involves heterogeneous VNFs requiring different amounts and types of resources, namely computational, storage, and transmission resources. However, their proposed algorithm is executed by the SDN controller and is a centralised solution. To the best of our knowledge, the algorithm proposed in Chapter 7 is the first distributed solution solving the dynamic SFCs placement problem at the edge-cloud multi-layer networks.

2.1.5 SFCs Placement in Optical Networks

As 5G is expected to support low latency (<1ms), high capacity, and high-speed (1Gbps) communications, optical transmission technology, which offers exactly these desired capabilities, must be combined with NFV and MEC technologies together for the 5G solutions. However, there are only a few works considering optical layer for the SFCs placement [1, 3, 46, 73].

In 2015, for the first time, the optical network was considered into an SFCs placement algorithm to minimise expensive OEO conversions in packet/optical DC networks [73]. In DC networks, optical technologies can perform VNF chaining for larger aggregated flows. Authors formulate the SFCs placement problem as a BIP model and design an efficient heuristic algorithm achieving near-optimal OEO conversion performance compared to BIP. In this work, SFCs placement is constrained by the computing, storage, and optical network resources, service order, affinity/anti-affinity constraints, and other requirements. However, this work is not fit for the hierarchical network consisting of DCs and MEC nodes.

The following three works are in the edge-cloud scenario. Authors in [1] propose an algorithm for dynamic SFCs placement in a metro-area network aiming to minimise both the average number of active nodes and the blocking probability. In this network, optical transparent switches are used and WDM technology is adopted, which imposes wavelength continuity constraints. It is to be noticed that latency is involved, which contributed from 1) context switching delay incurred for loading and saving the state of VMs, 2) propagation delay for each optical link, 3) Forward Error Correction (FEC) delay for encoding/decoding optical signals, 4) OEO conversion latency in

intermediate nodes of an SFC. However, processing, queueing, and transmission latency are not included.

Reference [3] studies the SFCs placement problem in Cloud-Radio Access Networks (C-RAN) and proposes a GA to minimise both the used CPU and service blocking ratio. Access Offices (AOs) and Central Office (CO) are connected via an optical ring using WDM or Space Division Multiplexing (SDM). In their work, optical network capacity is also involved in the resource constraints together with computing resources and storage resources. However, no exact solution like the MILP model is designed and no PFs are studied.

In addition to traditional approaches, the RL algorithm is used for the dynamic SFC placement in NFV-SDN enabled metro-core optical networks [46]. Authors create a MILP model as an environment for an RL system to optimise the resource allocation of SFCs in a multi-layer network (packet over flexi-grid optical layer). The objective is to maximise the number of successfully routed SFCs while minimising blocking probability, reconfiguration penalty, and power consumption. They develop the reward function and the agent that decides the SFC reconfiguration. In the multi-layer architecture, the service layer collects information on SFC requests. IP/Multi-Protocol Label Switching (MPLS) layer handles the creation of lightpaths between the chaining VNFs mapped from the service layer. Their solution is QoS-aware with propagation latency and service traffic volume taken into consideration. However, the RL algorithm used in this work is suitable for neither mathematical optimisation nor multi-agent system, which we will study in Chapter 6 and Chapter 8.

2.2 Literature Review on QoS-aware SFCs Scheduling

2.2.1 Traditional Ways Solving SFCs Scheduling

The SFCs scheduling problem has been studied by several works from different aspects adopting MILP, heuristic, meta-heuristic, or other traditional methods [30, 38, 48, 72, 74–76].

Authors in [30] emphasise fairness and resource utilisation. They model the SFC scheduling based on min-plus algebra theory and propose a weighted-based VNF scheduling algorithm to achieve fair scheduling. By sharing VNF instances among different traffic, assigning high weights to services requiring high capacities, and re-allocating the idle resources, the resource fragmentation can be reduced and the resource utilisation can be improved. Authors in [74] provide an adaptive SFC scheduling solution in a dynamic network. To address the mismatch problem that the arrival rate of data exceeds the maximum departure rate in a VNF instance, they split one SFC flow into several portions. A four-stage adaptive scheduling algorithm is proposed to make the trade-off between service performance and network management overhead. It maximises the available resource utilisation in common instances by absorbing more traffic and decreases the scaling frequency to reduce the management cost. Authors in [38] pay more attention to cost minimisation. They provide a MILP cost minimisation model involving capital

expenditures, operating expenditures, and link costs. A one-hop heuristic online algorithm is proposed to jointly optimise three steps in NFV-RA (e.g., VNFs-Chain Composition, VNFs-Forward Graph Embedding, and VNFs-Scheduling). The E2E delay is taken as a penalty signal in the objective function, and the link delay is generated by the uniform distribution.

SFC delays are included in [48], and authors study the SFCs scheduling problem with the MILP method to minimise the makespan of the overall VNF's schedule. The model is decomposed into four sub-problems: 1) the virtual link bandwidth allocation subproblem to assign traffic on each physical link; 2) the virtual machine allocation subproblem to map VNFs onto servers; 3) the transmit bitrates at the VNFs is calculated; 4) the starting time is decided for each VNF in the chain. Based on such decomposition, a GA-based method is developed for solving the problem efficiently and it aims to find the earliest completion of the schedule of each network service. Processing delay and transmission delay are included in the delay model. Authors in [48] study the SFCs scheduling problem to minimise the VNF transmission and processing delays. They develop a genetic algorithm-based heuristic method for solving this problem. However, they assume that the virtual link between two physical nodes can only handle one traffic flow at a time.

The joint SFC placement and scheduling problem has also been studied. Authors in [76] propose an SFC embedding and scheduling algorithm. A node resource ability metric is provided before the SFCs are placed onto the physical node. The node having the highest available resource ability value will be selected to embed the VNFs. If there is no proper node to support VNFs, the SFC will be rejected. Then, the virtual links between VNFs are mapped onto the physical links. After the scheduling is finished, the algorithm starts to reschedule SFCs including those rejected ones to enhance QoS performance and improve resource utilisation. If the rescheduling of the rejected SFC requests fails again, these requests will be thrown. A forwarding probability is used to measure the possibility of a physical node forwarding to its neighbouring node. There are two metrics evaluating the algorithm performance, one is the SFC acceptance ratio, and the other is the resource utilisation ratio. Different types of resources include computing resources, storage resources, and link bandwidth resources.

There are some papers considering MEC nodes in their studies. Authors in [72], study the SFCs scheduling problem on different MEC nodes and formulate it as a matching game between the VNFs and edge resources. The execution order of the VNFs can be achieved while minimising resource consumption and overall latency. Different types of resources such as computational, storage, and transmission resources are allocated to stochastic arrival SFC requests. They also consider the E2E delay involving processing, queueing, and transmission delay between two VNFs. The M/M/1 queue and Little's law are adopted to calculate such queueing delay. However, their solution is a centralised solution with an iterative learning algorithm executed by the SDN controller to converge to a Nash Equilibrium (NE). In [75], an affinity-based fair weighted scheduling heuristic is proposed, in the MEC scenario, which considers the access point selection

and the transmission between micro DCs.

However, all of the above works focus on the centralised solution. There is only one work developing the distributed solution. A multi-site cooperative throughput-optimal SFCs scheduling algorithm for edge cloud networks is developed in [77]. The authors provide a stochastic queueing-based system model to characterise the runtime traffic scheduling problem for SFCs. In the distributed settings, scheduler instances located at each site have access only to their local state, and the scheduler uses a distributed method to dynamically select where to send a packet and how many resources to allocate to each VNF instance at runtime. If no local VNF instance has enough resources to support traffic, the traffic will be sent to VNF instances on other sites. Both system utilisation and service quality are optimised. The scheduler can also balance load among different sites by forwarding packets when local traffic is overloaded. This solution is adaptable to high traffic dynamics since it is a fine-grained packet-level strategy that requires no prior knowledge of traffic and can leverage the resources that become available on the fly as a result of real-time sudden changes in network traffic. Hence, a packet-level simulator is developed to evaluate their solutions.

These heuristic algorithms (e.g., greedy algorithms) are fast and straightforward to build in general, but their performance is highly dependent on the features of the problem and lacks generality [50].

2.2.2 Reinforcement Learning (RL)-assisted SFCs Scheduling

Apart from heuristic/meta-heuristic methods, some works adopted the RL approach [49, 50, 78].

In [50], authors formulate the delay-aware SFCs scheduling problem as a job-shop problem and solve it using MILP and reinforcement learning. The scheduling actions include two parts: 1) For service, the starting time for each VNF being processed is determined; 2) For the NFV node, the scheduling sequence of all embedded VNFs is determined. Considering the action set depending on the previous VNF varies and each action execution time also varies at each decision-making state for the scheduling, a novel Q-learning algorithm is developed with a variable state-dependent action set and a customized reward function. Both the makespan (i.e., the time it takes from the execution of the first VNF to the completion of the last VNF for all services) and the various E2E latency requirements are considered. In the reward function, the shorter makespan leads to a large reward and the performance satisfaction leads to a positive reward. They also introduce a weight mechanism to reflect the service priority. To accommodate varying delay requirements, weight should be set as a large value for delay-sensitive services and a small value for delay non-sensitive services.

In [78], authors model the SFC requests as a graph and develop an online customised deep deterministic policy gradients (DDPG) based algorithm to maximise network utility for SFCs scheduling in the cost geo-diversity network. As the profit relates not only to the revenue but also to the QoS, both E2E delay and operation costs are taken into consideration. Although E2E

delay is considered, only processing delay is included. They redesign the exploration strategy, create a dual replay buffer structure, and use their formulation to guide replay buffer updates. The basic DDPG training agent composes of two parts: the actor-network and the critic-network. The actor's role is to define parameterised policy and generate actions (e.g., VNF scheduling) based on the network state (e.g., network topology, current VNF activation, flow rate). The critic's job is to evaluate current actions considering the reward obtained from the network. However, these two works are centralised requiring global information and are not fit for addressing privacy or scalability issues.

To the best of my knowledge, there are no distributed RL solutions, whereas the aforementioned RL approaches are all centralised. To fill this gap, a distributed delay-aware SFCs scheduling solution is proposed based on the multi-agent deep reinforcement learning approach in Chapter 8.

2.3 Literature Review on Multi-Objective RL

Due to the multi-objective characteristics of many real problems, Multi-objective Reinforcement Learning (MORL) has drawn great attention in recent years. It refers to the sequential decision-making problem with multiple objectives [79]. In MORL, each objective function has its reward signal, so the reward is not a scalar value in conventional RL problems but a vector value [79, 80]. Each agent is required to obtain action policies that can optimise multiple objectives. However, if these objectives are conflicting, there is no policy to optimise them at the same time. The maximisation of one objective can lead to the minimisation of the other. Under such circumstances, which is more challenging, the policy for different trade-offs among objectives is expected.

According to the number of learned policies, there are two main approaches for solving MORL: single-policy and multi-policy [79–81]. Single-policy methods aim to find the best single policy, which represents the preferences of users among the multiple objectives, while multi-policy methods can find a variety of solutions simultaneously [79–81].

The single-policy method is the simplest way because a multi-objective problem is transformed into a scalarised single-objective problem and is possible to be solved with any single-objective RL algorithms [80]. The scalarisation can be linear or nonlinear. When compared to multi-policy techniques, this method has the benefit of requiring less computational expenses [81]. However, it requires prior information about preference, which can be hard to get. Sometimes, even though the preference information is available initially, a slight change in the objective preference may lead to significant solution variations and produce undesired solutions.

Another class of approaches is a multi-policy method. Instead of finding a single policy for a specific preference, this method can generate multiple solutions simultaneously to satisfy various preferences [79]. It learns a set of Pareto dominating policies in a single run and obtains the

approximation of true PF, i.e., the set of solutions that Pareto dominates all the other solutions but are mutually incomparable [80]. The main drawback of this approach is its lack of scalability because a PF of optimal policies are represented by learning several individual policies and it grows significantly with the size of the domain [82]. Due to the simple implementation of scalarisation, the single-policy approach is the most common way adopted to find a set of trade-off solutions by solving several scalarisation functions over different preferences and combining their results [82].

For example, authors in [83], generate a set of uniformly spread weight vectors, use the weighted-sum approach to scalarise multi-objective functions and decompose the multi-objective travelling salesman problem into a set of scalar optimisation sub-problems. Each sub-problem is modelled as Pointer Network and solved by Deep Reinforcement Learning (DRL). As two sub-problems with adjacent weights could have very close optimal solutions, a neighbourhood-based parameter-transfer strategy is designed to solve a sub-problem assisted by the knowledge of its neighbouring sub-problem. The Actor-Critic method is used to train the sub-problem model with 1) an actor-network, modelled as the Pointer Network, which provides the probability distribution for choosing the following action; and 2) a critic-network that assesses the expected reward given a specific problem state. Motivated by their methods, in Chapter 6, we model the SFCs placement problem as a Pointer Network and improve their methods by finding more precise PFs.

Although the weighted-sum approach is straightforward to be implemented, it is not guaranteed that the mapping from weight space to objective space is isomorphic [80]. Moreover, the actions in PF's concave areas may not be chosen so that the PF cannot be accurately approximated [79]. Hence, several works are using the Chebyshev scalarisation function in MORL to find a Pareto approximate set [84, 85].

Authors in [84] study the way of finding the PF in situations where the shape of the front is not known beforehand, as is often the case. Instead of using a linear scalarisation function, they propose a non-linear scalarisation function, called the Chebyshev scalarisation function. They try to minimise the distance between some reference points and the feasible objective region. Most often, the utopia point in the PF is used as a reference. They conclude that there are three main advantages of the Chebyshev scalarisation method: 1) it can discover Pareto optimal solutions regardless of the shape of the front, 2) it can obtain a better spread amongst the set of Pareto optimal solutions, and 3) it is not particularly dependent on the actual weights used [84].

In reference [85], the multi-objective VM placement problem in DC networks is handled. It is challenging but crucial to find a proper weight for each objective because the inappropriate weights will cause the solution set to deviate from the Pareto optimal set [85]. Unlike most authors who simply set the weight for each objective to the same value, the authors in this work employ the Chebyshev scalarisation function in RL to determine the appropriate weights. They aim to reduce resource waste and energy consumption by finding a Pareto approximate set. As an initialisation, they create random scalarisation weight tuples. After that, combine the Chebyshev

scalarisation function with RL based on such initialisation. Finally, they will be able to select a suitable weight tuple from these randomly generated ones.

In multi-objective optimisation problems, feasible regions can be defined by constraints, and the task is to optimise the target function while satisfying these constraints. However, there is no closed-form solution for general constraints in RL [86]. Authors in [86], use an alternative penalty signal to guide the policy toward a constraint-satisfying solution, which can be effectively combined with existing RL algorithms. Precisely, they use the Lagrange relaxation technique and transform the constrained problem into an equivalent unconstrained problem.

Authors in [66] apply this method to the DRL for solving the SFCs placement problem, use the Lagrange relaxation technique and introduce the penalty signal to the reward function. The resulting agent is able to learn placement decisions with the aim of minimising the overall power consumption. It is the only work modelling this problem as the constrained combinatorial optimisation problem and using the sequence-to-sequence model to solve it. Motivated by this work, we firstly extend it to multi-objective, secondly improve its scalability by using Pointer Network, which removes the context vector calculation in the sequence-to-sequence model. In their model, they manually select the weight for each constraint and then combine them into a penalty signal, which may mislead the policy to sub-optimal solutions. Instead, we will use the constraint violation concept in [87] to construct the penalty signal.

2.4 Literature Review on Multi-Agent RL for Scheduling

DRL has presented great potential in the telecommunication area, especially for dealing with the dynamic problem as the best strategy can be determined according to the interaction with the environment [67]. However, most implementations of existing works are single-agent DRL-based and cannot cope with more and more complex situations. In such situations, multiple DRL agents are interacted with each other in a common environment to address the challenges like scalability, privacy, and partial-observability. These agents cooperate or compete to obtain the best overall performance, which is essential to establish self-organising, self-sustaining, and decentralised networks in the 5G and the beyond 5G networks [88, 89]. In this subsection, the related works of Multi-Agent Reinforcement Learning (MARL), especially those in the telecommunication area will be reviewed.

Several challenges are arising in the MARL, such as 1) non-stationarity; 2) scalability; 3) partial observability; 4) privacy and security [88]. When this approach is adopted in the telecommunication area, these issues must be suitable tackled. There are mainly two kinds of methodologies developed for overcoming the non-stationarity problem. The first one generalises the single-agent RL algorithms to the multi-agent setting, while the second one uses the centralised training and decentralised execution framework [88]. The latter one has become a standard paradigm in multi-agent systems because it provides a straightforward solution to the

partial observability and non-stationarity problems while allowing the decentralisation of agents' policies [88, 90]. It assumes a central controller collects information from all the other agents and learning policies for all the other agents during the training period to avoid non-stationarity. But during execution, these learned policies are decentralised executed, and the agent chooses actions using the local information only. This framework suits the telecommunication scenario well where centralised training can happen in the simulator or laboratory with no communication constraints, and execution can be done decentralised to save communication resources, reduce communication time, and protect agent's privacy [89].

The centralised training and decentralised execution framework is proposed simultaneously in the two works. Authors in [90], propose a counterfactual multi-agent policy gradient algorithm (COMA), a multi-agent actor-critic approach that employs a centralised critic to estimate the Q-function and decentralised actors to optimise the agents' policies. However, this method works only in the cooperative scenario and focuses only on discrete action space. Authors in the other work [91], create the multi-agent deep deterministic policy gradient (MADDPG) algorithm based on the actor-critic policy gradient method. Unlike COMA, MADDPG can be applied to both cooperative and competitive games and learn continuous policies efficiently. In the MADDPG, the critic has fully observations and uses extra information about other agents' policies for the training. After training is completed, only the actors take actions based on their local information during execution. This algorithm can be seen as a general-purpose MARL approach because it i) enables decentralised execution manner by using only local information at execution time for learned policies; ii) does not consider environments in which agents have explicit communication mechanisms or a differentiable model of the environment dynamics; iii) suits cooperative, competitive, and mixed cooperative-competitive environments since each agent have a centralised critic, allowing for agents with differing reward functions.

Several works adopt centralised training and decentralised execution framework and use MADDPG in the telecommunication area. In [92], the MADDPG algorithm is used to cooperate with the multi-channel access and task offloading of MEC in Industry IoT. Reference [93] studies the edge caching problem with the MADDPG-like algorithm. In this work, the cloud DC is taken as the centralised critic, each base station is considered as an agent hosting actor-network, and control channels allow the communication between the cloud DC and base stations. In [94], authors also propose MADDPG-like algorithms for cooperative edge caching. The central controller evaluates the actions taken by edge servers locally based on the global caching state and feeds back the evaluation results to edge servers to optimise the following actions. In their setting, actor-networks are deployed in the central controller, and each network is duplicated in its assigned edge server. With such implementation, the local actor-network parameters can be replaced by the central controller for updating. Simulation results of the above three works prove the effectiveness of the MADDPG method in the MARL scenario. However, none of them is NFV-related.

To the best of our knowledge, only two works are studying virtual resource allocation with the MARL approach. The first one studies the virtual network embedding (VNE) problem [95]. The authors propose a decentralised MARL algorithm that dynamically allocates physical network resources to multiple virtual networks in a coordinated way to improve the virtual network acceptance ratio. QoS requirements such as the packet drop rate and virtual link delay are satisfied. The learning environment consists of physical node agents and physical link agents. Each node agent manages node queue sizes while each link agent manages link bandwidths. It is assumed that each node agent has information on substrate node resource availability and all the resource utilisation of virtual nodes mapped onto it, and each link agent has information on substrate link resource availability and all the resource utilisation of virtual links mapped to it. Link agents also coordinate together to make sure that no virtual link can be mapped to more than one physical link. These agents dynamically adjust allocated resources to make sure that resources are not left underutilised. Their proposed method belongs to the first class of solving MARL problems: the generalisation of single-agent RL algorithms to the multi-agent setting. A decentralised Q-learning algorithm iteratively approximates the state action value and optimises the virtual nodes and virtual link mapping.

The second one studies the dynamic SFC-placement problem in IoT networks [96]. In this article, the whole IoT network is modelled as an agent, and the cooperation of multiple IoT networks is modelled through MARL. In the beginning, the cloud controller selects the IoT devices to serve in the current time slot. Then, the virtual network agents implement SFCs of the selected devices. The results of each DRL agent are combined in the central Q-table. Based on the combined decision, the network status (environment) is updated, the rewards returned, and the central policy updated accordingly. Each distributed DRL agent is a software element in the cloud server which provides distributed SFC mapping solutions. The performance is measured through the power consumption and the cost of nodes and links. To improve the long-term performance, two Q-networks are introduced, where one solves the SFC-placement problem while the other keeps track of long-term policy changes and adjusts the weights of the first Q-network. However, on the one hand, this solution cannot address the partial-observability problem, on the other hand, it faces a scalability problem for storing all agents' decisions in the central Q-table.

As the SFCs scheduling problem can be modelled as the job-shop scheduling problem (JSSP), this subsection provides JSSP-related works. Because centralised control of network states is challenging to be implemented in some practical scenarios, in [97, 98], various multi-agent RL algorithms have been presented to tackle reactive JSSPs. Compared to predictive scheduling, reactive scheduling is online control concerned with making local decisions independently [97]. It has several advantages, including the ability to: 1) respond effectively to unforeseen events (such as a machine breakdown) without the need for complete re-planning; 2) solve the large-scale scheduling problem in a reasonable time owing to its decentralised feature; 3) more practical in real-world situations where no global control can be implemented, and communication between

distributed working centres is impossible; 4) increase flexibility, save expenses, and may facilitate humans and agent-based machinery to collaborate [97, 98]. Hence, the JSSP can be formulated as a multi-agent MDP with changing action sets and then be solved in a distributed manner via inter-agent coordination. The agents' role is to select the next job from a list of jobs waiting to be processed at some resource.

Authors in [97] model the reactive JSSPs as multi-agent RL problems. Each resource is assigned an adaptive agent that makes job dispatching decisions independently of the other agents and uses an RL algorithm to improve its dispatching behaviour from trial and mistake. In each time step, the distributed RL agent observes the local state only, which contains condensed information about its associated resource and the jobs waiting there, and takes judgments after time intervals. They create a new multi-agent learning algorithm that incorporates data-efficient batch-mode RL, neural network-based value function approximation, and an optimistic inter-agent coordination mechanism to allow agents to learn in parallel. Because the immediate global costs are also taken into account, this learning rule builds a relationship between the local dispatching decisions and the overall optimisation goal. The expected long-term costs are minimised and the resource utilisation is maximised when the number of time steps can be minimised during which jobs are waiting for processing at the resources' queues (i.e., makespan). To do so, it must learn a decision policy that determines the optimum action for a given state. Although partial observability makes finding an optimal schedule more challenging, it allows for complete decentralisation in decision-making.

In [98], the reactive JSSP is also solved as a distributed problem with a large number of independent agents making decisions based on local observation. Authors create a multi-agent RL algorithm to solve reactive JSSPs featured by changing action sets (i.e., the list of jobs waiting at resource) and partially ordered transition dependencies. Each resource has an RL agent connected to it that uses probabilistic dispatching policies to determine which operations of the jobs queued at the respective resource should be processed next. They propose a small set of real-valued parameters for a compact representation. Using policy gradient RL, the agents adjust these parameters during training to optimise their dispatching policies and minimise the maximum makespan. Furthermore, they invent a lightweight communication mechanism that improves the capabilities of independently learning agents, allowing them to become partially aware of inter-agent dependencies.

The SFCs scheduling problem is more challenging compared to JSSPs [50]. Firstly, it is delay-aware JSSPs as different SFCs associated with different latency requirements [50]. Dispatching rules will consider resource capacity constraints, VNF ordering constraints, and the E2E latency constraints for each flow. Secondly, the same type of VNF instance on different nodes with different processing capacities increases the problem complexity. For example, processing VNFs at the MEC node leads to high queueing latency but low propagation latency, while, processing them at DC leads to low queueing latency but high propagation latency. It requires coordination

between agents or global training. In Chapter 8, we will solve these two challenges.

METHODOLOGY

In this chapter we provide the necessary methodology for the rest of the thesis. They include mathematical modelling tools like Mixed Integer Linear Programming (MILP) in section 3.1, Deep Reinforcement Learning (DRL) model in section 3.2 and Congestion Game in section 3.3; algorithms especially multi-agent reinforcement learning (MARL) algorithm in section 3.4; management tools like OpenStack and Open Source MANO (OSM) in section 3.5. These tools and methods are used for solving the QoS-aware NFV-RA problem in edge-cloud networks in this thesis.

3.1 Mixed Integer Linear Programming

The SFC placement problem has been proved to be NP-hard in [78, 99]. The non-deterministic polynomial-time (NP) theory belongs to computability theory and informally, the NP-hardness defines a class of problems that are at least as hard as the hardest problems in NP [100].

The computability theory originated in the work of Turing and others in 1936, is a standard computer model [101]. In this theory, the class P problems can be solved by some algorithms within a limited number of steps bounded by some fixed polynomial-time, while the class NP problems cannot be solved within polynomial-time by a non-deterministic Turing machine. For problems that fall into the class of NP-hard, there has been no algorithm discovered yet to achieve the optimal solution systematically [102].

To solve NP-hard problems, Integer Linear Programming (ILP) or Mixed-Integer Linear Programming (MILP) models can be applied. ILP investigates linear programming problems in which the variables are restricted to integers, while in MILP, only some of the variables are constrained to be integers, and others are allowed to be non-integers [103]. The general

problem is of the form: $\min c^T x$, subject to $Ax = b$, where $x \geq 0$ and $x_i \in \mathbb{Z} \ \forall i \in I$. There are many commercial-available optimization tools for solving ILP, and MILP models, such as Gurobi Optimiser [104].

Although the ILP and MILP approaches can obtain the optimal solutions, they suffer the main limitation of scalability. Authors of [103] provide proof that MILP problems are NP-Hard, and their computational complexity grows exponentially with problem size [47]. When the problem size is large, exact methods are no longer feasible. To improve the scalability, heuristic and meta-heuristic algorithms can be adopted for near-optimal solutions, which means there is a trade-off between the execution time and the quality of solutions [47]. Furthermore, designing an effective heuristic or meta-heuristic algorithm requires expertise in the problem.

In recent years, thanks to the development of deep reinforcement learning (DRL), solving NP-hard problems without the need for human involvement and not time-consuming has become possible [102]. DRL learns how to make decisions, which is in alignment with the requirements of mathematical optimisation problems. Benefiting from the deep neural network, DRL is capable of solving large-scale problems [19]. For example, in an SFCs placement problem with large-scale action spaces, DRL can exploit known information, explore new knowledge, evaluate actions by interacting with the environment and learns optimal policy. In addition, once the agent is trained, due to its strong generalization ability, it can solve problems that the algorithm never saw before [83].

3.2 Deep Reinforcement Learning

Reinforcement Learning (RL) is learning how to map situations to actions so as to maximise a numerical reward signal [105]. In RL, any decision-maker (learner) is called an agent, and anything outside the agent is called the environment. The agent senses the environment frequently and interacts with it to optimise its own goals. Figure 3.1 describes the interactions between the agent and the environment. At time step t , the state of the environment is denoted as S_t . The agent senses this state and performs a corresponding action A_t . Then, the environment state changes to S_{t+1} and a reward R_{t+1} is generated as a feedback to the agent [89].

Beyond the agent and the environment, there are two main concepts in RL: a policy and a value function. 1) A policy π is a learning agent's way of behaving at a given time, which defines how to map the perceived state to the action [105]. By interacting with the environment, the agent can optimise its policy; if the selected action is followed by a poor reward, then the policy will assign a low possibility to this action, while if it is followed by a good reward, then the policy will assign high possibility. The policy is the core of an RL agent as it alone is sufficient to determine behaviour [105]. 2) A value function calculates the long-term reward expectation [105]. To get a good result in the long run, the exploitation and exploration dilemma should be paid attention since we need to explore the states as much as possible and exploit the best ones from

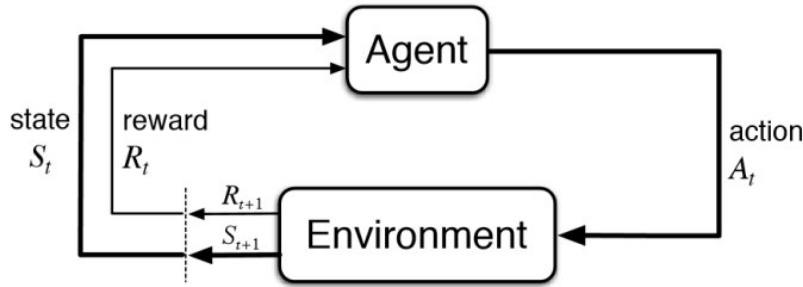


FIGURE 3.1. Overview of Reinforcement Learning.

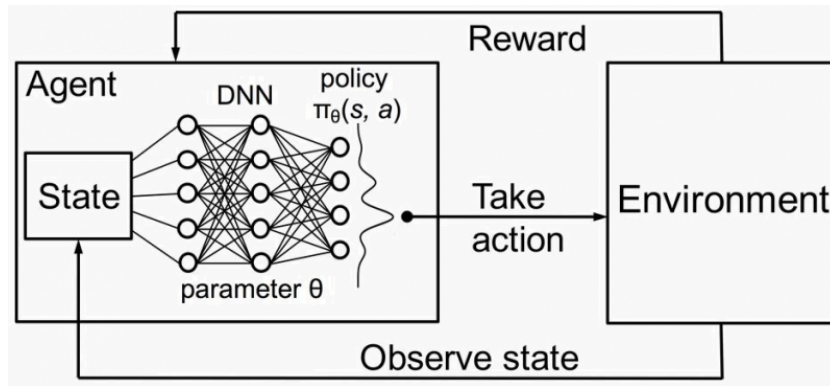


FIGURE 3.2. Overview of Deep Reinforcement Learning.

visited states.

By combining with Deep Learning (DL), RL achieves big success in recent years, for example, Google's DeepMind applied DRL to the Go play and the AlphaGo beats the best Go player [106]. The role of DL in DRL is to use the powerful representation capabilities of neural networks (NN) to suit the RL agent's strategy (mechanism shown in Figure 3.2) [107]. NN is composed of node layers, including the input layer, one or more hidden layers, and the output layer. Each neuron, connected with others, has an associated weight and bias as its own linear regression function [108]. Learning from the training data, the NN adjusts the weights and biases of each neuron to achieve outputs.

DRL is mainly used for making decisions in high-dimensional environments [89]. Thus, it can be applied to solve various optimization problems automatically [83] and recent advances of the DRL have been extended to solve NP-hard problems [89]. The action space of the DRL is the search space in mathematical optimisation problems, and it usually has a large dimensionality. In such a context, it usually relies on a policy-based learning method to effectively map an instance of the problem (state) to a solution (action) [109]. Because of the exploring characteristic of DRL training, the trained model has a strong generalisation property and can solve problems that it has never seen before.

DRL has also been applied to the research on multi-objective optimisation due to large practical problems having multiple objectives. MORL problems necessitate a learning agent obtaining action policies that can simultaneously optimise two or more objectives [79]. According to the number of learned policies, MORL algorithms can be divided into two classes. One is the single-policy MORL approach, and the other is the multiple-policy MORL approach [79]. In the single-policy MORL approach, the reward signal is a scalar and the best single policy representing users' preferences is going to be found. In the multi-policy MORL approach, the reward signal is a vector, and a set of policies that represents a set of objectives are learned together to approximate the true PF [79].

In this thesis, MORL is adopted to find the PF of the SFC placement problem. The simplest way to achieve multi-policy RL is a weighted sum, that is, converting vector reward to a set of weighted-sum scalar rewards. By running these scalar optimisation subproblems, the desired PF can be solved [83].

3.3 Distributed Control and Congestion Game

Fundamentally, edge computing architectures are based on existing distributed system technologies and established paradigms [15]. It means that not only centralised control, but also distributed control can be established. Compared to the centralised control where the edge nodes rely heavily on the central DC to manage and orchestrate edge resources and networking resources [15], the distributed control services reside on both edge nodes and DCs to provide more scalable and flexible control and management [15]. Distributed systems have a long and illustrious history on the internet. Interior gateway protocols like Open Shortest Path First (OSPF) and Intermediate System-Intermediate System (IS-IS) are fully distributed. BitTorrent and other distributed file-sharing applications have had a significant influence on content sharing. All of these technologies aim to improve their core performance by removing centralised entities [68].

The distributed control brings several advantages for the NFV-RA in edge-cloud networks. They can be summarised as follows: 1) the distributed scheme offers scalability for a large number of MEC nodes, the growth of the service requests, and the dynamic network changes; 2) single-point-of-failure can be avoided since many agents reduce the failure possibility for the entire resource allocation scheme; 3) privacy can be guaranteed without disclosing private information to the central controller or the third-party controller; 4) the distributed scheme provides solutions considering the user-specific and individualistic behaviour and capture the non-cooperative behaviour among competitive and individualistic entities, which is motivated by self-interest rather than a shared goal; 5) due to the NP-hardness of the NFV-RA problem, distributed solutions reduce the computational time to get competitive solutions [68, 71]. However, it also faces several drawbacks, for example, the overhead to synchronise among distributed

controllers [15].

For the distributed algorithm design, game theory is widely used [70]. It can be used to address the problem of choosing the optimal decisions for each player in the presence of competition, cooperation, or conflict [107]. A game model is usually made up of players, a set of strategies, and utility functions. In general, the players who make decisions are dependent on the choices of others. In game theory, players compete against each other in a series of turns to maximise their payoff until they reach the Nash Equilibrium (NE). A NE is a stable scenario in which no player has an incentive to deviate from its chosen strategy after considering the decisions of other opponents [107].

Among all the game models, congestion games have been proposed in many networking scenarios because of their ability to capture network resource congestion. Congestion games are a type of game in which a group of players competes against each other for a limited number of resources [110]. Each player's strategy in the congestion game consists of selected resources based on the cost of these resources, which can be reflected as the congestion over such resources. The payoff of each player is determined by the number of players who choose the same resource as it chooses. The congestion game also has a lot of appealing properties: 1) it possesses a Pure strategy Nash Equilibrium (PNE) [110]; 2) the Finite Improvement Property (FIP) makes every improvement path finite, allowing any improvement sequence to lead to a PNE [23]. Because it holds a potential function, which is a real-valued function that records the changes in the cost functions of players which unilaterally deviate from a strategy to another [111].

3.4 Multi-Agent Reinforcement Learning

Multi-agent Reinforcement Learning (MARL) has recently attracted great attention because there are more and more situations including the cooperation and competition among multiple agents, such as multiplayer games and unmanned aerial vehicles (UAVs) [89]. In MARL, a set of agents interact with the common environment and in some mechanisms, they can communicate with each other [89].

The MARL problem is generally formulated as a Stochastic Game [88]: the number of agents can be denoted as n , a discrete set of environmental states as S , and a set of actions for each agent as $A_i, i = 1, 2, \dots, n$. The joint action set including actions of all agents is represented as $A = A_1 \times A_2 \times \dots \times A_n$. The state transition probability function can be defined by $p : S \times A \times S \rightarrow [0, 1]$ and the reward function is specified as $r : S \times A \times S \rightarrow \mathbb{R}^n$.

Extending from single-agent to multi-agent environment, there are several challenges that must be considered: 1) non-stationarity due to the agents simultaneously acting; 2) centralised control or decentralised control when the joint action space grows exponentially with the number of agents; 3) full observability or partial observability; 4) Cooperative, competitive, or mixed environments [88, 89, 112, 113]. Unlike a single-agent system, the rewards and the environment

transition probabilities of an agent may not be stationary, as they depend on the other agents' actions and their updated policies, respectively. The Markov property does not hold anymore and the convergence property is not guaranteed in most MARL problems [114]. Moreover, in some privacy cases, each agent can only access its local observation and not be allowed to share such information with other agents [113]. Therefore, the centralised training and decentralised execution method, proposed independently in [90, 91], are widely used in recent research works to ease implementation and improve stability [115]. Compared to this method, the fully centralised method suffers from scalability problems, and the fully decentralised method cannot guarantee convergence.

The centralised learning and decentralised execution scheme has recently become a standard paradigm in MARL [89, 90]. Taking into account the ease of training process and convergence performance, all agents are trained together by a centralised method having fully observations, while considering the communication constraints and partially information observability during execution time, decentralised execution is adopted where each agent can take actions based on its own observations [89]. In both [91] and [90], authors propose an actor-critic policy gradient method for Q-learning, allowing the critic to use the policies of other agents for the training purpose, while, the actors to use their local information for the execution in a decentralised manner. There are two separate neural networks involved in the actor-critic algorithm: 1) actor-network: taking action according to the local observation and transferring to the critic-network for evaluation; 2) critic-network: using a Temporal Difference (TD)-error function to indicate the future tendency of the selected action [89].

As for the relationship among agents, there are four types: 1) cooperative; 2) competitive; 3) mixed cooperative and competitive; 4) self-interested [116]. In a fully cooperative setting, all agents usually share a common reward function, i.e., $R_1 = R_2 = \dots = R_n$ [116]. Authors in [90] solve a fully cooperative multi-agent task as a stochastic game. In a fully competitive setting, multi-agent tasks are typically modelled as zero-sum stochastic/Markov games. A mixed setting can be modelled as the general-sum stochastic game [117]. When each agent is self-interested, each agent maximises its own objective, and their rewards may be conflicting. The methods proposed in [91], can be applied to both cooperative and competitive environments.

3.5 Management Tools

3.5.1 OpenStack

OpenStack is a cloud computing platform developed by NASA for the control of large pools of computing, storage, and networking resources [119]. It enables administrators and researchers to deploy Infrastructure as a Service (IaaS) infrastructure. As shown from Figure 3.3, OpenStack provides a series of Application Programming Interfaces (APIs) to allow applications to access the underlying infrastructure resources and fulfil control and management functions. From

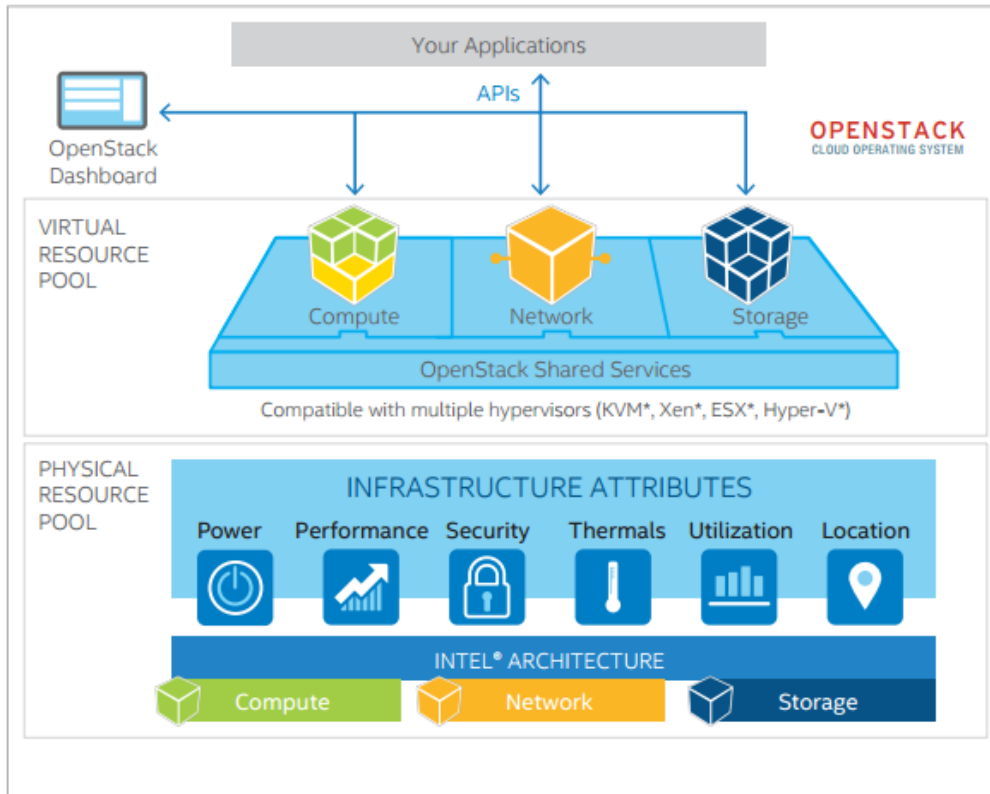


FIGURE 3.3. OpenStack Cloud Operating System [118].

this figure, we can also see an OpenStack dashboard where administrators control and user resources provision is supported [120]. In addition to resource provision, control, and management, OpenStack enables orchestration, fault management, and service management [120].

The main characteristics of OpenStack are 1) Scalable: as a worldwide solution implemented in various companies, it is capable of managing up to 1 million physical machines, 60 million virtual machines, and billions of stored objects [119]. 2) Compatible and Flexible: it supports most virtualization solutions of the market, including ESX, Hyper-V, KVM, LXC, QEMU, UML, Xen, and XenServer, and it can be applied in both private and public clouds [121]. 3) Open: it is an open-source technology whose codes can be modified, and it also provides a validation process for new standards [119]. With these features, OpenStack has great potential in the 5G communication system.

Figure 3.4 shows components of OpenStack. Among them, OpenStack Compute, Image, and Object are three main components [119]. OpenStack Compute, named Nova, is a management platform that controls the access, servers, and networks. It is essential for OpenStack as it provides IaaS by controlling the hypervisor services, handling the lifecycle management of instances, creating VLAN/DNS/Bridges and firewall rules, and determining where tasks should be executed [119, 123]. Glance is the OpenStack Image service providing storage services, recording

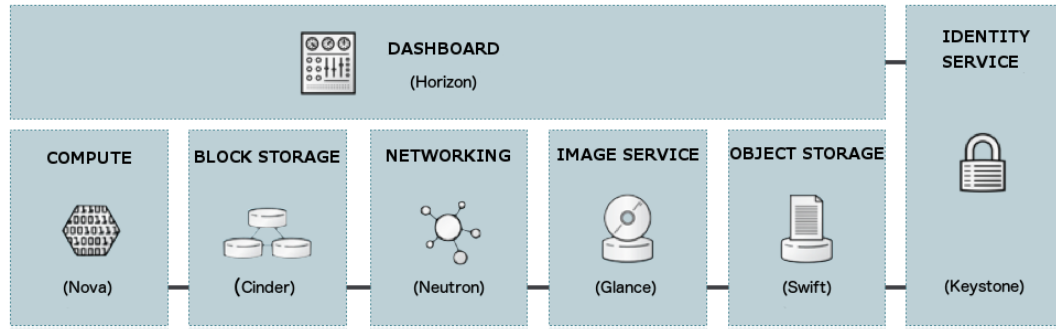


FIGURE 3.4. Components of OpenStack [122].

and distributing the images to VM disks, and allowing querying of VM image metadata and catalogue through APIs [121]. Swift is the OpenStack Object Storage used to store petabytes of data with multiple access points on distributed architecture to avoid Single Point of Failure (SPoF) [119].

Other components include the OpenStack dashboard (Horizon), authentication system (Keystone), networking services (Neutron) [119]. Neutron provides network connectivity among devices managed by Nova, which enables the management of dynamic host configuration protocol (DHCP), static Internet protocol (IP), and virtual area networks (VLAN) along with other advanced policies [121]. All of the configuration and management information generated by OpenStack are stored in the MySQL database management system by default [121]. In the implementation, MySQL is installed on the controller of the network, and it is essential for basic cloud services.

3.5.2 Open Source MANO

In this subsection, MANO will be explained in detail. The MANO framework is proposed by ETSI NFV Industry Specification Group (ISG) [18]. It enables an integrated and holistic approach to the management of resources and network services [24]. To be more specific, it controls the lifecycle and configuration of VNFs, NSs, network slices, manages and orchestrates the virtualized resources, and monitors the network performance [124]. In addition, a database is included in the MANO system for the storage of resource and network service-related information, such as the lifecycle properties of VNFs and NSs, data models for the NSs deployment, as well as resource status [21].

Open Source MANO (OSM) is a solution to the proposed MANO framework. It is an orchestration and management system and aims to support the broadest range of NFVI, Virtualised Infrastructure Managers (VIMs), WAN Infrastructure Managers (WIMs) as well as the broadest possible range of VNFs, NSs, and network slices [124]. VIMs control the VMs and containers while WIMs control the virtual links [124]. Hence, as we can see from Figure 3.5, OSM acts as an

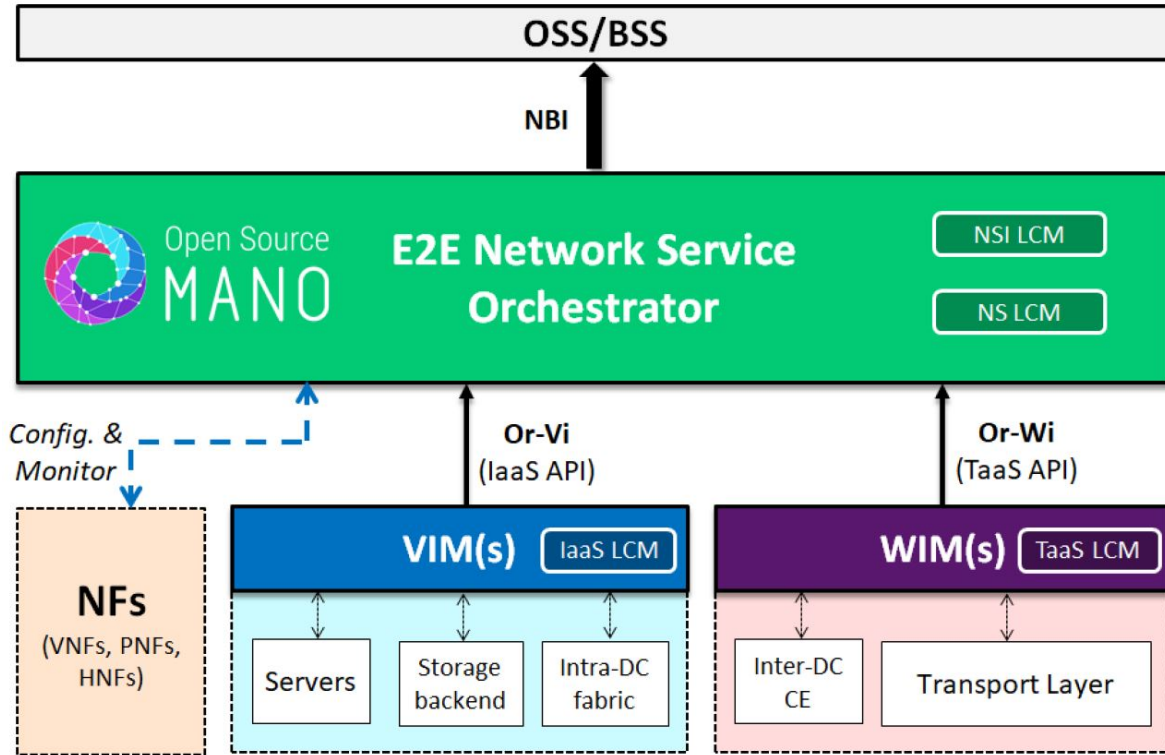


FIGURE 3.5. OSM in Service Platform View [124].

E2E NS orchestrator in the network.

The stages of the lifecycle of the NS in the E2E NS orchestrator include modelling, onboarding, creation, operation, and finalisation [124]. OSM's Information Model (IM) is capable of modelling, and the models can be used as the templates for NSs creation. Then, the NS and NF packages will be onboarded in OSM. After creating an NS, OSM will control the NS instance-related Read, Update, and Delete operations.

At the top of Figure 3.5, a unified Northbound Interface (NBI) facing OSS/BSS of the network is provided by the OSM. By integrating with the existing BSS, it provides a commercial gateway to an NS. With the existing OSS, it will treat an NS as a component in an existing service composition framework [12]. The NBI enables the full control of NSs. It can provide all the necessary abstractions for the control, operation, and supervision of the NSs and network slices lifecycle and hide unnecessary details of its constituent elements at the same time [124].

At the bottom of Figure 3.5, the connections between OSM and NFs, VIMs, and WIMs are shown. Through them, OSM gets access to the hosted functions and infrastructures. NSs are composed of VNFs, physical network functions (PNFs), and hybrid network functions (HNFs) in different domains, such as private infrastructure, public cloud, access network, and transport network. To enable the control of NSs spanning across different domains, there are broad types

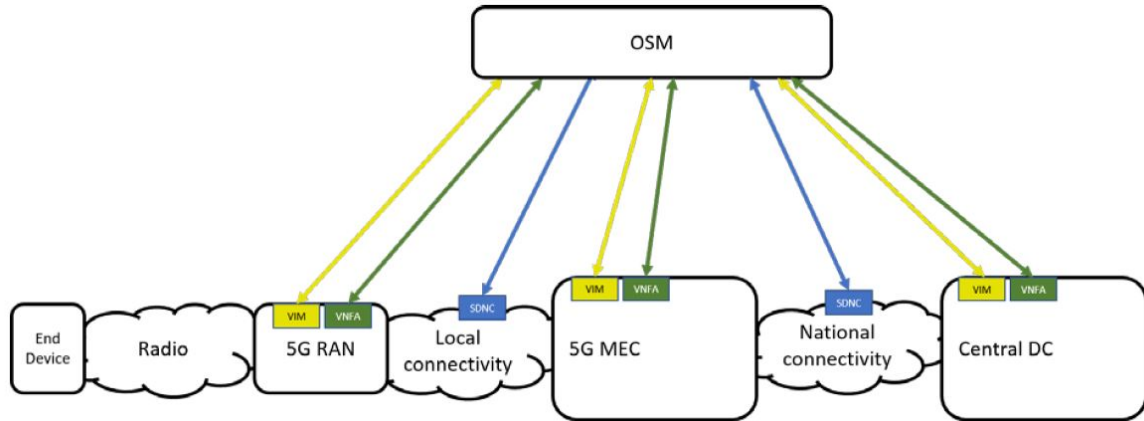


FIGURE 3.6. Next-Generation 5G Architecture [12].

of OSM Southbound, which can work with VIM interfaces (OpenStack, VMWare, OpenVIM, and Kubernetes), SDN interfaces (ODL, ONOs, and Floodlight), VNF interfaces and domain interface [12, 124].

Figure 3.6 shows how the OSM controls the E2E network and services in the 5G architecture. From the RAN to the MEC, from the MEC to the central DC, OSM controls the computing resources and VNFs through the connection with the VIM, which interacts with infrastructure to manage the lifecycle, and the VNFAccess (VNFA), which interacts with VNF instances to manage configuration and monitoring. In addition, OSM controls the networking resources and connections by interacting with the SDN Controller (SDNC) to configure and manage network infrastructures and connectivity [12].

The OSM fulfils its resource management and orchestration function from two aspects. On the one hand, it provides the location of VNFs, the topology of existing NSs, and the resource utilization of VMs, containers, and servers to the planning tools, which can decide the composition and resource allocation of new NSs [21, 124]. On the other hand, OSM's IM provides a mechanism for different providers to describe the internal topology, required resources, procedures, and lifecycle of their NFs in the NF Packages [124].

SINGLE-OBJECTIVE OPTIMISATION FOR SFCs PLACEMENT

The latency-aware SFCs placement problem is crucial to flexibly support ultra-low latency services and efficiently use limited resources in MEC nodes. In this chapter, this problem is studied in a multi-layer edge-cloud network scenario. Service requests with different data rates, E2E latency, and VNFs are placed on MEC nodes, edge DCs, and core DCs interconnected via optical links. By adopting different mathematical latency calculation methods for the MEC nodes and DCs in the designed MILP model, both MEC resources can be better utilised, and ultra-low latency requirements can be satisfied under different workloads. For solving large-scale problems, a *Data Rate-based Heuristic Algorithm* is designed, which can reach ≤ 1.5 approximation ratio under simulation scenarios and at least 1.7 times as much service acceptance ratio as the first-fit baseline algorithm.

4.1 Introduction

The combination of MEC, NFV, and optical technology is necessary to satisfy the low latency and high capacity requirements of the 5G network. As more and more upcoming network applications are time-critical (e.g., AR, smart manufacturing, real-time gaming), MEC technology can support them at a place closer to end-users, and optical networks can support high-speed transmission. Furthermore, the virtualisation capability brought by NFV guarantees flexible service processing according to their latency requirements.

To leverage the advantages provided by the combination of technologies, several challenges have to be addressed for solving the SFC placement problem in edge-cloud networks: 1) the hierarchical network characterised by different resource capacities in different network domains makes the resource allocation more difficult than before [125]; 2) the various QoS requirements of network services such as E2E latency and data rate need to be satisfied in the same network;

3) the optical layer and its related constraints should be considered because it supports high-capacity and high-speed communications (around 1 Gbps data rate) [3] and OEO conversion latency around 100 μ s cannot be ignored [126] for ultra-low latency services.

In this chapter, the challenges mentioned above are addressed. Firstly, a multi-layer MILP model including virtual function layer, IP layer, and optical layer is designed to minimise the total service E2E latency consisting of processing, queueing, transmission, propagation, and OEO conversion latency. Despite numerous efforts that have been made to address the latency-aware SFCs placement problem, the optical layer and the hierarchical network characteristic have not been studied together in the existing MILP models [8, 43, 46, 51, 52, 55]. However, considering both of them is essential to the problem in the realistic edge-cloud networks. The proposed MILP model in this chapter narrows such gaps by adding the optical layer and adopting different mathematical models for MEC nodes and DCs. Secondly, a heuristic algorithm called the *Data Rate-based Heuristic Algorithm* is designed based on the optimal resource allocation pattern obtained from the MILP model to solve a large-scale problem.

The main contributions can be summarised below as:

1) A novel multi-layer MILP model is designed, and the optical layer is added for the first time. Optical layer-related constraints, such as the wavelength resource limitation, the mapping of virtual link to optical link, and the wavelength continuity constraint, are properly modelled. Taking into consideration the computing resource hierarchical property, the M/M/1 queueing model is only used to calculate processing, queueing, and OEO conversion latency at MEC nodes. While the resource capabilities of DCs are larger, neither queueing nor OEO conversion latency is taken into account.

2) A heuristic algorithm is designed to serve more ultra-low latency services with limited MEC node resources. It can replace VNFs from MEC nodes to DCs when the MEC nodes are overloaded. Its performance is compared with the MILP model in the small-scale network and with the first-fit heuristic algorithm in the large-scale network. Simulation results prove that it can both maximise the MEC resource utilisation and improve the SAR for ultra-low latency services.

The rest of this chapter is organised as follows. Section 4.2 states the problem and formulates the MILP model. Section 4.3 introduces simulation settings and analyses the simulation results of the MILP model. Section 4.4 explains the *Data Rate-based Heuristic Algorithm* and discusses its performance. Finally, Section 4.5 concludes this work.

4.2 MILP Formulation for Single-Objective SFCs Placement

4.2.1 Network Topology and Problem Statement

We consider a three-level network topology, encompassing access, metro, and core networks, as illustrated in Figure 4.1. In this network, computing nodes are categorised as MEC nodes, Edge

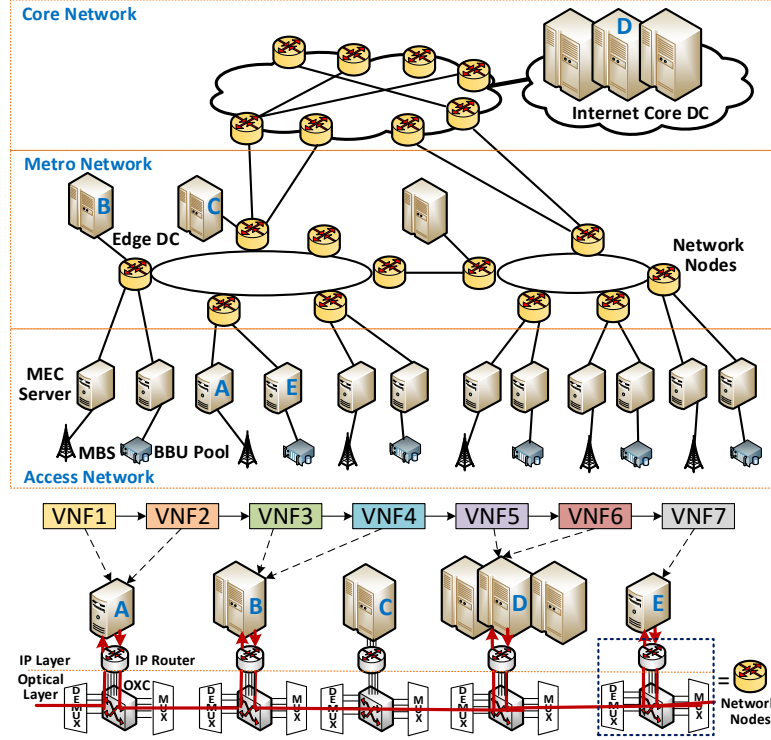


FIGURE 4.1. 5G Network Topology and SFCs Placement [2]

DCs (EDCs), and core DCs (CDCs) based on their locations and resource capacities. MEC server is expected to be located close to 4G/5G base stations (i.e., eNodeBs) or macro-base stations (MBS) [25, 127], while EDC and CDC are located at the metro and core network, respectively. EDC and CDC have far more resources than the MEC server, which has limited computing and storage resources. Other network nodes are switching nodes that have no computing resources. Each network node contains an IP router and an Optical Cross Connector (OXC) [128]. All the network nodes are connected through Wavelength Division Multiplexing (WDM) optical links.

The latency-aware SFCs placement problem can be stated as follows. Given the hierarchical 5G network topology, resource capacities, and all the SFC requests, we need to map VNFs on computing nodes and map virtual links to optical links to minimise the service E2E latency for all demands while satisfying all the constraints. The bottom of Figure 4.1 shows an example of the SFC placement. There are seven different VNFs in this SFC, and they are placed on MEC nodes (A and E), EDCs (B and C) and CDC (D) according to the placement results. If two neighbouring VNFs are placed at different computing nodes, after the data is processed by the first VNF, the electrical traffic will be converted to optical traffic before being aggregated with

other optical traffic by a multiplexer (MUX). Then, combined optical traffic will be transmitted via a single wavelength λ along the lightpath. When it reaches the destination node, optical traffic is dropped by a demultiplexer (DEMUX) and converted to electrical traffic, and then processed at the computing node where the second VNF is placed.

The E2E latency for the single service includes:

- 1) Processing latency: the time period data is processed at the computing nodes;
- 2) Queueing latency: the time period data is queued in an electronic buffer, which only happens at the MEC servers in our MILP model;
- 3) Transmission latency: the time period the whole packet is transmitted from node to link;
- 4) Propagation latency: the time period traffic transmitted in each WDM link;
- 5) OEO conversion latency: the time period signals converted from optical to electrical, and vice versa, are all called OEO conversion latency in this chapter.

4.2.2 MILP Formulation

4.2.2.1 Input Parameters

To formulate the latency-aware SFC placement problem, the network infrastructure is coded as follows.

The network infrastructure is modelled as a directed graph $G = (N, L^P)$, with N representing the set of nodes, and L^P representing the set of optical links. Among all of the nodes, N_v represents the node with computing resources, and N_{SWN} represents the node with switching capability only. N_v includes MEC servers, EDCs, and CDC, denoted as N_{MEC} , N_{EDC} , and N_{CDC} , respectively. MEC servers have much lower resources compared to DCs and are located near 5G base stations (i.e., eNodeBs) [25]. The switching node in this model includes the router and the OXCs.

A processing node $n \in N_v$ is characterised by the computing resource, buffer, and OEO conversion-related resource (including OE and EO), denoted as n^{cpu} , n^{buf} , and n^{oeo} , respectively. As CPU is usually defined as the bottleneck in most VNFs, while other hardware resources are relatively sufficient in most cases [43], the storage resource is not included. Switching node $n \in N_{SWN}$ has only OEO conversion-related resources. The different resource utilisation ratio on node n is u_n^{cpu} , u_n^{buf} , u_n^{oeo} , individually.

A physical link $(n, n') \in L^P$ indicates a link from node n to n' , with the length $len_{(n, n')}$. The total number of wavelength in the network is W and the w -th wavelength in the set of wavelength is $w \in [1, |W|]_z$. ($|\cdot|_z$ represents the number of elements in the set, and $[A, B]_z$ represents the set of integers from A to B). In the physical link (n, n') , the transmission capacity of w -th wavelength is $B_{(n, n')}^w$. In optical fibre, the light speed is lv . A lightpath (l, l') from node $l \in N_v$ to node $l' \in N_v$ belongs to the set of lightpath L^{lp} . $Q_{(l, l')}$ represents the number of lightpaths from node l to l' . We use $q_{(l, l')}$ to denote the q -th lightpath on (l, l') , $q \in [1, |Q_{(l, l')}|]_z$.

F defines the set of different types of VNFs, and $m \in F$ stands for the specific m type VNF. Different VNFs require different amount of computing resource, buffer, and OEO conversion-

related resource, denoted as β_m^{cpu} , β_m^{buf} , β_m^{oeo} , respectively. In addition, the scaling attribute of different VNF is different. We use δ_m to represent this attribute. The output data rate v_{output} for each flow is determined by the input data rate v_{input} and the scaling attribute, which can be computed by $v_{output} = \delta_m \cdot v_{input}$.

The symbols that represent the MILP model's input parameters are provided in Table. 4.1. These input variables include all the variables stated above, such as network topology, node and link resources, and VNF attributes. In addition, service requests are also the input variables. The set of service requests is K and the chain of VNFs makes up each service request $k \in K$. The number of VNFs required by the service request k is O_k . The o -th VNF in the SFC k is represented by $o \in [1, |O_k|]_z$. In addition, service request is also characterised by the source node $s \in N_v$ and destination node $d \in N_v$, data rate v^k , required E2E latency DR^k , and packet size TS^k .

The o -th VNF in the service request k has a data rate of $v_m^{k,o}$. For the first VNF in SFC, $v_m^{k,1} = v^k$. For the other VNF, $v_m^{k,o} = \delta_{m'}^{k,o-1} \cdot v_{m'}^{k,o-1}$. The computing resource, buffer, and OEO conversion-related resource required by the o -th VNF of service request k , can be represented by $C_m^{k,o,cpu}$, $C_m^{k,o,buf}$, $C_m^{k,o,oeo}$, respectively. The following three equations can be used to compute the required resource amount: $C_m^{k,o,cpu} = \beta_m^{cpu} \cdot v_m^{k,o}$, $C_m^{k,o,buf} = \beta_m^{buf} \cdot v_m^{k,o}$, and $C_m^{k,o,oeo} = \beta_m^{oeo} \cdot v_m^{k,o}$.

After receiving the service request, VNFs should be mapped on physical nodes, and virtual links between VNFs should be mapped to underlying physical links, while satisfying E2E latency requirement and resource capacity limitation [40]. $suit_{m,n}$ is a binary indicator representing whether the VNF can be supported by the node n or not.

4.2.2.2 Output Variables

The resulting placements are represented by the following output variables in Table. 4.2.

4.2.2.3 Objective

The objective function shown in equation (4.1) is the minimisation of service E2E latency D^{total} composed of processing latency DP^k , queueing latency DQ^k , transmission latency DT^k , propagation latency DG^k , and OEO conversion latency DC^k for all demands. It is worth mentioning that this objective can avoid congestion and leave enough resources at the MEC node to support ultra-low latency services because it can route traffic from MEC nodes to DCs when the $DQ^k + DC^k$ at the MEC node is larger than the $DT^k + DG^k$ caused by the transmission from the MEC node to the DC.

$$(4.1) \quad \min D^{total} = \sum_{k \in S} (DP^k + DQ^k + DT^k + DG^k + DC^k)$$

Table 4.1: Input Parameters of Single-objective Optimisation Model

G	directed network graph, $G = (N, L^P)$
N	the set of nodes, including MEC servers N_{MEC} , EDCs N_{EDC} , CDCs N_{CDC} , and switching nodes (SWN) N_{SWN}
L^P	the set of physical links
n	physical node, $n \in N$
(n, n')	physical link connecting node n and n' , $(n, n') \in L^P$
$len_{(n, n')}$	the length of physical link
w	the w -th wavelength in the set of wavelengths W
$n^{cpu}, n^{buf}, n^{oeo}$	computing resources, buffer, and OEO conversion related resources in node n
$u_n^{cpu}, u_n^{buf}, u_n^{oeo}$	computing resources, buffer, and OEO conversion related resources utilisation ratio on node n
$B_{(n, n')}^w$	transmission capacity of each wavelength
(l, l')	lightpath from node l to l' in the set of lightpaths L^p
$q(l, l')$	the q -th lightpath in the set of lightpaths $Q_{(l, l')}$
m	the m -th type of VNFs in the set of VNFs F
$\beta_m^{cpu}, \beta_m^{buf}, \beta_m^{oeo}$	computing resources, buffer and O/E/O conversion related resources required by the VNF
δ_m	the scaling attribute of the VNF
$suit_{m, n}$	a binary variable indicating whether the VNF can be supported by node n or not
k	the k -th service request in the set of all service requests K
$l_{s, d}^k$	the link between source and destination node of the service request
O^k	the number of VNFs required by the service request
o	the o -th VNF in the service request, $o \in [1, O_k]_z$
v^k	the data rate required by the service request
$v_m^{k, o}$	the data rate for the o -th VNF in the k -th service request, $v_m^{k, o} = \delta_m^{k, o-1} \cdot v_m^{k, o-1}$ if $o > 1$, else $v_m^{k, o} = v^k$
$C_m^{k, o, cpu}, C_m^{k, o, buf}, C_m^{k, o, oeo}$	computing resources, buffer and OEO conversion related resources required by the VNF, which are proportional to the required data rate, e.g. $C_m^{k, o, cpu} = \beta_m^{cpu} \cdot v_m^{k, o}$
DR^k	the latency required by the service
TS^k	the packet size required by the service
$z_n^{k, o, cpu}, z_n^{k, o, buf}, z_n^{k, o, oeo}$	average processing, queueing, and OEO conversion latency
$z_{n, max}^{k, o, cpu}, z_{n, max}^{k, o, buf}, z_{n, max}^{k, o, oeo}$	the maximum processing, queueing, and OEO conversion latency
$a_r, b_r, c_r, d_r, e_r, g_r$	coefficients for the r -th piecewise linearisation in all function sets R

Table 4.2: Output Variables of Single-objective Optimisation Model

$x_{m,n}^{k,o}$	a binary variable indicating whether the VNF in the service request is placed on node n or not
$y_{(l,l'),q}^{k,o,m}$	a binary variable indicating whether lightpath is selected on the path between the o -th and the $(o+1)$ -th VNF or not
$yw_{(l,l'),q,(n,n'),w}^{k,o,m}$	a binary variable indicating whether wavelength in (l,l') is selected on the path between the o -th and the $(o+1)$ -th VNF or not
$tw_{(l,l'),q,(n,n'),w}^{k,o,m}$	a binary variable indicating whether wavelength in (n,n') is selected on the path between the o -th and the $(o+1)$ -th VNF or not
\hat{f}_n^k	a binary variable indicating whether the node is used by the service chain or not, if $x_{m,n}^{k,1} = 1 \ \forall n \in N_{MEC}$ or $\sum_{o \in [1, O_k -1]_z} \sum_{(l,l') \in L^{lp}} \sum_{q \in Q_{(l,l')}} \sum_{w \in [1, W_k]_z} \sum_{a \in N} tw_{(l,l'),q,(n,a),w}^{k,o,m} + \sum_{o \in [2, O_k]_z} \sum_{(l,l') \in L^{lp}} \sum_{q \in Q_{(l,l')}} \sum_{w \in [1, W_k]_z} \sum_{a \in N} tw_{(l,l'),q,(a,n),w}^{k,o-1,m} = 1 \ \forall k \in K, n \in N_v$, it equals 1, otherwise, it equals 0

4.2.2.4 Constraints

a) VNF Placement

Constraint (4.2) ensures that the o -th VNF can only be mapped on one VNF instance on the node that can support it.

$$(4.2) \quad \sum_{n \in N} x_{m,n}^{k,o} \cdot suit_{m,n} = 1 \quad \forall m \in F, k \in K, o \in [1, |O_k|]_z$$

b) Node and Link Resources

Equations (4.3)-(4.5) guarantee that the required computing resources, buffer, and OEO conversion-related resources do not exceed the available resource capacities of physical nodes. The resource constraint for optical link is represented by (4.6).

$$(4.3) \quad \sum_{k \in K} \sum_{o \in [1, |O_k|]_z} x_{m,n}^{k,o} \cdot C_m^{k,o,cpu} \leq n^{cpu} \quad \forall n \in N$$

$$(4.4) \quad \sum_{k \in K} \sum_{o \in [1, |O_k|-1]_z} \sum_{(l,l') \in L^{lp}} \sum_{q \in Q_{(l,l')}} \sum_{w \in [1, |W_k|]_z} \cdot \sum_{a \in N} tw_{(l,l'),q,(a,n),w}^{k,o,m} \cdot C_m^{k,o,buf} \leq n^{buf} \quad \forall n \in N$$

$$(4.5) \quad \sum_{k \in K} \sum_{o \in [1, |O_k|-1]_z} \sum_{(l,l') \in L^{lp}} \sum_{q \in Q_{(l,l')}} \sum_{w \in [1, |W_k|]_z} \cdot \sum_{a \in N} tw_{(l,l'),q,(a,n),w}^{k,o,m} \cdot C_m^{k,o,oeo} \leq n^{oeo} \quad \forall n \in N$$

$$(4.6) \quad \sum_{k \in K} \sum_{o \in [1, |O_k|-1]_z} \sum_{(l,l') \in L^{lp}} \sum_{q \in Q_{(l,l')}} tw_{(l,l'),q,(n,n'),w}^{k,o,m} \cdot v_m^{k,o} \leq B_{(n,n')}^w \quad \forall n \in N$$

c) Link Mapping

Flow conservation constraint is presented in (4.7). Equation (4.8) makes sure that the wavelength continuity between two VNFs placed on different nodes. Equation (4.9) and (4.10) are the relationship constraints between lightpaths and optical fibre links.

$$(4.7) \quad \sum_{(l,l') \in L^p} \sum_{q \in Q_{(l,l')}} \sum_{w \in [1, |W_k|]_z} \sum_{a \in N} tw_{(l,l'),q,(n,a),w}^{k,o,m} - \sum_{(l,l') \in L^p} \sum_{q \in Q_{(l,l')}} \sum_{w \in [1, |W_k|]_z} \sum_{a \in N} tw_{(l,l'),q,(a,n),w}^{k,o,m} \\ = x_{m,n}^{k,o} - x_{m,n}^{k,o+1} \quad \forall k \in K, o \in [1, |O_k| - 1]_z, n \in N$$

$$(4.8) \quad \sum_{w \in [1, |W_k|]_z} yw_{(l,l'),q,w}^{k,o,m} = y_{(l,l'),q}^{k,o,m} \quad \forall k \in K, o \in [1, |O_k| - 1]_z, (l,l') \in L^p, q \in Q_{(l,l')}$$

$$(4.9)$$

$$tw_{(l,l'),q,(n,n'),w}^{k,o,m} \leq yw_{(l,l'),q,w}^{k,o,m} \quad \forall k \in K, o \in [1, |O_k| - 1]_z, (l,l') \in L^p, q \in Q_{(l,l')}, (n,n') \in L^p, w \in [1, |W_k|]_z$$

$$(4.10) \quad \sum_{n \in N} tw_{(l,l'),q,(n,n'),w}^{k,o,m} - \sum_{i \in N} tw_{(l,l'),q,(n',n),w}^{k,o,m} \\ = \begin{cases} yw_{(l,l'),q,w}^{k,o,m} & \text{if } n' = l' \\ -yw_{(l,l'),q,w}^{k,o,m} & \text{if } n' = l \\ 0 & \text{otherwise} \end{cases} \quad \forall k \in K, o \in [1, |O_k| - 1]_z, \\ (l,l') \in L^p, q \in Q_{(l,l')}, w \in [1, |W_k|]_z, n' \in N$$

d) Latency

The E2E latency includes processing, queueing, transmission, propagation, and OEO conversion latency, which can be calculated as follows. Considering that MEC servers are equipped with much fewer resources compared with DCs, the processing, queueing, and OEO conversion overheads are modelled as the M/M/1 queueing model in MEC nodes, whereas there are no such overheads in DCs [51, 129].

Processing latency DP^k only happens at the computing nodes and can be obtained by equation (4.12).

$$(4.11) \quad u_n^{cpu} = \left(\sum_{k \in K} \sum_{o \in [1, |O_k|]_z} x_{m,n}^{k,o} \cdot C_m^{k,o,cpu} \right) / n^{cpu} \quad \forall n \in N_v$$

$$(4.12) \quad DP^k = \sum_{n \in N} \sum_{o \in [1, |O_k|]_z} x_{m,n}^{k,o} \cdot \frac{1}{(1 - u_n^{cpu}) \cdot n^{cpu}} = \sum_{n \in N} \sum_{o \in [1, |O_k|]_z} x_{m,n}^{k,o} \cdot (a_r \cdot u_n^{cpu} + b_r) \\ = \sum_{n \in N} \sum_{o \in [1, |O_k|]_z} z_n^{k,o,cpu} \quad \forall k \in S$$

The Big-M method is used for the linearisation of the conduct of a binary variable and a continuous variable, and the upper bound of $1/(1 - u_n^{cpu})$ is $z_{n,max}^{k,o,cpu}$. Piecewise linearisation functions equation (4.13) and (4.14) are used to approximate the processing latency [8].

$$(4.13) \quad z_n^{k,o,cpu} \leq x_{m,n}^{k,o} \cdot z_{n,max}^{k,o,cpu} \quad \forall k \in K, o \in [1, |O_k|]_z, n \in N_v$$

$$(4.14) \quad a_r \cdot u_n^{cpu} + b_r \leq z_n^{k,o,cpu} + (1 - x_{m,n}^{k,o}) \cdot z_{n,max}^{k,o,cpu} \\ \forall k \in K, o \in [1, |O_k|]_z, n \in N_v$$

Queueing latency DQ^k and OEO conversion related latency DC^k can be calculated by equation (4.16) and (4.20), respectively. Similar to the linearisation process of processing latency, we use equation (4.17), (4.18), (4.21), and (4.22) to estimate the queueing and OEO conversion latency [8].

$$(4.15) \quad u_n^{buf} = \left(\sum_{k \in K} \sum_{o \in [1, |O_k| - 1]_z} \sum_{(l,a) \in L^{lp}} \sum_{q \in Q_{(l,a)}} \sum_{w \in [1, |W_k|]_z} \cdot \sum_{(n,n') \in L^p} tw_{(l,a),q,(n,n'),w}^{k,o,m} \cdot C_m^{k,o,buf} \right) / n^{buf} \quad \forall l \in N_v$$

$$(4.16) \quad DQ^k = \sum_{n \in N_{MEC}} \sum_{o \in [1, |O_k|]_z} z_n^{k,o,buf} \quad \forall k \in K$$

$$(4.17) \quad z_n^{k,o,buf} \leq x_{m,n}^{k,o} \cdot z_{n,max}^{k,o,buf} \quad \forall k \in K, o \in [1, |O_k|]_z, n \in N_v$$

$$(4.18) \quad c_r \cdot u_n^{buf} + d_r \leq z_n^{k,o,buf} + (1 - \hat{f}_n^k) \cdot z_{n,max}^{k,o,buf} \quad \forall k \in K, o \in [1, |O_k|]_z, n \in N_v, r \in [1, |R|]_z$$

$$(4.19) \quad u_n^{o eo} = \left(\sum_{k \in K} \sum_{o \in [1, |O_k| - 1]_z} \sum_{(l,a) \in L^{lp}} \sum_{q \in Q_{(l,a)}} \sum_{w \in [1, |W_k|]_z} \cdot \sum_{(n,n') \in L^p} tw_{(l,a),q,(n,n'),w}^{k,o,m} \cdot C_m^{k,o,o eo} \right) / n^{o eo} \quad \forall l \in N_v$$

$$(4.20) \quad DC^k = \sum_{n \in N_{MEC}} \sum_{o \in [1, |O_k|]_z} z_n^{k,o,o eo} \quad \forall k \in K$$

$$(4.21) \quad z_n^{k,o,o eo} \leq x_{m,n}^{k,o} \cdot z_{n,max}^{k,o,o eo} \quad \forall k \in K, o \in [1, |O_k|]_z, n \in N$$

$$(4.22) \quad e_r \cdot u_n^{o eo} + g_r \leq z_n^{k,o,o eo} + (1 - \hat{f}_n^k) \cdot z_{n,max}^{k,o,o eo} \quad \forall k \in K, o \in [1, |O_k|]_z, n \in N, r \in [1, |R|]_z$$

Transmission latency DT^k happens only at the source node of the lightpath.

$$(4.23) \quad DT^k = \sum_{o \in [1, |O_k| - 1]_z} \sum_{(l,l') \in L^{lp}} \sum_{q \in Q_{(l,l')}} \sum_{w \in [1, |W_k|]_z} \cdot \sum_{(n,n') \in L^p} tw_{(l,l'),q,(n,n'),w}^{k,o,m} \cdot TS^k / v_m^{k,o} \quad \forall k \in S$$

Propagation latency happens at every chosen physical link for the traffic routing.

$$(4.24) \quad DG^k = \sum_{o \in [1, |O_k| - 1]_z} \sum_{(l,l') \in L^{lp}} \sum_{q \in Q_{(l,l')}} \sum_{w \in [1, |W_k|]_z} \cdot \sum_{(n,n') \in L^p} tw_{(l,l'),q,(n,n'),w}^{k,o,m} \cdot len_{(n,n')} / ls^w \quad \forall k \in K$$

Constraint (4.25) ensures that the total latency experienced by all SFC functions does not exceed its E2E latency requirement.

$$(4.25) \quad (DP^k + DQ^k + DT^k + DG^k + DC^k) \leq DR^k \quad \forall k \in K$$

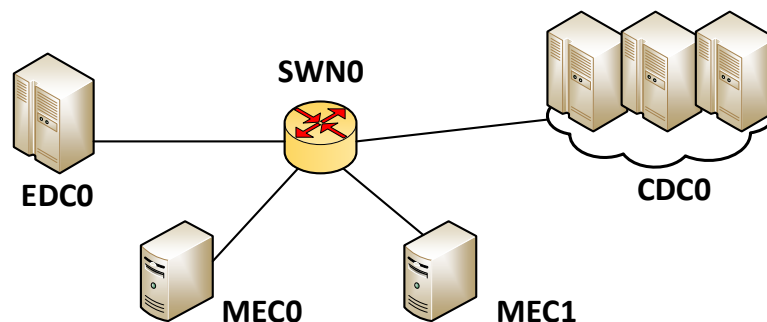


Figure 4.2: 5-Node Network Topology

4.3 MILP Simulation Evaluation

4.3.1 Simulation Setup

The MILP model is evaluated on a small-scale network with two MEC servers (MEC0 and MEC1), one EDC (EDC0), one CDC (CDC0), and one switching node (SWN0) shown in Figure 4.2. We assume that each MEC node has 512 CPU cores [1], EDC and CDC have 2560 and 5120 CPU cores, which are 5 and 10 times the CPU cores in the MEC node, respectively [55]. On MEC nodes, the buffer is set to 2048 units [3]. According to equation (4.20), the OEO conversion-related resources are set to 200 units to make sure that all the 1ms services can be accepted in our model even when all the resources are 98% used. There are 4 bidirectional physical links, each link with 4 wavelengths and a capacity of 25 Gbps for each wavelength. The length of physical links (N_{MEC0}, N_{SWN0}), (N_{MEC1}, N_{SWN0}), (N_{SWN0}, N_{EDC0}), and (N_{SWN0}, N_{CDC0}) are 10km, 10km, 25km and 300km, respectively.

There are 6 service types evaluated, including Cloud Gaming, Augmented Reality (AR), Voice over Internet Protocol (VoIP), Video Streaming, Massive Internet of Things (MIoT), Smart Manufacturing, and Non-real Time services. Table. 4.3 lists their percentage, data rate, E2E latency, and VNF chains requirements. The size of these services is randomly generated from [1MB, 2MB] according to uniform distribution [49]. For the source and destination setting, Smart Manufacturing and MIoT services have the same source and destination MEC node, AR services have the different MEC node as source and destination, while other types of services have the randomly chosen source and destination.

There are 11 VNF types taken into account, including Data Pre-processing (DP), Network Address Translation (NAT), Firewall (FW), Intrusion Detection (ID), WAN Optimiser (WO), Flow Monitor (FM), Video Transcoder (VT), Application Accelerator (AA), Learning (LR), Motion Control (MC) and Transmitter (TM). The computing resource and buffer required by different VNFs are in Table. 4.4. The OEO resources are all set to 0.001 to prevent exceeding the associated resource capacity. Each SFC includes 6 VNFs and their orders are shown in Table. 4.4. Among

all the VNFs, the 'Transmitter' function consumes no resource but is included in SFCs for the purpose of routing traffic to the destination because it can only be supported by the destination node.

The MILP model is solved by the Gurobi solver [130] on an IBM System, with 24GB RAM and dual-core AMD opteron processor.

Table 4.3: Service Requests Setting [1][2]

Service	Percentage	Data Rate	Latency	VNF Chain
Cloud Gaming	25%	4Mbps	80ms	NAT-FW-VT-WO-ID-TM
Augmented Reality	25%	100Mbps	1ms	NAT-FW-FM-VT-ID-TM
VoIP	1.5%	0.064Mbps	250ms	NAT-FW-FM-FW-NAT-TM
Video Streaming	25%	4Mbps	100ms	NAT-FW-FM-AA-ID-TM
MIoT	7.02%	100Mbps	5ms	NAT-FW-DP-LR-ID-TM
Smart Manufacturing	7.03%	100Mbps	1ms	NAT-FW-MC-TM-TM-TM
Non-real Time	9.45%	(4,100)Mbps	500ms	NAT-FW-WO-LR-ID-TM

Table 4.4: VNF CPU Resource Requirements [1, 3]

VNF	CPU	Buffer	Scale	VNF	CPU	Buffer	Scale
DP	0.003	0.006	0.5	NAT	0.00092	0.00092	1
FW	0.0009	0.00135	1	ID	0.0107	0.0107	1
WO	0.0054	0.0108	1	FM	0.0133	0.0399	1
VT	0.0054	0.0054	2	AA	0.003	0.003	0.5
LR	0.008	0.008	1	MC	0.008	0.016	1
TM	0	0	1				

4.3.2 Simulation Results and Analysis

The average total E2E latency results can be obtained by running the MILP model 10 times on the 5-Node network topology, as shown in Figure 4.3(a). It can be seen that the total latency grows slowly at the beginning because when the total number of service requests is less than 200, the MEC nodes have enough resources, and all the services can be processed at the MEC nodes. When the total number of service requests increases from 200 to 500, the total latency increases at a faster rate because firstly, the MEC nodes are overloaded and the processing latency at MEC nodes is increased, secondly, the transmission latency and propagation latency are added induced by routing services from MEC nodes to other nodes. The maximum number of service requests is 500 because the simulation result for larger than 500 requests cannot be reached by solving the MILP model on the IBM server with limited RAM capability (16 GB).

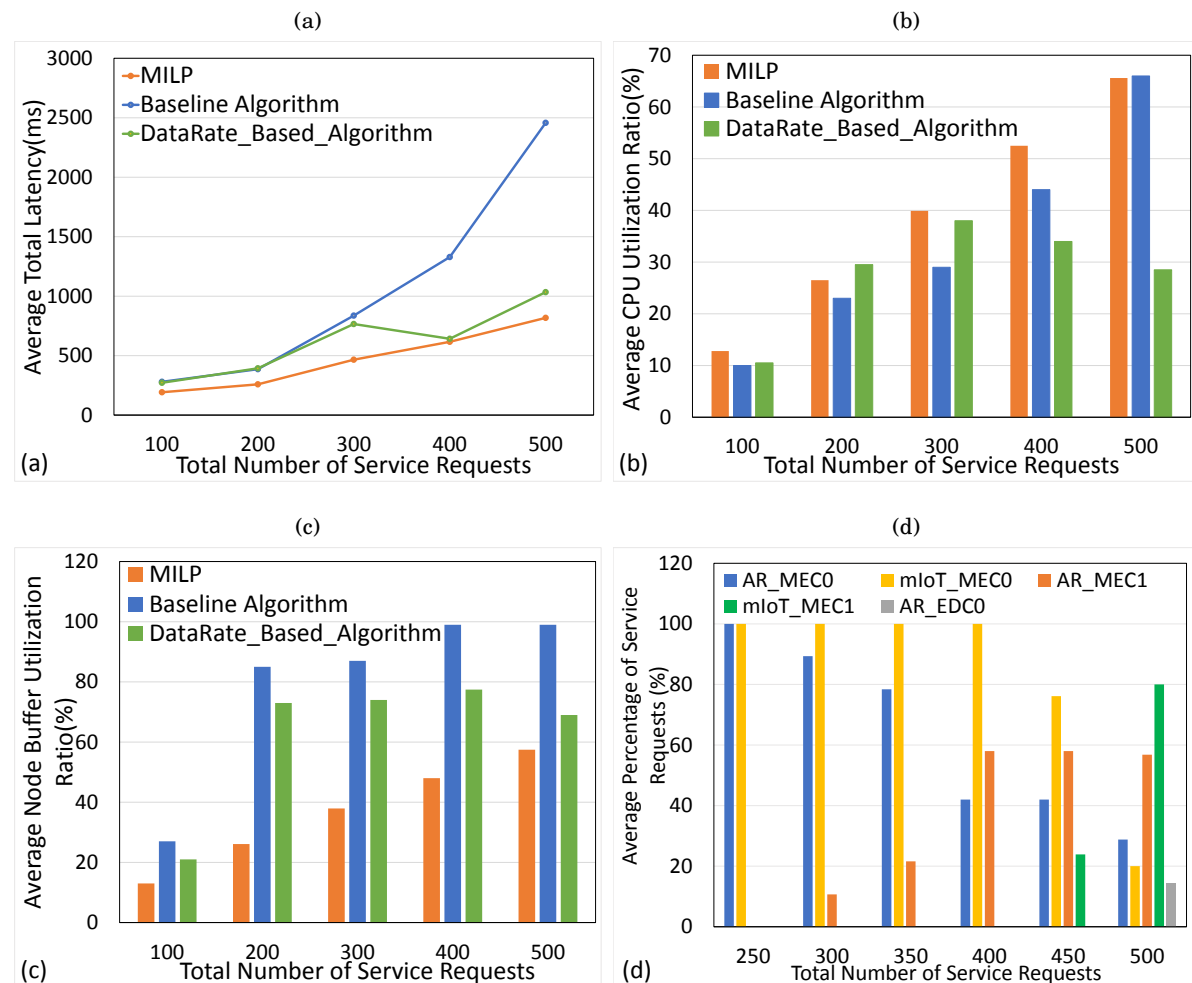


FIGURE 4.3. Simulation Results on 5-Node Topology: (a) Average Total Service E2E Latency (b) Average MEC CPU Utilisation Ratio (c) Average MEC Buffering Resource Utilisation Ratio (d) AR and MIoT Placement Solutions of MILP

The average CPU and buffer utilisation ratios of MEC nodes under different workload conditions are compared in Figure 4.3(b) and Figure 4.3(c), respectively. Both ratios rise steadily as the workload rises. When the number of services is increased to 500, the MEC node CPU utilisation ratio reaches the highest 65% and the buffer utilisation ratio achieves the highest 48%. Moreover, it is worth mentioning that the 38% buffer utilisation ratio at 300 service requests is the point at which the traffic is routed from MEC nodes to DCs because of the lowered overall E2E latency. To be more specific, when the buffer utilisation ratio exceeds 38%, the sum of transmission and propagation latency for routing traffic to DCs is less than the sum of queueing and OEO conversion latency on MEC nodes.

Although we use a simple 5-Node topology, the designed MILP model is still time-consuming. For example, it takes 12.8 hours to finish the 500 SFCs placement, which makes achieving optimal

solutions in large-scale network topology difficult. Hence, we are going to design a heuristic algorithm for large-scale problems in the next section.

4.4 Data Rate-based Heuristic Algorithm

4.4.1 Algorithm Design

By solving the MILP model with all the services employing the MEC0 as the source and destination node, we find that only the AR and MIoT traffic with 100 Mbps (the highest data rate in this simulation) are routed to the neighbouring MEC node or neighbouring EDC0. The traffic placement trends are plotted in Figure 4.3(d). When there are less than 250 service requests, all the services are placed at the MEC0 (source) node. It is worth mentioning that, as the number of service requests increases, AR is firstly routed to the MEC1 and then to the EDC0, while MIoT requests are only routed to MEC1. Other services, even those requiring 500ms E2E latency, are all assigned at the MEC0 node in the optimal simulation results. Therefore, we can conclude that *i)* it is the data rate requirement that affects resource allocation rather than the latency requirement, *ii)* the traffic should be routed to the nearest available node firstly and then routed to the DCs.

Inspired by this regularity, we design the corresponding heuristic algorithm (Algorithm 1) for the large-scale network. It is accomplished in the following two phases.

1) The first step is running the baseline approach to generate the initial solutions (node mapping solution $x_{n,initial}^{k,o,m}$ and link mapping solution $tw_{(n,n'),w,initial}^{k,o,m}$). The baseline algorithm prioritises SFCs that have lower E2E latency requirements. In detail, all the service requests are firstly sorted by their E2E latency requirements. Secondly, the VNF in each service request is placed at the nearest computing node with sufficient resources to the source node. Thirdly, the physical link with sufficient resources is selected for traffic routing between two VNFs. After all the VNFs are placed, the network status is updated. The SFC can be blocked either because of no computing nodes or physical links having enough resources to support the VNF processing or transmission, or the unsatisfied E2E latency requirement. We calculate the total E2E latency and SAR in the initial solutions as the baseline results.

2) The second step is running the *Data Rate-based Heuristic Algorithm*, which takes the initial solutions ($x_{n,initial}^{k,o,m}$ and $tw_{(n,n'),w,initial}^{k,o,m}$) as inputs and then replaces the services with the higher data rate first. In Algorithm 1, according to the regularity discovered by the MILP model, all services are sorted by the data rate in the descending order (line 3) and services with higher data rates will be preferentially replaced. Next, the current node mapping and link mapping solutions are initialised with initial input solutions and the best solutions are initialised with the current solutions (line 4).

For all the VNFs in all the SFC requests, after finding the initial node s supporting this VNF, the other nodes are sorted in ascending order according to the distance from the original

Algorithm 1: Data Rate-based Heuristic Algorithm

```

1 Input: Updated network status, VNF parameters, SFC requests, Initial
   solutions:  $x_{n,initial}^{k,o,m}, tw_{(n,n'),w,initial}^{k,o,m}$ 
2 Output:  $x_{n,best}^{k,o,m}, tw_{(n,n'),w,best}^{k,o,m}$  Service acceptance ratio, Objective: Minimum total latency
3 Sort all the service requests by the descending order of data rate
4 Initialise  $x_{n,current}^{k,o,m} \leftarrow x_{n,initial}^{k,o,m}, tw_{(n,n'),w,current}^{k,o,m} \leftarrow tw_{(n,n'),w,initial}^{k,o,m}, x_{n,best}^{k,o,m} \leftarrow x_{n,current}^{k,o,m},$ 
    $tw_{(n,n'),w,best}^{k,o,m} \leftarrow tw_{(n,n'),w,current}^{k,o,m}$ 
5 for All the SFC requests do
6   for All the VNFs of the SFC request do
7     Find the initial node  $s$  supporting this VNF
8     Sort all the  $N_v$  by the ascending order of the distance from node  $s$ 
9     for  $n$  in  $N_v$  do
10      Initialise  $block = 0$ 
11      if  $suit_{m,n} = 1$  and remaining resources on node  $n$  are enough to support this VNF then
12        Find the shortest path between node  $s$  and  $n$ 
13        if Remaining resource on the shortest path is not enough to support the
           transmission then  $block = 1$ 
14        end if
15      else
16         $block = 1$ 
17      end if
18      if  $block = 0$  then
19        Calculate  $dt_1$  and  $dg_1$  for routing function from node  $s$  to node  $n$ 
20        Calculate  $dq_0$  and  $dc_0$  when the VNF is placed on node  $s$ 
21        if  $(dt_1 + dg_1) < (dq_0 + dc_0)$  then
22           $x_n^{k,o,m} = 1, tw_{(n,n'),w}^{k,o,m} = 1, x_{n,current}^{k,o,m} = x_n^{k,o,m}, tw_{(n,n'),w,current}^{k,o,m} = tw_{(n,n'),w}^{k,o,m}$ 
23          Calculate  $dq_{best}$  and  $dc_{best}$ 
24          if  $(dt_1 + dg_1) < (dq_{best} + dc_{best})$  then
25             $x_{n,best}^{k,o,m} = x_{n,current}^{k,o,m}, tw_{(n,n'),w,best}^{k,o,m} = tw_{(n,n'),w,current}^{k,o,m}$ 
26            Update the network status
27          end if
28        end if
29      end if
30    end for
31  end for
32 end for
33 Initialise  $Service\_Block \leftarrow \emptyset$ 
34 for All SFC requests  $K$  do
35   Calculate the service latency
36   if service latency  $>$  required total latency then
37      $Service\_Block \leftarrow k$ 
38   end if
39 end for
40 Calculate the Minimum total E2E latency and Service acceptance ratio

```

node s (line 8). These nodes are the candidates for the new node to which the VNF will be placed. For each new node candidate, if enough computing resources and bandwidth resources are available on the shortest path between the original node and the new node, the induced transmission latency dt_1 and propagation latency dg_1 for routing the corresponding traffic will be calculated (line 19). The queueing latency dq_0 and OEO conversion latency dc_0 at the original node are also calculated (line 20). Next, the sum of transmission latency and propagation latency (i.e., $dt_1 + dg_1$) is compared to the sum of queueing latency and OEO conversion latency (i.e., $dq_0 + dc_0$) that the VNF on the original node experienced. If $dt_1 + dg_1 < dq_0 + dc_0$, the new node and the shortest physical link will be assigned as the current solutions (line 21 to 22). We also compare $dt_1 + dg_1$ to the lowest sum of queueing and OEO conversion latency obtained so far (i.e., $dq_{best} + dc_{best}$). If $dt_1 + dg_1 < dq_{best} + dc_{best}$, the current solutions are set as the best solutions, and the network status are updated accordingly (line 23 to 26). Such latency comparison can route traffic from the original node to a new node and reduce total E2E latency, which is called the traffic routing mechanism in this algorithm. Finally, the total E2E latency for all demands and SAR are calculated (line 34 to 40).

4.4.2 Algorithm Performance Analysis

The *Data Rate-based Heuristic Algorithm* is firstly run on the 5-Node topology (Figure 4.2) for the performance comparison with optimal solutions provided. All the SFC and VNF parameters are identical to those used in the MILP model simulation. Under various workloads, the results of the baseline algorithm (i.e., the first stage of Algorithm 1), Data Rate-based heuristic algorithm, and the MILP model are compared in terms of average total E2E latency, average MEC CPU utilisation ratio, and average MEC buffer utilisation ratio.

Figure 4.3(a) compares the total E2E latency performance of the MILP, baseline algorithm, and *Data Rate-based Heuristic Algorithm*. Among them, the baseline algorithm performs the worst with the highest E2E latency under all workload conditions. When the total number of service requests is less than 300, *Data Rate-based Heuristic Algorithm* has similar performance to that of the baseline algorithm. After 300 service request point, unlike the total E2E latency of the baseline algorithm, which increases dramatically, the performance of the *Data Rate-based Heuristic Algorithm* improves greatly with much lower E2E latency, which is similar to the optimal solution in the MILP model. Such improvement is achieved by routing services from MEC nodes to DCs, which greatly reduces the total E2E latency (traffic routing mechanism in Algorithm 1). Because the sum of transmission latency and propagation latency induced by routing traffic to the core network is less than the sum of queueing latency and OEO conversion latency at the edge nodes under high workload scenarios.

It is worth mentioning that there is a latency fluctuation of the *Data Rate-based Heuristic Algorithm*. Total E2E latency decreases between 300 and 400 service requests because the traffic is relocated from MEC nodes to DCs and MEC nodes' processing latency falls a lot. It can also be

reflected by the decreased MEC CPU utilisation ratio after 300 in Figure 4.3(b). The proposed algorithm performs the best at 400, with the closest latency to the optimum. After 400 service requests point, the latency increases again and becomes larger than the optimal solutions due to the limited traffic routing capability of Algorithm 1, which compares only the sum of transmission latency and propagation latency and the sum of processing latency and queueing latency. However, compared to the baseline algorithm, the increasing trend after 400 is moderate, which shows the effectiveness of this traffic routing mechanism. The approximation ratio ≤ 1.5 can be achieved in this proposed algorithm. Under the low (≤ 250 total service requests) or high (≥ 350 total service requests) workload scenarios, the algorithm can perform better with ≤ 1.25 approximation ratio.

Figure 4.3(b) and (c) show that the MILP model can place more VNFs at the MEC nodes with the highest average MEC CPU utilisation ratio and lowest average MEC buffer utilisation ratio, while the other two heuristic algorithms are capable of routing more traffic to metro and core networks, which increases the traffic routing back to MEC nodes and consumes more edge buffer resources. The buffer resource usage of the baseline algorithm is always the highest and rises significantly after 200 because it does not consider the source and destination information and route services to the core network even if their source and destination are the same MEC nodes, which significantly increases the edge node buffer usage. Compared to it, the MILP model and *Data Rate-based Heuristic Algorithm* can achieve a lower buffer utilisation ratio to reduce the total queueing latency. As fewer functions are placed on MEC nodes in both algorithms than in the MILP model, the MEC CPU utilisation ratios of the two algorithms are lower than those in the MILP model. But we can see that the maximum CPU utilisation ratios are all 65% in the three approaches in Figure 4.3(b). From Figure 4.3(b), we can also conclude that the *Data Rate-based Heuristic Algorithm* can effectively route traffic from MEC nodes to DCs (lowest CPU utilisation ratio for 400 and 500 service requests) and leave enough edge computing resources for the ultra-low latency services.

Then the baseline algorithm, *Data Rate-based Heuristic Algorithm* and the QoS improvement algorithm in [1] are run on the 35-Node topology shown in Figure 4.1. The box plots of three algorithms' total E2E latency results under 10 independent runs are compared in Figure 4.4(a). Under various workload circumstances, the baseline algorithm is the worst with the highest average total E2E latency. The QoS improvement approach can only reduce the total E2E latency when service requests are larger than 3000. For the rest of the time, it performs the same as the baseline algorithm. When compared to these benchmark algorithms, the *Data Rate-based Heuristic Algorithm* can achieve significantly lower latency. At the point of 4000 total service requests, the average latency of the *Data Rate-based Heuristic Algorithm* is one-seventh and one-fourth of that in the baseline algorithm and QoS improvement algorithm, respectively. Compared to 5-node simulation results (Figure 4.3(a)), a similar average latency fluctuation of the *Data Rate-based Heuristic Algorithm* happens on the 35-node topology between 2500 and 3000 service requests (Figure 4.4(a)) caused by the traffic relocation from MEC nodes to DCs.

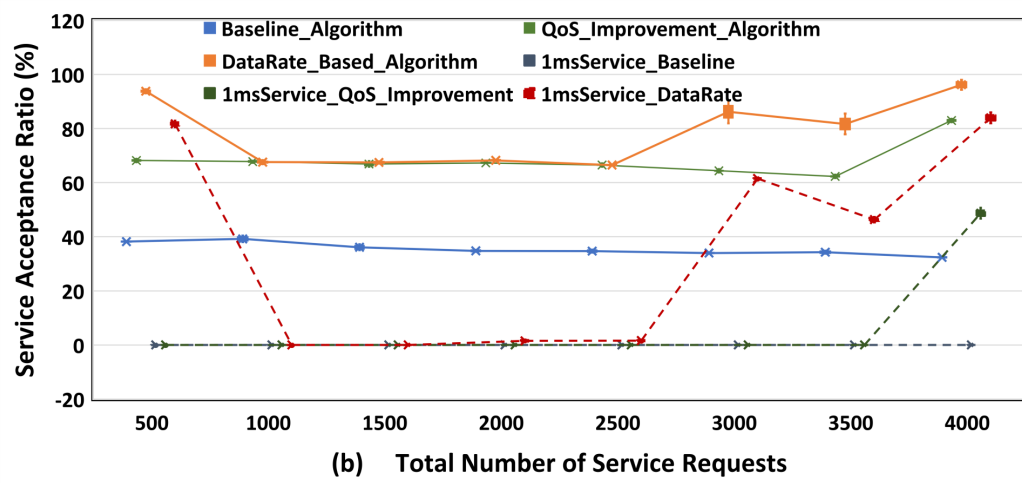
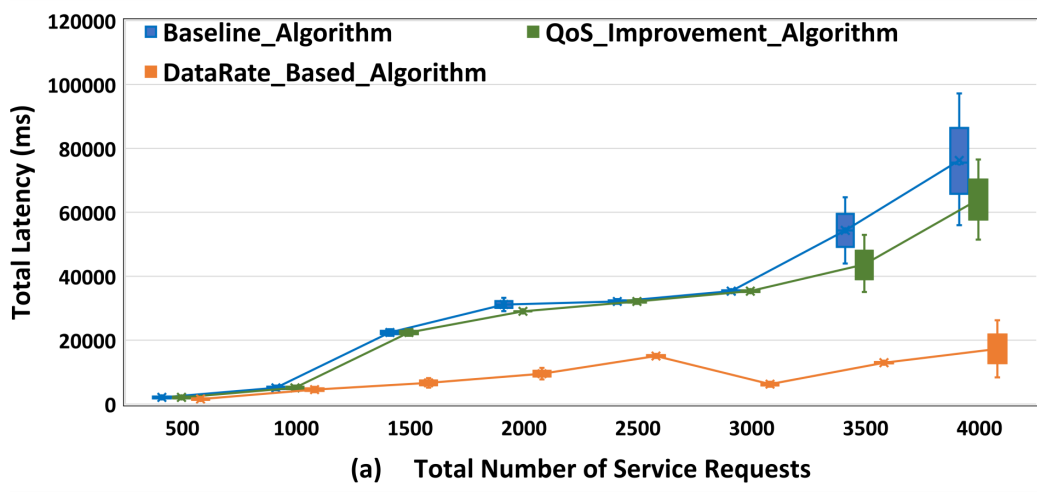


FIGURE 4.4. Simulation Results on 35-Node Topology: (a) Total Service E2E Latency, (b) Total SAR and 1ms SAR.

As for the statistical performance, most of the time, the latency variances of the three algorithms are slight and comparable. When there are 4000 service requests, *Data Rate-based Heuristic Algorithm* achieves 8947 latency variance for 50% results (from 12800 to 21747), the QoS improvement approach gets 14298 (from 55980 to 70278), and the baseline algorithm only gets 20610 (from 65775 to 86385). Based on these, it can be concluded that the baseline algorithm is the worst with broader distribution, while, the proposed *Data Rate-based Heuristic Algorithm* is the best.

Figure 4.4(b) compares the service acceptance performance of all the algorithms in 10 independent simulations using different seeds for service request generation. The baseline approach (solid blue line) is the worst, with less than 42% service acceptance ratio. The QoS improvement algorithm (solid green line) improves this ratio by more than 30%, and the *Data Rate-based*

Heuristic Algorithm (solid orange line) improves it furthermore. It can be seen that the average SAR of *Data Rate-based Heuristic Algorithm* is larger than 1.7 times compared to that of the baseline approach. The trends of the proposed *Data Rate-based Heuristic Algorithm* can be explained in detail as follows. When the workload is light, more than 90% of service requests can be accepted because MEC nodes have sufficient resources to support services. Then the SAR drops to roughly 67% since the remaining MEC node resources can not handle all ultra-low latency services. When the workload is high (i.e., between 2500 and 4000), the sum of transmission and propagation latency for routing traffic to DCs is less than the sum of queueing and OEO conversion latency at MEC nodes, the average SAR increases again to 90% because the traffic is routed to DCs, allowing MEC nodes to support more ultra-low latency services.

The ultra-low latency SARs derived by three algorithms are also compared in Figure 4.4(b). The baseline algorithm (dotted blue line) has a nearly zero ratio, which can only be improved by the QoS improvement algorithm when the total number of service requests hits 4000 (dotted green line). However, under low and high workload scenarios, the *Data Rate-based Heuristic Algorithm* produces considerably superior outcomes, with 81.7% and 83.92% ultra-low latency service accepted at the point of 500 and 4000 total service requests, respectively. Such a comparison demonstrates that the developed *Data Rate-based Heuristic Algorithm* can effectively deliver ultra-low latency services in edge-cloud networks. For both SAR and ultra-low latency SAR performance, all the algorithms are stable with condensed data in the smaller section of the box plot ($\leq 5\%$ SAR).

4.5 Summary

In this chapter, the single-objective optimisation for latency-aware SFCs placement problem was studied in hierarchical 5G networks. Firstly, we designed the multi-layer MILP model including the optical layer, IP layer, and virtual layer to minimise the E2E latency for all demands. The total E2E latency includes processing, queueing, transmission, propagation, and OEO conversion latency. Simulation results proved that the ultra-low latency requirements can be met, and the maximum MEC CPU utilisation ratio (around 65%) can be obtained. Secondly, we proposed a *Data Rate-based Heuristic Algorithm* to solve this problem in a large-scale network. Simulation results show it can achieve a ≤ 1.25 approximation ratio under low or high workload scenarios in small-scale networks and significantly outperform the other benchmark approaches in terms of total E2E latency and service acceptance ratio.

MULTI-OBJECTIVE OPTIMISATION FOR SFCs PLACEMENT

The SFCs placement problem is related to both the resource utilisation and the QoS of the network traffic [62]. It is not pragmatic to optimise the QoS while sacrificing other performance. Hence, in this chapter, the QoS-aware SFCs placement problem is expanded from single-objective to multi-objective by considering resource utilisation. First of all, a multi-objective MILP model is designed to minimise the total service E2E latency and the resource congestion ratio in the multi-layer networks involving the virtual layer, IP layer, and optical layer. In the following, simulation results prove that both objectives can be improved after adding the optical layer. At last, a non-dominated sorting genetic algorithm-II (NSGA-II) and a state-of-the-art constrained two-archive evolutionary algorithm (C-TAEA) are adapted to the SFCs placement problem to find non-dominated solutions for large-scale problems.

5.1 Introduction

The ultra-low latency 5G services impose stringent requirements on the network infrastructures. For those applications, the delay can be critical ranging between 1ms and 10 ms [11]. Hence, 5G-enable network infrastructures are deploying MEC nodes to host computing and storage resources at the edge to reduce the transmission latency [127]. However, the limited resource capacities of MEC nodes can lead to resource congestion during peak hours. Network and compute virtualisation provided by NFV architecture promises service flexibility to better utilise MEC nodes' resources. Typically, 5G services can be represented by SFCs interconnecting a sequence of VNFs [39]. The placement of SFCs has recently become a subject of extensive research in edge-cloud networks [25, 58, 131].

Considering both the resource utilisation and the QoS of the network traffic, previous single-

objective optimisation work in Chapter 4 is extended to multi-objective optimisation work in the three-level infrastructure, comprising the MEC nodes, EDCs, and CDCs. A multi-objective MILP model is designed to minimise the E2E latency for all service requests to satisfy QoS requirements and to minimise the computing resource congestion for all nodes to enhance load balance. These two objectives are contradictory. On the one hand, the first objective can minimise the E2E latency and place VNFs at the MEC nodes rather than DCs because of the reduced transmission and propagation latency. On the other hand, the second objective can reduce the edge resource utilisation and route VNFs to the DCs to ease the MEC burden. We also take into account multi-layer networks, which are made up of a virtual function layer, IP layer, and optical layer. In our approach, the queueing and transmission latency at switches can be further reduced by including the optical layer, which also increases the available communication capacity [3]. As a result, SFCs that require low E2E latency can be placed at remote locations, such as neighbouring MECs and DCs.

In this study, we utilise the Pareto-front approach to address the multi-objective QoS-aware SFCs placement problem. Pareto-front is a set of solutions that are non-dominated by each other [132]. In this problem, all Pareto optimal solutions are SFC placements that cannot be improved in any way without deteriorating another objective at the same time. Due to the NP-hardness of this problem, the MILP approach suffers from the scalability problem and spends huge time in finding non-dominated solutions. EA, which scales better than MILP, is widely used to address the multi-objective problems, especially for approximating a population of non-dominated solutions[133] because they can capture a number of solutions concurrently in a single run [134]. There are also some mechanisms involved in EAs to deal with constrained optimisation problems, such as the constrained dominance relation [135] and the balancing trade-off between convergence and diversity [133]. Inspired by these mechanisms, we modify the GA, the NSGA-II, and the constrained two-archive EA (C-TAEA) to fit this problem and study their performance.

To the best of our knowledge, the main contributions are: 1) The extension of our MILP model proposed in [131] to handle multi-objective SFCs placement and to enhance it with the loop-avoidance constraint in the optical layer. 2) The multi-objective heuristic-based GA, NSGA-II, and C-TAEA are designed to solve the constrained SFCs placement problem and deal with the lack of scalability of the MILP model.

The following of this chapter is organised as follows. Section 5.2 details the problem formulation and multi-objective MILP modelling. Section 5.3 introduces the GA, NSGA-II, and C-TATE algorithm designed for the SFCs placement problem. In the next, Section 5.4 analyses the benefits of multi-objectives and optical layer, and then the PFs approximation performance. In the end, Section 5.5 concludes the whole chapter.

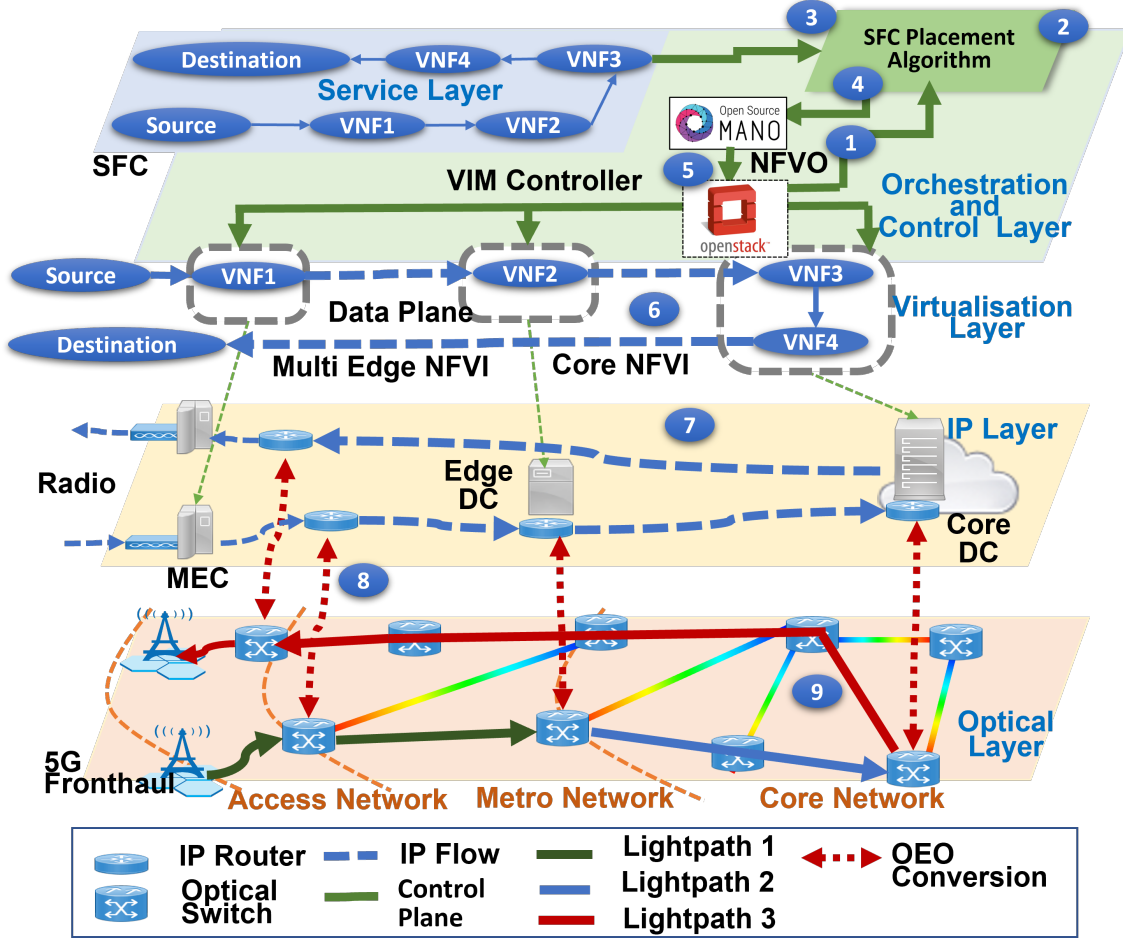


FIGURE 5.1. SFC Placement Example in Multi-layer Edge-Cloud Network.

5.2 MILP Formulation for Multi-Objective SFCs Placement

5.2.1 SFCs placement in Edge-Cloud Networks

Figure 5.1 shows the SFCs placement in multi-layer edge-cloud network. In this MFV-enabled infrastructure, NFVO collects network status and service chain requests and then sends them as the input parameters to the SFC placement algorithm, which makes decisions for the resource allocation. MEC nodes, EDCs, and CDC that hold different capacities of resources are connected via optical fibres and switching nodes, such as OXCs. On both the IP layer and optical layer, physical resources are virtualised and deployed with OpenStack. NFVO integrates VNFs across several MEC nodes and DCs to suit all end-user requirements while improving resource utilisation [41]. In the optical layer, WDM is adopted and optical transparent switches are used to assure wavelength continuity [1]. The SFCs placement problem refers to the optimal mapping of chained VNFs to computing nodes and virtual links between VNFs to the lightpath in physical

links without violating corresponding constraints in such a situation.

Figure 5.1 also depicts a SFC placement example. In this case, a single SFC contains 4 VNFs. According to the placement findings, one of the MEC nodes hosts VNF1, the EDC hosts VNF2, and the CDC hosts VNF3 and VNF4. The source and destination nodes of this SFC are also included. The signal will be at first converted from electronic to optical and then transmitted along optical fibres at the transmission end. Optical traffic will be converted to electronic signals at the receiving end and then processed at the VNF instance. OEO conversions exist across the optical layer and IP layer, as indicated by the red dotted arrow line. The green, blue, and red thick arrow lines show the optical signals transmitted from the transmitter to the receiver via lightpath.

To formulate the multi-objective SFC placement problem, the infrastructure is coded as follows.

A directed graph $G = (N, L^P)$ is used to model the network of nodes and physical links, where N represents the set of nodes, and L^P represents the set of optical links. Among all of the nodes, N_v represents node with computing resources, including MEC servers N_{MEC} , EDCs N_{EDC} , and CDC N_{CDC} . N_{SWN} represents node with switching capability only. MEC servers have limited computing and buffering resources and are placed close to 5G base stations (i.e., eNodeBs) [25]. In contrast, DCs are assumed to have unlimited resources. The switching node in this model includes the router and the OXCs.

There are two types of network nodes. One is computing node $n \in N_v$, which is characterised by the computing and buffering resources, indicated as n^{cpu} and n^{buf} , respectively. The other has only buffering resources and is called switching node $n \in N_{SWN}$. A lightpath (l, l') from node $l \in N_v$ to node $l' \in N_v$ belongs to the set of lightpath L^{lp} . The light speed in optical fibre is lv . An optical physical link $(n, n') \in L^P$ represents a link between node n and n' , with length of $len_{(n, n')}$. The total number of wavelengths in the network is W , and $w \in [1, |W|]_z$ is the w -th wavelength in the wavelength set. $|\cdot|$ denotes the number of elements in the set, and $[A, B]_z$ denotes the set of integers from A to B). In the physical link (n, n') , the transmission capacity of w -th wavelength is $B_{(n, n')}^w$.

The set of VNFs types is defined as M , and $m \in M$ is used to indicate the specific m type VNF. Different VNFs demand different amounts of computing and buffering resources denoted as β_m^{cpu} , and β_m^{buf} , respectively. Different VNFs has different scaling attribute, which is represented by δ_m . For each flow, the output data rate v_{output} is determined by the input data rate v_{input} and the scaling attribute, which can be computed using $v_{output} = \delta_m \cdot v_{input}$. Furthermore, $suit_{n, m}$ is a binary indicator that indicates whether or not the VNF m can be placed on the computing node n .

5.2.2 Multi-Objective MILP Formulation

5.2.2.1 Input Variables

Input variables contain all of the variables listed above, as well as service requests. We assume there is a set of service requests K . A chain of VNFs makes up each service request is $k \in K$. Each service request k requires the O_k number of VNFs in the chain. We use o to represent the o -th VNF in the SFC k , $o \in [0, |O_k| + 1]_{\mathbb{Z}}$, where $o = 0$ represents the source of SFC s^k and $o = |O_k| + 1$ represents the destination d^k . In addition, the service request is also characterised by the data rate v^k , required E2E latency DR^k , and packet size TS^k .

The o -th VNF in the service request k has a data rate of $v_m^{k,o}$. For the first VNF in SFC, $v_m^{k,1} = v^k$. For the other VNF, $v_m^{k,o} = \delta_{m'}^{k,o-1} \cdot v_{m'}^{k,o-1}$. The o -th VNF of service request k requires computational and buffering resources, which are represented by $C_m^{k,o,cpu}$ and $C_m^{k,o,buf}$, respectively. The following two equations can be used to compute the required computing and buffering resource amount: $C_m^{k,o,cpu} = \beta_m^{cpu} \cdot v_m^{k,o}$ and $C_m^{k,o,buf} = \beta_m^{buf} \cdot v_m^{k,o}$.

5.2.2.2 Output Variables

Table.5.1 shows the MILP model's output variables.

Table 5.1: Output Variables of Multi-objective Optimisation Model

$x_{n,m}^{k,o}$	A binary variable indicating whether the m type VNF in the service request is placed on the node n or not
$yw_{(l,l'),w}^{k,(o,o+1),m}$	A binary variable indicating whether w wavelength in the lightpath (l,l') is selected on the path between the o -th and $(o+1)$ -th VNF or not
$tw_{(l,l'),(n,n'),w}^{k,(o,o+1),m}$	A binary variable indicating whether w wavelength in the physical link (n,n') is selected on the path between the o -th and $(o+1)$ -th VNF or not
D^{total}	The total service E2E latency of all service requests
U^{cpu}	The total CPU utilisation ratio of all nodes
u_n^{cpu}	The CPU utilisation ratio on node n
u_n^{buf}	The buffer utilisation ratio on node n

5.2.2.3 Objectives

In this multi-objective problem, we design the first objective as the minimisation of the total service E2E latency (presented in equation (5.1)), which includes processing latency DP^k , queueing

latency DQ^k , transmission latency DT^k , and propagation latency DG^k of all demands.

$$(5.1) \quad \min D^{total} = \sum_{k \in [1, |K|]_z} (DP^k + DQ^k + DT^k + DG^k)$$

The second objective is to minimise the total congestion ratio (shown in equation (5.2)), which is determined as the sum of CPU utilisation ratio of all nodes. The CPU utilisation ratio of each node n is calculated using equation (5.3).

$$(5.2) \quad \min U^{cpu} = \sum_{n \in N_v} u_n^{cpu}$$

$$(5.3) \quad u_n^{cpu} = \left(\sum_{k \in [1, |K|]_z} \sum_{o \in [1, |O_k|]_z} x_{n,m}^{k,o} \cdot C_m^{k,o,cpu} \right) / n^{cpu} \quad \forall n \in N_v$$

5.2.2.4 Constraints

A) Placement: This constraint guarantees that the o -th VNF of service request k can only be mapped to one node in the network. Specifically, the 0-th VNF and $(|O_k| + 1)$ -th VNF can only be mapped to the required source and destination node, respectively.

$$(5.4) \quad \sum_{n \in N} x_{n,m}^{k,o} \cdot suit_{n,m} = 1 \quad \forall k \in [1, |K|]_z, o \in [0, |O_k| + 1]_z$$

B) Resource Capacity: Equation (5.5) and (5.6) ensure that the computing resource and buffer required by all the VNFs do not exceed the physical node's available resources.

$$(5.5) \quad \sum_{k \in [1, |K|]_z} \sum_{o \in [1, |O_k|]_z} x_{n,m}^{k,o} \cdot C_m^{k,o,cpu} \leq n^{cpu} \quad \forall n \in N_v$$

$$(5.6) \quad \sum_{k \in [1, |K|]_z} x_{l',m}^{k,1} \cdot C_m^{k,o,buf} + \sum_{k \in [1, |K|]_z} \sum_{o \in [1, |O_k|]_z} \sum_{l \in N} \sum_{w \in [1, |W|]_z} yw_{(l,l'),w}^{k,(o,o+1),m} \cdot C_m^{k,o,buf} \leq l'^{buf} \quad \forall l' \in N$$

Equation (5.7) ensures that the total bandwidth used by all transmissions does not exceed the physical link's transmission capacity.

$$(5.7) \quad \sum_{k \in [1, |K|]_z} \sum_{o \in [0, |O_k| + 1]_z} \sum_{(l,l') \in L^{lp}} tw_{(l,l'),(n,n'),w}^{k,(o,o+1),m} \cdot v_m^{k,o} \leq B_{(n,n')}^w \quad \forall (n,n') \in L^P, w \in [1, |W|]_z$$

C) Link Mapping: Equation (5.8) makes sure the flow conservation restriction.

$$(5.8) \quad \sum_{(l,l') \in L^{lp}} \sum_{w \in [1, |W|]_z} \sum_{n' \in N} tw_{(l,l'),(n,n'),w}^{k,(o,o+1),m} \cdot v_m^{k,o} - \sum_{(l,l') \in L^{lp}} \sum_{w \in [1, |W|]_z} \sum_{n' \in N} tw_{(l,l'),(n',n),w}^{k,(o,o+1),m} \cdot v_m^{k,o} \\ = x_{n,m}^{k,o} - x_{n,m'}^{k,o+1} \quad \forall k \in [1, |K|]_z, o \in [0, |O_k|]_z, n \in N$$

Equation (5.9) shows the wavelength continuity constraint from the o -th VNF to the $(o + 1)$ -th VNF along all the selected (n, n') links.

$$(5.9) \quad \sum_{w \in [1, |W|]_z} yw_{(l,l'),w}^{k,(o,o+1),m} \leq 1 \quad \forall k \in [1, |K|]_z, o \in [0, |O_k|]_z, (l, l') \in L^{lp}$$

Equation (5.10) makes sure that the physical link can be chosen only after the lightpath has been chosen.

$$(5.10) \quad \begin{aligned} tw_{(l,l'),(n,n'),w}^{k,(o,o+1),m} &\leq yw_{(l,l'),w}^{k,(o,o+1),m} \quad \forall k \in [1, |K|]_z, o \in [0, |O_k|]_z, \\ (l, l') &\in L^{lp}, (n, n') \in L^P, w \in [1, |W|]_z \end{aligned}$$

Equation (5.11) represents the relationship constraint between traffic flow in lightpath and physical link.

$$(5.11) \quad \sum_{n \in N} tw_{(l,l'),(n,n'),w}^{k,(o,o+1),m} - \sum_{n \in N} tw_{(l,l'),(n',n),w}^{k,(o,o+1),m} = \begin{cases} yw_{(l,l'),w}^{k,(o,o+1),m} & \text{if } n' = l' \\ -yw_{(l,l'),w}^{k,(o,o+1),m} & \text{if } n' = l \\ 0 & \text{otherwise} \end{cases}$$

$$\forall k \in [1, |K|]_z, o \in [0, |O_k|]_z, (l, l') \in L^{lp}, w \in [1, |W_k|]_z$$

Equation (5.8) and (5.11) aid in the avoidance of traffic loops for computing nodes, and equation (5.12) guarantees that no traffic loops exist for switching nodes.

$$(5.12) \quad \sum_{n \in N} tw_{(l,l'),(n,n'),w}^{k,(o,o+1),m} \leq 1 \quad \forall k \in [1, |K|]_z, o \in [0, |O_k|]_z, (l, l') \in L^{lp}, w \in [1, |W|]_z, n' \in N_{SWN}$$

D) Latency: The following equations can be used to calculate processing, queueing, transmission, and propagation latency. 1) Processing latency: it happens only at the computing nodes and the M/M/1 model is adopted to the calculation [8]. Since equation (5.13) is a conduct of a binary variable and a continuous variable, the Big-M method is used for the linearisation. $z_n^{k,o,cpu}$ is the average processing latency, $z_{n,max}^{k,o,cpu}$ is the upper bound of $1/(1 - u_n^{cpu})$, and b_e, c_e are parameters introduced for the linearisation (the total number of piecewise linearisation functions is E and $e \in [1, |E|]_z$).

$$(5.13) \quad \begin{aligned} DP^k &= \sum_{n \in N_v} \sum_{o \in [1, |O_k|]_z} x_{n,m}^{k,o} \cdot \frac{1}{(1 - u_n^{cpu}) \cdot n^{cpu}} = \sum_{n \in N_v} \sum_{o \in [1, |O_k|]_z} x_{n,m}^{k,o} \cdot (b_e \cdot u_n^{cpu} + c_e) \\ &= \sum_{n \in N_v} \sum_{o \in [1, |O_k|]_z} z_n^{k,o,cpu} \quad \forall k \in [1, |K|]_z \end{aligned}$$

$$(5.14) \quad z_n^{k,o,cpu} \leq x_{n,m}^{k,o} \cdot z_{n,max}^{k,o,cpu} \quad \forall k \in [1, |K|]_z, o \in [1, |O_k|]_z, n \in N_v$$

$$(5.15) \quad b_e \cdot u_n^{cpu} + c_e \leq z_n^{k,o,cpu} + (1 - x_{n,m}^{k,o}) \cdot z_{n,max}^{k,o,cpu} \quad \forall k \in [1, |K|]_z, o \in [1, |O_k|]_z, n \in N_v, e \in [1, |E|]_z$$

2) Queueing latency: it is assumed that DCs have unlimited buffering capacities, while MEC nodes have limited buffering resources in this model. As a result, the queueing latency is only taken into account at the MEC servers and M/M/1 model is used for the calculation [8]. Equation (5.17) calculates the buffering resource utilisation ratio on MEC server l' .

$$(5.16) \quad DQ^k = \sum_{n \in N_{MEC}} \sum_{o \in [1, |O_k|]_z} z_n^{k,o,buf} \quad \forall k \in [1, |K|]_z$$

$$(5.17) \quad u_{l'}^{buf} = \left(\sum_{k \in [1, |K|]_z} x_{l',m}^{k,1} \cdot C_m^{k,1,buf} + \sum_{k \in [1, |K|]_z} \sum_{o \in [1, |O_k|]_z} \sum_{l \in [N_v]} \right. \\ \left. \sum_{w \in [1, |W|]_z} yw_{(l,l'),w}^{k,(o,o+1),m} \cdot C_m^{k,o,buf} \right) / l'^{buf} \quad \forall l' \in N_{MEC}$$

In order to calculate the queueing latency, the same linearisation approach is used. The average queueing latency is $z_n^{k,o,buf}$, the maximum queueing latency is $z_{n,max}^{k,o,buf}$, and coefficients for the linearisation include g_e and j_e .

$$(5.18) \quad z_n^{k,1,buf} \leq x_{n,m}^{k,1} \cdot z_{n,max}^{k,1,buf} \quad \forall k \in [1, |K|]_z, n \in N_{MEC}$$

$$(5.19) \quad z_n^{k,o,buf} \leq \sum_{w \in [1, |W|]_z} \sum_{l \in N_v} yw_{(l,n),w}^{k,(o,o+1),m} \cdot z_{n,max}^{k,o,buf} \quad \forall k \in [1, |K|]_z, o \in [2, |O_k|]_z, n \in N_{MEC}$$

$$(5.20) \quad g_e \cdot u_n^{buf} + j_e \leq z_n^{k,1,buf} + (1 - x_{n,m}^{k,1}) \cdot z_{n,max}^{k,1,buf} \quad \forall k \in [1, |K|]_z, n \in N_{MEC}, e \in [1, |E|]_z$$

$$(5.21) \quad g_e \cdot u_n^{buf} + j_e \leq z_n^{k,o,buf} + \left(1 - \sum_{w \in [1, |W|]_z} \sum_{l \in N_v} yw_{(l,n),w}^{k,(o,o+1),m}\right) \cdot z_{n,max}^{k,o,buf} \quad \forall k \in [1, |K|]_z, o \in [2, |O_k|]_z, n \in N_{MEC}, e \in [1, |E|]_z$$

3) Transmission latency: Traffic is sent along the chosen lightpath in this multi-layer infrastructure. Under this scenario, transmission latency occurs solely at the source of the lightpath. To be more specific, traffic will be sent along the lightpath (l, l') if the o -th and $(o+1)$ -th VNFs are deployed at two different node l and l' , with transmission latency happens solely at node l .

$$(5.22) \quad DT^k = \sum_{o \in [1, |O_k|]_z} \sum_{(l,l') \in L^{lp}} \sum_{w \in [1, |W|]_z} yw_{(l,l'),w}^{k,(o,o+1),m} \cdot TS^k / B^w \quad \forall k \in [1, |K|]_z$$

4) Propagation latency: exists on all the optical links chosen for the traffic transmission.

$$(5.23) \quad DG^k = \sum_{o \in [0, |O_k|]_z} \sum_{(l,l') \in L^{lp}} \sum_{w \in [1, |W|]_z} \sum_{(n,n') \in L^p} tw_{(l,l'),(n,n'),w}^{k,(o,o+1),m} \cdot len_{(n,n')}/lv \quad \forall k \in [1, |K|]_z$$

5) E2E latency: constraint (5.24) ensures that the total latency experienced by all the chained VNFs does not exceed its service E2E latency requirement.

$$(5.24) \quad (DP^k + DQ^k + DT^k + DG^k) \leq DR^k \quad \forall k \in [1, |K|]_z$$

5.3 Algorithms for Non-dominated Solutions

Due to the NP-hard property of the SFCs placement problem and the lack of scalability of MILP based approaches [32, 56], it is crucial to design algorithms for solving large scale problems. Since the GA, NSGA-II, and Multi-objective Virtual Network Function Chain Placement (MO-VNFCP) Algorithm have been proposed for the SFCs resource allocation in recent works [36, 45, 56], we modify them to fit this SFCs placement problem. In addition, we implement a state-of-the-art

two-archive EA, which is designed for constrained multi-objective optimisation problems, for this SFCs placement problem.

To address the constraint challenge in multi-objective optimisation problems, we adopt the constraint-handling technique used in [136]. In this approach, the solution (x1), which constrained-dominates another one (x2), is chosen. The constrained domination criterion is true if any one of the following conditions is true: 1) if x1 is feasible and x2 is infeasible; 2) if x1 and x2 are infeasible, and x1 has a smaller constraint violation value [87]; 3) if x1 and x2 are feasible and x1 dominates x2 with the usual domination principle.

The sum of constraint violation can be calculated as follows:

$$(5.25) \quad \psi(\vec{x}) = \sum_{\kappa=1}^t \max(0, \xi_{\kappa}(\vec{x}))^2 + \sum_{\lambda=1}^p |\phi_{\lambda}(\vec{x})|$$

where \vec{x} is the vector of solutions $\vec{x} = [x_1, x_2, \dots, x_n]^T$, the values of each inequality constraint $\xi_{\kappa}(\vec{x}), \kappa = 1, 2, \dots, t$ and also each equality constraint $\phi_{\lambda}(\vec{x}), \lambda = 1, 2, \dots, p$ are normalized [87]. Due to this simple constraint-handling scheme, this approach can be coupled to a variety of algorithms, without introducing new parameters [87]. However, it may lead to premature convergence [87].

5.3.1 Heuristic-based GA & NSGA-II Algorithm

Our proposed heuristic-based GA and heuristic-based NSGA-II can be explained as follows:

1) *Objective*: Algorithms are designed to achieve the minimisation of total service E2E latency and the minimisation of total CPU utilisation ratio at the same time.

2) *Population*: It contains a fixed number of chromosomes. The initial population is created in three ways without violating the physical resource constraints: 1) a first fit placement heuristic that starts from the MEC to the EDC and then to the CDC; 2) a first fit placement heuristic that starts from the CDC to the MEC and then to the EDC; 3) a random VNF placement. In [56], they only use the first fit heuristic to generate feasible solutions for NSGA-II. However, according to [137], infeasible solutions can be valuable stepping stones between feasible regions of the search space, so we introduce infeasible solutions through the random generation.

3) *Coding of individuals*: An individual is characterised by a set of parameters known as Genes. Each individual is a node placement solution to this problem. The gene in the specific position represents the chosen node for the specific VNF in SFC. The gene's ordering is the same as the VNF's ordering in the SFC. Each integer is the selected node number. Figure 5.2 (a) and (b) show an example of such coding. There are two SFC requests and each one requires three VNFs. For the first SFC, its VNFs are mapped to MEC1, MEC2 and core DC, so its coding is 1, 2, and 7. While the second SFC is coded as 1, 2, 6.

4) *Parent Selection*: We introduce the constrained-domination criterion to parent selection. Two individuals are firstly randomly selected, and then the solution which constrained-dominates the other one is kept as one of the parents. The other is chosen based on the same procedure.

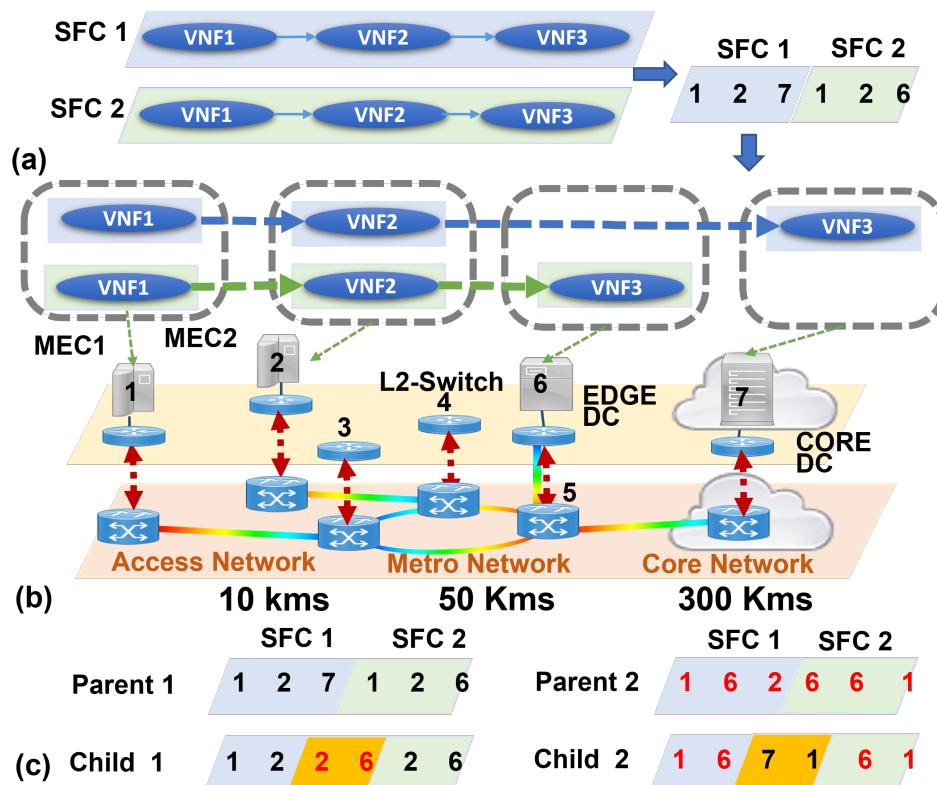


FIGURE 5.2. SFCs Coding in the Proposed NSGA-II. a) SFC Requests b) Example of Placement Codification of Two SFCs in 7-Node Simulation Topology. c) Example of Two-Point Crossover

Hence, we can make sure that the feasible and fittest solution can be kept and has more chance to pass its genes to the next generation.

5) *Crossover*: In this most significant phase, we use a two-point crossover where two points on the chromosomes are randomly selected. A child inherits elements between these two crossing points of one parent and inherits the remaining elements from the other parent, which is also shown in Figure 5.2 (c).

6) *Mutation*: Mutation allows to modify the offspring in order to diversify the population. To improve the convergence performance, we use a local search to find a neighbour solution that constrained-dominates the original solution. The neighbour solution is generated by changing the placement result for a randomly chosen VNF. In [56], mutation result is randomly chosen. Although it introduces diversity, it is harder to find a feasible solution and prevent convergence in our problem where there are plenty of complex constraints.

7) *Difference between GA and NSGA-II*: Compared to GA, the NSGA-II introduces the fast non-dominated sorting approach and a diversity preservation mechanism. At first, the usual coding, selection, crossover, and mutation are used to create offspring. The procedure is different after the initial generation. For each solution, the number of solutions which dominates the solution,

and the set of solutions that this solution dominates are calculated. The one dominated by the lower number of solutions is in the lower rank and is a better solution. Then for each solution, the crowding-distance is calculated as the sum of individual distance values corresponding to each objective [135]. Solutions in lower rank will be selected and solutions in the same rank but has a high crowding distance will be selected for the next generation [135].

5.3.2 Constrained Two-Archive Evolutionary Algorithm

Compared to GA and NSGA-II, C-TAEA provided in [133] is designed especially for addressing constrained multi-objective optimisation problems. It can simultaneously balance the convergence, diversity, and feasibility by maintaining two collaborative archives named convergence-oriented archive (CA) and diversity-oriented archive (DA). As the name indicated, the CA archive aims to improve the convergence performance and keep solutions in the feasible regions, while, the DA archive aims to improve the diversity features and discover the regions that have not been explored before. Mating parents are chosen separately from the CA and the DA during the reproduction process based on their evolution status. Following that, the offspring are used to update the CA and the DA.

The algorithm can be described as follows.

1) *Density Estimation*: Firstly, a density estimation method is introduced to divide the objective space into N sub-regions. To make it comparable to the modified GA and NSGA-II, we uniformly divide each objective coordinate into 100 divisions and generate 101 subregions with the weight vectors $(0, 1), (0.01, 0.99), \dots, (0.99, 0.01), (1, 0)$. Each solution \mathbf{x} of a population is associated with a unique sub-region whose index is determined by equation (5.26).

$$(5.26) \quad k = \underset{i \in 1, \dots, N}{\operatorname{argmin}} < \overline{\mathbf{F}}(\mathbf{x}), \mathbf{w}^i >$$

where $\overline{\mathbf{F}}(\mathbf{x})$ is the normalised objective vector of \mathbf{x} , and its i -th objective function is calculated by the following equation (5.27).

$$(5.27) \quad \overline{f}_i(\mathbf{x}) = \frac{f_i(\mathbf{x}) - z_i^*}{z_i^{nad} - z_i^*}$$

where $i \in 1, \dots, m$, z_i^* and z_i^{nad} are, respectively, the estimated ideal and nadir points, where $z_i^* = \min_{\mathbf{x} \in S} f_i(\mathbf{x})$, $z_i^{nad} = \max_{\mathbf{x} \in S} f_i(\mathbf{x})$ and S is the current solution set. The density of a sub-region is counted as the number of its associated solutions.

2) *Population Generation and Coding of individuals*: We utilise the same methods for population generation and individual coding in the GA and NSGA-II algorithm design.

3) *Update CA*: CA attempts to balance the convergence and diversity within the feasible region by firstly pushing the population towards the feasible region as much as possible. They produce a hybrid population H_c , a mix of the CA and the offspring population Q . Feasible solutions in H_c are chosen into a temporary archive S_c . Afterwards, the follow-up procedure is determined by

the size of S_c : 1) if S_c has the same size as N , it is directly used as the new CA. 2) if $|S_c| > N$, they divide solution S_c into several non-dominated levels using the NSGA-II fast non-dominated sorting approach. Then, they trim the worst population in each sub-region until S 's size equals N . 3) if $|S_c| < N$, they use the same fast non-dominated sorting approach to divide the infeasible solutions in H_c into several non-dominated levels. Solutions in the first several levels have a larger chance of surviving into the new CA.

4) *Update DA*: DA makes an effort to include as much diversity as possible. It takes the most current CA as a reference set and explores the DA's under-exploited areas to compensate. Initially, the DA combines itself with the offspring population Q to create a hybrid population H_d . Then, they associate each solution in H_d and the up-to-date CA with its corresponding sub-region separately. Afterwards, they iteratively analyse each subregion and decide whether or not the solutions in H_d should be kept in the new DA. For the currently investigating sub-region, if there are already solutions in CA, no solutions in H_d will be kept for this sub-region during this iteration. Otherwise, the best non-dominated solutions in H_d will be picked to survive the new DA. The investigation will be iterated until the DA is filled.

5) *Offspring Reproduction*: The restricted mating selection is utilised to exploit the elite knowledge from both archives for offspring reproduction. They construct a set H_m by combining the CA and the DA. Afterwards, they calculate the proportion of non-dominated solutions of the CA and the DA in H_m , respectively. If the CA's convergence status is better than the DA's, the first mating parent is chosen from the CA, otherwise, it comes from the DA. The proportion of non-dominated solutions in the CA determines whether the other mating parent is chosen from the CA or the DA. They use a binary tournament to select a mating parent. To be more specific, if all of the randomly chosen candidates are feasible, they are selected using the Pareto dominance principle; if only one of them is feasible, the feasible one will be selected; otherwise, the mating parent is chosen at random. Once the mating parents have been chosen, they employ the simulated binary crossover [138] and the polynomial mutation [139] for offspring reproduction.

5.4 Simulation-based Evaluation

5.4.1 Simulation Setup

Two network scenarios are considered: a 7-node topology shown in Figure 5.2 (b) and a 29-node topology in Figure 5.3. On the 7-node network, we solve the MILP model by Gurobi solver and run the proposed heuristic-based GA, NSGA-II, and C-TATE algorithms designed for the SFCs placement problem and the benchmark algorithm MOSA. On the 29-node network, we only run the above-mentioned algorithms. For the simulation, the MILP model and designed algorithms are running on an IBM server with an 8-core processor and 24GB RAM.

For the computing and buffering resource setting, we assume that each MEC node, EDC, and CDC has 280, 1400, 140000 CPU cores, individually [140–142], and 1120, 5600, 560000

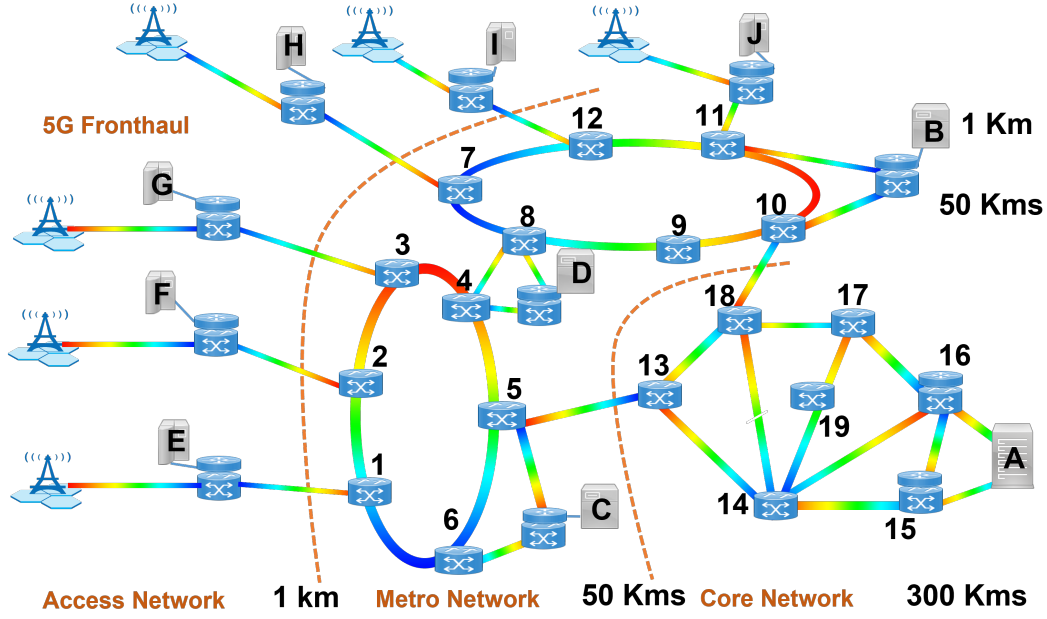


FIGURE 5.3. Large-Scale Simulation Topology: 29 Node Network

GB RAM resources, respectively in both small-scale (7-node) and large-scale (29-node) network scenarios [3]. For the link settings, we assume that, in 7-node topology, physical links from CDC to SWN, from EDC to SWN, from the MEC server to switching node (SWN), and between SWNs are 300km, 20km, 1km, and 2km, respectively. Each link has 6 wavelengths and a capacity of 10 Gbps [143]. While in the 29-node topology, physical links between the MEC server and the nearest SWN, between EDC and the nearest SWN, between CDC and the nearest SWN, between SWN and the neighbour SWN in the metro area, and between SWN and the neighbour SWN in the core area are 1km, 20km, 300km, 2km, and 5km, respectively. Each link has 11 wavelengths and a 10 Gbps capacity.

Based on the E2E latency and bandwidth requirements, We divide network services into five categories: 1) High bandwidth and low latency; 2) Low bandwidth and low latency; 3) High bandwidth and medium latency; 4) Low bandwidth and medium latency; and 5) Non-real time services. To simulate these five service classes, we use Augmented Reality (AR), mIoTs, Video Streaming (VS), Voice over Internet Protocol (VoIP), and Web services. Table 5.2 shows their proportion, data rate, latency, and VNFs requirements. For the AR and mIoTs services, the source and destination are the same MEC node. Others have their source and destination determined at random. There are six VNF types considered, including Firewall (FW), Network Address Translation (NAT), Video Transcoder (VT), Intrusion Detection (ID), WAN Optimizer (WO), and Traffic Monitoring (TM). The computing and buffering resources demanded by various VNFs, as well as their scaling vectors are listed in Table 5.3.

Table 5.2: Service Requests Setting [4–8]

Service	Percentage	Data Rate	Latency	VNF Chain
AR	39.5%	100Mbps	1ms	NAT-FW-TM-VT-ID
mIOT	10.05%	100Kbps	5ms	NAT-FW-ID
VS	39.5%	4Mbps	100ms	NAT-FW-TM-VT-ID
VoIP	1.5%	64Kbps	100ms	NAT-FW-TM-FW-NAT
Web	9.45%	100Kbps	500ms	NAT-FW-TM-WO-ID

Table 5.3: VNF required Resources and Properties[1, 3]

VNF	CPU	RAM	Scale	VNF	CPU	RAM	Scale
NAT	0.00092	0.00092	1	FW	0.0009	0.00135	1
TM	0.0133	0.0399	1	VT	0.0054	0.0054	2
WO	0.0054	0.0108	1	ID	0.0107	0.0107	1

5.4.2 Simulation Results and Analysis

5.4.2.1 MILP Results

In this subsection, the benefits of adding the optical layer and the different effects brought by the two objectives are concluded through the comparison among three MILP models. 1) **Model 1** and **Model 3** involve the optical layer by adding transmission latency *only at the source node* while keeping the wavelength continuity. **Model 2** does not involve the optical layer and transmission latency happens both at the source node and the intermediate switches. 2) **Model 1** and **Model 2** minimise the total service E2E latency whereas **Model 3** minimises the total congestion ratio.

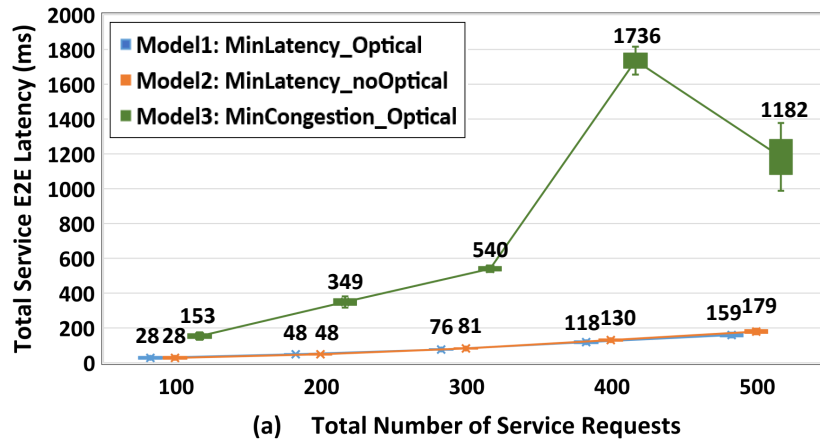


FIGURE 5.4. Total Service E2E Latency on 7-Node Topology.

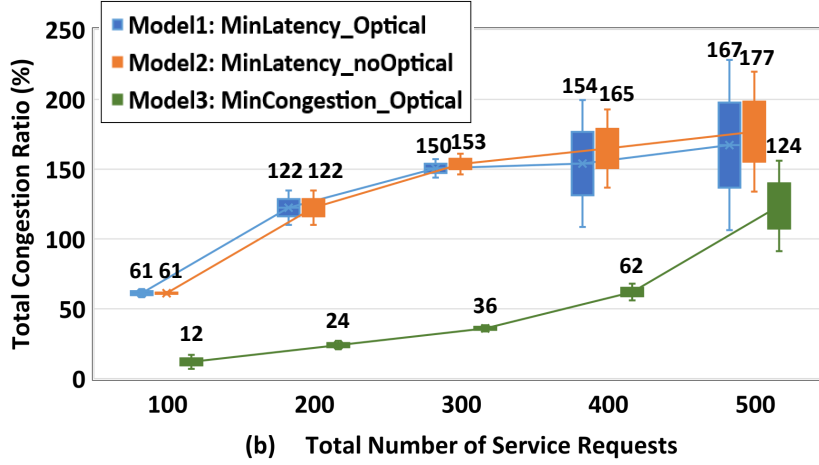


FIGURE 5.5. Total Congestion on 7-Node Topology.

Model 1 vs Model 2: By comparing findings across a 7-node topology presented in Figure 5.2 (b), we can confirm the advantages of adding the optical layer for reducing both service E2E latency and congestion. When the number of requests is fewer than 200 both **Model 1** and **Model 2** yield comparable results. However, when the service requests reach 400, **Model 1** achieves a 9% lower overall service E2E latency (Figure 5.4) and a 6% lower total congestion ratio (Figure 5.5) than **Model 2**. When there are 500 requests, **Model 1** achieves a lower total service E2E latency of 11% and a lower total congestion ratio of 5% than **Model 2**. As a consequence of the inclusion of the optical layer in **Model 1**, the total service E2E latency and congestion ratio can be reduced more efficiently by allowing VNFs to be deployed not only in the nearest MEC to users but DCs reachable through optical lightpaths. Furthermore, it is expected that the difference between **Model 1** and **Model 2** will be wider in a larger network topology serving larger service requests.

Model 1 vs Model 3: The trade-off between minimising total service E2E latency and total congestion ratio can be confirmed by the results in Figure 5.4 and Figure 5.5. By minimising the total congestion ratio, the total service E2E latency is significantly increased by **Model 3**. As expected, **Model 1** achieves up to 85% lower service E2E latency and 85% higher overall congestion ratio than **Model 3** for up to 200 service requests. The network becomes congested when the number of requests reaches 400 or 500, resulting in a narrowing of the performance gap between **Model 1** and **Model 3**. Because these two objectives are contradictory, using a multi-objective method to solve the SFCs placement problem is critical. In the following subsections, we will study the non-dominated solutions to the multi-objective SFCs placement problem.

Apart from the average latency and congestion results, Figure 5.4 and Figure 5.5 also present the statistical performance of the three models. Based on the box plots in Figure 5.4, it can be concluded that MILP models with latency minimisation as the objective (**Model 1** and **Model 2**)

can achieve more stable and robust results in terms of latency performance compared to the model with congestion minimisation as the objective (**Model 3**). The 50% latency variance of **Model 1** and **Model 2** are less than 5ms for different number of service requests. But this value of **Model 3** is always larger and even reaches 80ms and 194ms for 400 and 500 service requests, respectively. A similar conclusion can be drawn from Figure 5.5. **Model 3** minimising congestion ratio outperforms the other two models minimising total service E2E latency in terms of congestion statistic performance. For the different number of service requests, **Model 3**'s box width is always narrow compared with the other two models' box widths, which indicates more condensed data and a more robust feature.

5.4.2.2 Simulation Results for Algorithms

In this subsection, approximated non-dominated solutions obtained by the heuristic-based GA, NSGA-II, MO-VNFCP, and C-TAEA algorithms running on both 7-node and 29-node network topology are presented and analysed.

1) Simulation Results on 7-node Topology

To provide an idea of how the non-dominated front changes with the increase of the total number of service requests, Figure 5.6 (a)-(d) show the non-dominated fronts obtained via different approaches for 100, 200, 300, and 400 service requests on the 7-node network, respectively. In the simulation, we set the population size to 100 and the mutation rate to 0.30 for the heuristic-based GA and heuristic-based NSGA-II algorithms. For the comparison purpose, we run heuristic-based GA, heuristic-based NSGA-II, MO-VNFCP algorithms proposed in [36], and C-TAEA 100 iterations.

Among all the algorithms, the MO-VNFCP performs the worst. It can be seen that all the non-dominated solutions calculated by the MO-VNFCP algorithm are dominated by other algorithms' non-dominated solutions. Because it highly depends on the initial solutions and improves a little with the non-dominated generation within 100 iterations (black triangle). When the number of services increases, it performs even worse, which can be seen from the average distance between the MO-VNFCP solutions and the GA solutions. Compared to it, the heuristic-based GA and NSGA-II algorithms improve the performance a lot with better and more non-dominated solutions (yellow circle and orange star). However, their capabilities of exploring PF are still limited in the highly constrained problem, which means the randomness introduced by the mutation can be largely possible in the infeasible area and the new generation will not be selected for the new population generation.

The C-TAEA performs the best. It can effectively reduce the total congestion ratios and achieve better non-dominated solutions (dark green cross) since it introduces the diversity mechanism to explore the unvisited area. In this model, instead of running VNFs at MEC nodes, C-TAEA can route VNFs from MEC nodes to DCs to further reduce the congestion ratio. However, it takes more time than other algorithms to get such good solutions. Table. 5.4 summarises the running

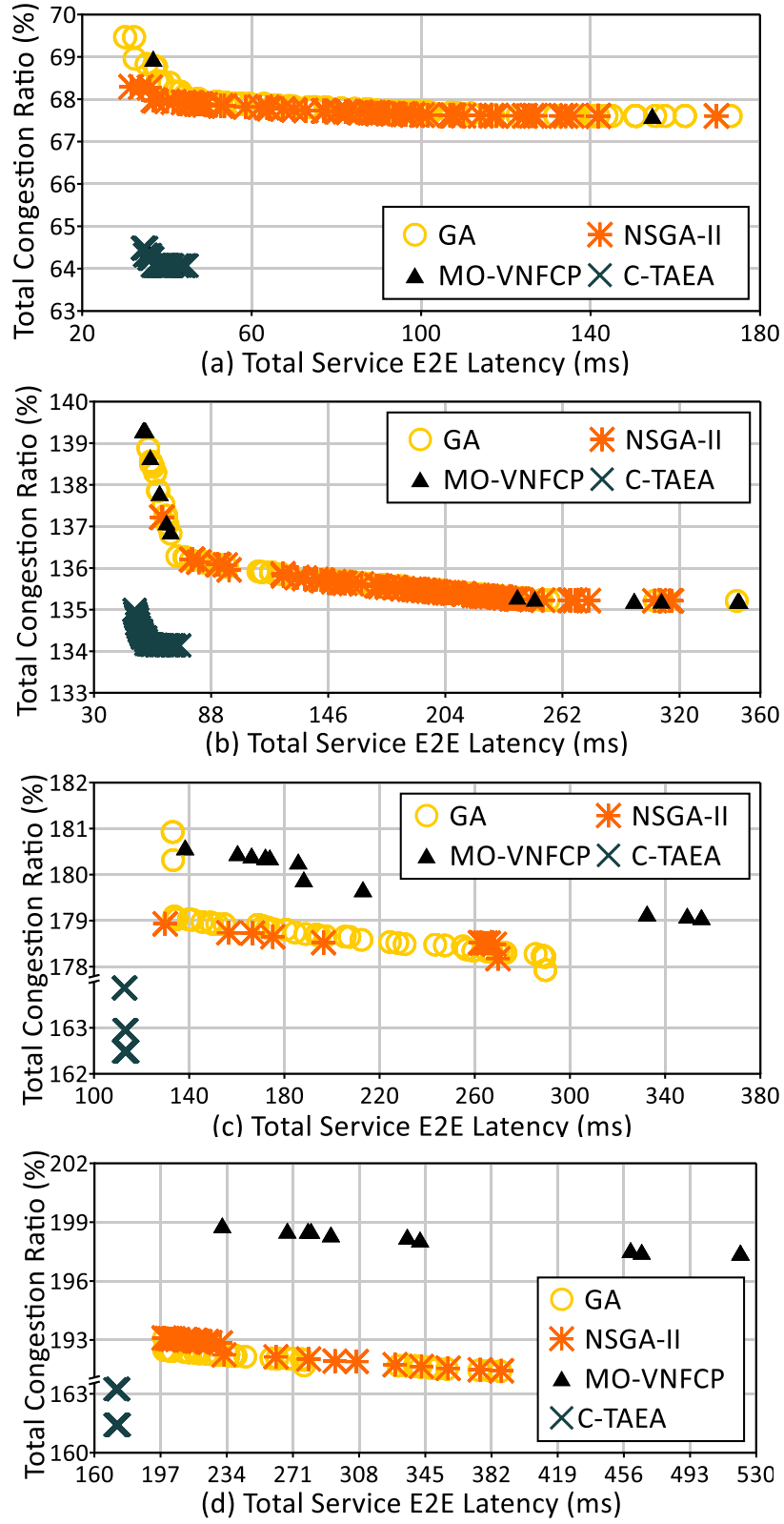


FIGURE 5.6. Non-dominated Solutions on Small-Scale Network: (a) 100 Service Requests Situation, (b) 200 Service Requests Situation, (c) 300 Service Requests Situation, (d) 400 Service Requests Situation.

time performance of different approaches on the 7-node network topology. From this table, we can find that 1) the GA and NSGA-II take a similar time to run 100 iterations, 2) MO-VNFCP costs on average 10% running time of these two algorithms, 3) C-TAEA almost doubles the running time of GA and NSGA-II.

Table 5.4: Time Consumption for Small Scale Network

Total	7-Node Network Topology			
Service	GA	NSGA-II	MO-VNFCP	C-TAEA
100	112min	153min	25min	182min
200	203min	210min	54min	357min
300	239min	234min	70min	526min
400	289min	312min	86min	611min

2) Simulation Results on 29-node Topology

In this subsection, the four algorithms mentioned above are tested on the large-scale (29-node) network. Figure 5.7 (a)-(d) shows the non-dominated fronts obtained for 300, 600, 900, and 1200 service requests, respectively. Table. 5.5 demonstrates the time consumption of different algorithms for 300, 600, 900, and 1200 service requests. From Figure 5.7 and Table. 5.5, we can make a conclusion that 1) although the MO-VNFCP spends the lowest time for the simulation, its performance is the worst with all the solutions dominated by other algorithms' solutions, 2) the heuristic-based GA and heuristic-based NSGA-II performs similar in terms of both the non-dominated fronts and the running time, and these two algorithms can achieve better non-dominated fronts than the MO-VNFCP, 3) the C-TAEA algorithm achieves the best non-dominated fronts under the different total number of service requests. However, it consumes more than two times the running time of GA and NSGA-II.

Table 5.5: Time Consumption for Large Scale Network

Total	29-Node Network Topology			
Service	GA	NSGA-II	MO-VNFCP	C-TAEA
300	1239min	1323min	247min	2955min
600	3551min	3211min	700min	6862min
900	8542min	8722min	1793min	18375min
1200	13672min	13692min	2755min	28219min

5.5 Summary

In this chapter, the multi-objective optimisation for latency-aware SFCs placement problem was investigated in multi-layer networks. The single objective MILP model presented in Chapter 4

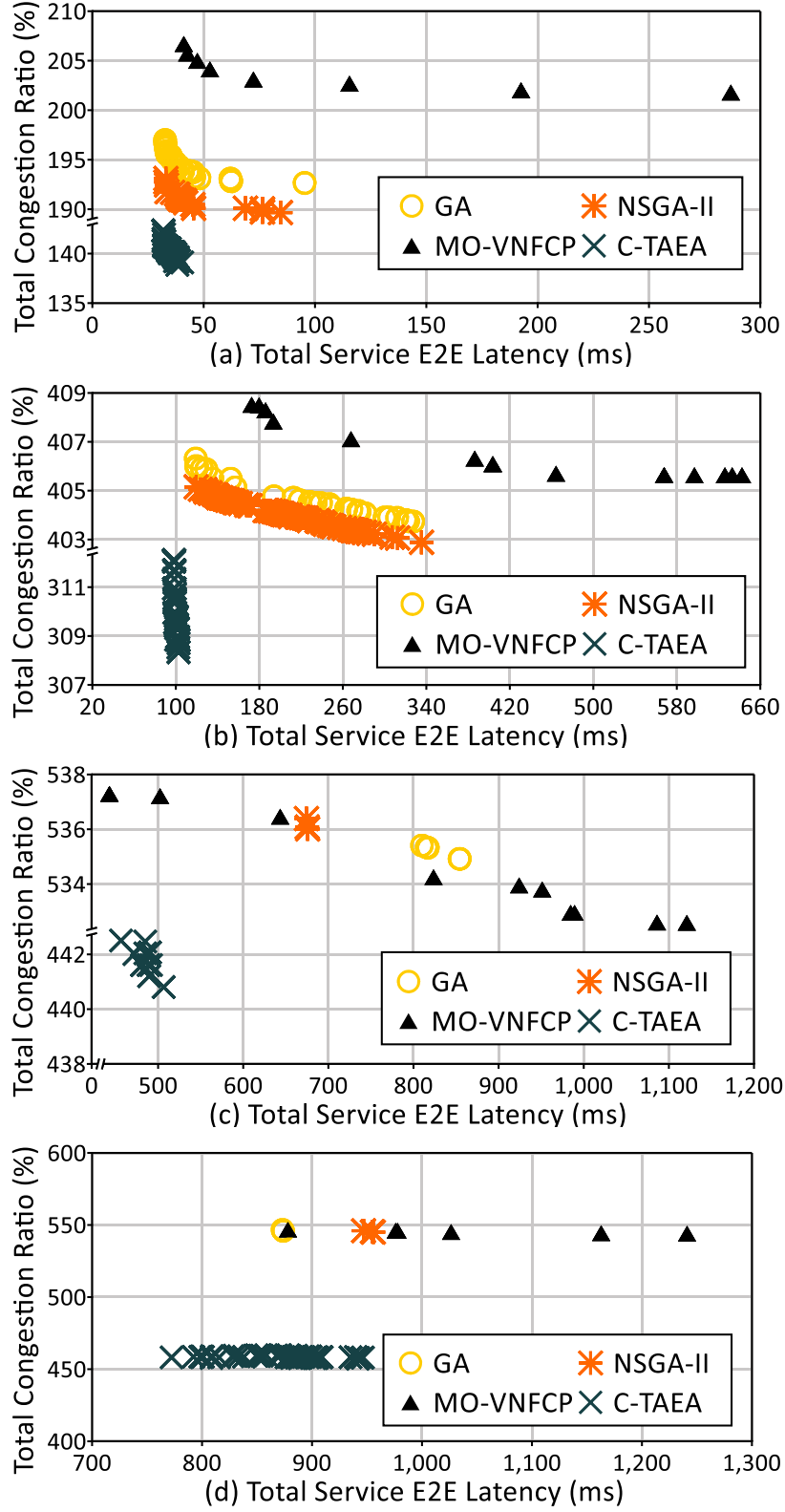


FIGURE 5.7. Non-dominated Solutions on Large-Scale Network: (a) 300 Service Requests Situation, (b) 600 Service Requests Situation, (c) 900 Service Requests Situation, (d) 1200 Service Requests Situation.

was first extended to multi-objective by adding the total CPU congestion minimisation. Then, this model is enhanced with loop avoidance constraints on intermediate switches. In the next, the MILP model with and without an optical layer are evaluated on small-scale networks, with findings demonstrating that, by adding the optical layer, the total service E2E latency can be reduced by ten times, despite the model complexity increasing. Finally, four algorithms were adapted to solve the SFCs placement problem and validated on both small-scale and large-scale networks. The C-TAEA algorithm outperforms the others in terms of the Pareto Fronts approximation performance, but the heuristic-based GA and heuristic-based NSGA-II consume less running time.

MULTI-OBJECTIVE DRL-ASSISTED SFCs PLACEMENT

The SFCs placement solution should meet a variety of QoS criteria, reduce resource congestion at the network edge, and improve service acceptance performance. This chapter proposes a novel approach for addressing these challenges by solving a multi-objective SFCs placement problem using the Pointer Network. This approach is a DRL-based algorithm, called *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm*, that overcomes the limitations of traditional heuristic and meta-heuristic approaches. Then, we run this algorithm iteratively with a set of weights to generate non-dominated solutions with significantly greater hypervolume values than those obtained using existing state-of-the-art algorithms. Furthermore, using chosen weights from non-dominated solutions to execute our algorithm individually can avoid edge resource congestion and attain 98% low-latency SARs during high-workload periods. Finally, the proposed approach is found to be suitable for pragmatic service implementation while reaching 100% of SARs in the deployed use cases on the testbed.

6.1 Introduction

To meet the various and stringent needs of QoS, the fast deployment of 5G and beyond 5G network services will necessitate flexible and efficient resource utilisation. Network service will be implemented as an SFC that connects a series of VNFs [39]. The SFCs placement problem is described as the optimal resource allocation for SFCs deployments while meeting the QoS requirements. Although several studies have been conducted on the SFCs placement problem in edge-cloud networks [3, 39, 58, 60, 131], there are still three major challenges not properly investigated. 1) Strict and diverse QoS criteria must be met to place SFCs in the same infrastructure. A typical example is an E2E latency from 500 ms (e.g., web services) to 1 ms (e.g.,

AR) [4]. 2) MEC nodes with limited resource capacities can cause significant congestion and a high rejection ratio of low-latency services during peak hours [143]. 3) Scalability of the problem due to a large number of computing nodes in edge-cloud networks and the growing number of services. This problem has been proved to be NP-hard, with the computational complexity exponentially increasing as the problem size grows [47].

To address all the aforementioned challenges, a DRL-based model for the multi-objective SFCs placement problem is proposed in the multi-layer edge-cloud networks formed by the virtual function layer, IP layer, and optical layer. In this model, on the one hand, the total E2E latency is minimised to satisfy QoS requirements, on the other hand, the computing resource congestion is minimised for all nodes to improve load balance. The MILP model provided in chapter 5 is taken as the environment in this DRL approach.

To tackle the scalability barrier in the multi-objective SFCs placement problem, instead of using the MILP technique, most existing approaches focus on heuristic and EAs [3, 39, 58, 60]. However, these approaches are still limited in their scalability when dealing with high dimensional and constrained multi-objective problems. In fact, EAs need a large number of iterations and long computation time [83]. Furthermore, in order to discover feasible solutions, both heuristic and EA approaches require expert knowledge of the problem [66, 87]. Thanks to recent advances in DRL, it is now possible to link high-dimensional states and actions to solve NP-hard problems without human interaction in an acceptable time scale [83, 102]. In this chapter, we design and test a DRL-based model using a Pointer Network technique, with an attention mechanism as a "pointer" to pick a member of the input as the output, to handle the multi-objective SFCs placement problem [144]. Our proposed model employs a sequence-to-sequence structure that consists of two Recurrent Neural Networks (RNNs), referred to as encoder and decoder, respectively [83]. The input SFC requests are encoded into a code vector using the encoder RNN. Then, such a code vector is decoded into appropriate placement solutions by the RNN decoder.

The multi-objective optimisation problem can be solved in one of two ways. The first is to find all of the Pareto optimal solutions (Pareto Front (PF)) or a representative subset [145]. In this problem, Pareto optimal solutions include all non-dominated SFCs placement solutions that cannot be enhanced in any way without deteriorating another goal. The second is to combine all of the objective functions into a single composite objective function or to keep only one function as objective and utilise the others as constraints [132]. The weighted-sum approach, which introduces different weights to each individual function belongs to the second group [80, 145]. It has been extensively adopted because of its easy implementation and ability to express the preferences in the selected weights [80]. In our mathematical model, VNFs can be placed at DCs during peak hours and MEC nodes' congestion can be relieved by giving the first objective a low weight and the second one a high weight. Unlike many other researchers who choose the weights according to their prior knowledge, we will choose the weights based on achieved non-dominated

solutions. Because the mapping from weight space to objective space is not guaranteed to be isomorphic [80], we utilise the Chebyshev scalarisation approach to approximate the PFs. It has several advantages, such as it can 1) search solutions in both convex and non-convex areas, 2) get a better spread among the set of non-dominated solutions, and 3) be independent on the actual weights used [84]. In this chapter, we are going to apply the non-linear Chebyshev function to the actor-critic algorithm to solve the multi-objective SFCs placement problem.

The following are the main contributions of this chapter:

1) **Modelling:** A DRL model based on the Pointer Network technique is designed with the MILP models proposed in chapter 5 as its environment to solve large-scale multi-objective SFCs placement problems in edge-cloud networks.

2) **Algorithm Design:** A novel DRL algorithm, called *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm*, is developed for training. Under various workload scenarios, it can be used to estimate PFs, improve the MEC resource usage and enhance the ultra-low latency SAR.

3) **Simulation Evaluation:** The proposed algorithm is evaluated on both small and large scale networks using simulation. In terms of PFs approximation, it outperforms other state-of-the-art methods proposed in [3, 36, 56, 83, 133], with the highest hypervolume values. In terms of edge resource usage and SAR, it outperforms the First-Fit algorithm and the Data-Rate-based algorithm in [131].

4) **Experimental Evaluation:** The proposed algorithm is integrated and experimentally evaluated with an OSM for SFCs placement in an OpenStack enabled testbed that emulates a long haul optical edge-cloud network. Experimental results validate that it outperforms the First-Fit algorithm and the Data-Rate-based algorithm [131] with respect to the service acceptance performance.

The rest of this chapter is organised as follows. In Section 6.2, problem formulation and DRL modelling are presented. Then, the proposed DRL algorithm is explained in Section 6.3. In the next section, we proceed with our simulation setup and results. Then, our experimental testbed is detailed and experimental results are analysed in Section 6.5. Finally, Section 6.6 summarises the highlights of this chapter.

6.2 Multi-Objective SFCs Placement Problem Formulation

6.2.1 SFCs placement in Edge-Cloud Networks

The multi-layer edge-cloud network considered can be modelled as a directed graph $G = (N, L^P)$, where N is the set of nodes, and L^P is the set of optical links. Among all of the nodes, N_v indicates a set of nodes with computing resources, whereas N_{SWN} indicates a set of nodes with just switching capability. N_v includes MEC servers N_{MEC} , EDCs N_{EDC} , and CDC N_{CDC} . MEC servers, in comparison to DCs, have limited resources and are placed located near 5G base stations (i.e., eNodeBs) [25]. In this model, the switching node includes the router and the OXCs.

A computing node $n \in N_v$ is characterised by the computing resources and buffer, denoted as n^{cpu} and n^{buf} , respectively. Switching node $n \in N_{SWN}$ has only buffering resources. A optical link $(n, n') \in L^P$ denotes a link with a length of $len_{(n, n')}$ from node $n \in N$ to $n' \in N$. The total number of wavelength in the network is W , and the w -th wavelength in the collection of wavelength is $w \in [1, |W|]_z$. (The number of elements in the set is denoted by $|\cdot|$, and the set of integers from A to B can be represented by $[A, B]_z$). In the physical link (n, n') , the transmission capacity of w -th wavelength is $B_{(n, n')}^w$, and the light speed is lv . The set of lightpath L^P includes links (l, l') connecting computing node $l \in N_v$ to the other computing node $l' \in N_v$.

Moving to the virtual layer, the set of distinct types of VNFs is represented as M , and the specific type VNF is represented as $m \in M$. Different VNFs require different amounts of computing and buffering resources, denoted as β_m^{cpu} , and β_m^{buf} , respectively. Besides, different VNFs has different scaling attribute denoted as δ_m . For each flow, the output data rate v_{output} is determined by the input data rate v_{input} and the scaling attribute, which can be calculated by $v_{output} = \delta_m \cdot v_{input}$. Furthermore, $suit_{n, m}$ is a binary indication that indicates whether or not the VNF m can be placed on the node n .

In the offline scenario, the set of service requests K is known in advance. Each service request $k \in K$ is made up of a chain of VNFs. O_k denotes the number of required VNFs in service k , and the $o \in [0, |O_k| + 1]_z$ VNF is used to represent the o -th VNF in the SFC k , where $o = 0$ is the source of SFC s^k and $o = |O_k| + 1$ is the destination d^k . In addition, the other service-related features include the data rate v^k , required E2E latency DR^k , and packet size TS^k .

Considering that different VNFs have different scaling attribute, data rate $v_m^{k, o}$ for the o -th VNF in the service request k can be different from the v^k . For the first VNF in SFC, $v_m^{k, 1} = v^k$. For the other VNF, $v_m^{k, o} = \delta_{m'}^{k, o-1} \cdot v_{m'}^{k, o-1}$. The computing resources and buffer required by the o -th VNF of service request k , can be represented by $C_m^{k, o, cpu}$ and $C_m^{k, o, buf}$, respectively. These required resource amount can be affected by the data rate as they are calculated by the following two equations: $C_m^{k, o, cpu} = \beta_m^{cpu} \cdot v_m^{k, o}$ and $C_m^{k, o, buf} = \beta_m^{buf} \cdot v_m^{k, o}$ in our model.

The SFCs placement solutions include: 1) the node mapping variable $x_{n, m}^{k, o}$, indicating whether or not the m type VNF in the service request is placed on the node n ; 2) the lightpath mapping variable $y_{(l, l'), w}^{k, (o, o+1), m}$, showing whether or not the w wavelength in the lightpath (l, l') is selected for the traffic routing from the o -th to the $(o+1)$ -th VNF; 3) the link mapping variable $tw_{(l, l'), (n, n'), w}^{k, (o, o+1), m}$, indicating whether or not w wavelength is selected in the physical link (n, n') on the path between the o -th and $(o+1)$ -th VNF. Based on these solutions, the total service E2E latency of all service requests D^{total} and the total CPU utilisation ratio U^{cpu} can be obtained.

6.2.2 DRL Model Formulation

The DRL approach is adopted to solve this multi-objective SFCs placement problem. To simplify the problem and save the running time [83], we use the scalarisation method and decompose it into a series of subproblems with different weights. The number of subproblems is I and the i -th

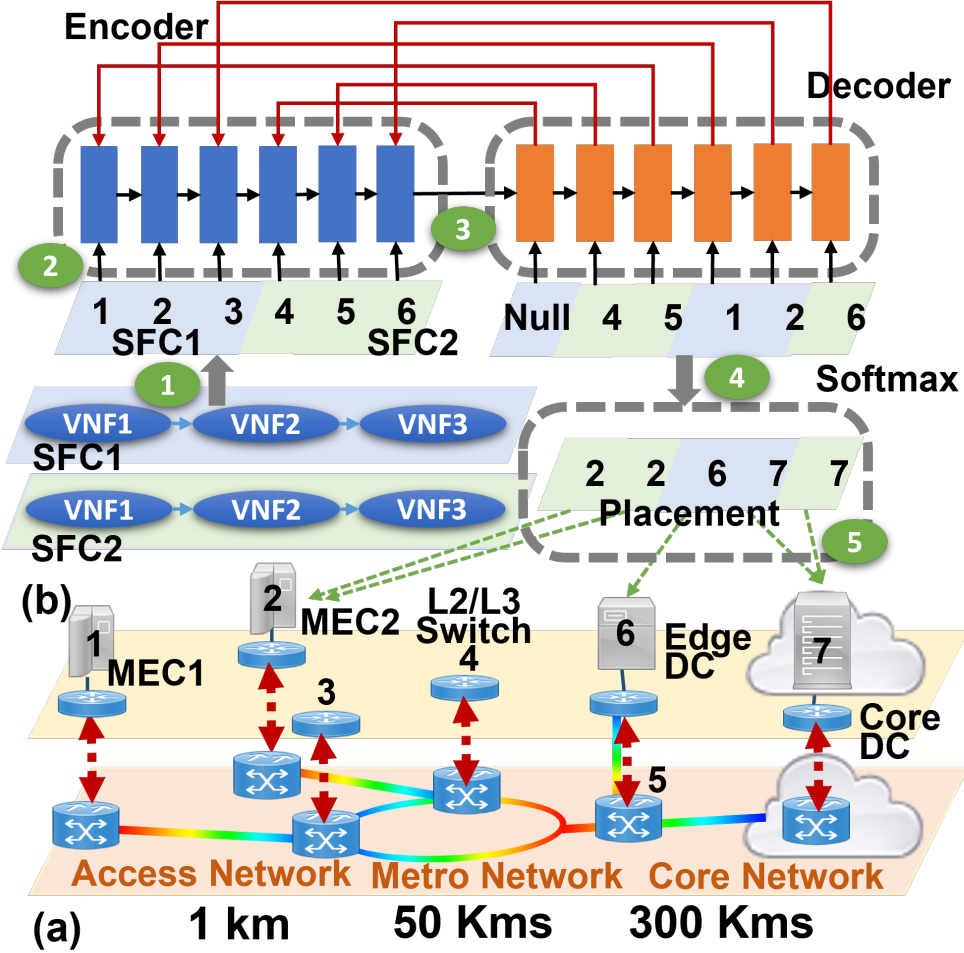


FIGURE 6.1. Encoding and decoding of SFC: a) 7-Node Simulation Topology. b) Pointer Network Architecture.

subproblem is represented as $i \in [1, |I|]_Z$. A neural network (NN) is used to model each subproblem with θ_i as the weight. The optimised subproblem is represented by θ_i^* . The MILP model designed in Chapter 5 is taken as the environment for the DRL model, and the two objectives are defined in the same way.

Each SFCs placement subproblem is described as a Pointer Network with two RNNs, an encoder and a decoder, both of which are made up of Long Short-Term Memory (LSTM) cells. Figure 6.1 (b) shows the architecture of the Pointer Network (the encoder is represented by blue cells on the left while the decoder is orange cells on the right). The encoder's input is a list of all service requests $\mathcal{R}_i = \{r_i^1, \dots, r_i^k, \dots, r_i^{|K|}\}$. Each service request comprises of the needed VNFs $r_i^k = \{f_i^{k,1}, \dots, f_i^{k,o}, \dots, f_i^{k,|O_k|}\}$. To make the model design process easier, the \mathcal{R}_i is represented as $\mathcal{R}_i = \{f_i^1, \dots, f_i^{|K| \times |O_k|}\}$. The coming two SFCs requests are therefore encoded as "1,2,3,4,5,6" in the Figure 6.1 (b) step 2. The state in this model comprises all of the service requests as well as

their characteristics $\langle s_i^k, d_i^k, v_i^k, TS_i^k, DR_i^k \rangle$. The decoder's output is a list of placement solutions (actions) specifying which host node the VNF should be placed on, $\mathcal{H}_i = \{h_i^1, \dots, h_i^{|K| \times |O_k|}\}$. As shown in Figure 6.1 (b), the VNFs should be placed chosen in step 3 are placed to the computing nodes in step 4.

When solving the i -th subproblem, the probability of allocating a VNF f relies on the placement solutions of previous allocated VNFs at each stage. Such conditional probability is shown in Equation (6.1).

$$(6.1) \quad p_i(\mathcal{H}_i | \mathcal{R}_i) = \prod_{f_i=1}^{|K| \times |O_k|} p_i(h_i^f | h_i^{(<f)}, \mathcal{R}_i) \quad \forall i \in [1, |I|]_z$$

In Pointer Network, the attention mechanism is utilised to determine the aforementioned probability. To be more precise, at first, the encoder turns the input sequence to a high-dimensional vector, which is then fed to the generating network. Next, using the attention mechanism, the generating network produces a pointer to the input items. Encoder hidden states are $(e_{1,i}, \dots, e_{|K| \times |O_k|, i})$, and decoder hidden state at decoding step t is $d_{t,i}$. The Pointer Network selects a member of the input sequence as the output according to the pointer provided by the attention mechanism [144]. The pointer is a softmax probability distribution with dictionary size equal to the length of the input [144]. At each output time t , the attention vector u_i^t is obtained by equation (6.2), and the vector u_i^t is then softmax normalised (equation (6.3)) to be an output distribution over the dictionary of inputs [144]. v_i , $W_{1,i}$, and $W_{2,i}$ are all learnable parameters in the output model.

$$(6.2) \quad u_{f,i}^t = v_i^T \tanh(W_{1,i} \cdot e_{f,i} + W_{2,i} \cdot d_{t,i}) \quad \forall f \in [1, |K| \times |O_k|]_z, i \in [1, |I|]_z$$

$$(6.3) \quad p(H_t | H_1, \dots, H_{t-1}, \mathcal{R}) = \text{softmax}(u^t)$$

The stochastic policy gradient technique is used to learn the placement policy that maps all the provided service requests (state) to host nodes (action) and optimises the parameters of a Pointer Network denoted as θ_i . The stochastic policy that generalises a placement strategy for all service requests in the i -th subproblem is indicated as $\pi_{\theta_i}(\mathcal{H}_i | \mathcal{R}_i)$, which sets a high probability to the placement solutions with low reward and a low probability to the placement solutions with high reward. Equation (6.1) can be used for such policy calculation.

Given all of the solutions for placement, the expected total service E2E latency $J_D^\pi(\theta_i | \mathcal{R}_i)$ and the expected total congestion $J_U^\pi(\theta_i | \mathcal{R}_i)$ can be defined as follows:

$$(6.4) \quad J_D^\pi(\theta_i | \mathcal{R}_i) = \mathbb{E}_{\mathcal{H}_i \sim \pi_{\theta_i}(\cdot | \mathcal{R}_i)} [D^{total}(\mathcal{H}_i)] \quad \forall i \in [1, |I|]_z$$

$$(6.5) \quad J_U^\pi(\theta_i | \mathcal{R}_i) = \mathbb{E}_{\mathcal{H}_i \sim \pi_{\theta_i}(\cdot | \mathcal{R}_i)} [U^{cpu}(\mathcal{H}_i)] \quad \forall i \in [1, |I|]_z$$

Due to the fact that the SFCs placement problem is a constrained optimisation problem with no closed-form solution for general constraints [86], we introduce a penalty signal to the reward

function to indicate the degree of constraint dissatisfaction and guide the solution to feasible regions. We utilise a constraint violation idea given in [87] to construct the penalty signal and make the problem an unconstrained problem. It is calculated in equation (6.6) based on the values of each inequality constraint $\xi_\kappa(\mathcal{H}_i), \kappa = 1, 2, \dots, \epsilon$ and the values of each equality constraint $\phi_\lambda(\mathcal{H}_i), \lambda = 1, 2, \dots, \tau$.

$$(6.6) \quad \psi(\mathcal{H}_i) = \sum_{\kappa=1}^{\epsilon} \max(0, \xi_\kappa(\mathcal{H}_i))^2 + \sum_{\lambda=1}^{\tau} |\phi_\lambda(\mathcal{H}_i)| \forall i \in [1, |I|]_z$$

Equation (6.7) gives the expectation of the penalty signal linked with a policy.

$$(6.7) \quad J_C^\pi(\theta_i|\mathcal{R}) = \mathbb{E}_{\mathcal{H}_i \sim \pi_{\theta_i}(\cdot|\mathcal{R}_i)} [\psi(\mathcal{H}_i)] \forall i \in [1, |I|]_z$$

The objective function is the sum of the expected weighted-sum reward and the penalty signal in each subproblem. The weight for the total service E2E latency minimisation is denoted as $\alpha_i^D \in [0, 1]$, and the weight for the total congestion minimisation is represented as $\alpha_i^U \in [0, 1]$. For all the subproblems, uniformly spread weights vector are applied (1,0), (0.99,0.01), ..., (0.01,0.99), (0,1). In such a setting, the DRL will determine which policy minimises the weighted-sum objective while fulfilling all the constraints. Equation (6.8) is the new objective function defining the quality of the policy.

$$(6.8) \quad \min_{\theta_i} J_L^\pi(\theta_i|\mathcal{R}_i) = \min_{\theta_i} [\alpha_i^D \cdot J_D^\pi(\theta_i|\mathcal{R}_i) + \alpha_i^U \cdot J_U^\pi(\theta_i|\mathcal{R}_i) + J_C^\pi(\theta_i|\mathcal{R}_i)]$$

The stochastic gradient descent (SGD) approach is used to optimise the parameters of NN. The gradient of the objective is calculated by the log-likelihood method with a baseline estimator $b_{\theta_{v,i}}$, which is parameterised by weights $\theta_{v,i}$. The baseline function is independent of policy π and calculates the expected goal of reducing gradient variance [109]. The penalised objective value achieved in each training iteration is $L_i^\pi(\mathcal{H}_i|\mathcal{R}_i)$. It can be computed by equation (6.10), where $D_i^\pi(\mathcal{H}_i|\mathcal{R}_i)$ is the total latency and $U_i^\pi(\mathcal{H}_i|\mathcal{R}_i)$ is the total computing resource utilisation ratio.

$$(6.9) \quad \nabla_{\theta_i} J_L^\pi(\theta_i|\mathcal{R}_i) = \mathbb{E}_{\mathcal{H}_i \sim \pi_{\theta_i}(\cdot|\mathcal{R}_i)} [(L_i^\pi(\mathcal{H}_i|\mathcal{R}_i) - b_{\theta_{v,i}}(\mathcal{R}_i)) \cdot \nabla_{\theta_i} \log \pi_{\theta_i}(\mathcal{H}_i|\mathcal{R}_i)]$$

$$(6.10) \quad L_i^\pi(\mathcal{H}_i|\mathcal{R}_i) = \alpha_i^D \cdot D_i^\pi(\mathcal{H}_i|\mathcal{R}_i) + \alpha_i^U \cdot U_i^\pi(\mathcal{H}_i|\mathcal{R}_i) + \psi(\mathcal{H}_i|\mathcal{R}_i)$$

Furthermore, the gradient is estimated using Monte Carlo sampling with B samples and sampling a single as indicated in equation (6.11).

$$(6.11) \quad \nabla_{\theta_i} J_L^\pi(\theta_i) \approx \frac{1}{B} \sum_{j=1}^B (L_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}) - b_{\theta_{v,i}}(\mathcal{R}_{i,j})) \cdot \nabla_{\theta_i} \log \pi_{\theta_i}(\mathcal{H}_{i,j}|\mathcal{R}_{i,j})$$

Throughout the training phase, another NN critic is used to assess the expected reward. The baseline is trained using SGD on the mean squared error objective between the actual observed reward obtained from the environment and its predictions $b_{\theta_{v,i}}(\mathcal{R}_{i,j})$.

$$(6.12) \quad \mathcal{L}(\theta_v) = \frac{1}{B} \sum_{j=1}^B \|b_{\theta_v}(\mathcal{R}_{i,j}) - L_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j})\|^2$$

6.3 Multi-Objective DRL Algorithm

In this section, we propose a *Multi-Objective DRL algorithm* for the SFCs placement problem. In this algorithm, the complete problem is divided into multiple scalarised subproblems and the non-dominated solutions can be obtained after solving all of the subproblems. It is worth mentioning that we train each subproblem using the homotopy optimisation technique [82]. Due to the existence of many local minimum throughout the landscape of the mean squared error loss $\mathcal{L}(\theta_{v,i})$, a flatter value metric loss $\mathcal{L}(\theta_{h,i})$ is introduced to guide policy to the global minimum. The homotopy path ω connecting $\mathcal{L}(\theta_{v,i})$ and $\mathcal{L}(\theta_{h,i})$ provides better chances of discovering the global optimal parameters θ^* . Equation (6.13) presents the new loss function. By gradually raising the value ω from 0 to 1, the loss function is shifted from $\mathcal{L}(\theta_{v,i})$ to $\mathcal{L}(\theta_{h,i})$. $\mathcal{L}(\theta_{v,i})$ first guarantees the prediction is near to the true expected total reward and then $\mathcal{L}(\theta_{h,i})$ offers auxiliary pull along the direction with better utility.

$$(6.13) \quad (1 - \omega) * \mathcal{L}(\theta_{v,i,\omega}) + \omega * \mathcal{L}(\theta_{h,i,\omega}) = \frac{1}{B} \sum_{j=1}^B [(1 - \omega) * ||b_{\theta_{v,i,\omega}}(\mathcal{R}_{i,j}) - L_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j})||^2 + \omega * (b_{\theta_{v,i,\omega}}(\mathcal{R}_{i,j}) - L_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}))]$$

Algorithm 2 describes the proposed *Multi-Objective DRL algorithm*. The input parameters include the index of subproblems I , training steps S , homotopy path Ω , service requests \mathcal{R}_i , batch size B , and MILP models for subproblems. For each subproblem, the output parameters contain the optimised parameters $\theta_{i,\omega}^*$ and $\theta_{v,i,\omega}^*$. For the first two subproblems, the expected total E2E latency and total congestion are learned by Algorithm 3 (line 4-21), whereas, for the remaining subproblems, Algorithm 4 is adopted to learn these two objectives (line 23-33).

At first, the weights for two objectives are calculated for each subproblem (line 6, line 8 and line 23). Then, each subproblem is trained along the homotopy path. When $\omega = 0$, two parameters $\theta_{i,\omega}$, $\theta_{v,i,\omega}$ are initialised randomly (line 12 and line 26). With the increase of ω , the $\theta_{i,\omega}$ and $\theta_{v,i,\omega}$ are fed with the optimal $\theta_{i,\omega-1}^*$ and $\theta_{v,i,\omega-1}^*$ (line 15-16 and line 29-30). This homotopy technique is effective because it applies the optimisation result from the previous phase in each update step.

Inspired by the asynchronous advantage actor-critic (A3C) algorithm proposed in [109], we design an *Actor-Critic SFCs Placement Algorithm* for the two subproblems with $\alpha_i^D = 1, \alpha_i^U = 0$ and $\alpha_i^D = 0, \alpha_i^U = 1$, and a *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* for the rest subproblems based on SGD. During the training procedure, T instances of all service requests are produced. For training, both algorithms have two networks: 1) An actor-network, which is the Pointer Network, offers the probability distribution for picking the host nodes and calculates the placement solution for the next VNF (line 5-7 in both Algorithm 3 and Algorithm 4). 2) A critic-network, which assesses the expected reward given a certain issue state, maps an input sequence $\mathcal{R}_{i,j}$ into a baseline prediction $b_{\theta_{v,i}}(\mathcal{R}_{i,j})$. The critic-network is based on the same

Algorithm 2: Multi-objective DRL Algorithm

```

1 Input: the number of subproblems  $I$ ; the number of homotopy path  $\Omega$ ; the model of
   subproblem including MILP environment; the training steps  $S$ ; batch size  $B$ 
2 Output: the optimal params  $\theta_{i,\omega}^*$  and  $\theta_{v,i,\omega}^*$  for each subproblem
3 for  $i = 0, \dots, I - 1$  do
4   if  $i == 0$  or  $i == 1$  then
5     if  $i == 0$  then
6        $\alpha_i^D = 1, \alpha_i^U = 0,$ 
7     else
8        $\alpha_i^D = 0, \alpha_i^U = 1,$ 
9     end if
10    for  $\omega = 0, 1/\Omega, 2/\Omega, \dots, 1$  do
11      if  $\omega == 0$  then
12        initialise params  $\theta_{i,\omega}, \theta_{v,i,\omega}$  randomly,
13         $J_D^\pi(\theta_{i,\omega}|\mathcal{R}_i), J_U^\pi(\theta_{i,\omega}|\mathcal{R}_i), \theta_{i,\omega}^*, \theta_{v,i,\omega}^* \leftarrow \text{Algorithm 3}(\theta_{i,\omega}, \theta_{v,i,\omega}, \alpha_i^D, \alpha_i^U, \omega)$ 
14      else
15         $\theta_{i,\omega} \leftarrow \theta_{i,\omega-1/\Omega}^*, \theta_{v,i,\omega} \leftarrow \theta_{v,i,\omega-1/\Omega}^*$ 
16         $J_D^\pi(\theta_{i,\omega}|\mathcal{R}_i), J_U^\pi(\theta_{i,\omega}|\mathcal{R}_i), \theta_{i,\omega}^*, \theta_{v,i,\omega}^* \leftarrow \text{Algorithm 3}(\theta_{i,\omega}, \theta_{v,i,\omega}, \alpha_i^D, \alpha_i^U, \omega)$ 
17      end if
18    end for
19    if  $i == 1$  then
20       $J_D^{min} = \min(J_D^\pi(\theta_{i,0}|\mathcal{R}_i), J_D^\pi(\theta_{i,1}|\mathcal{R}_i)), J_U^{min} = \min(J_U^\pi(\theta_{i,0}|\mathcal{R}_i), J_U^\pi(\theta_{i,1}|\mathcal{R}_i))$ 
21    else
22    else
23       $\alpha_i^D = 1 - (i - 1) * 0.01, \alpha_i^U = 0 + (i - 1) * 0.01,$ 
24      for  $\omega = 0, 1/\Omega, 2/\Omega, \dots, 1$  do
25        if  $\omega == 0$  then
26          initialise params  $\theta_{i,\omega}, \theta_{v,i,\omega}$  randomly,
27           $\theta_{i,\omega}^*, \theta_{v,i,\omega}^* \leftarrow \text{Algorithm 4}(\theta_{i,\omega}, \theta_{v,i,\omega}, \alpha_i^D, \alpha_i^U, \omega, J_D^{min}, J_U^{min})$ 
28        else
29           $\theta_{i,\omega} \leftarrow \theta_{i,\omega-1/\Omega}^*, \theta_{v,i,\omega} \leftarrow \theta_{v,i,\omega-1/\Omega}^*$ 
30           $\theta_{i,\omega}^*, \theta_{v,i,\omega}^* \leftarrow \text{Algorithm 4}(\theta_{i,\omega}, \theta_{v,i,\omega}, \alpha_i^D, \alpha_i^U, \omega, J_D^{min}, J_U^{min})$ 
31        end if
32      end for
33    end if
34 end for
35 return  $\theta_{i,\omega}^*$  and  $\theta_{v,i,\omega}^*$  for all subproblems

```

Algorithm 3: Actor-Critic SFCs Placement Algorithm

```

1 Input: the initialised params  $\theta_{i,\omega}, \theta_{v,i,\omega}$ ; the model of subproblem including service requests  $\mathcal{R}_i$ 
   and MILP environment; the weights for objective  $\alpha_i^D, \alpha_i^U$ ; the path  $\omega$ ; the training steps  $S$ ; batch
   size  $B$ 
2 Output: the optimal params  $\theta_{i,\omega}^*$  and  $\theta_{v,i,\omega}^*$  for each subproblem
3 for  $s = 1, \dots, S$  do
4   generate  $T$  instances of service requests  $\mathcal{R}_{i,j}$ 
5   while not terminated do
6     select  $h_i^f$  for the VNF  $r_i^f$  according to probability calculated by equation (6.1)
7   end while
8   sample the input  $\mathcal{R}_{i,j}$  for  $j \in 1, \dots, B$ 
9   sample the correspondence policy  $\pi_{\theta_{i,j}}$  for  $j \in 1, \dots, B$ 
10  compute  $D_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}), U_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}), \psi_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}), L_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j})$  for  $j \in 1, \dots, B$ 
11  calculate  $\nabla_{\theta_{i,\omega}} J_L^\pi(\theta_{i,\omega})$  by equation (6.11)
12  calculate  $\mathcal{L}(\theta_{v,i,\omega})$  by equation (6.13)
13   $\theta_{i,\omega} \leftarrow \text{ADAM}(\theta_{i,\omega}, \nabla_{\theta_{i,\omega}} J_L^\pi(\theta_{i,\omega}))$ 
14   $\theta_{v,i,\omega} \leftarrow \text{ADAM}(\theta_{v,i,\omega}, \nabla_{\theta_{v,i,\omega}} \mathcal{L}(\theta_{v,i,\omega}))$ 
15 end for
16 return  $\theta_{i,\omega}^*, \theta_{v,i,\omega}^*, J_D^\pi(\theta_{i,\omega}|\mathcal{R}_i), J_U^\pi(\theta_{i,\omega}|\mathcal{R}_i)$  for all subproblems
    
```

design as the Pointer Network’s encoder to encode an input sequence. The critic output, a baseline prediction, is then decoded from the encoder’s hidden state.

The objective function to be optimised is the only difference between the two designed Actor-Critic Algorithms. In Algorithm 3, the objective function is equation (6.13). In Algorithm 4, the Chebyshev scalarisation method [85] is adopted. It is based on minimising the distance between a point and the Utopian point (J_D^{\min}, J_U^{\min}) in terms of the factor that maximises such distance. Therefore, the objective function in Algorithm 4 is $(1 - \omega) * \mathcal{D}(\theta_{v,i,\omega}) + \omega * \mathcal{D}(\theta_{h,i,\omega})$ when the *latency – difference* is greater than the *balance – difference*, and is $(1 - \omega) * \mathcal{U}(\theta_{v,i,\omega}) + \omega * \mathcal{U}(\theta_{h,i,\omega})$ when the *latency – difference* is smaller than the *balance – difference*. The *latency – difference* and the *balance – difference* are calculated in line 11-12 based on the J_D^{\min} and the J_U^{\min} calculated by line 19-21 in Algorithm 2.

The proposed *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* has a time complexity of $\mathcal{O}(|I| \cdot |K| \cdot |O_k| \cdot d_h^2)$, where $|I|$ stands for the number of subproblems, $|K|$ for the number of service requests, $|O_k|$ for the number of VNFs in each service request, and d_h^2 for the time complexity of the RNN [83].

6.4 Simulation-based Evaluation

6.4.1 Simulation Setup

The simulation network topologies, resource capacities, VNF properties, SFC-related parameters, and other simulation settings are the same as those in Section 5.4 of Chapter 5. We run the

Algorithm 4: Chebyshev-assisted Actor-Critic SFCs Placement Algorithm

```

1 Input: the initialised params  $\theta_{i,\omega}, \theta_{v,i,\omega}$ ; the model of subproblem including service requests  $\mathcal{R}_i$ 
   and MILP environment; the weights for objective  $\alpha_i^D, \alpha_i^U$ ; the path  $\omega$ ; the minimum expected
   objective:  $J_D^{min}$  and  $J_U^{min}$ ; the training steps  $S$ ; batch size  $B$ 
2 Output: the optimal params  $\theta_{i,\omega}^*$  and  $\theta_{v,i,\omega}^*$  for each subproblem
3 for  $s = 1, \dots, S$  do
4   generate  $T$  instances of service requests  $\mathcal{R}_{i,j}$ 
5   while not terminated do
6     select  $h_i^f$  for the VNF  $r_i^f$  according to probability calculated by equation (6.1)
7   end while
8   sample the input  $\mathcal{R}_{i,j}$  for  $j \in 1, \dots, B$ 
9   sample the correspondence policy  $\pi_{\theta_{i,j}}$  for  $j \in 1, \dots, B$ 
10  compute  $D_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}), U_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}), \psi_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j})$  for  $j \in 1, \dots, B$ 
11   $latency - difference = \alpha_i^D * (D_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}) - J_D^{min}),$ 
12   $balance - difference = \alpha_i^U * (U_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}) - J_U^{min}),$ 
13  if  $latency - difference > balance - difference$  then
14     $\nabla_{\theta_{i,\omega}} J_D^\pi(\theta_{i,\omega}) \leftarrow 1/B \sum_{j=1}^B (D_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}) + \psi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}) - b_{\theta_{v,i,\omega}}(\mathcal{R}_{i,j})) \cdot \nabla_{\theta_{i,\omega}} \log \pi_{\theta_{i,\omega}}(\mathcal{H}_{i,j}|\mathcal{R}_{i,j})$ 
15     $(1-\omega) * \mathcal{D}(\theta_{v,i,\omega}) + \omega * \mathcal{D}(\theta_{h,i,\omega}) = \frac{1}{B} \sum_{j=1}^B [(1-\omega) * ||b_{\theta_{v,i,\omega}}(\mathcal{R}_{i,j}) - \alpha_i^D * D_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}) -$ 
16     $\psi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j})||^2 + \omega * (b_{\theta_{v,i,\omega}}(\mathcal{R}_{i,j}) - \alpha_i^D * D_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}) - \psi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}))]$ 
17     $\theta_{i,\omega} \leftarrow ADAM(\theta_{i,\omega}, \nabla_{\theta_{i,\omega}} J_D^\pi(\theta_{i,\omega}))$ 
18     $\theta_{v,i,\omega} \leftarrow ADAM(\theta_{v,i,\omega}, \nabla_{\theta_{v,i,\omega}} \mathcal{D}(\theta_{v,i,\omega}))$ 
19  else
20     $\nabla_{\theta_{i,\omega}} J_U^\pi(\theta_{i,\omega}) \leftarrow 1/B \sum_{j=1}^B (U_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}) + \psi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}) - b_{\theta_{v,i,\omega}}(\mathcal{R}_{i,j})) \cdot \nabla_{\theta_{i,\omega}} \log \pi_{\theta_{i,\omega}}(\mathcal{H}_{i,j}|\mathcal{R}_{i,j})$ 
21     $(1-\omega) * \mathcal{U}(\theta_{v,i,\omega}) + \omega * \mathcal{U}(\theta_{h,i,\omega}) = \frac{1}{B} \sum_{j=1}^B [(1-\omega) * ||b_{\theta_{v,i,\omega}}(\mathcal{R}_{i,j}) - \alpha_i^U * U_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}) -$ 
22     $\psi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j})||^2 + \omega * (b_{\theta_{v,i,\omega}}(\mathcal{R}_{i,j}) - \alpha_i^U * U_i^\pi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}) - \psi(\mathcal{H}_{i,j}|\mathcal{R}_{i,j}))]$ 
23     $\theta_{i,\omega} \leftarrow ADAM(\theta_{i,\omega}, \nabla_{\theta_{i,\omega}} J_U^\pi(\theta_{i,\omega}))$ 
24     $\theta_{v,i,\omega} \leftarrow ADAM(\theta_{v,i,\omega}, \nabla_{\theta_{v,i,\omega}} \mathcal{U}(\theta_{v,i,\omega}))$ 
25  end if
26 end for
27 return  $\theta_{i,\omega}^*$  and  $\theta_{v,i,\omega}^*$  for all subproblems

```

proposed Multi-Objective DRL Algorithm and other state-of-the-art algorithms for both small-scale (Figure 6.1 (a) 7-node) and large-scale (Figure 5.3 29-node) networks. For the simulation, the benchmark approaches are running on an IBM server with 24GB RAM and an 8-core CPU processor, while DRL-related training and testing are carried out on a CORSAIRONE PRO workstation equipped with an 8-core CPU processor and TensorFlow.

6.4.2 Simulation Results and Analysis

In both 7-node and 29-node network, to approximate the PFs, we run six algorithms: *GA*, *NSGA-II*, *MO-VNFCP Algorithm*, *Constrained Two Archive Evolutionary Algorithm (C-TAEA)*, *Linear-Multi-Objective Deep Reinforcement Learning (MODRL) Algorithm (LMODRL)* and *Chebyshev-MODRL Algorithm (CMODRL)*. The first three benchmark algorithms are based on the approaches used in [3, 36, 56] to place SFCs, respectively. The *C-TAEA* algorithm is chosen as the

benchmark because it is specifically designed for addressing constrained multi-objective optimisation problems and it exhibits a strong capacity to balance convergence, diversity, and feasibility at the same time [133]. These four algorithms have been modified to match this multi-objective SFCS placement problem, and their detailed algorithm designs are listed in Chapter 5. The fifth benchmark comes from [83], which solves each sub-problem of the multi-objective DRL problem using the linear weighted-sum scalarisation approach. The last algorithm is proposed in this chapter (Algorithm 2), which is based on the non-linear Chebyshev scalarisation technique. Three EA-based algorithms conduct 100 generations for 100 populations, the *MO-VNFCP Algorithm* runs 100 iterations, while the last two DRL algorithms handle 100 weights uniformly distributed sub-problems in both 7-node and 29-node network topologies.

6.4.2.1 Non-dominated Fronts on Small Scale Networks

Figure 6.3 shows the non-dominated fronts obtained for addressing multi-objective SFCS placement of 100, 200, 300, and 400 service requests on the 7-node network. The *MO-VNFCP Algorithm* performs poorly in all cases since it can only find neighbouring solutions for the initial solutions. In comparison, GA, NSGA-II, and C-TAEA can discover more and better non-dominated solutions. However, these four algorithms rely heavily on the initially generated feasible solutions in this highly restricted situation. However, these human-designed algorithms cannot guarantee the quality of solutions and these obtained feasible solutions are neither near to optimum nor represents true PFs distribution. As a result, despite the existence of mechanisms to provide randomness, such as the mutation process, these algorithms are unable to successfully escape the local minimum and reach feasible solutions.

As shown in the four graphs in Figure 6.3, two DRL-based algorithms can successfully overcome the limitations of human-designed algorithms to find solutions that are significantly closer to the global optimum and closer to true PFs. Especially in the congestion area, it can find solutions that vastly minimise traffic congestion. Although there are 100 weights set in the *Linear-MODRL Algorithm*, it can only discover a few non-dominated solutions and fails to find some solutions in the non-convex areas. In most cases, the solutions discovered are identical to those found with (0, 1) and (1, 0) weight vectors, demonstrating that the mapping from weight space to objective space is not perfectly isomorphic.

Undoubtedly, the proposed *Chebyshev-MODRL Algorithm* outperforms all other algorithms. When compared to the first four algorithms, it approximates the optimal values very well. It can discover more non-dominated solutions than the *Linear-MODRL Algorithm* because it is not limited by the setting weights and capable of solving problems in non-convex areas. In Figure 6.3 (c) and (d), we can find that the *Linear-MODRL Algorithm* has a lot of solutions (blue squares) that are located close to each other at the left bottom, which makes it difficult to represent non-dominated fronts well. Nevertheless, the solutions of the *Chebyshev-MODRL Algorithm* spread widely. In further, all solutions that are closer to minimum total E2E latency, minimum

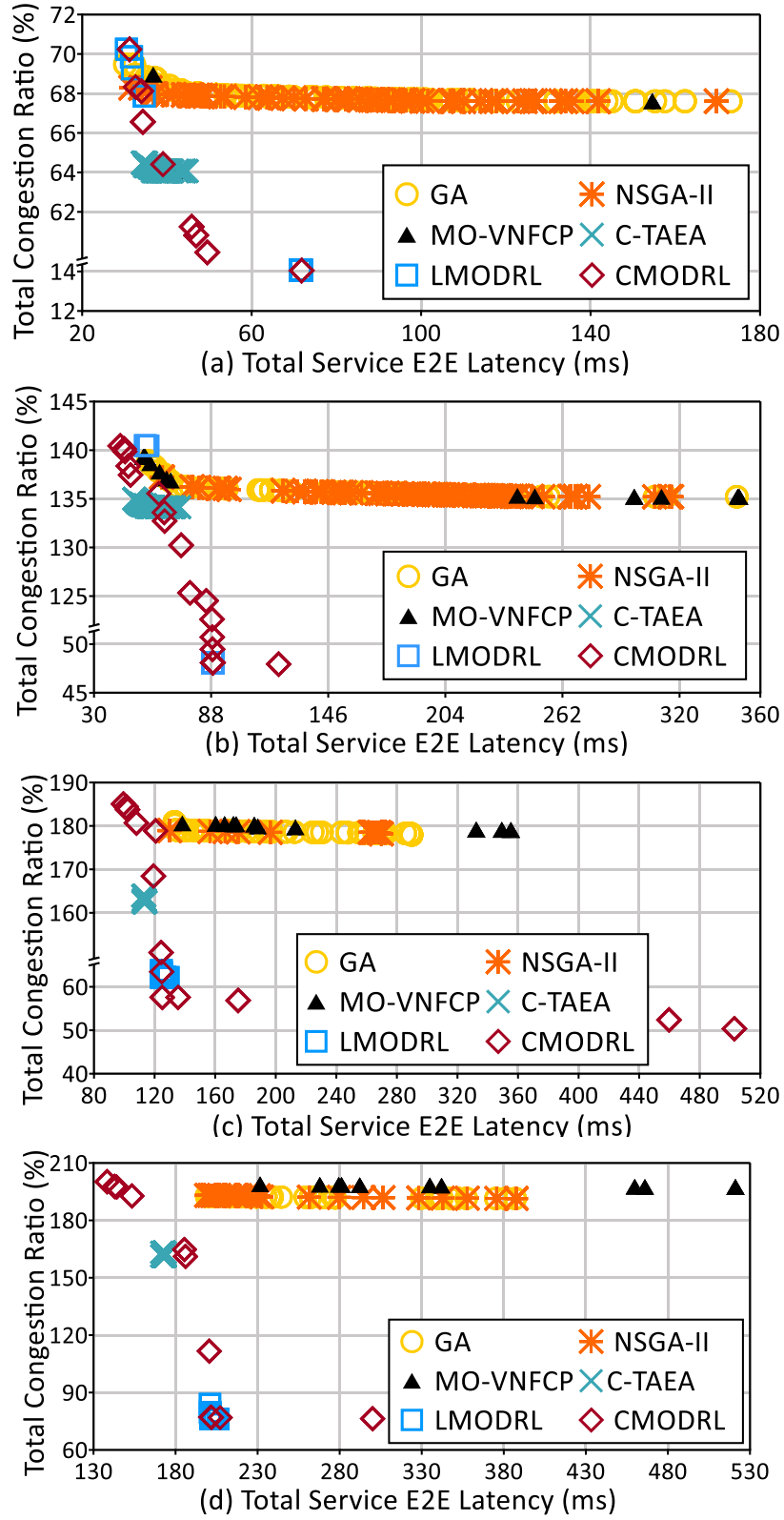


FIGURE 6.2. Non-dominated Solutions on Small-Scale Network: (a) 100 Service Requests Situation, (b) 200 Service Requests Situation, (c) 300 Service Requests Situation, (d) 400 Service Requests Situation.

total congestion, and utopia point [146] can be found, which can better reflect PFs.

Hypervolume Indicator (HV) is used to evaluate the performance of six algorithms by measuring the area dominated by non-dominated fronts [36]. However, this volume is infinite for a minimisation problem. Hence, we use a reference point with the maximum objective values generated by all six algorithms as its coordination. Under such circumstance, higher HV values indicate better performance [36]. From Table 6.1, it can also be deduced that two *MODRL*-based algorithms beat the other benchmarks, with the *Chebyshev-MODRL Algorithm* providing the highest HV values in most scenarios.

Table 6.1: Algorithm Performance Indicator

Total	Small Scale Network					
Service	GA	NSGA-II	MO-VNFCP	C-TAEA	LMODRL	CMODRL
100	0.02772	0.02865	0.02396	0.04499	0.56851	0.57269
200	0.03223	0.03211	0.03179	0.04180	0.68814	0.69581
300	0.13733	0.13686	0.11527	0.20688	0.78492	0.79181
400	0.25591	0.25567	0.22765	0.35321	0.68230	0.68270
Total	Large Scale Network					
Service	GA	NSGA-II	MO-VNFCP	C-TAEA	LMODRL	CMODRL
300	0.05827	0.07086	0.01609	0.28949	0.67689	0.67245
600	0.02599	0.02762	0.0188	0.22229	0.66945	0.67881
900	0.04245	0.06004	0.02074	0.17774	0.67818	0.71030
1200	0.10380	0.08265	0.07422	0.17014	0.61477	0.64358

6.4.2.2 Non-dominated Fronts on Large Scale Networks

Figure 6.3 shows the non-dominated fronts achieved for placing 300, 600, 900, and 1200 service requests on the 29-node network using multi-objective SFC algorithm. Six algorithms perform similarly on the large-scale network to those on the small-scale network. In all cases, the *MO-VNFCP Algorithm* performs worst with minimal HV values (as indicated in Table 6.1). Although the GA and NSGA-II improve the performance, they are unable to effectively eliminate congestion. *C-TAEA Algorithm* can release congestion by providing more solutions with a lower congestion ratio because it involves the mechanism to explore new areas. Two DRL algorithms significantly enhance the performance with considerably higher HV values (Table 6.1). *Linear-MODRL Algorithm* can achieve better non-dominated fronts in terms of both solution numbers and spread performance in large-scale networks than in small-scale networks. In most cases, the *Chebyshev-MODRL Algorithm* is superior to the *Linear-MODRL Algorithm* because it further improves the non-dominated fronts and gives lower latency and lower congestion solutions.

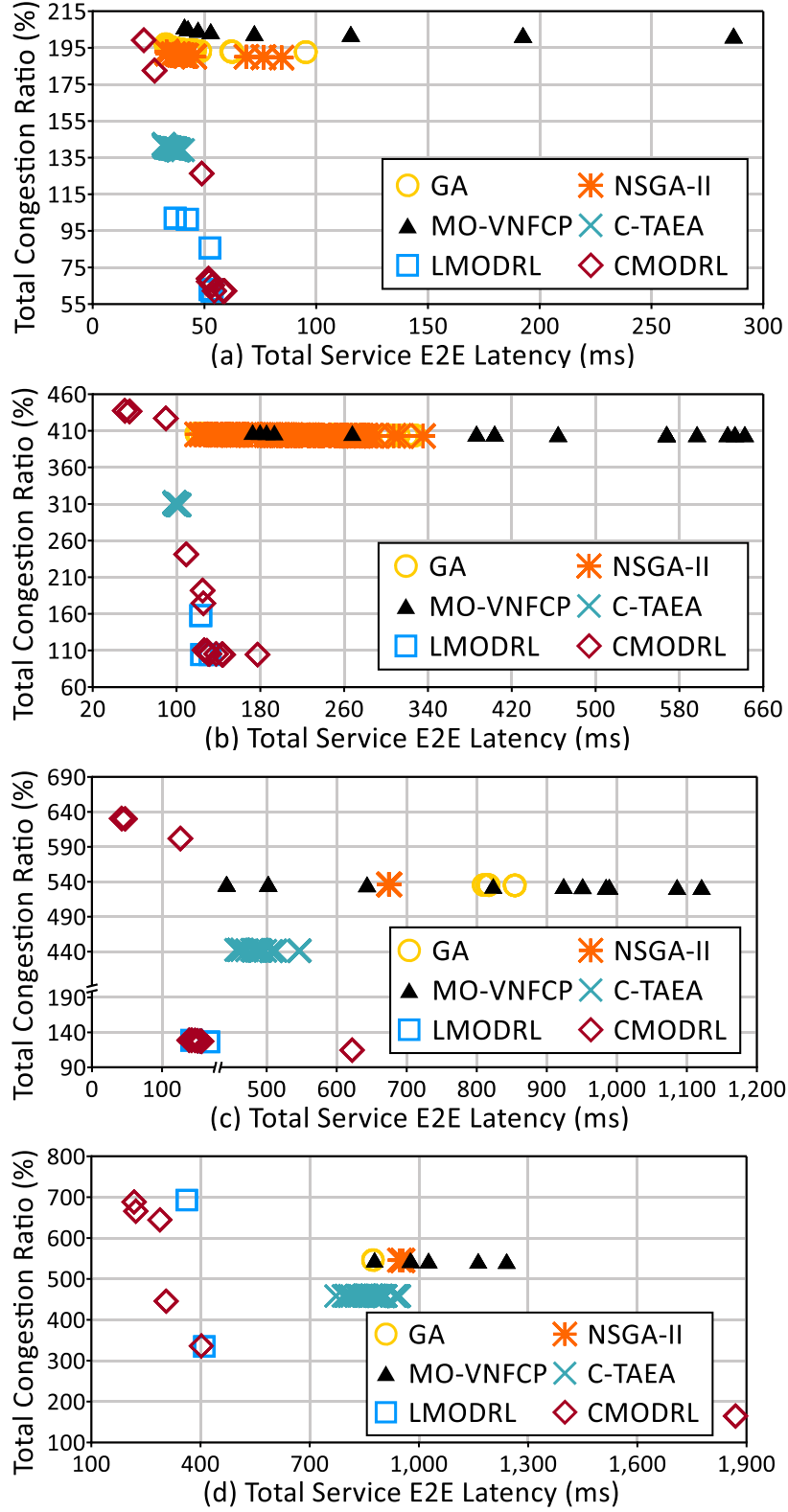


FIGURE 6.3. Non-dominated Solutions on Large-Scale Network: (a) 300 Service Requests Situation, (b) 600 Service Requests Situation, (c) 900 Service Requests Situation, (d) 1200 Service Requests Situation.

6.4.2.3 Time Consumption Performance

The running time for all algorithms on 7-node and 29-node architecture is shown in Table 6.2. In both scenarios, it can be seen that: 1) *GA* and *NSGA-II* take about the same amount of time to run 100 iterations, 2) *MO-VNFCP* consumes about a quarter of the consumption time that *GA* and *NSGA-II* use, but its performance is the worst (lowest HV values in Table 6.1), 3) although *C-TAEA* produces better non-dominated solutions than the *GA* and *NSGA-II*, it takes more than twice running time of *GA* and *NSGA-II*, 4) two DRL algorithms have the best performance, with training time accounting for just 2% of the total running time of *GA* and *NSGA-II*.

The execution time for each approach grows when the network scales from 7-node to 29-node and the number of service requests triples. On a large-scale network, two DRL algorithms cost less than seven times what they do on a small-scale network. However, *GA*, *NSGA-II*, *MO-VNFCP*, and *C-TAEA* algorithms take ten to fifty times longer than those on the small-scale network. For example, the *GA* and *NSGA-II* simulations take 9 days for 1200 service requests on the 29-node network, the *C-TAEA* simulation takes 19 days, while two DRL approach simulations take just 4 hours.

Based on the results of the simulation, it can be concluded that the proposed *Chebyshev-MODRL Algorithm* outperforms other state-of-the-art benchmarks in terms of the PF exploration capacity and scalability for solving large-scale problems.

Table 6.2: Time Consumption of Different Approach

Total	Small Scale Network					
Service	GA	NSGA-II	MO-VNFCP	C-TAEA	LMODRL	CMODRL
100	112min	153min	25min	182min	9.65min	8.23min
200	203min	210min	54min	357min	14.23min	13.70min
300	239min	234min	70min	526min	26.28min	26.8min
400	289min	312min	86min	611min	36.91min	35.95min
Total	Large Scale Network					
Service	GA	NSGA-II	MO-VNFCP	C-TAEA	LMODRL	CMODRL
300	1239min	1323min	247min	2955min	34.95min	36.45min
600	3551min	3211min	700min	6862min	123min	95min
900	8542min	8722min	1793min	18375min	139min	143min
1200	13672min	13692min	2755min	28219min	249min	245min

6.4.2.4 Weighted-sum Results on Large Scale Network

On the 29-node network, we run the 1) *First-Fit Algorithm*, which places SFCs to the nearest available nodes; the 2) *Data-Rate-based Algorithm*, which starts with the results of the first algorithm and then routes VNFs from MEC nodes to DCs when the MEC nodes are overloaded

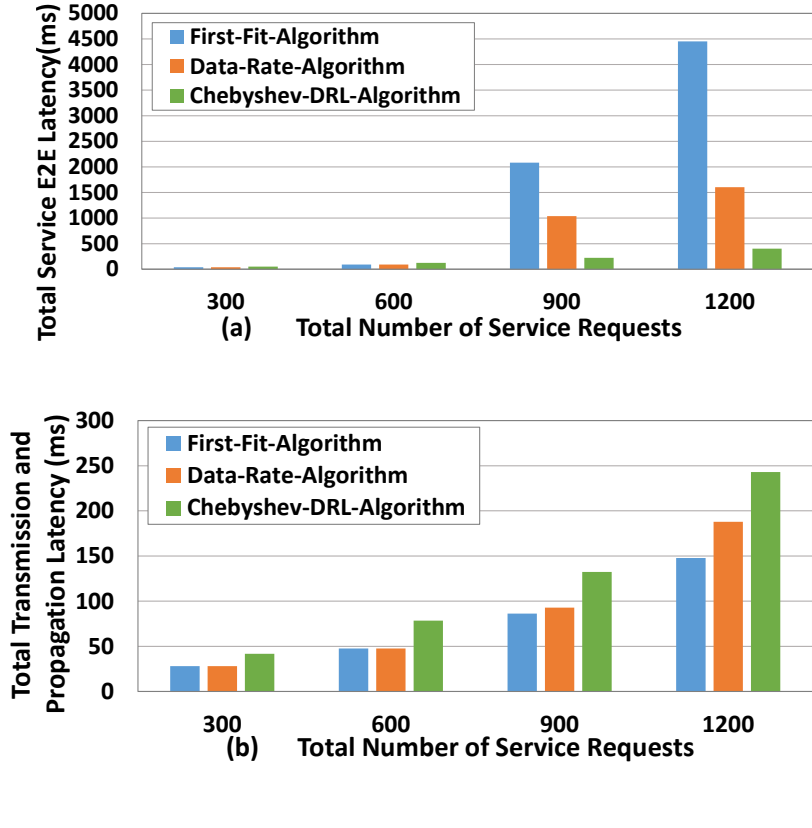


FIGURE 6.4. Simulation Results on 29-Node Network: (a) Average Total Service E2E Latency, (b) Average Total Transmission and Propagation Latency.

[131]; and the 3) *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* designed in this chapter (Algorithm 4). Under various workload scenarios, the chosen weights for Algorithm 4 are the weights of the points closest to the Utopia points [146], and 10 independent simulation experiments are carried out with different random seeds for the service request generation.

The average latency results of the three methods are illustrated in Figure 6.4 (a) and (b). It can be observed that the *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* has the lowest average total service E2E latency (green bar in Figure 6.4 (a)) and can route more services from MEC nodes to DCs, resulting in increased transmission and propagation latency (green bar in Figure 6.4 (b)). Because of such replacement, they can utilise more network resources than the first two algorithms. Furthermore, when there are more than 900 service requests, the overall service E2E latency for the *First-Fit Algorithm* grows dramatically owing to the massive increase in processing latency at MEC nodes when they are overloaded (more than 70% utilisation ratio shown in Figure 6.6 (d)).

Figure 6.5 are box plot latency results of 10 independent simulation runs. As the total number of service requests increases, three algorithms become less robust (Figure 6.5 (a)). Among them, the *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* performs the best with

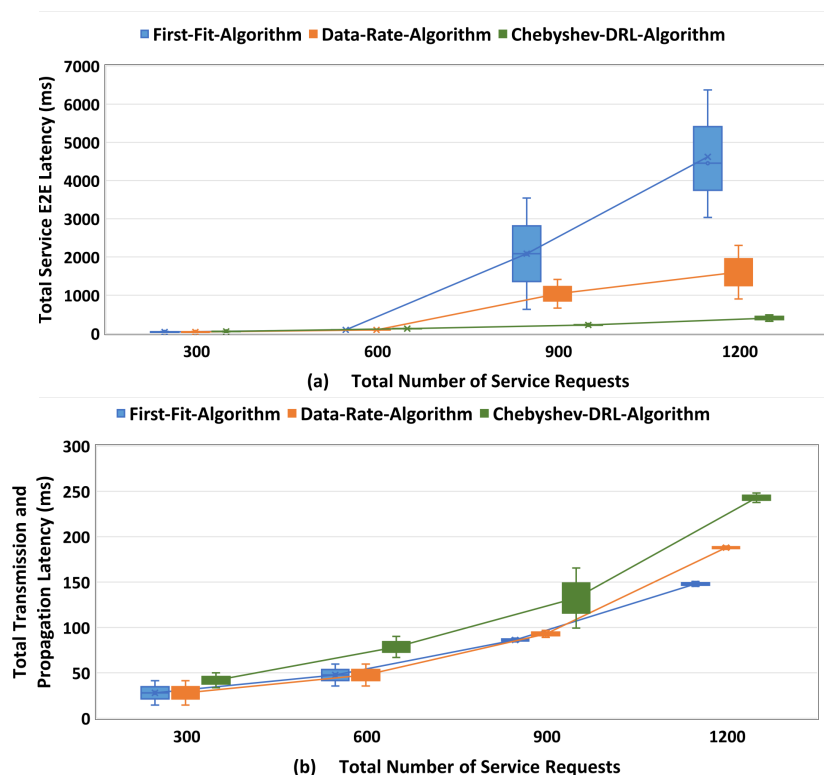


FIGURE 6.5. Statistical Simulation Results on 29-Node Network: (a) Total Service E2E Latency, (b) Total Transmission and Propagation Latency.

the narrowest box width, and the *First-Fit Algorithm* performs the worst with the broadest box width. For example, the *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* can efficiently control 50% latency variance within 52ms (from 392ms to 444ms), while, the *Data-Rate-based Algorithm* and the *First-Fit Algorithm* can only reach 702ms (from 1251ms to 1953ms) and 1669ms (from 3744ms to 5413ms) latency variance, respectively. The reason is that the DRL algorithm is more general than the heuristic algorithm and can lead to robust performance. In Figure 6.5 (b), the observed box plots show no specific trend for the tested three algorithms in terms of total transmission and propagation latency.

The CPU utilisation results of the three methods are shown in Figure 6.6 (c) and (d). The *First-Fit Algorithm* performs the poorest with the highest average congestion ratio. When the MEC nodes are overloaded, the *Data Rate-based Algorithm* can release the congestion by replacing VNFs from MEC nodes to EDCs, keeping the edge CPU utilisation ratio around 75%. The overall congestion performance and average edge CPU utilisation can be improved using the *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm*. On the one hand, when compared to the *First-Fit Algorithm* and the *Data Rate-based Algorithm*, it can successfully replace VNFs from MEC nodes to CDCs, reducing total congestion by 312% and 208% for 1200 service requests, respectively. On

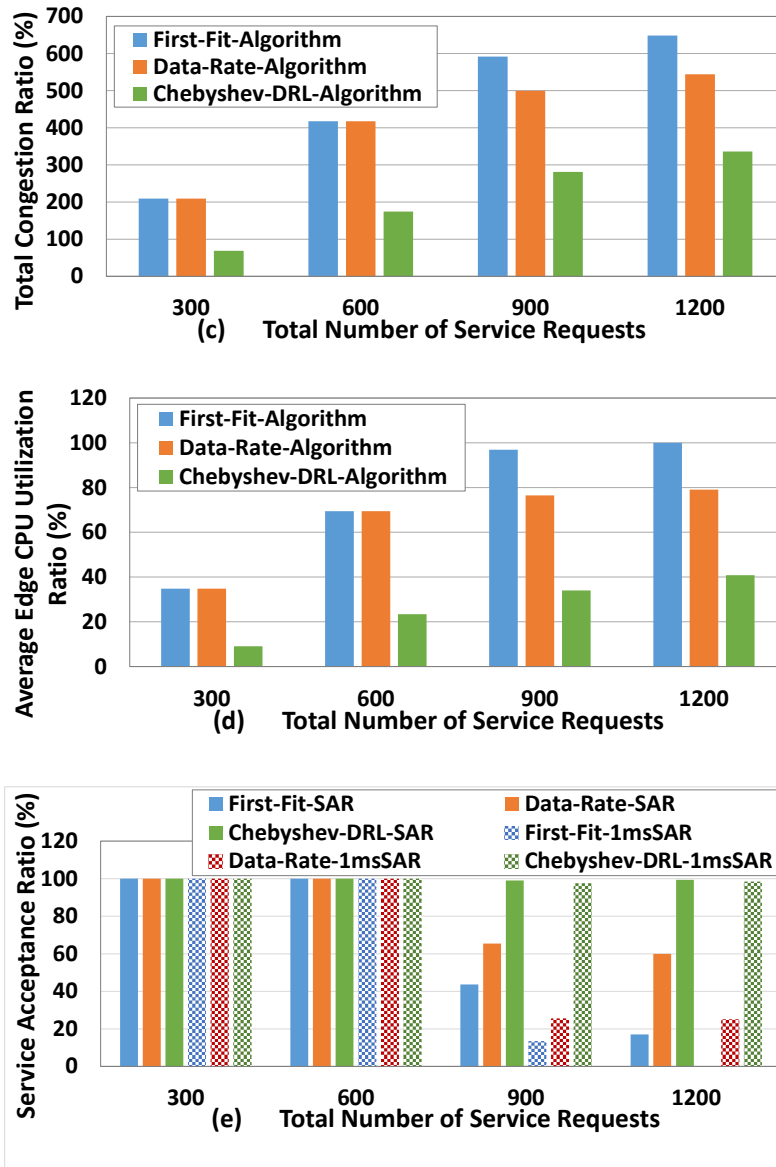


FIGURE 6.6. Simulation Results on 29-Node Network: (c) Average Total Congestion, (d) Average Edge CPU Utilisation Ratio, (e) Average SAR.

the other hand, it can keep the utilisation ratio of edge CPU below 41%, relieve the MEC nodes' burden, and keep enough edge resources for the future ultra-low latency services.

Figure 6.6 (e) shows the average SAR performance of three algorithms. When there are less than 900 service requests all three methods can reach 100% SAR. It can be seen that the *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* is the best for the rest of the time because it can effectively place VNFs at CDCs and accepts more than 99% service requests. Remarkably, the average SAR performance of 1ms services is compared among the three algorithms. When there are more than 900 service requests, more than 75% 1ms service requests will be rejected by the first two algorithms. Under the 1200 requests scenario, the *First-Fit Algorithm* performs significantly worse, and not one 1ms service can be accepted. It is worth mentioning that the *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* (Algorithm 4) can achieve 98% 1ms service requests under various workload scenarios because of its capability to intelligently support services with diverse QoS requirements.

Figure 6.7 (c), (d), and (e) depict the statistical performance of three algorithms with respect to total congestion, average edge CPU utilisation ratio, and SAR, individually. For the CPU utilisation, by comparing the box width of three algorithms with the number of service requests varying from 300 to 1200, it can be concluded that the proposed *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* outperforms the other two methods since it can get smaller distribution and lower mean value. It can limit the congestion variance and reduce congestion effectively under various workloads. Figure 6.7 (e) proves that the *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* is robust in comparison to other baseline approaches in terms of both overall service acceptance and ultra-low latency service acceptance performance. Because its SAR and 1ms SAR results have the smallest box width, the smallest data variance, and the highest mean value. To be more specific, its SAR variances and 1ms SAR variances are 0ms and its mean values are 100% for all the considered scenarios. Since the highest SAR value, the lowest SAR value, the mean SAR value, and the 50% SAR variance equal that of the 1ms SAR for the same algorithm, it indicates that all the rejected service requests are ultra-low latency service requests for all the considered scenarios.

Table 6.3 compares the time consumption performance of these three algorithms on the 29-node topology. The *First-Fit SFCs Placement Algorithm* takes the least amount of time but cannot guarantee the quality of solutions. Once trained, the *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* runs faster than the *Data-Rate-based SFCs Placement Algorithm* in the large-scale static network, requiring just 0.16-0.18 seconds to assign a single service request.

It is desired to adapt the designed algorithm for online simulation, taken into account the limitations of offline algorithms (e.g., not working for users' unpredictable behavior [147]). According to the time consumption results in Table 6.3, it can be expected that our proposed algorithm should be able to complete the resource allocation per request within one second. Furthermore, it satisfies the reported instantiation time for online single service provisioning

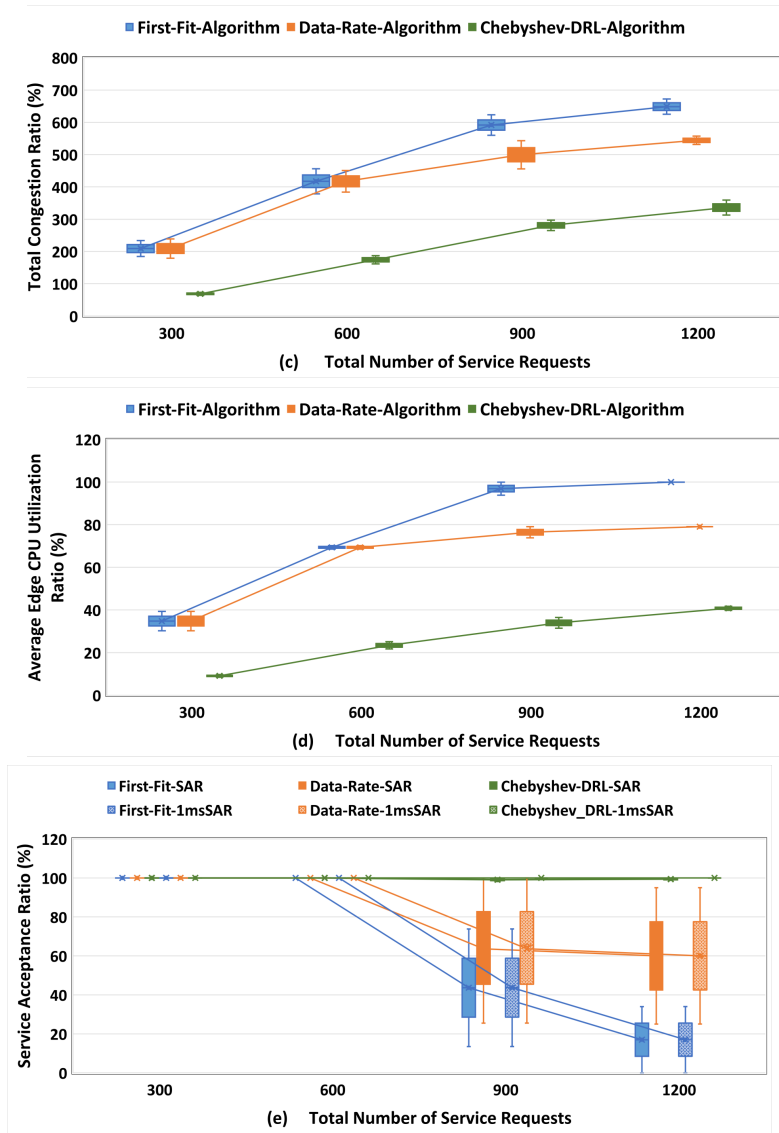


FIGURE 6.7. Statistical Simulation Results on 29-Node Network: (c) Total Congestion, (d) Average Edge CPU Utilisation Ratio, (e) SAR.

[148]. Based on Table 4 in [148], the time it takes to instantiate a service should be less than 120sec, and our proposed algorithm takes just 0.83% time of such target requirement. In the future, the fully live-adaption environment, which includes unseen service requests, can be modelled using the same constraints proposed in this chapter to train the *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm*.

Table 6.3: Running Time of Weighted-sum Approaches

Service Requests	300	600	900	1200
First-Fit Algorithm	2.97sec	5.67sec	7.27sec	11.24sec
Data-Rate Algorithm	332.91sec	1341.56sec	3461.92sec	7684.07sec
Chebyshev-DRL Algorithm	53.19sec	104.59sec	145.67sec	217.11sec

6.5 Testbed Experimentation-based Evaluation

6.5.1 Experiment Description

In addition to simulation evaluation, the experimental evaluation is carried out. The proposed OpenStack-based experiment testbed is established using the logical diagram of the optical layer depicted in Figure 6.8 (a). In this testbed, two CORSAIRONE PRO computers holding 8-core processors act as DCs, while two IBM x3455 servers holding 4-core processors as MEC nodes. Four computing nodes are connected to the Corsa DP2100 switch via 10 Gbps Ethernet cables on top of the optical fibres. At computing nodes and layer-2 switch, Intel multi-mode small form-factor pluggable (SFP) transceivers are used as the OEO conversion-related resources. Moreover, two optical fibres ranging 100km and 50km are included as well.

All of the NFVIs described (shown in Figure 6.8 (c) and (e)) are controlled by the OSM implemented on the EDC (DC1), which has less CPU resources after running OSM than the other computer. The other one is designed as the CDC (DC2). Figure 6.8 (b) shows how these computing nodes are controlled by the OSM controller, as well as their CPU capacity and current CPU utilisation. To reflect the transmission latency on 10km and 150km optical fibres between DC2 and switch, and MEC node and switch, respectively, extra latency of 0.73ms and 0.05ms are introduced at DC2 and MEC nodes, individually. We assign separate IP addresses to various SFCs' ports and build two different lightpaths (shown in Figure 6.8 (a) and (f)) for SFC from the same source and destination.

Firewall (FW), Deep Packet Inspection (DPI), VLC-media player (VLC), and Web Service (Web) are four types of VNF instances implemented at four VMs on each computing node. On MEC nodes, VMs have 2 CPU cores dedicated to hosting VLC and 1 CPU core dedicated to other functions. While, on DC, VMs are given with 4 CPU cores for VLC and 2 CPU cores for other functions. These instances are all controlled by the OSM. Examples of VNF instances on the

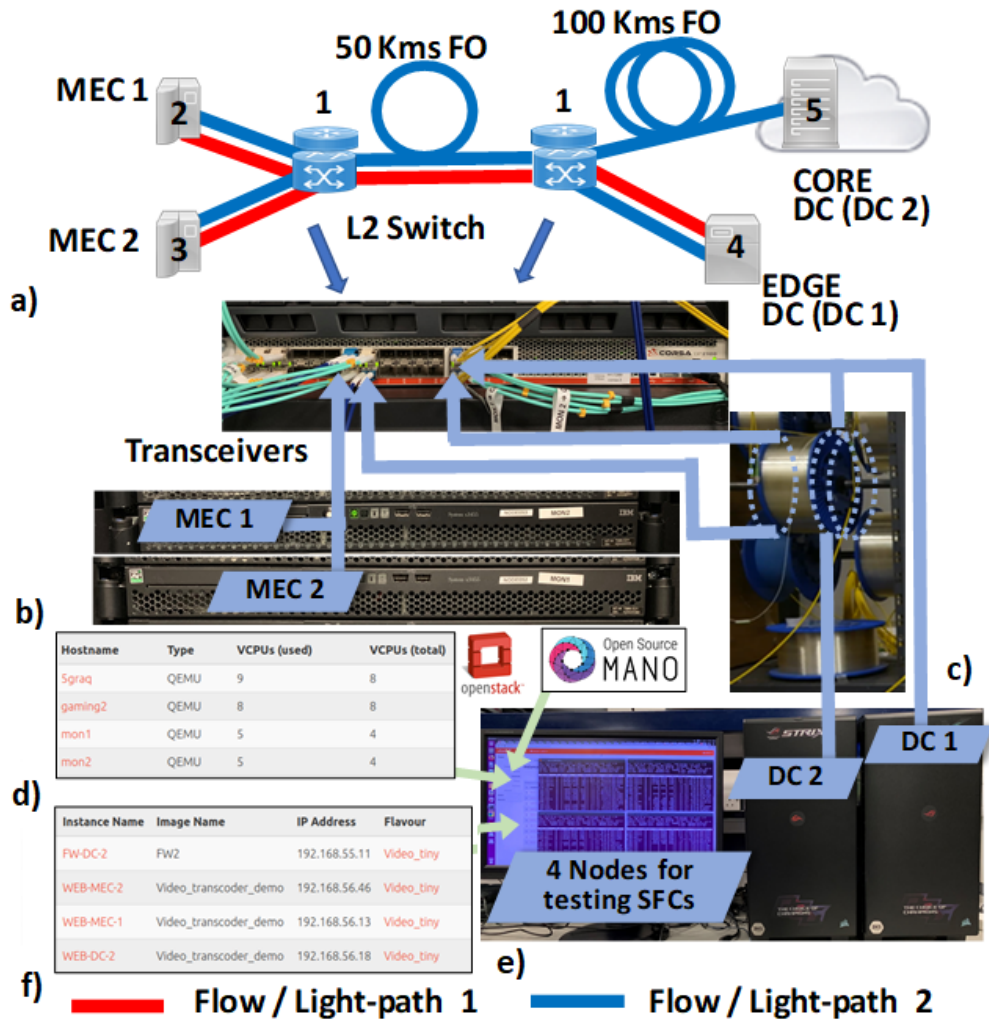


FIGURE 6.8. Experiment Testbed: a) Logical Diagram of Optical Layer, b) Control of Computing Nodes, c) NFVIs Part I (MEC nodes, Switches, and Optical Fibres), d) Control of VNF instances, e) NFVIs Part II (DC nodes and Monitors), f) Lightpath Labels.

OSM are shown in Figure 6.8 (d). In this diagram, one FW is instantiated on DC2, and three Web created on MEC nodes and DC. Each VNF instance has its own IP address for routing traffic through SFCs. In this experiment, we explore two types of service chains. One is the video streaming service requiring FW, DPI, and VLC function in order, 550ms E2E latency and 270 Kbyte packet for transmission. The other is the Web service requiring FW, DPI, and Web in order, 2s E2E latency and 64-byte packet for transmission.

At DC1 (controller), all the SFC placement algorithms are executed. First, the OSM gathers all of the VMs' current 'VCPUs(used)'. This information and SFC requests are taken as inputs to the algorithm. After getting input parameters, these algorithms are executed to produce node mapping and link mapping results. Then SFCs are created by NFVO according to these obtained results. To compare each node's CPU utilisation ratio and the service E2E latency between the simulation and experiment, these two indicators are calculated by SFC placement algorithms for the simulation, and recorded during the experiment as the real experimental outcomes.

6.5.2 Experiment Results and Analysis

Prior to the experiment, we conduct a pre-experiment test to gather information for the simulation environment settings. When there is no active service, the CPU utilisation ratio of each node is monitored. Accordingly, the CPU capacity is set to be 7.41, 7.88, 3.91, and 3.93 for DC1, DC2, MEC1, and MEC2. According to this pre-test, the maximum number of operating VLC instances is 5 and 2 for each DC and MEC node. We then run the maximum number of VLC instances and find that the CPU utilisation ratio is 36% and 64% for each MEC node and DC, respectively. Based on the results of this test, we set the VNF CPU resource consumption attribute β_m^{cpu} to be 0.003, 0.003, 0.003, and 0.006 for FW, DPI, Web, and VLC, respectively. For 8, 12, 16, and 20 service requests, we solve the multi-objective SFCs placement problem in the experiment. All SFC requests are evenly split between the FW-DPI-VLC video service and the FW-DPI-Web web service.

The simulation results and experimental results of two objectives obtained by three algorithms are presented in Figure 6.9 (a) and (b) as dotted line and solid line, respectively. It can be seen that they have comparable tendencies in Figure 6.4 (a) and (b). To be more specific, *First-Fit SFCs Placement Algorithm* performs the worst, having the highest total service E2E latency and total congestion. *Data Rate-based SFCs Placement Algorithm* and *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* performs better with lower total E2E latency and lower total congestion ratio. However, these two algorithms' total congestion results (solid green line and solid red line) are similar since the MEC CPU capacity and DC CPU capacity are comparable in the experiment environment.

In Figure 6.9 (c), the average edge CPU utilisation ratios of these methods are compared. Both for the simulation and experiment, the *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* always obtains the lowest average edge CPU utilisation ratios (dotted and solid red

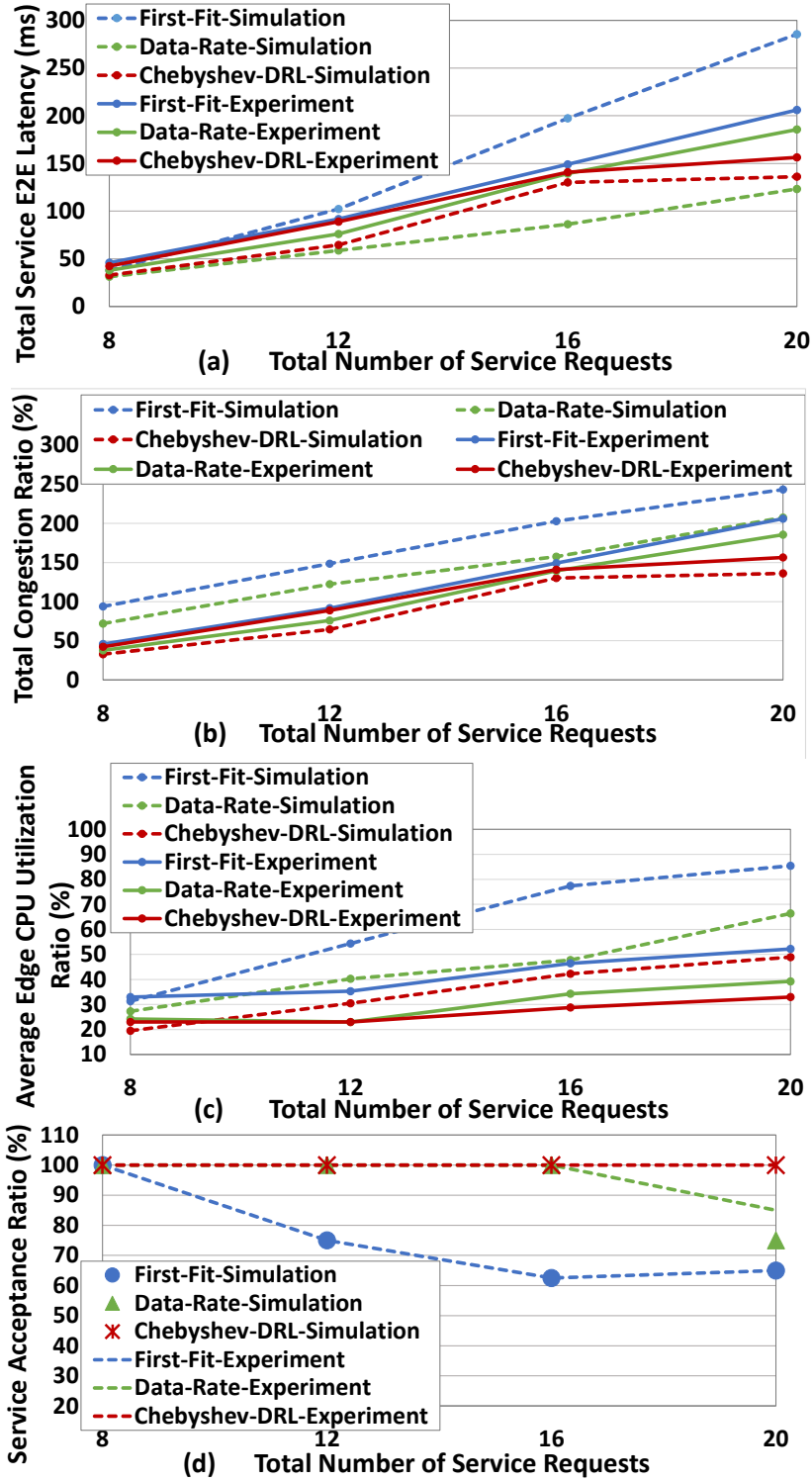


FIGURE 6.9. Simulation and Experiment Results on Real Testbed: (a) Total service E2E latency, (b) Total Congestion Ratio, (c) Average Edge CPU Utilisation Ratio, (d) SAR.

lines), demonstrating that it is the best algorithm for moving service from MEC nodes to DCs in a variety of workload circumstances. From this figure, it can also be found that the experiment CPU usage results are different from the simulation because the real CPU cores allocation differs from the M/M/1 queueing model adopted in the simulation.

Figure 6.9 (d) presents the SAR performance for simulation (dots) and experiment (line). The *First-Fit SFCs Placement Algorithm* (blue line) always accepts lower service requests than the *Data Rate-based SFCs Placement Algorithm* (green line) and also lower service requests than the *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* (red line). Because the first algorithm places a large number of services at MEC nodes for the high workload scenario, which is greater than the maximum number of supported services. Even though the second algorithm can improve such performance, some services are still be rejected in 20 SFC situations. Notably, the third algorithm consistently reaches 100% SAR. It can also be found that the simulation and experiment SAR of the second algorithm are different when there are 20 service requests. The reason for this is that different methods to SAR calculation are employed. In the simulation, when the calculated E2E latency exceeds the requirement or there is a CPU resource shortage, the video streaming service request is rejected. However, because the CPU utilisation ratio never reaches 100% in reality, the video streaming service request will not be refused in the experiment owing to the CPU resource shortage.

6.6 Summary

In this chapter, the SFCs placement problem in multi-layer edge-cloud network was handled using a multi-objective optimisation strategy to meet high capacity and low latency service requirements. Initially, a Pointer Network-based DRL model was designed. Following that, a novel *Chebyshev-assisted Actor-Critic SFCs Placement Algorithm* was proposed to achieve above 0.55 HV of PFs outperforming other state-of-the-art algorithms, and to deal with various workload scenarios achieving 99% total SAR and 98% 1ms SAR. Finally, experiments were carried out on a real testbed that consisted of multiple OpenStack-based NFVIs (e.g., MECs) hosted in servers linked by optical fibres. The results of simulation and real testbed experiments showed that the proposed algorithm can satisfy a wide range of QoS criteria and prevent congestion in different workload conditions. Notably, it achieved a 100% SAR in the tested scenario.

DISTRIBUTED GAME THEORY-BASED SFCs PLACEMENT

To overcome the disadvantages suffered from the centralised control of SFCs placement, such as scalability, privacy, and single point failure, a decentralised control method is proposed in this chapter. The latency-aware SFCs placement problem is first modelled as the congestion game to measure the effects of resource congestion on packet processing latency, optical-to-electrical (OE) conversion latency, and electrical-to-optical (EO) conversion latency. In this model, all SFC requests arriving simultaneously are players competing against each other to minimise their own E2E latency and resource consumption costs. Secondly, a distributed online algorithm is designed with the simulation results showing its 100% service acceptance performance in the simulation scenario. In the weighted-sum objective function, by setting lower resource consumption weights for ultra-low latency services, these services can be routed to edge nodes, and network operators can earn more. Thirdly, a real testbed experiment is carried out for SFCs with different packet sizes. Experimental results prove the online capability of the proposed algorithm as it converges to Nash Equilibrium in 40 seconds, and all the E2E latency criteria can be met if the packet size is small.

7.1 Introduction

Faced with the requirements of massive and various applications, the 5G or beyond 5G network is expected to provide high capacity and low latency services [149]. Because the combination of optical transmission and switching technologies can supply both high data rate and low latency solutions, the optical network is a significant enabler [150]. MEC is a promising enabler for ultra-low latency services as it brings computing resources from the core network to the edge network [151]. With virtualisation technologies, on-demand computing and network resources

can be offered on elastic optical networks for 5G services. For example, in SFCs, VNFs can be instantiated on any available resources, and virtual links can be mapped to any available physical links. By jointly considering optical network, MEC, and NFV technologies, flexible, efficient, and dynamic resource allocation can be achieved while catering to diverse application QoS requirements.

In this chapter, the latency-aware SFCs placement problem is studied by considering computing resources at MEC servers and DCs, network resources such as wavelength, optical-to-electronic (OE) conversion and electronic-to-optical (EO) conversion resources, and constraints in multi-layer networks. Regarding this problem, many works focus on the centralised methods including ILP, heuristic, meta-heuristic and RL algorithms [1, 8, 46, 51, 73, 152], however, these centralised approaches cannot address the requirements of fast, efficient, and scalable SFCs deployment in edge-cloud networks with a large number of MEC servers and the increasing number of services [38]. On the one hand, due to the NP-hardness of this problem, the ILP model is not scalable [21], heuristic algorithms cannot get high-quality solutions [153], meta-heuristic algorithms require a large iteration process to get good results [153], and RL algorithms which depend on tabular methods cannot handle high dimensional problems very well [83]. On the other hand, centralised methods also have disadvantages such as private information disclosure and the machine breakdown problem [68, 71]. Although some DRL approaches can handle the scalability problem, they suffer the disadvantages mentioned above.

To overcome the aforementioned drawbacks of the centralised approach, the distributed scheme is believed to be better for the latency-aware SFCs placement in edge-cloud networks [68]. In the distributed scheme, the proposed algorithm can run on many computing nodes based on local information. Therefore, *i*) scalability problem, *ii*) privacy problem, *iii*) single-point failure problem can all be addressed. Game theory is widely used in the development of distributed algorithms. There are several works using game theory for resource allocation in NFV-enabled networks. A graph partitioning game is used in [70] to assist the distributed algorithm design for VNF chaining in DCs. The expenses of processing and communication deployment are kept to a minimum with latency constraints. Authors in [69] propose a game theory-based heuristic algorithm for load balancing in VNF operation and routing. Authors in [71] solve the static SFC composition problem by modelling it as a congestion game and designing a distributed heuristic algorithm to minimise the congestion, latency, and cost all at the same time.

However, those methods proposed in the above works are not fit for solving the SFC placement problem in the multi-layer edge-cloud network because neither optical network nor edge computing resource is considered. Only one work presents a mixed-strategy gaming-based dynamic solution for the latency-aware SFCs placement problem in the elastic optical inter DC network taking into account computing, wavelength, and OEO conversion resources [154]. However, their model is not fully distributed but hierarchical. There is a central controller providing placement schemes for each service request. Hence, to the best of our knowledge, there is no distributed

online solution for the latency-aware SFCs placement problem that considers both optical network resources and edge computing resources.

To fill this gap, in this chapter, we design a game model and propose distributed online algorithm for latency-aware SFCs placement in multi-layer edge-cloud networks. The main contributions can be summarised below as:

1) The SFCs placement problem is modelled as a congestion game in which each service request behaves as a player attempting to reduce its own E2E latency and resource consumption costs. Due to the property of the congestion game, the processing latency, OE, and EO conversion latency are all affected by the number of users who share the same resource.

2) For each service request, a distributed online algorithm is developed to improve its own objective function and reach Pure strategy Nash Equilibrium (PNE). Its performance is tested in both discrete event simulation and real testbed experiments. By minimising both the latency and cost, 100% SAR can be achieved. The lower the weights set for network services, the higher the edge resources utilisation can be guaranteed and the higher network payoff can be achieved.

This chapter is structured as follows: Resource competition among service requests in 5G networks is introduced in Section 7.2. Section 7.3 formulates the SFCs placement problem as the congestion game and mathematically proves the existence of potential function and PNE. Section 7.4 provides the distributed online algorithm. Simulation setup, results, and analysis are described in Section 7.5. An experimental testbed is introduced, and experimental results are analysed in Section 7.6. The final section concludes the whole chapter.

7.2 Distributed SFCs Placement Problem Formulation

The network topology considered in this chapter is presented in Figure 7.1. In this topology, MEC servers are located at the Macro Base Station (MBS) in the 5G network [127]. While EDCs and CDCs are placed in the metro and core network, respectively. Compared to the EDCs and CDCs, MEC servers have fewer computing, OE, and EO conversion resources. All the computing nodes and switching nodes in the access, metro, and core network are connected via optical fibre links and OXCs, which means wavelength continuity should be satisfied [1].

The game-based latency-aware SFCs placement problem in this network topology can be defined as how to find appropriate computing nodes to instantiate the VNFs and appropriate optical fibre links to interconnect them while satisfying the E2E latency and resource capacity constraints of all the involved players [70]. Thus, each network service request is a selfish player in this game who strives to improve its own objective function while competing with other players for resources.

According to 3GPP, each network service request generated by the UE will be sent to UPF and third-party functions such as FW and DPI [151]. UPF is responsible for setting the data path between the UE and the data network and routing traffic towards the desired application

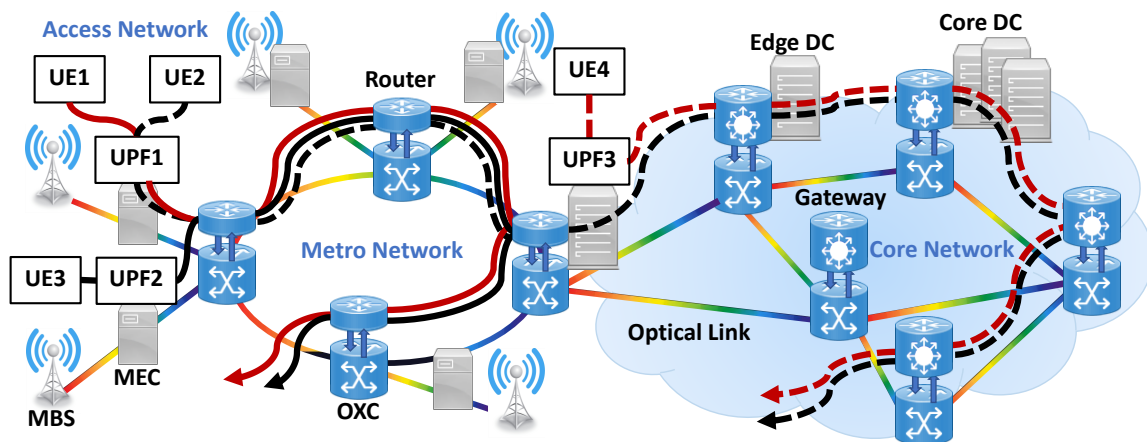


FIGURE 7.1. 5G Network Topology and Resource Competition.

based on network information [151]. Network services routed along the same path compete for the computing and network resources. Figure 7.1 also illustrates an example of such competition. There are four network service routing configurations. Network services requested by UE1, UE2, UE3 and UE4 shall be forwarded to their nearest available UPFs first [151]. Then, the UPF decides the resource allocation for each request assisted by algorithms and the traffic will be routed to the network accordingly. It can be seen that the network service from UE1 (red solid line) competes for the resources with network services from UE2 (black dotted line) in the core network and UE3 (black solid line) in the metro network. Instead of having UPF in MEC servers only, we also include UPF in EDC. The traffic from UE4 (red dotted line) is routed by UPF3 to the core network and this service competes with the traffic from UE2 (black dotted line) in the core network.

SFCs that request the same ordered VNFs but different sources, destinations, data rates, packet sizes, and E2E latencies are players in our game. From the source to the destination, computing resources, OE and EO conversion resources in various places can be chosen for these SFCs. Because processing latency, OE, and EO conversion latency are calculated using the M/M/1 queueing model, the higher the number of SFCs accessing the same resource, the worse the congestion is, the longer the latency will be. As a result, we should avoid using resources that have much traffic and congestion.

7.3 Game Model

In this section, firstly, the congestion game of SFCs is formulated with all the parameters related to the network, service requests, and the game model itself. Secondly, the formulated congestion game is proved to be a potential game holding both pure Nash Equilibrium (PNE) and Finite

Improvement Property (FIP) properties, which helps for the later distributed algorithm design.

7.3.1 Congestion Game Formulation

1) Input Parameters of network and SFC requests

The network and SFC requests are coded in Table. 7.1. The network is represented as a directed network (N, L) , where N is a set of nodes including MEC servers N_{MEC} , edge DCs N_{EDC} , core DCs N_{CDC} , and switching nodes N_{SWN} . L denotes a set of physical links. The physical link connecting physical node n and m can be represented as (n, m) , with length denoted by $len_{(n, m)}$. In the optical link system, W defines the set of wavelength.

For all the service requests, there are total O types of VNFs. o is used to define the o -th VNF in each SFC and the set of required VNFs in the same SFC F can be defined as $F = \{f_1, f_2, \dots, f_O\}$. We assume that each computing node has total O types of VNF instances with CPU resources $r_{o, n}^{CPU}$. Similarly, we use r_n^{OE} , r_n^{EO} and $r_{(n, m)}^w$ to indicate the OE conversion resource on node n , EO conversion resource on node n , and wavelength w on link (n, m) , respectively. Each kind of resource has its own unit price.

Table 7.1: Network Parameters and SFC Parameters

(N, L)	directed network including the set of nodes N and physical links L . N is composed by MEC nodes N_{MEC} , edge DCs N_{EDC} , core DCs N_{CDC} and switching nodes N_{SWN}
n	the node n in the network, $n \in N$
(n, m)	the physical link connecting node n and m , $(n, m) \in L$
$len_{(n, m)}$	the length of the physical link (n, m)
W	the set of wavelengths in the network
$r_{(n, m)}^w$	the wavelength w on the link (n, m) , $w \in W$
O	the total number of VNFs in the SFC
o	the o -th VNF in the SFC request, $o \in [1, O]_z^{1\ 2}$
F	the set of required VNFs in the same SFC, $F = \{f_1, f_2, \dots, f_O\}$
R	the set of resources in the network
$r_{o, n}^{CPU}, r_n^{OE}, r_n^{EO}$	the CPU on the o -th VNF in the node n , OE conversion and EO conversion resource in the node n , $r_{o, n}^{CPU}, r_n^{OE}, r_n^{EO} \in R$
v	the data rate of the network service
$\beta_o^{CPU}, \beta_o^{OE}, \beta_o^{EO}$	CPU, OE and EO conversion resources required by the o -th VNF. All the VNFs have the same β_o^{OE} and the same β_o^{EO} (e.g., $\beta_1^{OE} = \beta_2^{OE}$)
$\mu_{o, n}^{CPU}, \mu_n^{OE}, \mu_n^{EO}, \mu_{(n, m)}$	the unit price of using CPU, OE and EO resources on node n , and the unit price of using link (n, m)

2) Game Model Parameters

Table. 7.2 contains all the parameters defining the congestion game for service requests. The set of players S includes all the received SFC requests s at the same time. Each SFC is a player i in this game characterised by its source a^i , destination b^i , E2E latency τ^i , the set of VNFs F^i , data rate v^i , holding time T^i , and ρ^i indicating how the service request cares about resource consumption cost. The resource required by each player i depends on the resource type, the VNF type, and the traffic data rate, for example, the CPU required by the o -th VNF for the i -th player can be calculated by $\theta_o^{i,CPU} = \beta_o^{CPU} \cdot v^i$.

Each player has its own strategy ψ^i of choosing the CPU, OE, EO resource and wavelength. We use ψ to represent strategies for all the players and ψ^{-i} to represent strategies for all the other players.

Table 7.2: Game Model Parameters

S	the set of players including all the network service requests received at the same time
i	the player i in the game, $i \in S$
P	the total number of players in the game
$s(a^i, b^i, \tau^i, F^i, v^i, T^i, \rho^i)$	service request of the player i including source a^i , destination b^i , E2E latency τ^i , the set of VNFs F^i , data rate v^i , holding time T^i , and parameter ρ^i indicating how the service request cares about resource consumption cost
$\theta_o^{i,CPU}, \theta_o^{i,OE}, \theta_o^{i,EO}$	the CPU, OE and EO conversion resources required by the player i , $\theta_o^{i,CPU} = \beta_o^{CPU} \cdot v^i$, $\theta_o^{i,OE} = \beta_o^{OE} \cdot v^i$, $\theta_o^{i,EO} = \beta_o^{EO} \cdot v^i$
ψ^i	the strategy for the player i is the CPU, OE and EO conversion resources chosen by each VNFs
$\psi = (\psi^1, \dots, \psi^P)$	the strategy profile of all the players
$\psi^{-i} = \psi \setminus \psi^i$	the strategies of other players, $\psi^{-i} = (\psi^1, \dots, \psi^{i-1}, \psi^{i+1}, \dots, \psi^P)$
$\epsilon_o^{i,CPU}, \epsilon_o^{i,OE}, \epsilon_o^{i,EO}$	the CPU, OE and EO conversion resources on the node ϵ is chosen by the o -th VNF for the player i
$\epsilon_{(a^i,1)}^i, \epsilon_{(o,o+1)}^i, \epsilon_{(O,b^i)}^i$	the physical link ϵ is chosen by the player i for the traffic from the source node a^i to the first VNF, between VNFs or from the last VNF to the destination node b^i
$\alpha_{o,n}^{CPU}, \alpha_n^{OE}, \alpha_n^{EO}$	the number of players choosing the o -th VNF CPU, the OE and EO conversion resource on node n

3) Latency Parameters

The packet processing latency dp within each VNF instance, OE conversion latency do and EO conversion latency de at each computing nodes experienced by the player i in the chosen $r_{o,n}^{CPU}$, r_n^{OE} and r_n^{EO} resource can be calculated by (7.1)-(7.3), respectively. As shown in these equations, these latency are also influenced by other players' strategies.

$$(7.1) \quad dp_{r_{o,n}^{CPU}}^{(\psi^i, \psi^{-i})} = 1/(r_{o,n}^{CPU} - \beta_o^{CPU} \cdot \sum_{j \in \alpha_{o,n}^{CPU}(\psi^i, \psi^{-i})} v)$$

$$(7.2) \quad do_{r_n^{OE}}^{(\psi^i, \Psi^{-i})} = 1/(r_n^{OE} - \beta_o^{OE}) \cdot \sum_{j \in \alpha_n^{OE}(\psi^i, \Psi^{-i})} v$$

$$(7.3) \quad de_{r_n^{EO}}^{(\psi^i, \Psi^{-i})} = 1/(r_n^{EO} - \beta_o^{EO}) \cdot \sum_{j \in \alpha_n^{EO}(\psi^i, \Psi^{-i})} v$$

The propagation latency dg for the player i on the chosen link (n, m) can be calculated by (7.4), which includes the propagation latency for the traffic between VNFs $(o, o+1)$, between the source node and the first VNF $(a^i, \epsilon_1^{i, CPU})$, and between the last VNF to the destination node $(\epsilon_o^{i, CPU}, b^i)$.

$$(7.4) \quad dg_{(n,m)}^{(o,o+1)} = len_{(n,m)}/v, \quad dg_{(n,m)}^{(a^i, \epsilon_1^{i, CPU})} = len_{(n,m)}/v, \quad dg_{(n,m)}^{(\epsilon_o^{i, CPU}, b^i)} = len_{(n,m)}/v$$

Equation (7.5)-(7.8) are processing latency DP^{ψ^i} , OE conversion latency DO^{ψ^i} , EO conversion latency DE^{ψ^i} , and propagation latency DG^{ψ^i} experienced by the player i , respectively.

$$(7.5) \quad DP^{(\psi^i, \Psi^{-i})} = \sum_{r_{o,n}^{CPU} \in \psi^i} dp_{r_{o,n}^{CPU}}^{(\psi^i, \Psi^{-i})}$$

$$(7.6) \quad DO^{(\psi^i, \Psi^{-i})} = \sum_{r_n^{OE} \in \psi^i} do_{r_n^{OE}}^{(\psi^i, \Psi^{-i})}$$

$$(7.7) \quad DE^{(\psi^i, \Psi^{-i})} = \sum_{r_n^{EO} \in \psi^i} de_{r_n^{EO}}^{(\psi^i, \Psi^{-i})}$$

$$(7.8) \quad DG^{\psi^i} = dg_{(n,m)}^{(a^i, \epsilon_1^{i, CPU})} + \sum_{o \in [1, |O|]_z} dg_{(n,m)}^{(o,o+1)} + dg_{(n,m)}^{(\epsilon_o^{i, CPU}, b^i)}$$

4) Cost Parameters

In this model, the cost for using CPU resource CP^{ψ^i} , OE conversion resource CO^{ψ^i} , EO conversion resource CE^{ψ^i} , and link resources CL^{ψ^i} are considered. Equation (7.9)-(7.12) calculate these latency for each player i , individually.

$$(7.9) \quad CP^{\psi^i} = \sum_{o \in [1, |O|]_z} \mu_{o, \epsilon_o^{i, CPU}}^{CPU} \cdot \theta_o^{i, CPU}$$

$$(7.10) \quad CO^{\psi^i} = \sum_{o \in [1, |O|]_z} \mu_{\epsilon_o^{i, OE}}^{OE} \cdot \theta_o^{i, OE}$$

$$(7.11) \quad CE^{\psi^i} = \sum_{o \in [1, |O|]_z} \mu_{\epsilon_o^{i, EO}}^{EO} \cdot \theta_o^{i, EO}$$

$$(7.12) \quad CL^{\psi^i} = v \cdot [\mu_{(a^i, \epsilon_1^{i, CPU})} + \sum_{o \in [1, |O|]_z} \mu_{(\epsilon_o^{i, CPU}, \epsilon_{o+1}^{i, CPU})} + \mu_{(\epsilon_o^{i, CPU}, b^i)}]$$

5) Objective Function

Equation (7.13) is the weighted-sum objective function for each player i minimising both E2E latency and resource consumption cost. Various ρ weights are assigned to services with different latency needs.

$$(7.13) \quad C(\psi^i, \Psi^{-i}) = DP^{(\psi^i, \Psi^{-i})} + DO^{(\psi^i, \Psi^{-i})} + DE^{(\psi^i, \Psi^{-i})} + DG^{\psi^i} + \rho^i \cdot (CP^{\psi^i} + CO^{\psi^i} + CE^{\psi^i} + CL^{\psi^i})$$

6) Resource Constraints

The VNF can only be mapped on the node with sufficient computing, OE and EO conversion resource (equation (7.14)-(7.16)). The virtual link can also only be mapped to the physical link which has enough bandwidth resources (equation (7.17)).

$$(7.14) \quad r_{o,n}^{CPU} - \beta_o^{CPU} \cdot \sum_{j \in \alpha_{o,n}^{CPU}(\psi^i, \Psi^{-i})} v > 0, \quad \forall n \in N, o \in [1, |O|]_z$$

$$(7.15) \quad r_n^{OE} - \beta_o^{OE} \cdot \sum_{j \in \alpha_n^{OE}(\psi^i, \Psi^{-i})} v > 0, \quad \forall n \in N$$

$$(7.16) \quad r_n^{EO} - \beta_o^{EO} \cdot \sum_{j \in \alpha_n^{EO}(\psi^i, \Psi^{-i})} v > 0, \quad \forall n \in N$$

$$(7.17) \quad r_{(n,m)}^w - \sum_{j \in \alpha_{(n,m)}^{i,w}(\psi^i, \Psi^{-i})} v > 0, \quad \forall (n, m) \in L$$

7.3.2 Proof of Potential Game

The congestion game can be specified as $G = (S, R, \psi^i, (C^i)_{i \in [1, |P|]_z})$ with all parameters and constraints given above, including the set of players, the resources, the strategies, and the payoff functions associated with resource competition status.

Definition 1: a strategy profile Ψ_\star^i is a Nash Equilibrium (NE) if, $\forall i \in [1, |P|]_z$, we have: $C^i(\psi_\star^i, \Psi_\star^{-i}) \leq C^i(\psi^i, \Psi_\star^{-i})$

Definition 2: for any congestion game, there is a potential function $\Phi(\psi^i, \Psi^{-i})$ which satisfies: $\Phi(\psi^i, \Psi^{-i}) - \Phi(\psi^{i'}, \Psi^{-i}) = C(\psi^i, \Psi^{-i}) - C(\psi^{i'}, \Psi^{-i})$

Proposition: In this model the potential function is:

$$(7.18) \quad \begin{aligned} \Phi(\psi^i, \Psi^{-i}) = & \sum_{i \in [1, |P|]_z} \hat{c}(\psi^i) + \sum_{r_{o,n}^{CPU} \in R} \sum_{j=1}^{\alpha_{o,n}^{CPU}(\psi^i, \Psi^{-i})} \frac{1}{r_{o,n}^{CPU} - \beta_o^{CPU} \cdot j \cdot v} \\ & + \sum_{r_n^{OE} \in R} \sum_{j=1}^{\alpha_n^{OE}(\psi^i, \Psi^{-i})} \frac{1}{r_n^{OE} - \beta_o^{OE} \cdot j \cdot v} + \sum_{r_n^{EO} \in R} \sum_{j=1}^{\alpha_n^{EO}(\psi^i, \Psi^{-i})} \frac{1}{r_n^{EO} - \beta_o^{EO} \cdot j \cdot v} \end{aligned}$$

where $\hat{c}(\psi^i) = DG^{\psi^i} + \rho^i \cdot (CP^{\psi^i} + CO^{\psi^i} + CE^{\psi^i} + CL^{\psi^i})$. The potential function is a real value function that tracks the changes in each player's objective function when the player unilaterally modifies its strategy [71].

In the following, the designed congestion game is proved to be a finite ordinal potential game admitting a potential function and PNE.

Proof:

$$\begin{aligned}
(7.19) \quad & \Phi(x, \boldsymbol{\psi}^{-k}) - \Phi(y, \boldsymbol{\psi}^{-k}) = \hat{c}^k(x) - \hat{c}^k(y) + \sum_{r^h \in R} \sum_{j=1}^{\alpha^h(x, \boldsymbol{\psi}^{-k})} \frac{1}{r^h - \beta^h \cdot j \cdot v} - \sum_{r^h \in R} \sum_{j=1}^{\alpha^h(y, \boldsymbol{\psi}^{-k})} \frac{1}{r^h - \beta^h \cdot j \cdot v} \\
& = \hat{c}^k(x) - \hat{c}^k(y) + \sum_{r^h \in R} \left(\sum_{j=1}^{\alpha^h(\boldsymbol{\psi}^{-k})} \frac{1}{r^h - \beta^h \cdot j \cdot v} + A_{r^h \in (x)} \cdot \frac{1}{r^h - \beta^h \cdot [\sum_{j \in \alpha^h(\boldsymbol{\psi}^{-k})} v + v]} \right) \\
& \quad - \sum_{r^h \in R} \left(\sum_{j=1}^{\alpha^h(\boldsymbol{\psi}^{-k})} \frac{1}{r^h - \beta^h \cdot j \cdot v} + A_{r^h \in (y)} \cdot \frac{1}{r^h - \beta^h \cdot [\sum_{j \in \alpha^h(\boldsymbol{\psi}^{-k})} v + v]} \right) \\
& = \hat{c}^k(x) - \hat{c}^k(y) + \sum_{r^h \in (x)} \left(A_{r^h \in (x)} \cdot \frac{1}{r^h - \beta^h \cdot [\sum_{j \in \alpha^h(\boldsymbol{\psi}^{-k})} v + v]} - A_{r^h \in (y)} \cdot \frac{1}{r^h - \beta^h \cdot [\sum_{j \in \alpha^h(\boldsymbol{\psi}^{-k})} v + v]} \right) \\
& = \hat{c}^k(x) - \hat{c}^k(y) + \sum_{r^h \in x} \frac{1}{r^h - \beta^h \cdot \sum_{j \in \alpha^h(x, \boldsymbol{\psi}^{-k})} v} - \sum_{r^h \in y} \frac{1}{r^h - \beta^h \cdot \sum_{j \in \alpha^h(y, \boldsymbol{\psi}^{-k})} v} = C^k(x, \boldsymbol{\psi}^{-k}) - C^k(y, \boldsymbol{\psi}^{-k})
\end{aligned}$$

where r^h represents $r_{o,n}^{CPU}$, r_n^{OE} or r_n^{EO} , and β^h represents β_o^{CPU} , β_o^{OE} or β_o^{EO} in a generic way. $A_{r^h} = 1$ if this resource is used by the strategy, otherwise, it is 0.

According to the *COROLLARY 2.2* in [155], every finite ordinal potential game possess a pure-strategy NE.

7.4 Distributed Algorithm

In the previous section, the proposed congestion game model for resource competition among service requests is mathematically proved to be a finite ordinal potential game. Using the Finite Improvement Property (FIP) of such finite ordinal potential game [155], we can construct a distributed algorithm for all players. This appealing property permits the unilateral improvement path that starts from any initial strategy and then converges to the NE in a finite number of iterations [71]. The improvement path is the sequence of unilateral strategy modifications made by each player to get better results in the game model. While, in this SFCs placement problem, the improvement path means adjusting VNFs mapping and virtual links mapping for each SFC request in the edge-cloud networks.

Algorithm 5 shows our designed distributed algorithm for the online SFCs placement. The network parameters involving network topology and resource capacities and service request parameters including source, destination, data rate, ordered VNFs, etc are input parameters for this algorithm. The main steps are 1) At the beginning, for all the service requests received at the same time, the same number of actors [156] will be created to process each of these requests (line 3). 2) For each actor, after receiving the necessary information, such as the service request associated with its requirements and the current network status like resource utilisation, it will

Algorithm 5: Distributed Algorithm

```

1 Input: Network Information: (N,L), R,  $\beta_o^{CPU}$ ,  $\beta_o^{OE}$ ,  $\beta_o^{EO}$ ,  $P$  number of Service Requests:
    $s(a^i, b^i, \tau^i, F^i, v^i, T^i, \rho^i)$ 
2 Output: Resource Allocation Solutions for all the Service Requests:  $\epsilon_o^{i,CPU}$ ,  $\epsilon_o^{i,OE}$ ,  $\epsilon_o^{i,EO}$ ,
    $\epsilon_{(a^i,1)}^i$ ,  $\epsilon_{(o,o+1)}^i$ ,  $\epsilon_{(O,b^i)}^i$ 
3 Create  $P$  number of actors;
4 Send the  $p$  – th service request and network information to the  $p$  – th actor;
5  $iteration = 1$ ;
6  $\star$  In each actor run the following algorithm  $\star \backslash$ 
7 while The algorithm is not converged do
8   if  $iteration \neq 1$  then
9     Receive all the resource allocation solutions from other SFCs of the previous
       iteration;
10    Update the network status;
11  end if
12  for all the resources in the network do
13    Calculate the objective function;
14    if objective function is the minimum then
15      return  $\epsilon_o^{i,CPU}$ ,  $\epsilon_o^{i,OE}$ ,  $\epsilon_o^{i,EO}$ ,  $\epsilon_{(a^i,1)}^i$ ,  $\epsilon_{(o,o+1)}^i$ ,  $\epsilon_{(O,b^i)}^i$ 
16    end if
17  end for
18  Send the minimum resource allocation solution to other actors;
19  Calculate the potential function;
20   $iteration = iteration + 1$ 
21 end while
    
```

find the initial placement solution for the service request at the first iteration. All the required VNFs are deployed at random nodes with sufficient resources on the shortest path from source to destination in this solution. Such shortest path is found by the Dijkstra algorithm while considering the transmission capacity and wavelength continuity. 3) Then, the initial solution found by each actor will be broadcast to all the other actors. 4) In the following iterations, the actor will update the network status based on all the received solutions before determining the optimal solution leading to the player's minimum objective function. 5) The actor will then send all the other actors the best solution in this iteration. 6) After each iteration, the potential function will be calculated (line 19). To decide whether the algorithm is converged or not, the value difference of potential function between two neighbouring iterations will be compared. If the algorithm does not converge, all the actors will repeat the steps from 4) to 6) until it does. When the algorithm converges, the final placement solution will be used to route each network service.

7.5 Simulation-based Evaluation

7.5.1 Simulation Setup

Figure 7.1 shows the 17-node network topology considered for the simulation. In this topology, computing nodes featuring different geographical locations are classified into MEC, EDC, and CDC nodes. The CPU, OE and EO conversion resource capacities are different on different nodes. Table. 7.3 summarizes these resource capacities on different nodes. Each MEC node is equipped with 512 CPU cores [1]. While, each EDC and CDC is equipped with 2560 and 5120 CPU cores, being 5 and 10 times of CPU cores in the MEC node, respectively [55]. The EO and OE conversion resources are set according to [154]. Table. 7.3 also lists the price for each type of resource on different nodes. In our setting, we set a higher price for resources at MEC servers compared to that on EDC and CDC nodes because their capacities are limited in MEC nodes and they can provide better QoS performance. Apart from MEC, EDC, and CDC nodes, others are switching nodes with no computing capability.

Table 7.3: Node Resource Capacity and Price

Type	VNF1 CPU	VNF2 CPU	VNF3 CPU	OE or EO	CPU Price	OE or EO Price
MEC	128	128	256	6400	3	1.5
EDC	640	640	1280	32000	2	1
CDC	1280	1280	2560	64000	1	0.5

In this topology, all the nodes are connected via Wavelength Division Multiplexing (WDM) fibre links. Each link has 4 wavelengths with a capacity of 25 Gbps per wavelength. The price for each link is 1. The length of each link in the access, metro, and core network is set to be 10km, 25km, and 100km, respectively.

All network services require a data rate of 100 Mbps and three VNFs (i.e. FW, DPI, and VLC Media Player (VLC)) in order. For these three VNFs, β_o^{CPU} is set to be 0.01, 0.01, and 0.02 accordingly. All of the VNFs have β_o^{OE} and β_o^{EO} of 0.1. Each service's E2E latency is generated from 1ms, 5ms, 80ms, 100ms, and 500ms randomly [1]. The source and destination for 1ms services are the same MEC server or EDC, while they are randomly chosen for the other services. Each service's holding time is determined by uniform distribution in the range of (300s and 900s). In the designed congestion game, players fighting for resources are 10 service requests arriving at the same time. In the simulations, we use Simpy [157] as the discrete event generator to produce these service requests in Poisson distribution with $\lambda = 0.1$.

7.5.2 Simulation Results and Analysis

To evaluate the performance of our proposed distributed algorithm, we conduct the distributed online simulation in the setup simulation environment mentioned above. Different weights are chosen for the objective function equation (7.13) in the distributed algorithm. When this objective function considers only E2E latency with $\rho^i = 0$, it is the baseline algorithm called Latency_Minimisation algorithm. The simulation results (i.e., CPU, OE, and EO conversion resources utilisation, network payoff, and SAR) of the distributed algorithm with different weights and of the Latency_Minimisation algorithm are compared in this subsection.

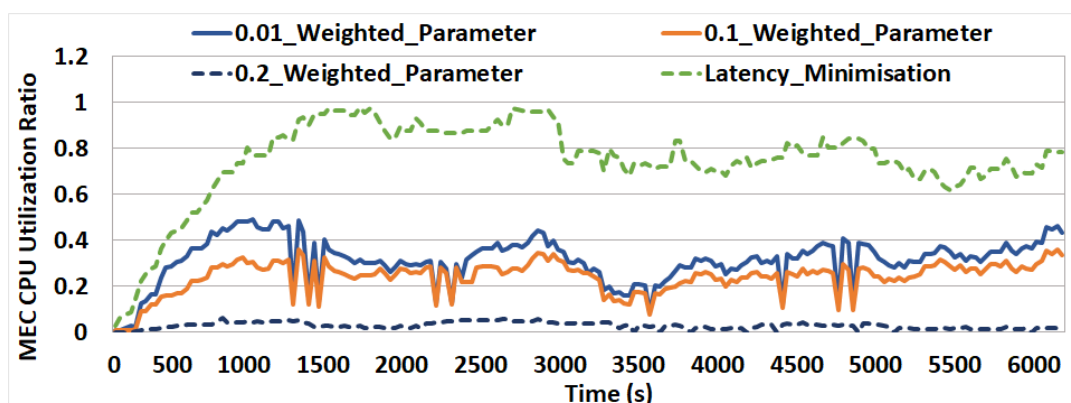


FIGURE 7.2. Average CPU Utilisation in MEC Servers.

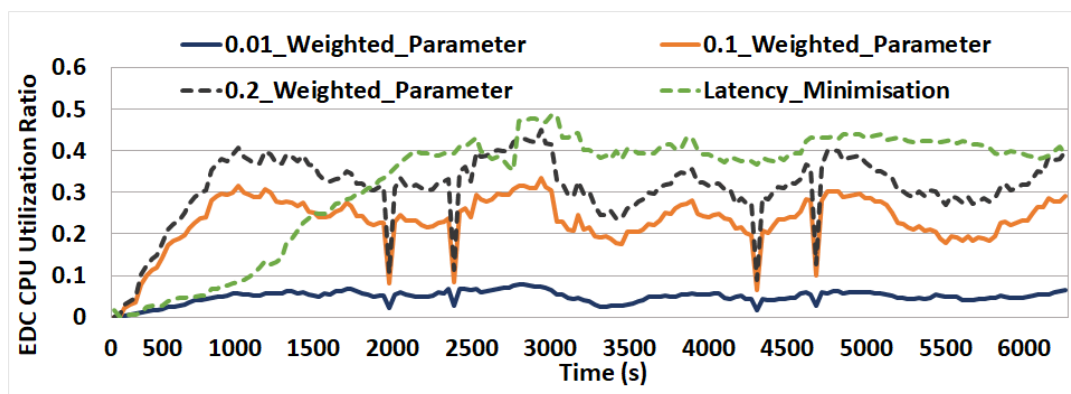


FIGURE 7.3. Average CPU Utilisation in Edge DC.

To be more specific, we compare the results of three distinct ρ value choices for the algorithm with (7.13) as the objective function: 1) 1ms service: $\rho = 0.01$, 5ms service: $\rho = 0.03$; 2) 1ms service: $\rho = 0.1$, 5ms service: $\rho = 0.3$; 3) 1ms service: $\rho = 0.2$, 5ms service: $\rho = 0.4$. other services always assume $\rho = 1$ in these three configurations. The reason for such a setting is that ultra-low latency services are willing to pay a higher price for a higher QoS, whereas others prefer to spend as

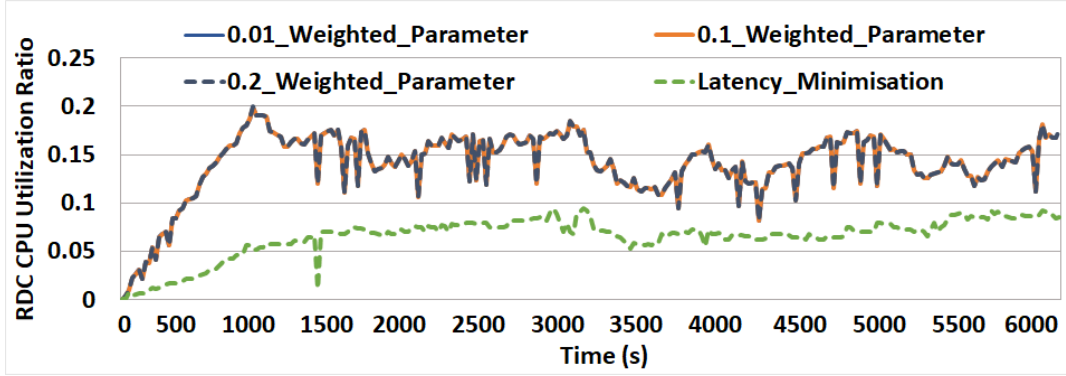


FIGURE 7.4. Average CPU Utilisation in Core DC.

little as possible. The ρ value of 1ms services is used to name each algorithm. For example, if $\rho = 0.01$ for 1ms service, this algorithm will be called as 0.01_Weighted_Parameter algorithm.

Figure 7.2, Figure 7.3 and Figure 7.4, respectively, demonstrate the dynamic average CPU utilisation ratios (i.e. the average CPU utilisation ratio of FW, DPI, and VLC VNF instance on the same node) obtained by the aforementioned four algorithms of MEC servers, EDC and CDC. Compared to other approaches, the baseline method, Latency_Minimisation algorithm, can route more services to MEC nodes and consume more MEC CPU resources. Especially around the 1700s and 2800s, 98% MEC CPU can be exploited (green dashed line in Figure 7.2). Other approaches can only assume a maximum of 50% MEC CPU. As shown in Figure 7.2 and Figure 7.3, the MEC CPU utilisation ratio of the Latency_Minimisation algorithm falls while its EDC CPU utilisation ratio increases after the 2770s. Hence, it can be concluded that the Latency_Minimisation algorithm can route traffic from MEC to EDC when the MEC server is overloaded. However, it always uses fewer CPU resources on CDC than the other three options since it cares about latency only and forwarding packets to CDC will definitely introduce more transmission and propagation latency.

For algorithms that consider both latency and cost, all the non-real-time services are forwarded to the CDC, resulting in the identical CDC CPU utilisation results shown in Figure 7.4. The algorithm with a lower ρ value can deploy more ultra-low latency services on the MEC server. However, even when the ρ value is set to 0.01, the MEC CPU usage ratio is still lower than that of the baseline algorithm (solid blue line is always below the dotted green line in Figure 7.2). In Figure 7.3, it can be found that algorithms with a lower ρ value achieve a higher EDC CPU utilisation ratio. Because moving VNFs from the MEC server to the EDC saves money in our simulation setting, the more network service cares about cost, the higher ρ value will be, and the more ultra-low latency network services will be placed at the EDC. It is worth noting that these approaches' CPU utilisation ratios are not always constant. For example, the average MEC CPU utilisation of algorithm with $\rho = 0.01$ and $\rho = 0.1$ drops dramatically around the 1500s, 2300s, 4500s, and 4800s (presented in Figure 7.2). Such sudden shifts are caused by two factors: 1) the

release of resources after services are completed, and 2) incoming services do not use the released resources at the same time.

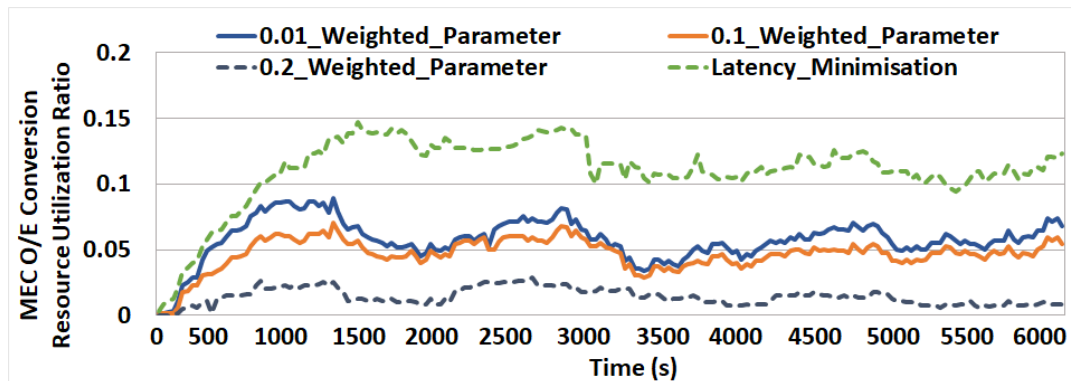


FIGURE 7.5. Average OE Resource Utilisation in MEC Servers.

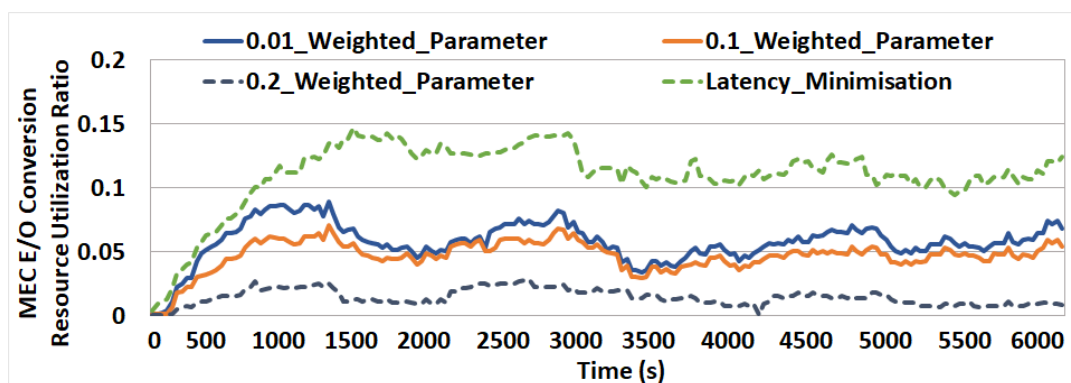


FIGURE 7.6. Average EO Resource Utilisation in MEC Servers.

Throughout the simulation, the average OE and EO conversion resource usage ratios of MEC servers (shown in Figure 7.5 and Figure 7.6, respectively) are nearly the same for each algorithm. Furthermore, each method's MEC CPU, MEC OE, and EO conversion resource utilisation follows a similar pattern. The algorithm, which can handle more services at the edge, consumes more OE and EO conversion resources on MEC nodes. There are no sudden changes in Figure 7.5 and Figure 7.6, unlike the average MEC CPU utilisation ratios of 0.01_Weighted_Parameter algorithm and 0.1_Weighted_Parameter algorithm. Because the average CPU utilisation ratio at the MEC server is determined by the average of three VNFs' resource usage, whereas the average OE and EO conversion resource utilisation ratios are based solely on the average of MEC nodes' conversion resource usage, for example, suppose the VNF2 and VNF3 are all placed at the destination node. In that case, the VNF2's resource release will affect the average CPU utilisation but not affect the average EO conversion resource utilisation since the traffic routing happens at the same node without EO conversion.

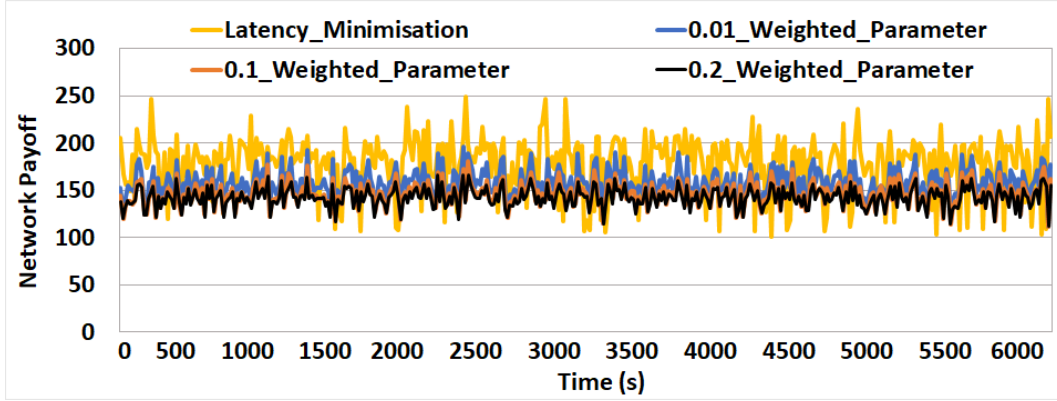


FIGURE 7.7. Network Payoff.

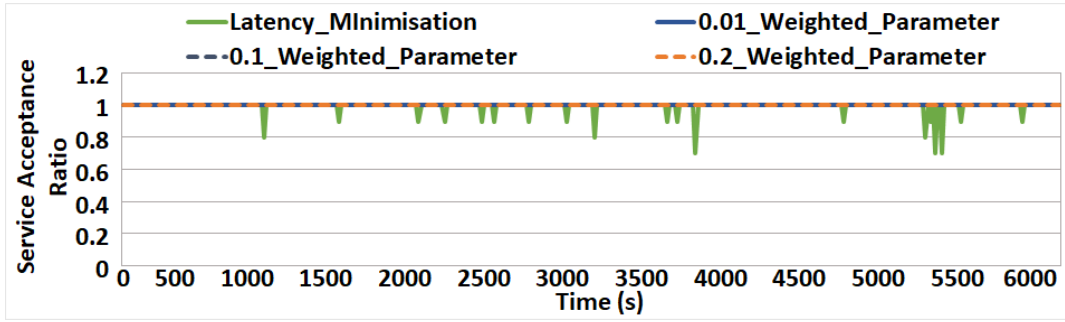


FIGURE 7.8. Service Acceptance Ratio.

Apart from resource usage results, the network payoff performance and the service acceptance performance are also studied. Figure 7.7 presents the network payoff of each algorithm. For algorithms that consider latency and cost jointly, the smaller the ρ value is, the larger the network payoff will be. Although the Latency_Minimisation algorithm can deploy more services on MEC servers, which are more expensive than EDC and CDC, its network payoff performance is not always superior. Because when the MEC server is overloaded, ultra-low latency services are rejected, which can also be proved by results in Figure 7.8. As we can see from this graph, the Latency_Minimisation algorithm's SAR is not always 100%, but the other algorithms can obtain 100% SAR at any moment. Taken advantage of 100% service acceptance performance, the 0.01, 0.1, and 0.2_Weighted_Parameter algorithm can achieve more stable network payoffs than the Latency_Minimisation algorithm. To sum up, the 0.01_Weighted_Parameter approach is the best in our simulation since it can always achieve 100% SAR, consume more resources at edge nodes, and earn more profits than other algorithms.

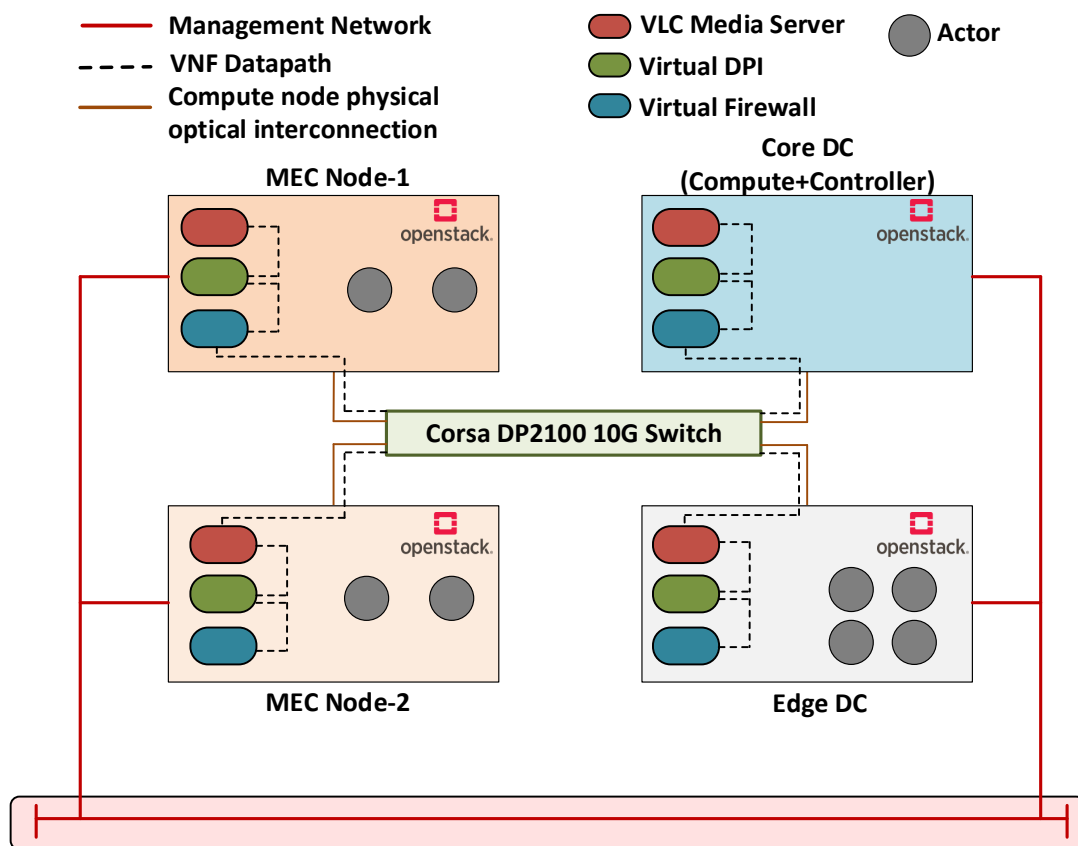


FIGURE 7.9. Experimental Testbed.

7.6 Experiment-based Evaluation

The real testbed experiment is carried out to validate our proposed online distributed algorithm. Both the experimental setup and the corresponding results are reported here.

Figure 7.9 depicts the OpenStack-based experimental testbed, which consists of two CORSAIRONE PRO computers with 8-core processors acting as EDC and CDC, and two IBM x3455 servers with 4-core processors acting as MEC servers. The OE and EO conversion resources in this testbed are Intel multi-mode SFP transceivers, and all of the computing nodes are connected to the Corsa DP2100 switch through 10 Gbps Ethernet links on top of the optical fibres. Because all the network services need the FW, DPI, and VLC in order, three VMs are built on each node, each holding an FW, DPI, and VLC Media Player instance. Service-related parameters like data rate and latency requirements are the same as those in the simulation. We add extra latency of 0.05ms, 0.12ms, and 1.47ms at the MEC server, EDC, and CDC during the experiment to reflect the transmission latency produced by 10km, 25km, and 300km optical fibres between MEC node and switch, EDC and switch, and CDC and switch, respectively.

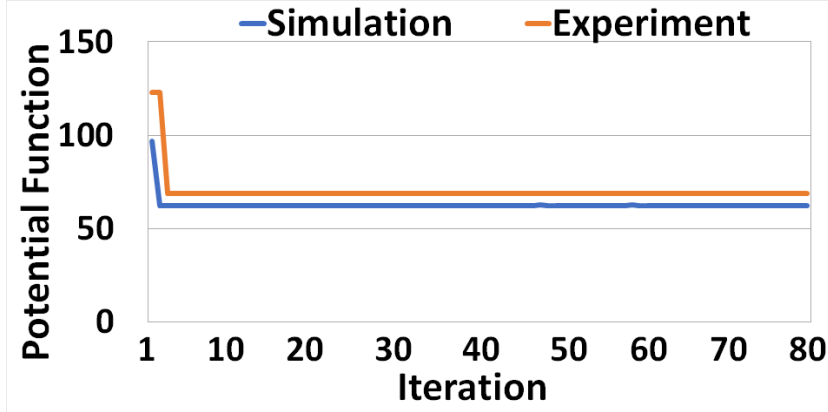


FIGURE 7.10. Potential Function on 5-Node Network.

Table 7.4: Application Latency and Ping Latency Results

Request (source,destination)	Latency Requirement	Application Latency	Ping Latency
(EDC,MEC2)	100ms	14.27ms	1.868ms
(MEC1,CDC)	100ms	26.55ms	15.95ms
(MEC1,MEC1)	1ms	14.7ms	0.77ms
(EDC,MEC1)	80ms	12.175ms	0.868ms
(EDC,MEC2)	80ms	14.22ms	1.91ms
(MEC2,MEC2)	1ms	13.85ms	0.632ms
(EDC,CDC)	100ms	26.65ms	17.35ms
(MEC1,MEC1)	1ms	14.7ms	0.77ms

To run our proposed algorithm, actors (implemented as VMs) are established on the MEC servers and the EDC. The actors collect the network topology and current CPU utilisation ratios of all the VNFs through the management links. Firstly, each actor receives one service request from a total of eight requests that arrive at the same time. The actors then run `0.01_Weighted_Parameter` algorithm to determine which strategy is the best for each service. Following each iteration of the algorithm, actors communicate with one another to learn about the strategies of other services and keep track of the current network status. The algorithm converges towards NE after the second iteration, which takes just 40 seconds, as shown in Figure 7.10.

When the algorithm converges, these eight service requests are routed according to the algorithm's devised resource allocation strategy. The results are shown in Table. 7.4. The time it takes to send a 270 Kbyte packet (the compressed size of MPEG video with the resolution of 640x480) along the chain Source-FW-DPI-VLC-Destination is named the application latency. The time it takes to send a 64-byte ping packet following the same chain is called the ping latency. The experimental results show that the ultra-low latency 5G services can meet their

latency requirements when transmitting 64-byte packets; however, if transmitting 270-Kbyte video packets, even when all VNFs are put at the edge node, the latency is greater than 10ms. To support 1ms services, since it is challenging to achieve 1ms E2E latency for enormous packet transmission, we recommend sending traffic with small packet sizes unless the transmission technologies are much improved to lower the packet transmission latency in the future.

7.7 Summary

In this chapter, the latency-aware SFCs placement problem was first modelled as a congestion game to minimise the E2E latency and resource consumption cost for each network service request. After proving the existence of potential function and pure NE theoretically, a distributed online algorithm was designed to find an optimal solution. For ultra-low latency services, we specify a lower cost weight in the goal function than the non-real-time services so that ultra-low latency services can be routed to edge nodes and network operators can profit more. Simulation findings demonstrate that algorithms with an objective function incorporating both latency and cost minimisation can reach a 100% SAR. The lower the weighted parameter in the objective function, the higher the edge resources utilisation and the greater the network payoff can be attained. The experiments show that the designed algorithm can be employed online with a convergence time of 40 seconds, and the E2E latency requirements for network services with small packet sizes can all be satisfied. We intend to introduce a dynamic pricing mechanism for resource allocation based on resource utilisation status in the future, which can further improve network payoff performance.

MULTI-AGENT DRL FOR SFCs PLACEMENT AND SCHEDULING

To fully address the NFV-RA problem in edge-cloud networks, the joint problem of SFCs placement and scheduling is investigated. As the number of edge nodes and services are numerous and the network status is dynamic and sometimes unpredictable, online scalable and dynamic solutions are needed. Motivated by these requirements, the multi-agent DRL (MADRL) approach is adopted to enable all the computing nodes to make dynamic placement and scheduling decisions in a distributed way. Firstly, this joint problem is formulated as a MADRL model. Secondly, the multi-agent deep deterministic policy gradient (MADDPG) algorithm is applied to two scenarios: 1) fully cooperation scenario where all agents have the same objective to maximise the accepted service requests; 2) self-interest scenario where edge agents maximise the ultra-low latency service acceptance ratio, while DC agents maximise all the service acceptance ratio. Thirdly, simulations are carried out and simulation findings prove the self-interest design can accept more ultra-low latency services than the fully-coordinated scenario. This is an ongoing work, fully distributed training solutions based on Graph Neural Network (GNN) will be added in the future.

8.1 Introduction

In real-world edge-cloud networks, the whole environments are always dynamic [158] as service requests demanding various QoS arrive in a stochastic and unpredictable way. To cope with such dynamic and diversified service requests, computing resources and network resources should be supplied in a flexible and intelligent way. Thanks to the NFV technology, network services can be processed flexibly on any network location. However, considering that ultra-low latency services are mainly processed at MEC nodes, the limited MEC resources must be used efficiently under

different workload scenarios.

To efficiently provide computing resources for network services in an online dynamic process, the latency-aware SFCs resource allocation problem should be investigated on the edge-cloud networks. The resource allocation involves both the placement and scheduling process. The SFCs placement problem has been extensively studied, however, there are few works studying the SFCs scheduling problem and among them, most assume that the SFCs have already been placed and take the SFCs placement solutions as input parameters [48–50, 77]. Compared with solving them separately, solving them together can definitely provide more realistic and adaptive solutions for the variable network status because these two sub-problems have an impact on each other [38].

The joint SFCs placement and scheduling problem can be formulated as a flexible Job Shop Scheduling Problem (JSSP), which allows a VNF to be processed by any same type of VNF instances in the network. In this model, one VNF instance can only process one VNF requirement at a time and each VNF processing is only assigned to one VNF instance, similarly, one link can only transmit one traffic at a time and each traffic transmission is only assigned to one link. The JSSP is proved to be an NP-hard combinatorial optimisation problem [159] and scalable approaches are expected to solve this problem in large-scale edge-cloud networks. There are only two works studying the joint problems by proposing simple heuristic algorithms [38, 43], which are easy to be implemented but cannot guarantee the solution quality [153]. Although meta-heuristic algorithms have been applied for the offline SFCs scheduling problem [48], it is infeasible for real-time scheduling, especially when solving large-scale problems [153].

RL-based scheduling methods have several successful applications on JSSP and could find competitive solutions for JSSP benchmark problems [158]. In NFV-enabled edge-cloud networks, the state space is huge and correlated with the number of computing nodes, the number of physical links, and the number of arriving requests. The general RL approach, which maintains a look-up table to store policies, is not capable of dealing with large infinite state space [33]. To handle high-dimension problems, the DRL approach utilising a deep neural network (DNN) as an approximation function can be adopted. It can on the one hand, efficiently handle a large number of real-time arriving requests especially in large, frequently transferred network state space, on the other hand, can flexible scale with different network topologies and can also be easily applied to different application scenarios [153]. Hence, the dynamic SFCs placement and scheduling problem formulated as flexible JSSP is supposed to be viewed as a Markov decision process (MDP) and solved via the DRL approach.

Most works study the SFCs scheduling problem by providing a centralise algorithm [48–50]. However, it is not practical in real-world edge-cloud networks. Sometimes a central controller simply does not exist or may be costly to install, sometimes the central controller needs to communicate with each agent and the communication overhead at the single controller increases [160]. Moreover, considering the huge amount of edge nodes with computing resources, the centralised approach will degrade the scalability of the whole communication system as well as

robustness to malicious attacks [160].

To overcome the drawbacks of centralised control, the multi-agent DRL approach is utilised to make both MEC nodes and DCs agents make decisions by interacting with the environment in a distributed scheme. Taken into consideration that network state and traffic typically exhibit unpredictable variations due to stochastically arriving requests with different QoS requirements [33]. The proposed MADRL approach is required to handle real-time network variations and various service requests and perform adaptive scheduling for SFC requests with different QoS requirements.

The main contributions can be summarised as follows.

1) The joint SFCs placement and scheduling problem is first modelled as a flexible JSSP problem. Then, the JSSP is viewed as a sequential decision-making problem and a multi-agent DRL model is proposed to tackle the dynamic network scenario.

2) The MADDPG algorithm is implemented to solve the multi-agent SFCs placement and scheduling problem. It is applied to two scenarios: 1) fully cooperation scenario where all agents maximise the accepted service requests, 2) self-interest scenario where MEC agents maximise the accepted ultra-low latency service requests and DC agents maximise the overall service requests.

This chapter is organised as follows: Section 8.2 defines the joint SFCs placement and scheduling problem in edge-cloud networks and formulates this joint problem with a MADRL model. Section 8.3 provides the MADDPG algorithm for a fully cooperation scenario and a self-interest scenario. Simulation settings, results, and analysis are given in Section 8.4. The final section concludes the whole chapter.

8.2 Multi-Agent DRL Formulation for SFCs Scheduling

8.2.1 SFCs Placement and Scheduling Formulation

Given the resource constraints, the network function execution order for a particular service, and the E2E latency requirements, the scheduling problem is characterised as a series of scheduling decisions for network services through the activated VNFs [48]. The joint problem needs to decide 1) on which VNF instances the traffic should proceed, 2) on which links the traffic should be transmitted, and 3) in which time slot the traffic should be executed.

Figure 8.1 shows the example of SFCs placement and scheduling in the edge-cloud network. There are three different SFC requests composed of four VNFs in a chain (Figure 8.1 (a)). $f_m^{k,o}$ represents that the o -th VNF required by the service request k is a m type VNF, which can only be placed to the VNF instance of m type at each node. In SFC1 (blue), the processing order is VNF1 \rightarrow VNF2 \rightarrow VNF3 \rightarrow VNF2; in SFC2 (orange), the processing order is VNF1 \rightarrow VNF2 \rightarrow VNF3 \rightarrow VNF1; in SFC3 (green), the processing order is VNF3 \rightarrow VNF1 \rightarrow VNF2 \rightarrow VNF3. In Figure 8.1 (b), the network topology and SFCs placement results are shown. MEC nodes, EDC, and CDC connected by optical links and switches, host all three types of VNF instances,

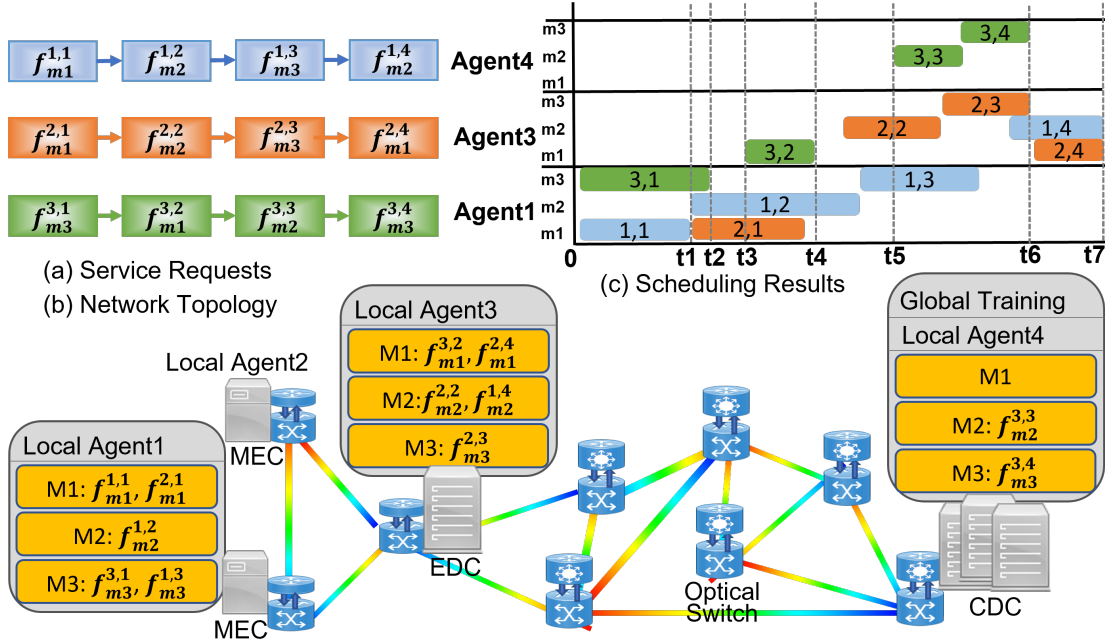


FIGURE 8.1. SFCSs Placement and Scheduling.

$M1$, $M2$, and $M3$. Each computing node owns its local agent where the scheduling algorithm is running. The CDC also owns the function of global training using the global information to ease the centralised training. In this example, the SFCSs placement and scheduling decisions are taken at the same time.

Apart from choosing the nodes to support VNFs and links to transmit traffic, in the scheduling problem, the start time for processing and transmission should also be worked out. In Figure 8.1 (c), the horizontal axis represents the scheduling latency of SFCSs. The total scheduling latency includes queueing latency for processing, processing latency, queueing latency for transmission, transmission latency, and propagation latency. In this subfigure, both $f_{m1}^{1,1}$ and $f_{m2}^{1,2}$ are placed at the same MEC1, and the $f_{m2}^{1,2}$ starts processing right after the $f_{m1}^{1,1}$ finished processing at $t1$. While for SFC3, the $f_{m3}^{3,1}$ and $f_{m1}^{3,2}$ are placed at different nodes, so the $f_{m1}^{3,2}$ starts processing at $t3$ after the transmission and propagation finished, which takes $t3 - t2$. For these three SFCs, the total scheduling latency for the SFC1 and the SFC2 are both $t7$ and for the SFC3 is $t6$.

The processing latency depends on the VNF type and the allocated resources. Each type of VNF instance has its own processing latency with unit resources [161]. The same type of VNF instance on the MEC nodes requires a longer processing time than that on the DCs because more resources are allocated to the VNFs on DC than VNFs on MEC. In this example, the MEC nodes and DCs equipped with different amounts of computing resources have different processing times for the same type of VNF, for example, the processing time of VNF1 is $t1$ at MEC1 and $t4 - t3$ at EDC. The VNF requests of different SFCSs are supposed to be processed by VNF instances in

serial mode. For example, on the VNF1 instance hosting at MEC1, the processing of $f_{m1}^{2,1}$ needs to wait until the processing of $f_{m1}^{1,1}$ is completed, which results in the queueing latency for processing for SFC2. Similarly, the queueing latency for transmission happens because the traffic needs to wait for other traffic finished transmission. In the algorithm, once the SFC starts transmitting, it takes up all the bandwidth resources until the transmission ends.

In the NFV-enabled network infrastructure, we represent the network as a connected graph $G(N, L^P)$, where N is the set of nodes, and L^P is the set of links that connect every two nodes, $\forall (n, n') \in L^P, n, n' \in N$. The set of nodes includes the computing nodes (MEC servers and DCs) $N_v = N_{MEC} \cup N_{DCs}$ and the switching nodes N_{SWN} . In fact, each computing node has a resource capacity, which contains the computing resources (i.e., CPU) and buffering resources, denoted as r_n^{cpu} and r_n^{buf} , respectively. Each link has length $len_{(n,n')}$. It contains the number of $|W|$ wavelengths and the transmission capacity for each wavelength $w \in [1, |W|]_z$ is denoted as $r_{(n,n')}^w$. Considering the optical layer in our model, we use L^{lp} to represent the set of lightpath and each lightpath connects two computing nodes $\forall (l, l') \in L^{lp}, l, l' \in N_v$. lv represents the light speed in the fibre.

We use $|M|$ to represent the number of different VNF types in the system. Each VNF $m \in [1, |M|]_z$ is associated with the specific scaling attribute δ_m , which can be used to calculate the output data rate $v_{output} = v_{input} \cdot \delta_m$. Different types of VNF have their own processing time tp_m . Each VNF instance on the computing node has a certain amount of computing resource capacities $r_{m,n}^{cpu}$ and we assume that only one instance of each type VNF is placed on each node and can be shared by different services.

To deal with the real-time network variations caused by the stochastic arrival of service requests, we introduce the concept of constant time unit $\Delta t = 1\mu s$, time slot τ , which is the integral multiple of a time unit $\tau = i \cdot \Delta t, i \in \mathbb{N}$, and the total time horizon T , during which the arriving service requests are accepted, processed or rejected. The beginning of each time slot is the time step denoted as t . Then, we use K to represent all the SFC requests in the time horizon, and $K(t)$ to represent the set of SFCs at time step t including already existing and newly arrival SFCs. For each service $k \in K(t)$, it is featured by the source node $sn^k(t)$, destination node $dn^k(t)$, the required $o \in [0, |O^k| + 1]_z$ VNF in the chain, the E2E latency DR^k , the arriving time AT^k , the data rate v^k , and the traffic size TS^k . To simplify the model, we use $o = 0$ and $o = |O^k| + 1$ to represent the source and destination node required by the SFC, and $m = 0$ and $m = |M| + 1$ to indicate the 'type' of source and destination, respectively. The o VNF in the request k demands the CPU capacity $c_m^{k,o,cpu}(t) = v_m^{k,o} \cdot tp_m / TS^k$ [161]. Considering different E2E latency requirements, each service is given a priority factor ρ^k , with the higher value showing the lower E2E latency requirement. The packet arrival rate λ^k is modelled as a Poisson process.

At each time step t , the network status such as the current CPU resource utilisation ratio $u_n^{cpu}(t)$, buffer utilisation ratio $u_n^{buf}(t)$, bandwidth resource utilisation ratio $u_{(n,n')}^w(t)$ are calculated. The available CPU capacity for the m type VNF instance $ar_{n,m}^{cpu}(t)$, available buffer capacity

of the node $ar_n^{buf}(t)$, available bandwidth capacity of link $ar_{(n,n')}^w(t)$, the residual time of request RT^k , the residual processing time $drp_m^{k,o}(t)$, and the residual transmission and propagation time $drt_{(l,l')}^{k,o}(t)$ for the VNF o in service request k are updated for the resource allocation at this time step.

8.2.1.1 Constraints

1) Placement constraint: the VNF can only be mapped on one node at time t .

$$(8.1) \quad \sum_{n \in N_v} x_{n,m}^{k,o}(t) = 1$$

2) Resource constraints: equation (8.2) ensures that at time step t , the CPU capacity required by all the m type VNFs on the node n cannot exceed the CPU capacity of m type VNF instance on node n .

$$(8.2) \quad \sum_{k \in K_n(t)} \sum_{o \in [1, |O^k|]_z} \alpha^k \cdot x_{n,m}^{k,o}(t) \cdot C_m^{k,o,cpu}(t) \leq r_{n,m}^{cpu} \quad \forall n \in N_v, m \in [1, |M|]_z$$

The allocated resource for all types of VNF instance on node n cannot exceed its CPU resource capacity.

$$(8.3) \quad \sum_{m \in [1, |M|]_z} r_{n,m}^{cpu} \leq r_n^{cpu} \quad \forall n \in N_v$$

Constraint (8.4) shows that, at time step t , the required buffering resource should not exceed the available buffering resources on node l' . Service chains buffered at node l' when there is not enough computing capacity to serve the SFC.

$$(8.4) \quad \sum_{k \in K_n(t)} \alpha^k \cdot x_{l'm}^{k,1} \cdot PS^k + \sum_{k \in K_n(t)} \sum_{o \in [2, |O^k|]_z} \sum_{l \in N_v} \sum_{w \in [1, |W|]_z} \alpha^k \cdot yw_{(l,l'),w}^{k,(o,o+1)}(t) \cdot PS^k \leq r_{l'}^{buf} \quad \forall l' \in N_v$$

Equation (8.5) illustrates the optical link resource constraint.

$$(8.5) \quad \sum_{k \in K_n(t)} \sum_{o \in [0, |O^k|]_z} \sum_{(l,l') \in L^l p} \alpha^k \cdot tw_{(l,l'),(n,n'),w}^{k,(o,o+1)}(t) \cdot v_m^{k,o} \leq r_{(n,n')}^w \quad \forall (n,n') \in L^P, w \in [1, |W|]_z$$

3) Link mapping constraints: flow conservation, the egress flow amount shall be equal to its ingress flow

$$(8.6) \quad \sum_{(l,l') \in L^l p} \sum_{w \in [1, |W|]_z} \sum_{n' \in N_v} \alpha^k \cdot tw_{(l,l'),(n,n'),w}^{k,(o,o+1)}(t) \cdot v_m^{k,o} - \sum_{(l,l') \in L^l p} \sum_{w \in [1, |W|]_z} \sum_{n' \in N_v} \alpha^k \cdot tw_{(l,l'),(n',n),w}^{k,(o,o+1)}(t) \cdot v_m^{k,o} \\ = x_{n,m}^{k,o}(t) - x_{n,m'}^{k,o+1}(t) \quad \forall k \in K(t), o \in [0, |O^k|]_z, n \in N_v$$

Wavelength continuity should be satisfied when the previous and latter VNF are placed on different nodes

$$(8.7) \quad \sum_{w \in [1, |W|]_z} \alpha^k \cdot yw_{(l,l'),w}^{k,(o,o+1)}(t) \leq 1 \quad \forall k \in K(t), o \in [0, |O^k|]_z, (l,l') \in L^l p$$

Table 8.1: Input Parameters of Multi-agent DRL Model

T	the time horizon
Δt	time unit, in this paper, each time unit is $1 \mu s$
$\tau = i \cdot \Delta t, i \in \mathbb{N}$	the total time horizon is divided into discrete time slot, which is an integral multiple of time unit
$G(N, L^P)$	the network topology consisting of a set of nodes N and a set of physical fiber links L^P
$n \in N$	physical node
$N_v \in N$	the set of computing nodes with computing and buffer resources
$N_{MEC} \cup N_{DCs} = N_v$	MEC servers, Edge DCs, and Core DCs
$N_v \cup N_{SWN} = N$	Switching nodes with only switching capability
$r_n^{cpu}, r_n^{buf}, n \in N_v$	CPU processing capacity and buffer capacity on computing node n
$u_n^{cpu}(t), u_n^{buf}(t)$	cpu and buffer resource utilization ratio on computing node n at time t
$(n, n') \in L^P, n \in N, n' \in N$	physical link connecting node n and n'
$len_{(n, n')}$	the length of physical link
lv	the light speed in the fibre
W	the number of wavelengths in the network
$w \in [1, W]_z$	the w_{th} wavelength in the set of wavelengths
$r_{(n, n')}^w, (n, n') \in L^P$	transmission capacity of the w_{th} wavelength in the physical link (n, n')
$u_{(n, n')}^w(t)$	bandwidth resource utilization ratio on link (n, n') at time t
L^{lp}	the set of lightpath in the network
$(l, l') \in L^{lp}, l \in N_v, l' \in N_v$	the lightpath connecting computing node l and l'
M	the number of different VNF types
$m \in [0, M + 1]_z$	the m_{th} type of VNF, 0 and $ M + 1$ represent the start and destination dummy function, which should be placed at source node and destination node but require no resource, just for the simplicity of modelling
$\delta_m = v_{output}/v_{input}, m \in [0, M + 1]_z$	the scaling attribute of the VNF, which can be used to calculate the output data rate
$r_{m, n}^{cpu}$	the computing capacity of m_{th} type VNF instance on node n , the SFC can be processed immediately if there's enough resource, otherwise, it needs to wait
tp_m	the packet processing time of m_{th} type VNF with unit resource
$ar_{n, m}^{cpu}(t), n \in N_v$	available cpu capacity at m_{th} type VNF instance on node n at time t
$ar_n^{buf}(t), n \in N_v$	available buffer capacity at node n at time t
$ar_{(n, n')}^w(t), (n, n') \in L^P$	available bandwidth capacity on w_{th} wavelength of physical link (n, n') at time t

Table 8.2: Service Requests of Multi-agent DRL Model

K	all the service requests in the time horizon
$K(t)$	the set of SFCs at time t including already exist SFCs and newly arrival SFCs
$sn^k(t), k \in K(t)$	source node of service request k
$dn^k(t), k \in K(t)$	destination node of service request k
$O^k, k \in K(t)$	the number of VNFs required by the service request k
$o, o \in [0, O^k + 1]_z$	the o_{th} VNF in the service request k , 0 represents the start node and $ O^k + 1$ represents the destination node
$DR^k = i_{dr}^k \cdot \tau, i_{dr}^k \in \mathbb{N}, k \in K(t)$	the E2E latency requirement should be integer multiples of time slot
$AT^k = i_{at}^k \cdot \tau, i_{at}^k \in \mathbb{N}, k \in K(t)$	the arriving time of service request k should be integer multiples of time slot
$RT^k, k \in K(t)$	the residual time of service request k
$K_n(t), n \in N_v$	the set of SFCs observed by agent n at time t , including the newly arriving SFCs has source node n , the SFCs just finished its VNF processing at node n , and SFCs routed to this node n
$\lambda^k, k \in K(t)$	the packet arrival rate of service request k , as a Poisson process with arrival rate (packet/s)
$v^k, k \in K(t)$	the data rate required by the service request k
$v_m^{k,o} = \delta_m \cdot v_m^{k,o-1}, if o > 1; = v^k, if o \leq 1$	the data rate for the o_{th} VNF in the service request k
TS^k	the traffic size of the service request k
$c_m^{k,o,cpu}(t) = v_m^{k,o} \cdot tp_m / TS^k$	cpu resource capacity required by the o_{th} VNF in the service request k
TS^k	buffer resource required by the VNF in the service request k
ρ^k	the priority of service request k , the k with the lowest E2E latency has the highest priority
$drp_m^{k,o}(t)$	the residual processing time of the o_{th} VNF in the service request k
$drt_{(l,l')}^{k,o}(t)$	the residual transmission and propagation time of the o_{th} VNF in the service request k

Table 8.3: Output Variables of Multi-agent DRL Model

$x_{n,m}^{k,o}(t), k \in K_n(t)$		a binary variable indicating whether the node n is chosen for processing the o_{th} VNF in the k_{th} service request at time t
$yw_{(l,l'),w}^{k,(o,o+1)}(t), o \in [0, O^k]_z, k \in K_l(t)$	\in	a binary variable indicating whether the lightpath (l, l') is chosen for the $(o, o+1)$ virtual link mapping for the k_{th} service request at time t
$tw_{(l,l'),(n,n'),w}^{k,(o,o+1)}(t), o \in [0, O^k]_z, k \in K_l(t)$	\in	a binary variable indicating whether the physical link (n, n') is chosen for the $(o, o+1)$ virtual link mapping for the k_{th} service request at time t
$stdp_{n,m}^{k,o}, o \in [0, O^k + 1]_z, k \in K_n(t)$		the starting time of the o_{th} VNF for processing at m type VNF instance on node n
$stdt_{(l,l'),w}^{k,(o,o+1)}, o \in [0, O^k]_z, k \in K_n(t)$	\in	the starting time of the o_{th} VNF for transmission at (l, l') lightpath
α^k		a binary variable indicating if the service request k is accepted or not, it equals 1 if the E2E latency requirement and other constraints are satisfied, otherwise, it equals 0 and rejected

The physical link can be selected only when the lightpath is selected.

(8.8)

$$\alpha^k \cdot tw_{(l,l'),(n,n'),w}^{k,(o,o+1)}(t) \leq \alpha^k \cdot yw_{(l,l'),w}^{k,(o,o+1)}(t) \quad \forall k \in K(t), o \in [0, |O^k|]_z, (l, l') \in L^l p, w \in [1, |W|]_z, (n, n') \in L^p$$

Constraint (8.9) defines the relationship between the lightpath and the physical link.

$$(8.9) \quad \sum_{n \in N} \alpha^k \cdot tw_{(l,l'),(n,n'),w}^{k,(o,o+1)} - \sum_{n \in N} \alpha^k \cdot tw_{(l,l'),(n',n),w}^{k,(o,o+1)} = \begin{cases} \alpha^k \cdot yw_{(l,l'),w}^{k,(o,o+1)} & \text{if } n' = l' \\ -\alpha^k \cdot yw_{(l,l'),w}^{k,(o,o+1),m} & \text{if } n' = l \\ 0 & \text{otherwise} \end{cases}$$

$$\forall k \in K(t), o \in [0, |O^k|]_z, (l, l') \in L^l p, w \in [1, |W_k|]_z$$

Equation (8.10) can help avoid loops for switching nodes.

$$(8.10) \quad \sum_{n \in N} \alpha^k \cdot tw_{(l,l'),(n,n'),w}^{k,(o,o+1)}(t) \leq 1 \quad \forall k \in K(t), o \in [0, |O^k|]_z, (l, l') \in L^l p, w \in [1, |W|]_z, n' \in N_{SWN}$$

3) Latency constraints: The E2E latency contains the queueing latency for processing $dqp^{k,o}$, the processing latency $dp^{k,o}$, the queueing latency for transmission $dqt^{k,(o,o+1)}$, the transmission latency $dt^{k,(o,o+1)}$, the propagation latency $dg^{k,(o,o+1)}$.

In this model: 1) The queueing latency for processing depends on the agent's decision, if it is chosen to be processed now, there is no queueing latency, if not, it needs to wait for other VNFs to finish processing. 2) The processing latency is the packet processing latency of the m type VNF (shown in equation (8.11)) because only the VNF provided with enough computing resources can be processed. 3) The queueing latency for transmission also depends on the agent's decision, if it is chosen to be transmitted now, there is no such latency, if not, it needs to wait for other VNFs to

finish transmission and propagation. These two latencies are 0 for 0 and $|O^k| + 1$ VNF, which are included for the model simplification. 4) The transmission latency happens only at computing nodes (shown in equation (8.12)). 5) The propagation latency of the o_{th} VNF in the k service request can be calculated by equation (8.13).

$$(8.11) \quad dp^{k,o} = \sum_{n \in N_v} x_{n,m}^{k,o}(t) \cdot tp_m \quad \forall k \in K_n(t), o \in [0, |O^k| + 1]_z$$

$$(8.12) \quad dt^{k,(o,o+1)} = \sum_{(l,l') \in L^{lp}} \sum_{w \in [1, |W|]_z} yw_{(l,l'),w}^{k,(o,o+1)}(t) \cdot TS^k / r_{(n,n')}^w \quad \forall k \in K_n(t), o \in [0, |O^k|]_z$$

$$(8.13) \quad dg^{k,(o,o+1)} = \sum_{(l,l') \in L^{lp}} \sum_{w \in [1, |W|]_z} \sum_{(n,n') \in L^p} tw_{(l,l'),(n,n'),w}^{k,(o,o+1)}(t) \cdot len_{(n,n')}/lv \quad \forall k \in K_n(t), o \in [0, |O^k|]_z$$

The total scheduling E2E latency for service k , completion time is the time sfc finished after completion of its last operation.

$$(8.14) \quad D^k = stdp_{n,m}^{k,|O^k|+1} + dqdp^{k,|O^k|+1} + dp^{k,|O^k|+1} - AT^k \quad \forall k \in K$$

The total scheduling E2E latency experienced by all functions in the service request should not exceed its E2E latency requirement.

$$(8.15) \quad D^k \leq DR^k \quad \forall k \in K$$

4) Scheduling constraints: Equation (8.16) guarantees that at most one VNF of all SFCs can be processed on m type VNF instance on node n at time t . Equation (8.17) guarantees that at most one VNF of all SFCs can be transmitted on the w_{th} wavelength of link (n, n') at time t . At the beginning of each time slot, the agent updates the resource availability.

$$(8.16) \quad \alpha^k \cdot x_{n,m}^{k,o}(t) + \sum_{k' \in K_n(t), k' \neq k} \sum_{o \in [1, |O^k|]_z} \alpha^{k'} \cdot x_{n,m}^{k',o}(t) \leq 1 \quad \forall k \in K_n(t), o \in [1, |O^k|]_z, n \in N_v, m \in [1, |M|]_z$$

$$(8.17) \quad \alpha^k \cdot yw_{(l,l'),w}^{k,(o,o+1)}(t) + \sum_{k' \in K_n(t), k' \neq k} \sum_{o \in [1, |O^k|]_z} \alpha^{k'} \cdot yw_{(l,l'),w}^{k',(o,o+1)}(t) \leq 1 \quad \forall k \in K_n(t), o \in [0, |O^k|]_z, (l, l') \in L^{lp}, w \in [1, |W|]_z$$

Forwarding a packet cannot start until the VNF finishes its processing.

$$(8.18) \quad stdp_{n,m}^{k,o} + dqdp^{k,o} + dp^{k,o} \leq stdt_{(n,l'),w}^{k,(o,o+1)} \quad \forall k \in K_n(t), o \in [0, |O^k|]_z$$

Processing a VNF on a downstream VM cannot start until the traffic is forwarded from an upstream VM through a virtual link connecting the two VMs.

$$(8.19) \quad stdt_{(l,n),w}^{k,(o-1,o)} + dqt^{k,(o-1,o)} + dt^{k,(o-1,o)} + dg^{k,(o-1,o)} \leq stdp_{n,m}^{k,o} \quad \forall k \in K_n(t), o \in [1, |O^k| + 1]_z$$

8.2.1.2 Performance Metric

1) Average service quality can be calculated separately for service requests with the same E2E latency requirements.

$$(8.20) \quad \sum_{k \in K} \alpha^k \cdot (1 - D^k / DR^k)$$

2) The node CPU utilisation ratio at time t

$$(8.21) \quad u_n^{cpu} = \left(\sum_{m \in [1, |M|]_z} x_{n,m}^{k,o}(t) \cdot C_m^{k,o,cpu}(t) \right) / r_n^{cpu} \quad \forall n \in N_v$$

The average CPU utilisation ratio of the whole network

$$(8.22) \quad ave^{cpu} = \sum_{n \in N_v} u_n^{cpu} / |N_v|$$

The optical link resource utilisation ratio at time t can be calculated as equation (8.23).

$$(8.23) \quad u_{(n,n')}(t) = \left(\sum_{n \in N_v} \sum_{k \in K_n(t)} \sum_{o \in [0, |O^k|]_z} \sum_{(l,l') \in L^l_P} \sum_{w \in [1, |W|]_z} \alpha^k \cdot tw_{(l,l'),(n,n'),w}^{k,(o,o+1)}(t) \cdot v_m^{k,o} \right) / \left(\sum_{w \in [1, |W|]_z} r_{(n,n')}^w \right) \quad \forall (n,n') \in L^P$$

8.2.2 DRL Formulation for SFCs Placement and Scheduling

To deal with the joint SFCs placement and scheduling problem in edge-cloud networks, we consider a multi-agent DRL setting defined by a tuple $\langle Ag, S, A, P, R, O \rangle$. Each computing node is an agent in this model and a set of agents $\langle Ag = (1, \dots, |N_v|) \rangle$ is defined by a joint set of states $\langle S(t) = (s_1(t), \dots, s_n(t), \dots, s_{|N_v|}(t)) \rangle$, a set of joint agent actions $A(t) = (a_1(t), \dots, a_n(t), \dots, a_{|N_v|}(t))$, and a set of observations $O(t) = (o_1(t), \dots, o_n(t), \dots, o_{|N_v|}(t))$ for each agent at each time slot t .

At the beginning of each time slot, each agent has both its local state including available resources, waiting SFCs to be processed and their properties, SFCs residual time, SFCs residual VNFs, and also the global states like the network status (network topology and bandwidth resource usage). $|N_v|$ elementary actions are assumed to be executed concurrently, the agent on node n must decide scheduling, which VNF to be processed at time t , which should be forwarded, which should be rejected. For each agent the dimension of action space is $|K_n(t)| \cdot (|N_v| + 1)$. At each time slot, the agent first finds the feasible action set based on the current system state $s_n(t)$, then chooses an action $a_n(t)$ from the feasible action set according to the current scheduling policy. 0 means the VNF still waiting at node n to be decided at the next time slot. If n equals the numbering of the agent, it means the VNF starts processing at the agent, otherwise, the VNF starts being forwarded to the n node.

A reward function $R : S \times A(s(t), a(t), s(t+1))$ is fed back to the agent for updating the scheduling policy through the RL algorithm, denoting the reward for executing $a(t)$ in $s(t)$ and transitioning to $s(t+1)$. Each agent n receives rewards based on the state and the agent's action: $r_n : S \times A_n \mapsto \mathbb{R}$.

We calculate the accumulated reward for continuous state-action pairs before an episode (i.e., a sequence of agent-environment interactions between initial and completion states) finishes, rather than giving the agent an instant reward after taking action at each time slot. Such accumulated reward is then returned to the agent in order to assist in the improvement of its VNF scheduling policy.

The state transition probabilities $P: S \times A \times S(s(t+1)|s(t), a(t))$, denoting the probability that the system arrives at $s(t+1)$ upon executing $a(t)$ in $s(t)$. A joint policy is a set of local policies $\pi|S \times A = (\pi_1, \dots, \pi_n, \dots, \pi_{|N_v|})$.

8.3 Multi-Agent DRL Algorithm

To efficiently utilise the MEC resources for supporting ultra-low latency services, the MADDPG algorithm is selected for training the multi-agent DRL model. This is a framework of centralised training with decentralised execution which allows the policies to use extra information to ease training and enables agents to use local information at execution time [91]. Since each agent Q-value is learned separately in the MADDPG, agents can have arbitrary reward structures [91]. Hence, we propose 1) a fully-cooperation scenario where both MEC agents and DC agents maximise the overall accepted service requests. 2) a self-interest scenario where MEC agents maximise the accepted ultra-low service requests, while DC agents maximise the overall accepted service requests.

In the fully cooperative environment, equation (8.24) represents the objective of all agents, to maximise the overall accepted service requests while satisfying all the constraints.

$$(8.24) \quad Obj_n = \max_{k \in K_n} \alpha^k - \psi \quad \forall n \in N_v$$

in which ψ represents the constraint violation [87].

In the self-interest environment, the DC agents hold the same objective function as that in the cooperation environment, while MEC agents use equation (8.25) to calculate the objective of maximising the accepted ultra-low latency services.

$$(8.25) \quad Obj_n = \max_{k \in K_n} \alpha^k \cdot \rho^k - \psi \quad \forall n \in N_{MEC}$$

For MADDPG, the set of all agent policies is denoted by $\pi = (\pi_1, \dots, \pi_n, \dots, \pi_{|N_v|})$, for these $|N_v|$ agents, policies are parameterised by $\theta = (\theta_1, \dots, \theta_n, \dots, \theta_{|N_v|})$. If all the agents' actions are known, the environment remains stationary even if the policies change. $P(s'|s, a_1, \dots, a_{|N_v|}, \pi'_1, \dots, \pi'_{|N_v|}) \forall \pi_n \neq \pi'_n$. The main idea is to directly adjust the parameter θ of the policy in order to maximise the objective, the objective is the expected return for agent n : $J(\theta_n) = \mathbb{E}[R_n]$. The gradient of the expected return for agent n is $\nabla_{\theta_n} J(\theta_n) = \mathbb{E}_{s \sim p^\mu, a_n \sim \pi_n} [\nabla_{\theta_n} \log \pi_n(a_n | o_n) \mathbf{Q}_n^\pi(\mathbf{x}, a_1, \dots, a_{|N_v|})]$, in which $\mathbf{Q}_n^\pi(\mathbf{x}, a_1, \dots, a_{|N_v|})$ is a centralised action value function that accepts all of the agents' actions as input, along with some state information \mathbf{x} , and outputs the Q-value for agent n . This action

value function is learned separately so agents can have arbitrary reward structures, including conflicting rewards in competitive environments. In simple case, the state information consists of the observations of all agents $\mathbf{x} = (o_1, \dots, o_{|N_v|})$.

The experience replay buffer $D(\mathbf{x}, \mathbf{x}', a_1, \dots, a_{|N_v|}, r_1, \dots, r_{|N_v|})$ constraints the tuples regarding experiences of all agents, recording experiences of all agents. $|N_v|$ continuous deterministic policies can be denoted as μ_{θ_n} . The gradient of deterministic policies is shown in equation (8.26).

$$(8.26) \quad \nabla_{\theta_n} J(\mu_n) = \mathbb{E}_{x, a \sim D} [\nabla_{\theta_n} \mu_n(a_n | o_n) \nabla_{a_n} Q_n^\mu(\mathbf{x}, a_1, \dots, a_{|N_v|} | a_n = \mu_n(o_n))]$$

The loss function is updated as equation (8.27), in which $\mu' = \mu_{\theta'_1}, \dots, \mu_{\theta'_{|N_v|}}$ is the set of target policies with delayed parameter θ'_n .

$$(8.27) \quad \begin{aligned} L(\theta_n) &= \mathbb{E}_{\mathbf{x}, a, r, \mathbf{x}'} [Q_n^\mu(\mathbf{x}, a_1, \dots, a_{|N_v|}) - y]^2, \\ y &= r_n + \gamma Q_n^{\mu'}(\mathbf{x}', a'_1, \dots, a'_{|N_v|})|_{a'_j = \mu'_j(o_j)} \end{aligned}$$

Algorithm 6 presents the modified MADDPG algorithm for joint SFCs placement and scheduling. Each DRL agent is divided into two parts, critic-network and actor-network. Hence, there are $|N_v|$ actors and $|N_v|$ critics. In the critic-network, the agent is responsible for value estimation and gives an approximation that allows the agent to make the action in the next step. In contrast, actor-network estimates the behaviour distribution when the agent arrives at a new state and relies on the approximation given by the critic-network to take actions.

The input parameters to this algorithm include the network topology, the network status, and arriving service requests. The output parameters include the placement and scheduling results, the service acceptance performance, and resource usage. The main loop starts from line 3. In the beginning, a random process is initialised to explore actions. At the time step $t = 0$, the network will be reset to the initialised status where there is no service processing and all the servers and links are in an idle situation (line 6). Then services are coming according to the Poisson distribution (line 9).

After receiving service requests, each agent selects actions based on current policy, the network moves to the next step by executing these actions, and reward and the new state are observed. Then, the tuple (x, A, R, x') is stored in a replay buffer D . The training process for each pair actor-critic network relies on these buffered tuples. According to the experience replay technique, the agent randomly samples a mini-batch to train the neural network. The critic-network is updated by minimising the loss function, while, the actor-network is updated according to the sampled policy gradient. At the end of each iteration, we update the target neural network.

8.4 Simulation-based Evaluation

8.4.1 Simulation Setup

The simulation network topologies, resource capacities, VNF properties, SFC-related parameters, and other simulation settings are the same as those in Section 5.4 of Chapter 5. The MADDPG

Algorithm 6: MADDPG Algorithm for SFCs Placement and Scheduling

```

1 Input: Network Topology  $G(N, L^P)$ , Network Status  $ar_{n,m}^{cpu}(t)$ ,  $ar_n^{buf}(t)$ ,  $ar_{(n,n')}^w(t)$ , Service
   requests  $k$  and their properties
2 Output: Placement and scheduling solutions:  $x_{n,m}^{k,o}(t)$ ,  $yw_{(l,l'),w}^{k,(o,o+1)}(t)$ ,  $tw_{(l,l'),(n,n'),w}^{k,(o,o+1)}(t)$ ,
    $stdp_{n,m}^{k,o}$ ,  $stdt_{(l,l'),w}^{k,(o,o+1)}$ , Service acceptance performance  $\alpha^k$  and  $D^k$ , and Resource
   utilisation performance  $u_n^{cpu}(t)$ ,  $u_n^{buf}(t)$  and  $u_{(n,n')}^w(t)$ 
3 for  $episode = 1$  to  $10000$  do
4   Initialise a random process for action exploration;
5   if  $episode$  is integral multiple of  $T$  then
6     Reset the network state  $x$ ;
7   end if
8    $t = 0.1 * (episode \% T)$ ;
9   Read service requests coming at time  $t$ ;
10  For each agent  $n$ , select action  $a_n(t)$  w.r.t the current policy and exploration;
11  Execute actions  $A(t)$  and observe reward  $R$  and new state  $x'$ ;
12  Store  $(x, A, R, x')$  in replay buffer  $D$ ;
13   $x \leftarrow x'$ ;
14  for agent  $n = 1$  to  $N_v$  do
15    Sample a random minibatch of  $S$  samples from  $D$ ;
16    Set  $y^j = r_n^j + \gamma Q_n^{\mu'}(\mathbf{x}', a_1^j, \dots, a_{|N_v|}^j) |_{a_k' = \mu_k'(o_k^j)}$ ;
17    Update critic by minimising the loss  $L(\theta_n) = \frac{1}{S} \sum_j [(y^j - Q_n^\mu(\mathbf{x}^j, a_1^j, \dots, a_{|N_v|}^j))^2]$ ;
18    Update actor using the sampled policy gradient:
        $\nabla_{\theta_n} J \approx \frac{1}{S} \sum_j \nabla_{\theta_n} \mu_n(o_i^j) \nabla_{a_n} Q_n^\mu(\mathbf{x}^j, a_1^j, \dots, a_{|N_v|}^j) |_{a_n = \mu_n(o_n^j)}$ ;
19  end for
20  Update target network parameters for each agent  $n$ :  $\theta_n' \leftarrow \tau \theta_n + (1 - \tau) \theta_n'$ 
21 end for
    
```

algorithm runs on the 29-node network shown in Figure 5.3. In the self-interest environment, accepting 1ms, 5ms, 100ms, and 500ms service requests at MEC nodes can generate 500, 100, 5, and 1 reward, respectively. Vice versa, rejecting services lead to corresponding negative rewards. While accepting any services at CDC get 100 rewards and rejecting any services get -100 rewards. In the cooperation environment, accepting services generates 100 rewards and rejecting services leads to -100 rewards at all nodes. A service can only be accepted if all the chained VNFs are finished processing, the traffic is transmitted from the source node to the destination node, and the E2E latency is within the requirement.

In the dynamic network environment, we adopt the Markov modulated process [77] to simulate service requests arrivals with two states low and high, represented by arrival rates λ_l and λ_h , respectively. The probability of transition from low to high state is denoted as p_h , and the probability of transition from high to low state is denoted as p_l . We set the $\lambda_l = 1/240\mu s$,

$\lambda_h = 1/24\mu s$, $p_l = 0.56$, $p_h = 0.4$ [77].

The MADDPG algorithm is implemented with PyTorch on an IBM System, with 24GB RAM and dual-core AMD opteron processor.

8.4.2 Simulation Results and Analysis

In this subsection, the simulation results are compared and analysed. For both self-interest and cooperation scenarios, we design two training schemes: 1) each agent takes action for a single waiting service request at each time slot and the first coming service request will be processed first (named as the *self-interest1* and the *cooperation1* training patterns); 2) each agent takes action for five waiting service request at the same time for each time slot and the first five coming service requests will be processed first (named as the *self-interest5* and the *cooperation5* training patterns).

Figure 8.2 compares the resource utilisation performance for all the training patterns. It can be found that training five service requests simultaneously can get better results than training a single service request. To be more specific, in Figure 8.2 (a), the *self-interest5* can achieve the highest average CPU utilisation ratio of MEC nodes (around 50%), the *cooperation5* achieves the second-highest (around 40%), while the *self-interest1* and the *cooperation1* use lower edge CPU resources. In addition, we consider the CPU usage for a single MEC node (MEC0). In Figure 8.2 (c), we can find that better results can always be obtained by training five service requests than by training one. However, for this MEC0 agent, the *cooperation5* performs better than the *self-interest5* in terms of both stability and CPU utilisation ratio. Figure 8.2 (b) compares the CPU utilisation ratio on the CDC, which also demonstrates that the *cooperation5* and the *self-interest5* are capable of utilising more DC resources than the other two schemes. Furthermore, the link resource usage performance is also taken into account. Figure 8.2 (d) shows the obvious bandwidth utilisation difference between the 5-services training pattern (0.11%) and 1-service training pattern (0.02%).

From these four sub-figures, it can be concluded that 1) a 5-services training pattern can get better overall resource utilisation performance than a 1-service training pattern because more services can be processed and transmitted instead of waiting at the queueing list; 2) there is no significant difference between the *self-interest* and the *cooperation* environment in terms of the resource usage, for example, the *self-interest5* and the *cooperation5* can get comparable link bandwidth utilisation ratio.

Figure 8.3 presents the obtained rewards for MEC nodes and CDC. It can be seen from Figure 8.3 (a) and (b) that the *self-interest5* training pattern can earn the highest rewards for both MEC nodes and CDC. In the Figure 8.3 (a), the *self-interest5* achieves around 30000 reward, the *cooperation5* achieves around 18000 reward, followed by the *self-interest1* and *cooperation1* achieving around 4800 reward. In the Figure 8.3 (b), the *self-interest5* achieves around 4000 reward, the *cooperation5* achieves around 2500 reward, followed by the *self-interest1*

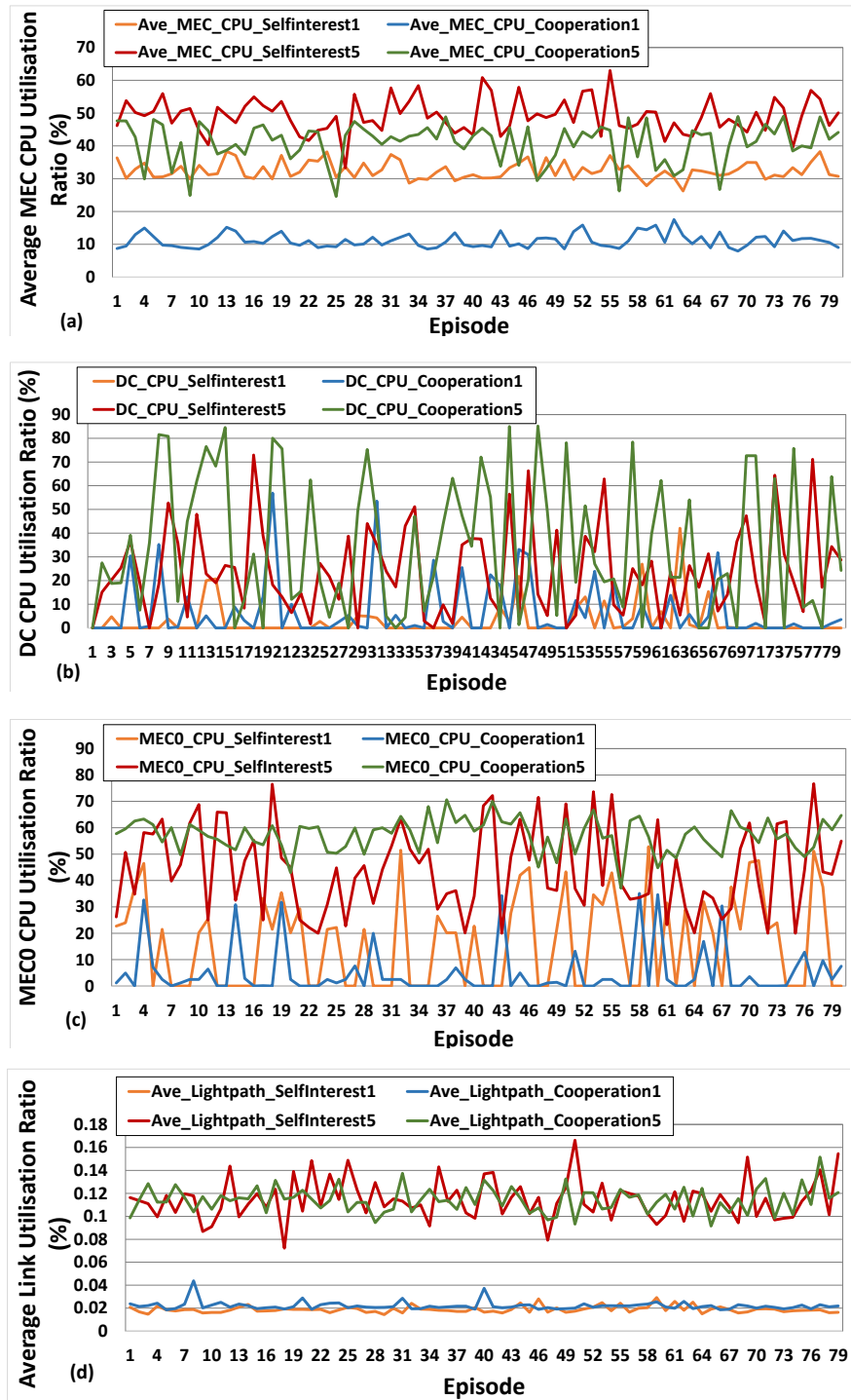


FIGURE 8.2. Resource Utilisation Ratio: (a) Average MEC Node CPU Utilisation Ratio, (b) CDC CPU Utilisation Ratio, (c) MEC0 Node CPU Utilisation Ratio, (d) Average Link Bandwidth Utilisation Ratio.

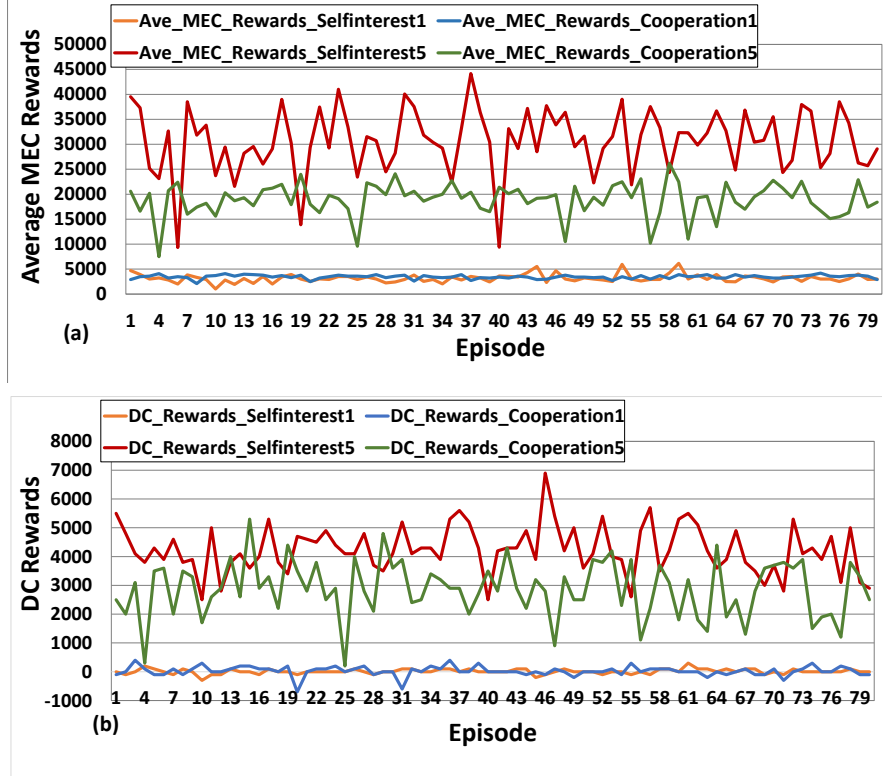


FIGURE 8.3. Rewards: (a) Average MEC Rewards, (b) CDC Rewards.

and *cooperation1* achieving around 0 reward. The reward difference is not obvious when training a single service in both the self-interest and the cooperation environment. However, when the number of trained services increases, the self-interest environment outperforms the cooperation one. Especially for the MEC nodes, the *self-interest5* can process more ultra-low latency services (also proved in the Figure 8.4) and gain almost double rewards compared to the *cooperation5*.

The SAR results for all types of services and 1ms services are compared in Figure 8.3 (a) and (b), respectively. The 5-services training scheme can make decisions for five services simultaneously and, therefore, get higher SAR than the 1-service training scheme. Among all the training schemes, the *self-interest5* gets the best overall performance with the highest 25% SAR and 23% 1ms SAR, the *cooperation5* gets the second-highest 20% SAR and 21% 1ms SAR, while the other two 1-service training schemes get around 5% SAR and 5% 1ms SAR. More service requests can be accepted in the self-interest environment than in the cooperation environment because different services can give different rewards, and MEC agents can be trained to accept more ultra-low latency services and earn more rewards in the self-interest scenario. While in the cooperation scenario, because all kinds of services produce the same reward, the MEC agent will not be trained to get more 1ms service requests and these ultra-low latency services may be rejected due to occupied resources by other service requests.

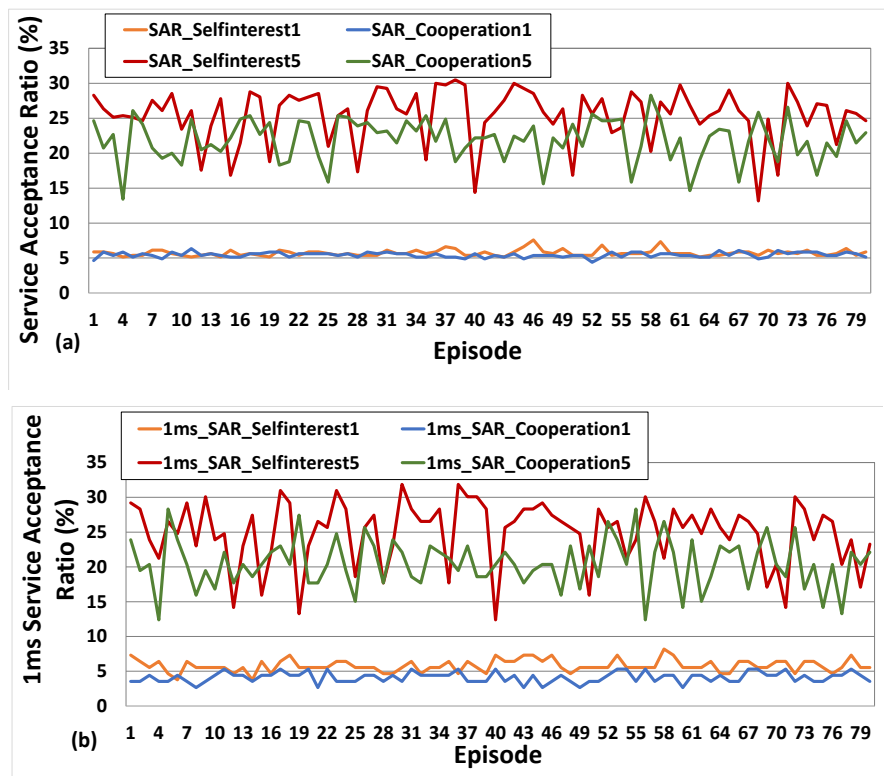


FIGURE 8.4. Service Acceptance Performance: (a) All Service Acceptance Performance, (b) 1ms Service Acceptance Performance.

From all the results, we can conclude that training more service requests at the same time can produce higher rewards for different agents, reach higher service acceptance performance for different services, and use more node and link resources to avoid resource wastage compared to process a single service request each time. Between two 5-services training patterns, the *self-interest5* can produce better results in terms of rewards and SAR than the *cooperation5* since MEC agents can be trained to use limited resources for ultra-low latency services in the self-interest environment. However, the simulation results prove the scalability limitation of the multi-agent RL, which is caused by the interaction among agents after taking actions. In the future, the algorithm will be trained for more episodes to see the convergence performance of the MARL.

8.5 Summary

In this chapter, to cope with the high capacity and low latency 5G service requirements in densely deployed edge nodes environment, the joint SFCs placement and scheduling problem is solved by a multi-agent DRL approach. The proposed approach enables the agents to make their own decisions with local information, which satisfies both the privacy and scalability requirements.

This is ongoing work, and at the current stage, the MADDPG algorithm is modified to solve this joint problem. Simulation results prove that the self-interest setup can achieve higher rewards for different agents and higher service acceptance performance for various QoS services.

CONCLUSION AND FUTURE WORK

This thesis focuses on the resource allocation for SFCs in multi-layer edge-cloud networks. Different methodologies and techniques, ranging from MILP, DRL, and game theory modelling to heuristic, meta-heuristic, and DRL algorithms, are adopted to solve the QoS-aware SFCs placement and scheduling problems from different aspects. The proposed multi-layer models and algorithms can effectively improve the service acceptance performance, especially for ultra-low latency services, with limited edge resources. Based on these research works, six directions can be explored in the future, including 1) machine-learning assisted dynamic price mechanism for SFCs placement, 2) online resource allocation for SFCs in the real testbed, 3) fully distributed resource allocation for open multi-agent systems, 4) fuzzy logic control for practical resource allocation, and 5) resource allocation digital twin.

9.1 Summary and Achievements

The combination of MEC and NFV plays a crucial role in the 5G and beyond 5G networks as edge computing provides resources at the network edge to host VNFs for more flexible and dynamic network services with more diversified QoS requirements [78]. To deploy the NFV-based network infrastructure, resource allocation is one of the main challenges of demanded network services [21], thus, the NFV-Resource Allocation (NFV-RA) problem has drawn great attention from both industry and academics in recent years.

However, not many of the state-of-the-art research works solve this problem in edge-cloud networks and the following challenges are not properly addressed: 1) support services with various QoS requirements, especially those demanding large capacities and low latency requirements in the same network infrastructure; 2) efficiently utilise limited resources at edge nodes under

different workload scenarios; 3) solve large scale problems involving the rising number of service requests and a large number of edge nodes; 4) different relationship among edge nodes and DCs, such as competence, cooperative and self-interested.

In this thesis, the NFV-RA problem has been studied in the edge-cloud networks and the aforementioned challenges have been well addressed. The main contributions include: 1) define the QoS-aware SFCs placement and scheduling problems in the multi-layer networks consisting of the virtual layer, IP layer and optical layer; 2) introduce the optical layer into the model design by considering the optical resources and constraints to support low latency and large-capacity communications; 3) design different models for the MEC nodes and DCs to capture their different resource capacity and geo-location features and improve the edge resource utilisation and ultra-low latency service acceptance performance under different workload scenarios; 4) provide a comprehensive solution for the SFCs placement and scheduling problems under different situations by proposing a wide range of algorithms based on different methods such as MILP, heuristic algorithms, EAs, game theory, and DRL algorithms; 5) both simulation and real testbed experiment evaluation are carried out to verify the performance of proposed algorithms, with results showing the improvement in terms of QoS, resource utilisation, congestion control, service acceptance ratio, as well as revenue.

Chapters 4-6 provide centralised offline solutions for the QoS-aware SFCs placement problem. In chapter 4, a single-objective MILP optimisation model is proposed to minimise the total service E2E latency to satisfy the QoS requirements. The proposed Data Rate-based heuristic algorithm can route SFCs from MEC nodes to the DCs when the workload is very high to ease congestion of MEC nodes and accept more ultra-low latency services. Then, this model is expanded to multi-objective by adding a total congestion minimisation objective in chapter 5. With two objectives, the resource usage between edge and cloud can be consistently balanced under different workload scenarios. The EAs are adapted to this multi-objective problem to find non-dominated solutions. However, both MILP and EAs are not scalable. Hence, in chapter 6, the DRL approach is adopted for solving the large-scale multi-objective problems. A pointer network is used for modelling and a Chebyshev-assisted Actor-Critic algorithm is proposed to find the non-dominated solutions. By selecting different weights, it can improve both the MEC resource usage and the service acceptance performance under different scenarios.

Chapter 7 provides distributed online solution for the SFCs placement problem. The SFC requests are modelled as a congestion game competing for the computing and network resources. In the objective function, both E2E latency and cost are included. We set a high price for the MEC nodes' resources and a low price for the DCs' resources. Therefore, services requiring ultra-low E2E latency care more about the QoS and would pay more to be processed at the MEC nodes to increase the payoff for network operators. Both chapter 6 and chapter 7 involve the real testbed experiments to validate the proposed algorithms.

Chapter 8 offers centralised training and decentralised execution online solutions for the

SFCs scheduling problem. This is an ongoing work, and in this stage, the DRL model is designed and a MADDPG algorithm is adapted to solve this problem.

In summary, two stages of the QoS-aware NFV-RA problem (i.e., the SFCs placement problem and the SFCs scheduling problem) in multi-layer edge-cloud networks have been studied in depth. For small-scale, large-scale, centralised, distributed, offline, online, cooperative, and competitive circumstances, the corresponding multi-layer models and algorithms are designed based on different methods. These proposed models and algorithms are evaluated via simulation and real testbed experiments to show their effective improvement on the metrics of service acceptance ratio, E2E latency, resource utilisation, congestion ratio, and payoff. Notably, they can significantly enhance the service acceptance performance for the ultra-low latency services with limited edge resources.

9.2 Future Work

For the future 6G network of the 'Internet of Intelligence', the resource allocation problem for SFCs at the network edge is still a very hot topic, which can help to fully unleash the potential of the 6G network for the continued rise of mobile devices, IoT devices, and interactive applications. However, the combination of MEC and NFV is still at the early stage and no effort should be spared in this area for foreseeing future demands. In this subsection, some future directions are provided to help gain more benefits from the NFV-enabled edge-cloud networks.

1) Machine-learning assisted dynamic price mechanism for SFCs placement. Cost minimisation or revenue maximisation has been widely studied for the SFCs placement problem. Although, different prices are set for resources at different locations in the edge-cloud networks or multi-domain networks [71, 78, 162, 163], the prices for resources and VNF instances are fixed. Such a setting can ease the problem complexity but may not gain all benefits from NFV. If there is a dynamic price mechanism that is capable of attracting more services at low workload scenarios and earning more money at high workload scenarios, the network payoff can be further improved. Especially with the assistance of ML approaches, the price can be charged intelligently. Future research can take this as a direction and find the best ML approach for resource allocation.

2) Online resource allocation for SFCs in a real testbed. Based on our best knowledge, there is no research work carried out testbed experiments for online SFCs placement or scheduling. Some focus on the implementation of SFC in the network [164, 165], others design placement algorithms and test their offline performance [163, 166, 167]. To realise this goal, the node resource and network resource utilisation ratios need to be monitored dynamically; the online algorithm needs to dynamically collect network status and SFC requests; the VNFs need to be automatically instantiated, scaled, and deleted; and the SFCs need to be automatically chained, set up, and deleted under the control of the algorithm. As the SFCs can now be supported by the OpenStack and OpenDaylight [164, 166], the aforementioned functions should be developed to

practically support the flexible and dynamic SFCs.

3) Fully distributed resource allocation for open multi-agent systems. Although centralised training and decentralised execution methods can guarantee the convergence of the multi-agent DRL, there are still some problems that can not be solved, such as the single point failure and privacy issues. In addition, this kind of method also suffers from the limitation of prior coordination for the joint training, which restricts the flexibility of the multi-agent system requiring quickly adaptive coordination [168]. Moreover, the number of agents can be variable instead of the fixed number, for example, agent breakdown. To support more flexible and scalable multi-agent systems, fully distributed algorithms are required for the open multi-agent system. Research has been carried out using Graph Neural Network (GNN) to realise such distributed multi-agent learning [168, 169]. Their proposed methods could be modified to the resource allocation problem for SFCs.

4) Fuzzy logic control for practical resource allocation. Considering the variation of a practical system parameter, the uncertainty of workload, as well as the nonlinearity characteristics of system performance, it is difficult and sometimes impossible to achieve the accurate modelling [170]. For example, in chapter 6, the results of the simulation and testbed experiment are different when the same SFCs placement algorithm is utilised because firstly, the CPU utilisation ratio does not linearly increase with the placed VNFs, secondly, the queueing model used for the simulation is mean oriented and not precise for the practical scenario [170]. Hence, the model-independent fuzzy logic control requiring less knowledge of the networks but providing more robust solutions needs to be designed. The design objective is to convert human knowledge and experience into a set of control rules that can be used to govern resource allocation without models. Authors in [171] study the resource allocation in virtualised environments for web services and develop a fuzzy control for QoS assurance with respect to response time. Authors in [172] propose a VM monitor to allocate resources for VMs in cloud computing dynamically. To provide scalability, they provide a multi-agent version of the fuzzy controller that can guarantee application service level agreements (SLAs) and save operating costs. Motivated by the good performance of their solutions, future research should consider fuzzy control as an approach for the practical SFCs resource allocation problems.

5) Resource Allocation Digital Twin. Faced with the increased network complexity owing to virtualised infrastructure and stringent QoS requirements, the dynamic resource allocation for SFCs requires the real-time interaction between simulation and real network. Digital Twin (DT), therefore, attracts both attention of academics and industry for its capability of digitally simulating the network's behaviours, predicting the time-varying performance, and providing the real-time interaction between physical and virtual domains to enhance efficient time-varying network management [173]. It is a virtual representation of the physical system utilising optimisation algorithms and ML algorithms to study the dynamics of the given physical system [174]. DT has already been applied to the resource allocation area. In [173], researchers

develop a GNN-based network DT for network slicing to get the real-time E2E slicing latency under different resource utilisation situations before making any resource allocation decisions. In [175], authors design a federated learning-based DT of edge networks for real-time data analysis and network resource optimisation. The proposed DT model based on the historical running data of devices aims to improve communication efficiency and reduce energy costs for IoT devices. In the future, the potential of DT can be further fulfilled by combining with SFCs in edge-cloud networks.



APPENDIX A

This appendix includes, on the one hand, detailed designs for Neural Networks in Chapter 6 and Chapter 8, on the other hand, the training performance of the proposed DRL agents in Chapter 6 and Chapter 8.

A.1 Hyperparameters and Training Performance for Multi-Objective DRL

The main hyperparameters configured in the model are presented in the following Table. A.1.

Table A.1: Hyperparameter Configuration for Pointer Network

Learning rate (agent)	0.0001
Batch size	1
Number of layer	1
Encoder/Decoder layers x Hidden size	1x128
Dimension of actor/critic input	Number of VNFs \times Number of SFCs
Dimension of actor output	Number of Computing Nodes
Embedding size	10
Learning rate (baseline)	0.1
The number of training steps	800
Gradients clipped to the norm	1

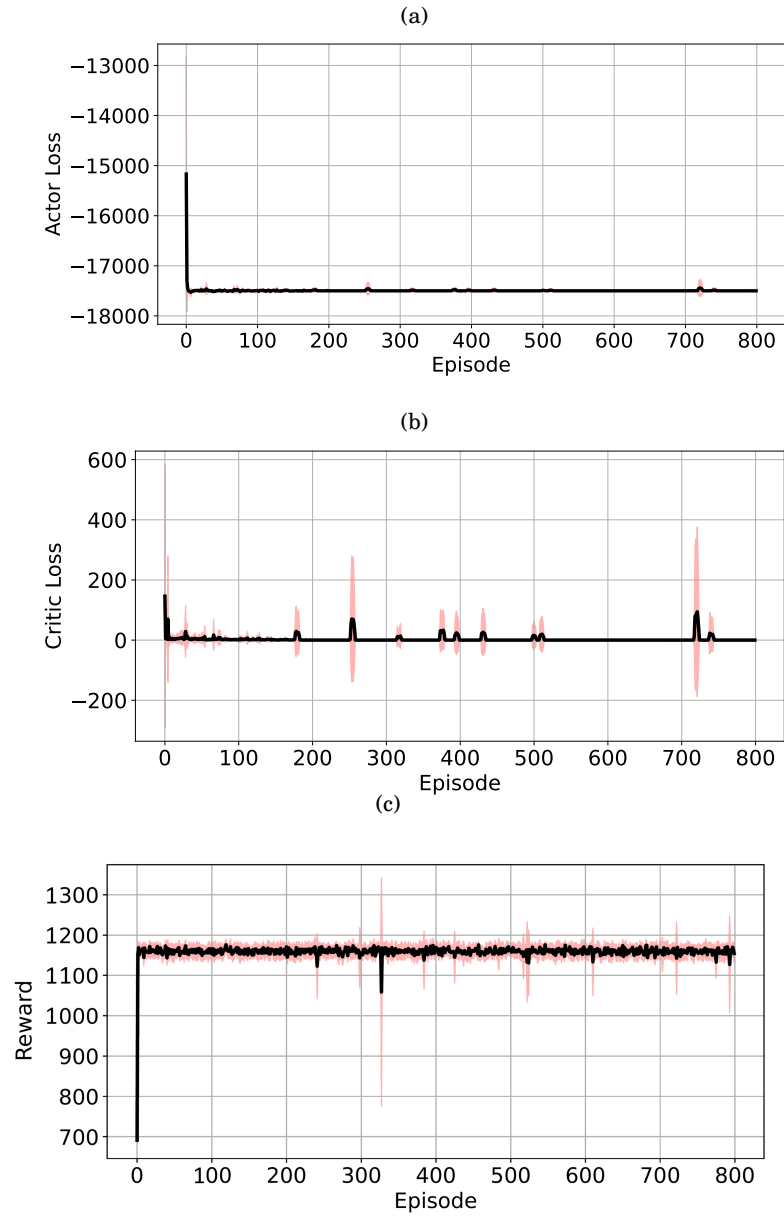


FIGURE A.1. Training Performance for Multi-Objective DRL: (a) Actor Loss Performance, (b) Critic Loss Performance, (c) Reward Performance.

I employ a one-layer RNN with the hidden size of 128 for both encoder and decoder because the encoder-decoder size needs to be sufficiently large to cope with the large features of the SFCs placement problem. The actor-network contains an encoder and a decoder. The Xavier initialised method is used to initialise the weights for the encoder. The actor is trained using the Adam optimizer, with a learning rate of 0.0001. The batch size is 1 in this model because the Stochastic Gradient Descent (SGD) is used for training. The critic is also trained using Adam and the learning rate of baseline in the critic-network is set to be 0.1. The gradients are clipped to the norm by a value of 1.

The whole training process for the multi-objective DRL includes a separate training process for each sub-problem with different weights for total service E2E latency D and for total congestion ratio U . In this appendix, one of the solutions in Pareto Front with $\alpha_i^D = 0.6$ and $\alpha_i^U = 0.4$ for 1200 service requests in a large scale network is chosen and its training process is visualised in Figure A.1.

In this sub-problem, the model is trained for 10 runs and each run contains 800 episodes. For each episode, a certain T number of service requests will be generated and placed on the network. The input state dimension for both actor and critic network depends on the total number of required VNFs, which can be calculated by the number of SFC requests and the number of VNFs in each SFC request, and the output dimension of the actor-network depends on the number of computing nodes, on which VNFs can be placed, in the network. During the training process, the actor loss, critic loss, and episode reward are computed for the network weights update.

The actor loss is based on policy gradients. Figure A.1 (a) shows the average loss and its standard deviation for the actor-network. The critic loss is calculated from the mean squared error loss to a flatter value metric loss. Figure A.1 (b) shows the average loss and its standard deviation for the critic-network. These two losses are minimised and after around 10 episodes they are stable. The oscillation may be caused by the updated critic loss function following the homotopy technique in each step. The final training critic loss is close to 0. Figure A.1 (c) presents the averaged episode reward and the standard deviation. Episode reward eventually converged to the maximum (around 1180) as the training progressed, which has the opposite trend to the loss function. The oscillation may be caused by the randomness of action selection, the penalty signal, and the varied loss minimization function.

A.2 Hyperparameters and Training Performance for Multi-Agent DRL

The main hyperparameters configured in the model are presented in the following Table. A.2. For both cooperation and self-interest scenarios, these hyperparameters are the same. In the proposed MADDPG algorithm, each agent is represented by an actor-critic network. In this model design, the hyperparameter configuration is the same for all types of agents ranging from the MEC agent

to EDC and CDC agent. The actor-network contains 3 layers with each layer activated by the ReLU function. The input size for the NN depends on observed queueing SFC requests at each agent, while, the output size for the NN can be calculated by adding the number of computing nodes and 2. In this specific training model with 10 computing nodes, the action size is 12. If the output equals the index of the computing nodes, the queued VNF will be transmitted to the node accordingly or processed at the current node, otherwise, it will be queued at the current node or rejected. The critic-network contains 4 layers and the input/output size are shown in the table. Both actor and critic network are trained using Adam with a learning rate of 0.001 and 0.01, individually.

Table A.2: Hyperparameter Configuration for Multi-Agent DRL

Training step for each episode	1000
Buffer size	500000
Batch size	256
Noise	0.1
Discount rate	0.95
Actor network: Learning rate	0.001
Actor network: 1st layer input/output size	[<i>Number of Queued SFCs</i> , 64]
Actor network: 2nd layer input/output size	[64, 64]
Actor network: 3rd layer input/output size	[64, <i>Number of Computing Nodes</i> +2]
Critic network: Learning rate	0.01
Critic network: 1st layer input/output size	[1610, 64]
Critic network: 2nd layer input/output size	[64, 64]
Critic network: 3rd layer input/output size	[64, 64]
Critic network: 4th layer input/output size	[64, 1]
Target network updating rate	0.01

During the training process, for each training episode, there are 1000 training steps. The noise for actor-network is set to be 0.1. At the start, the action will be chosen randomly with 10% probability and chosen by the NN with 90% probability. Then, this value will be reduced by 0.005 for each training step. Figure A.2 shows the training performance for the 5-service self-interest situation, which is the best among all the situations. The actor loss and critic loss are separated to avoid scaling. From this figure, it can be found that the training is not converged

after 500 training episodes. Both the average value (black line) and the standard deviation (red area) calculated based on 10 runs are non-stationary for actor loss, critic loss, and episode reward. This is caused by the non-stable feature of the multi-agent RL. Although the MADDPG takes the global environment for training, it may need much more training episodes to reach a stable performance, which is time-consuming for the large-scale SFCs placement problem. As this is ongoing work, more reliable solutions need to be explored in the future.

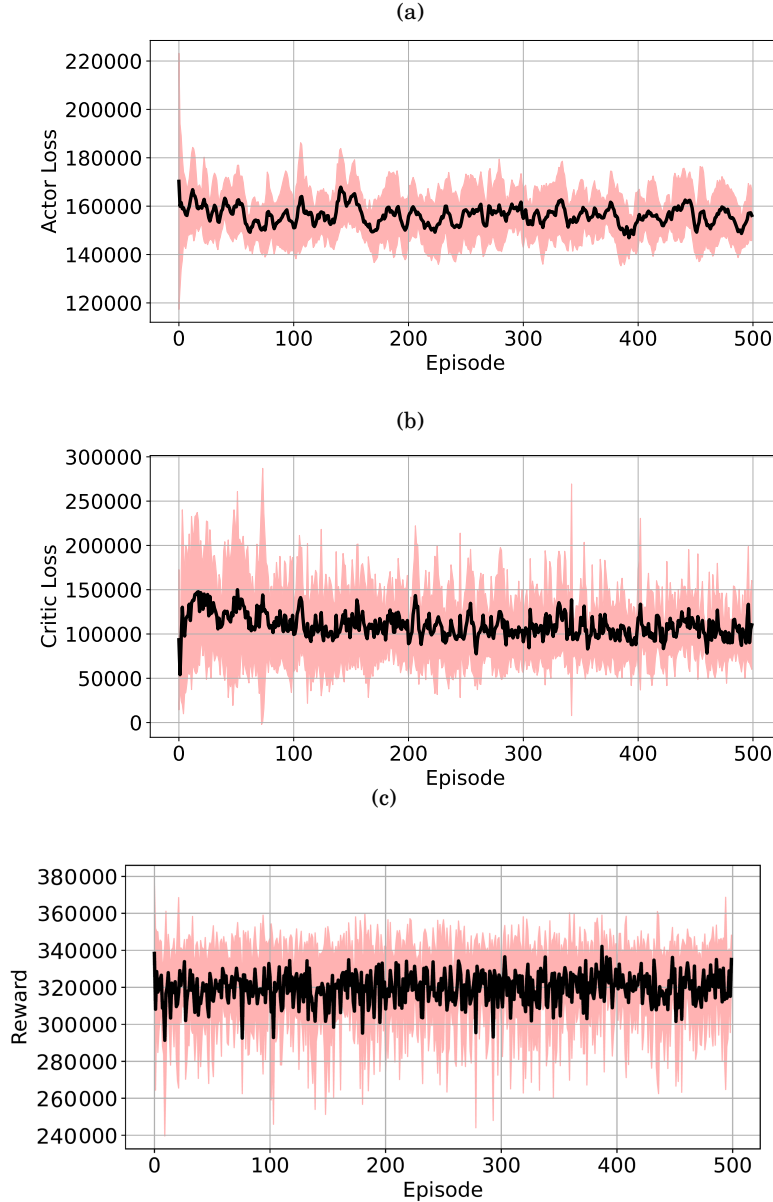


FIGURE A.2. Training Performance for Multi-Agent DRL: (a) Actor Loss Performance, (b) Critic Loss Performance, (c) Reward Performance.

BIBLIOGRAPHY

- [1] L. Askari, A. Hmaity, F. Musumeci, and M. Tornatore, "Virtual-network-function placement for dynamic service chaining in metro-area networks," in *2018 International Conference on Optical Network Design and Modeling (ONDM)*, pp. 136–141, May 2018.
- [2] A. Gupta, M. Tomatore, B. Jaumard, and B. Mukherjee, "Virtual-mobile-core placement for metro network," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pp. 308–312, June 2018.
- [3] L. Ruiz, R. J. Durán, I. De Miguel, P. S. Khodashenas, J.-J. Pedreno-Manresa, N. Merayo, J. C. Aguado, P. Pavon-Marino, S. Siddiqui, J. Mata, *et al.*, "A genetic algorithm for vnf provisioning in nfv-enabled cloud/mec ran architectures," *Applied Sciences*, vol. 8, no. 12, p. 2614, 2018.
- [4] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing-resource sharing on the placement of chained virtual network functions," *IEEE Transactions on Cloud Computing*, 2019.
- [5] L. Askari, A. Hmaity, F. Musumeci, and M. Tornatore, "Virtual-network-function placement for dynamic service chaining in metro-area networks," in *2018 International Conference on Optical Network Design and Modeling (ONDM)*, pp. 136–141, IEEE, 2018.
- [6] A. Chilwan and Y. Jiang, "Modeling and delay analysis for sdn-based 5g edge clouds," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–7, IEEE, 2020.
- [7] A. Gupta, M. Tomatore, B. Jaumard, and B. Mukherjee, "Virtual-mobile-core placement for metro network," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pp. 308–312, IEEE, 2018.
- [8] A. Baumgartner, V. S. Reddy, and T. Bauschert, "Combined virtual mobile core network function placement and topology optimization with latency bounds," in *2015 Fourth European Workshop on Software Defined Networks*, pp. 97–102, IEEE, 2015.
- [9] T. Mastrangelo, "5g: A network transformation imperative," *Intel white paper, Intel, CA, December*, 2015.

- [10] B. Blanco, J. O. Fajardo, I. Giannoulakis, E. Kafetzakis, S. Peng, J. Pérez-Romero, I. Trajkovska, P. S. Khodashenas, L. Goratti, M. Paolino, *et al.*, “Technology pillars in the architecture of future 5g mobile networks: Nfv, mec and sdn,” *Computer Standards & Interfaces*, vol. 54, pp. 216–228, 2017.
- [11] N. Alliance, “5g white paper,” *Next generation mobile networks, white paper*, vol. 1, 2015.
- [12] R. Andy, M. Antonio, and e. Ahmed, EISawaf, “Osm white paper: Osm deployment and integration,” feb 2020.
- [13] P. K. Agyapong, M. Iwamura, D. Staehle, W. Kiess, and A. Benjebbour, “Design considerations for a 5g network architecture,” *IEEE Communications Magazine*, vol. 52, no. 11, pp. 65–75, 2014.
- [14] I. Mesogiti, E. Theodoropoulou, K. Filis, G. Lyberopoulos, R. C. Palancar, N. S. Linares, D. Camps-Mur, J. Gutierrez, and A. Tzanakaki, “Network services slas over 5g infrastructure converging disaggregated network and compute resources,” in *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 1–5, IEEE, 2018.
- [15] “Edge computing next steps in architecture, design and testing,” sep 2020.
- [16] R. Gouareb, V. Friderikos, and A.-H. Aghvami, “Virtual network functions routing and placement for edge cloud latency minimization,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2346–2357, 2018.
- [17] G. P. A. W. Group *et al.*, “View on 5g architecture,” *White Paper, July*, 2016.
- [18] V. Sciancalepore, F. Giust, K. Samdanis, and Z. Yousaf, “A double-tier mec-nfv architecture: Design and optimisation,” in *2016 IEEE Conference on standards for communications and networking (CSCN)*, pp. 1–6, IEEE, 2016.
- [19] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, “Optimal vnf placement via deep reinforcement learning in sdn/nfv-enabled networks,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 263–278, 2019.
- [20] G. ETSI, “Network functions virtualisation (nfv): Architectural framework,” *ETSI Gs NFV*, vol. 2, no. 2, p. V1, 2013.
- [21] J. G. Herrera and J. F. Botero, “Resource allocation in nfv: A comprehensive survey,” *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.
- [22] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, “Network function virtualization: State-of-the-art and research challenges,” *IEEE Communications surveys & tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

-
- [23] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.
- [24] J. Cosmas, N. Jawad, M. Salih, S. Redana, and O. Bulakci, "5g ppp architecture working group view on 5g architecture," 2019.
- [25] L. Yala, P. A. Frangoudis, and A. Ksentini, "Latency and availability driven vnf placement in a mec-nfv environment," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2018.
- [26] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, *et al.*, "Mec in 5g networks," *ETSI white paper*, vol. 28, pp. 1–28, 2018.
- [27] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.
- [28] G. I. P. Association *et al.*, "5g vision-the 5g infrastructure public private partnership: the next generation of communication networks and services," *White Paper, February*, 2015.
- [29] L. Bing, Z. Yunyong, and X. Lei, "An mec and nfv integrated network architecture," *ZTE Communications*, vol. 15, no. 2, pp. 19–25, 2019.
- [30] B. Yi, X. Wang, and M. Huang, "A generalized vnf sharing approach for service scheduling," *IEEE Communications Letters*, vol. 22, no. 1, pp. 73–76, 2017.
- [31] N. Huin, B. Jaumard, and F. Giroire, "Optimal network service chain provisioning," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1320–1333, 2018.
- [32] S. Khebbache, M. Hadji, and D. Zeghlache, "Scalable and cost-efficient algorithms for vnf chaining and placement problem," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, pp. 92–99, IEEE, 2017.
- [33] Y. Xiao, Q. Zhang, F. Liu, J. Wang, M. Zhao, Z. Zhang, and J. Zhang, "Nfvdeep: Adaptive online service function chain deployment with deep reinforcement learning," in *Proceedings of the International Symposium on Quality of Service*, pp. 1–10, 2019.
- [34] H. A. Alameddine, L. Qu, and C. Assi, "Scheduling service function chains for ultra-low latency network services," in *2017 13th International Conference on Network and Service Management (CNSM)*, pp. 1–9, IEEE, 2017.
- [35] A. Hmaity, M. Savi, L. Askari, F. Musumeci, M. Tornatore, and A. Pattavina, "Latency-and capacity-aware placement of chained virtual network functions in fmc metro networks," *Optical Switching and Networking*, vol. 35, p. 100536, 2020.

- [36] S. Lange, A. Grigorjew, T. Zinner, P. Tran-Gia, and M. Jarschel, "A multi-objective heuristic for the optimization of virtual network function chain placement," in *2017 29th International Teletraffic Congress (ITC 29)*, vol. 1, pp. 152–160, IEEE, 2017.
- [37] W. Wang, P. Hong, D. Lee, J. Pei, and L. Bo, "Virtual network forwarding graph embedding based on tabu search," in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1–6, IEEE, 2017.
- [38] L. Wang, Z. Lu, X. Wen, R. Knopp, and R. Gupta, "Joint optimization of service function chaining and resource allocation in network function virtualization," *IEEE Access*, vol. 4, pp. 8084–8094, 2016.
- [39] Y. T. Woldeyohannes, A. Mohammadkhan, K. Ramakrishnan, and Y. Jiang, "Cluspr: Balancing multiple objectives at scale for nfv resource allocation," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1307–1321, 2018.
- [40] J. Cao, Y. Zhang, W. An, X. Chen, Y. Han, and J. Sun, "Vnf placement in hybrid nfv environment: Modeling and genetic algorithms," in *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 769–777, IEEE, 2016.
- [41] R. Mijumbi, J. Serrat, J.-L. Gorricho, S. Latré, M. Charalambides, and D. Lopez, "Management and orchestration challenges in network functions virtualization," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 98–105, 2016.
- [42] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pp. 7–13, IEEE, 2014.
- [43] Q. Zhang, Y. Xiao, F. Liu, J. C. Lui, J. Guo, and T. Wang, "Joint optimization of chain placement and request scheduling for network function virtualization," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 731–741, IEEE, 2017.
- [44] Q. Jin, S. Ge, J. Zeng, X. Zhou, and T. Qiu, "Scarl: Service function chain allocation based on reinforcement learning in mobile edge computing," in *2019 Seventh International Conference on Advanced Cloud and Big Data (CBD)*, pp. 327–332, IEEE, 2019.
- [45] W. Rankothge, F. Le, A. Russo, and J. Lobo, "Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 343–356, 2017.
- [46] S. Troia, R. Alvizu, and G. Maier, "Reinforcement learning for service function chain reconfiguration in nfv-sdn metro-core optical networks," *IEEE Access*, vol. 7, pp. 167944–167957, 2019.

- [47] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "Energy efficient algorithm for vnf placement and chaining," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 579–588, IEEE, 2017.
- [48] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Transactions on Communications*, vol. 64, no. 9, pp. 3746–3758, 2016.
- [49] T. Wang, J. Zu, G. Hu, and D. Peng, "Adaptive service function chain scheduling in mobile edge computing via deep reinforcement learning," *IEEE Access*, vol. 8, pp. 164922–164935, 2020.
- [50] J. Li, W. Shi, N. Zhang, and X. Shen, "Delay-aware vnf scheduling: A reinforcement learning approach with variable action set," *IEEE Transactions on Cognitive Communications and Networking*, 2020.
- [51] D. B. Oljira, K.-J. Grinnemo, J. Taheri, and A. Brunstrom, "A model for qos-aware vnf placement and provisioning," in *Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017 IEEE Conference on*, pp. 1–7, IEEE, 2017.
- [52] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, and R. Boutaba, "Delay-aware vnf placement and chaining based on a flexible resource allocation approach," in *Network and Service Management (CNSM), 2017 13th International Conference on*, pp. 1–7, IEEE, 2017.
- [53] Y. Li, L. T. X. Phan, and B. T. Loo, "Network functions virtualization with soft real-time guarantees," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, IEEE, 2016.
- [54] D. Cho, J. Taheri, A. Y. Zomaya, and L. Wang, "Virtual network function placement: towards minimizing network latency and lead time," in *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 90–97, IEEE, 2017.
- [55] B. Martini, F. Paganelli, P. Cappanera, S. Turchi, and P. Castoldi, "Latency-aware composition of virtual functions in 5g," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, pp. 1–6, IEEE, 2015.
- [56] S. Khebbache, M. Hadji, and D. Zeghlache, "A multi-objective non-dominated sorting genetic algorithm for vnf chains placement," in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–4, IEEE, 2018.
- [57] M. Gamal, M. Abolhasan, J. Lipman, R. P. Liu, and W. Ni, "Multi objective resource optimisation for network function virtualisation requests," in *2018 26th International Conference on Systems Engineering (ICSEng)*, pp. 1–7, Dec 2018.

- [58] F. B. Jemaa, G. Pujolle, and M. Pariente, “Qos-aware vnf placement optimization in edge-central carrier cloud architecture,” in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2016.
- [59] C. Han, S. Xu, S. Guo, X. Qiu, A. Xiong, P. Yu, K. Guo, and D. Guo, “A multi-objective service function chain mapping mechanism for iot networks,” in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 72–77, IEEE, 2019.
- [60] C. Zhang, X. Wang, Y. Zhao, A. Dong, F. Li, and M. Huang, “Cost efficient and low-latency network service chain deployment across multiple domains for sdn,” *IEEE Access*, vol. 7, pp. 143454–143470, 2019.
- [61] T. T. Nguyen, “A multi-objective deep reinforcement learning framework,” *arXiv preprint arXiv:1803.02965*, 2018.
- [62] P. Sun, J. Lan, J. Li, Z. Guo, and Y. Hu, “Combining deep reinforcement learning with graph neural networks for optimal vnf placement,” *IEEE Communications Letters*, 2020.
- [63] H. Chai, J. Zhang, Z. Wang, J. Shi, and T. Huang, “A parallel placement approach for service function chain using deep reinforcement learning,” in *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, pp. 2123–2128, IEEE, 2019.
- [64] J. Pei, P. Hong, K. Xue, D. Li, D. S. Wei, and F. Wu, “Two-phase virtual network function selection and chaining algorithm based on deep learning in sdn/nfv-enabled networks,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1102–1117, 2020.
- [65] Z. Ning, N. Wang, and R. Tafazolli, “Deep reinforcement learning for nfv-based service function chaining in multi-service networks,” in *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*, pp. 1–6, IEEE, 2020.
- [66] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, “Virtual network function placement optimization with deep reinforcement learning,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 292–303, 2019.
- [67] N. Yuan, W. He, J. Shen, X. Qiu, S. Guo, and W. Li, “Delay-aware nfv resource allocation with deep reinforcement learning,” in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–7, IEEE, 2020.
- [68] H. Halabian, “Distributed resource allocation optimization in 5g virtualized networks,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 627–642, 2019.

-
- [69] M. Obadia, J.-L. Rougier, L. Iannone, V. Conan, and M. Brouet, "Revisiting nfv orchestration with routing games," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 107–113, IEEE, 2016.
 - [70] A. Leivadreas, G. Kesidis, M. Falkner, and I. Lambadaris, "A graph partitioning game theoretical approach for the vnf service chaining problem," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 890–903, 2017.
 - [71] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra, "Exploiting congestion games to achieve distributed service chaining in nfv networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 2, pp. 407–420, 2017.
 - [72] A. Abouaoumar, S. Cherkaoui, Z. Mlika, and A. Kobbane, "Service function chaining in mec: A mean-field game and reinforcement learning approach," *arXiv preprint arXiv:2105.04701*, 2021.
 - [73] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for nfv chaining in packet/optical datacenters," *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, 2015.
 - [74] G. Shen, Q. Li, Y. Jiang, Y. Wu, and J. Lv, "A four-stage adaptive scheduling scheme for service function chain in nfv," *Computer Networks*, vol. 175, p. 107259, 2020.
 - [75] D. Bhamare, M. Samaka, A. Erbad, R. Jain, and L. Gupta, "Exploring microservices for enhancing internet qos," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3445, 2018.
 - [76] C. Haotong, J. Du, H. Zhao, D. Luo, N. Kumar, L. Yang, and R. Yu, "Resource-ability assisted service function chain embedding and scheduling for 6g networks with virtualization," *IEEE Transactions on Vehicular Technology*, 2021.
 - [77] M. Blöcher, R. Khalili, L. Wang, and P. Eugster, "Letting off steam: Distributed runtime traffic scheduling for service function chaining," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 824–833, IEEE, 2020.
 - [78] L. Gu, D. Zeng, W. Li, S. Guo, A. Zomaya, and H. Jin, "Deep reinforcement learning based vnf management in geo-distributed edge computing," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 934–943, IEEE, 2019.
 - [79] C. Liu, X. Xu, and D. Hu, "Multiobjective reinforcement learning: A comprehensive overview," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 385–398, 2014.

- [80] K. Van Moffaert and A. Nowé, “Multi-objective reinforcement learning using sets of pareto dominating policies,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3483–3512, 2014.
- [81] T. T. Nguyen, N. D. Nguyen, P. Vamplew, S. Nahavandi, R. Dazeley, and C. P. Lim, “A multi-objective deep reinforcement learning framework,” *Engineering Applications of Artificial Intelligence*, vol. 96, p. 103915, 2020.
- [82] R. Yang, X. Sun, and K. Narasimhan, “A generalized algorithm for multi-objective reinforcement learning and policy adaptation,” *arXiv preprint arXiv:1908.08342*, 2019.
- [83] K. Li, T. Zhang, and R. Wang, “Deep reinforcement learning for multiobjective optimization,” *IEEE Transactions on Cybernetics*, 2020.
- [84] K. Van Moffaert, M. M. Drugan, and A. Nowé, “Scalarized multi-objective reinforcement learning: Novel design techniques,” in *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 191–199, IEEE, 2013.
- [85] Y. Qin, H. Wang, S. Yi, X. Li, and L. Zhai, “Virtual machine placement based on multi-objective reinforcement learning,” *Applied Intelligence*, pp. 1–14, 2020.
- [86] C. Tessler, D. J. Mankowitz, and S. Mannor, “Reward constrained policy optimization,” *arXiv preprint arXiv:1805.11074*, 2018.
- [87] E. Mezura-Montes and C. A. C. Coello, “Constraint-handling in nature-inspired numerical optimization: past, present and future,” *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.
- [88] A. Feriani and E. Hossain, “Single and multi-agent deep reinforcement learning for ai-enabled wireless networks: A tutorial,” *arXiv preprint arXiv:2011.03615*, 2020.
- [89] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, “Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications,” *IEEE transactions on cybernetics*, 2020.
- [90] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, “Counterfactual multi-agent policy gradients,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.
- [91] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” in *Advances in neural information processing systems*, pp. 6379–6390, 2017.

-
- [92] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6201–6213, 2020.
- [93] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Deep multi-agent reinforcement learning based cooperative edge caching in wireless networks," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.
- [94] Y. Zhang, B. Feng, W. Quan, A. Tian, K. Sood, Y. Lin, and H. Zhang, "Cooperative edge caching: A multi-agent deep learning based approach," *IEEE Access*, vol. 8, pp. 133212–133224, 2020.
- [95] R. Mijumbi, J.-L. Gorricho, J. Serrat, M. Claeys, F. De Turck, and S. Latré, "Design and evaluation of learning algorithms for dynamic resource management in virtual networks," in *2014 IEEE network operations and management symposium (NOMS)*, pp. 1–9, IEEE, 2014.
- [96] H. A. Shah and L. Zhao, "Multi-agent deep reinforcement learning based virtual resource allocation through network function virtualization in internet of things," *IEEE Internet of Things Journal*, 2020.
- [97] T. Gabel and M. Riedmiller, "Adaptive reactive job-shop scheduling with reinforcement learning agents," *International Journal of Information Technology and Intelligent Computing*, vol. 24, no. 4, pp. 14–18, 2008.
- [98] T. Gabel and M. Riedmiller, "Distributed policy search reinforcement learning for job-shop scheduling tasks," *International Journal of production research*, vol. 50, no. 1, pp. 41–61, 2012.
- [99] W. Zhou, Y. Yang, M. Xu, and H. Chen, "Accommodating dynamic traffic immediately: A vnf placement approach," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.
- [100] WolframMathWorld, "Np-hard problem," nov 2020.
- [101] S. Cook, "The p versus np problem," *The millennium prize problems*, pp. 87–104, 2006.
- [102] R. Solozabal, J. Ceberio, and M. Takáč, "Constrained combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:2006.11984*, 2020.
- [103] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.
- [104] G. OPTIMIZATION, "Inc. gurobi optimizer reference manual, 2015," URL: <http://www.gurobi.com>, p. 29, 2014.

- [105] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [106] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, “Mastering the game of go with deep neural networks and tree search,” *nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [107] S. Yang, F. Li, S. Trajanovski, R. Yahyapour, and X. Fu, “Recent advances of resource allocation in network function virtualization,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 295–314, 2020.
- [108] I. C. Education, “Neural networks,” aug 2020.
- [109] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, “Neural combinatorial optimization with reinforcement learning,” *arXiv preprint arXiv:1611.09940*, 2016.
- [110] R. W. Rosenthal, “A class of games possessing pure-strategy nash equilibria,” *International Journal of Game Theory*, vol. 2, no. 1, pp. 65–67, 1973.
- [111] M. Mavronicolas, I. Milchtaich, B. Monien, and K. Tiemann, “Congestion games with player-specific constants,” in *International Symposium on Mathematical Foundations of Computer Science*, pp. 633–644, Springer, 2007.
- [112] L. Busoniu, R. Babuska, and B. De Schutter, “Multi-agent reinforcement learning: A survey,” in *2006 9th International Conference on Control, Automation, Robotics and Vision*, pp. 1–6, IEEE, 2006.
- [113] A. OroojlooyJadid and D. Hajinezhad, “A review of cooperative multi-agent deep reinforcement learning,” *arXiv preprint arXiv:1908.03963*, 2019.
- [114] P. Naghizadeh, M. Gorlatova, A. S. Lan, and M. Chiang, “Hurts to be too early: Benefits and drawbacks of communication in multi-agent learning,” in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 622–630, IEEE, 2019.
- [115] Y. S. Nasir and D. Guo, “Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2239–2250, 2019.
- [116] K. Zhang, Z. Yang, and T. Başar, “Multi-agent reinforcement learning: A selective overview of theories and algorithms,” *arXiv preprint arXiv:1911.10635*, 2019.
- [117] J. Hu and M. P. Wellman, “Nash q-learning for general-sum stochastic games,” *Journal of machine learning research*, vol. 4, no. Nov, pp. 1039–1069, 2003.

- [118] R. Shiveley, “Openstack: A unique value proposition for flexibility and affordability,” may 2015.
- [119] O. Sefraoui, M. Aissaoui, and M. Eleuldj, “Openstack: toward an open-source solution for cloud computing,” *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, 2012.
- [120] “Openstack the most widely deployed open source cloud software in the world,” sep 2020.
- [121] T. Rosado and J. Bernardino, “An overview of openstack architecture,” in *Proceedings of the 18th International Database Engineering & Applications Symposium*, pp. 366–367, 2014.
- [122] O. Özkan, “Components of openstack,” dec 2020.
- [123] F. Wuhib, R. Stadler, and H. Lindgren, “Dynamic resource allocation with management objectives—implementation for an openstack cloud,” in *2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualization management (svm)*, pp. 309–315, IEEE, 2012.
- [124] R. Andy, G. Andres, and e. Antonio, Elizondo Armengol, “Osm scope, functionaloty, operation and interrattion guidelines,” feb 2019.
- [125] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.
- [126] V. Bobrovs, S. Spolitis, and G. Ivanovs, “Latency causes and reduction in optical metro networks,” in *Optical Metro Networks and Short-Haul Systems VI*, vol. 9008, p. 90080C, International Society for Optics and Photonics, 2014.
- [127] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, “On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.
- [128] B. Chen, Z.-M. Jiang, R. K. Teng, X.-H. Lin, M. Dai, and H. Wang, “An energy efficiency optimization method in bandwidth constrained ip over wdm networks,” in *Information, Communications and Signal Processing (ICICS) 2013 9th International Conference on*, pp. 1–4, IEEE, 2013.
- [129] T. Zhao, S. Zhou, X. Guo, Y. Zhao, and Z. Niu, “Pricing policy and computational resource provisioning for delay-aware mobile edge computing,” in *Communications in China (ICCC), 2016 IEEE/CIC International Conference on*, pp. 1–6, IEEE, 2016.

- [130] G. Optimization, “Gurobi optimizer version 7.0. 2,” 2017.
- [131] Y. Bi, C. Colman-Meixner, R. Wang, F. Meng, R. Nejabati, and D. Simeonidou, “Resource allocation for ultra-low latency virtual network services in hierarchical 5g network,” in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2019.
- [132] A. Konak, D. W. Coit, and A. E. Smith, “Multi-objective optimization using genetic algorithms: A tutorial,” *Reliability Engineering & System Safety*, vol. 91, no. 9, pp. 992–1007, 2006.
- [133] K. Li, R. Chen, G. Fu, and X. Yao, “Two-archive evolutionary algorithm for constrained multiobjective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 303–315, 2018.
- [134] E. Zitzler and L. Thiele, “An evolutionary algorithm for multiobjective optimization: The strength pareto approach,” *TIK-report*, vol. 43, 1998.
- [135] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga-ii,” *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.
- [136] H. Jain and K. Deb, “An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: handling constraints and extending to an adaptive approach,” *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 602–622, 2013.
- [137] A. E. Brownlee and J. A. Wright, “Constrained, mixed-integer and multi-objective optimisation of building designs by nsga-ii with fitness approximation,” *Applied Soft Computing*, vol. 33, pp. 114–126, 2015.
- [138] K. Deb, R. B. Agrawal, *et al.*, “Simulated binary crossover for continuous search space,” *Complex systems*, vol. 9, no. 2, pp. 115–148, 1995.
- [139] K. Deb, M. Goyal, *et al.*, “A combined genetic adaptive search (geneas) for engineering design,” *Computer Science and informatics*, vol. 26, pp. 30–45, 1996.
- [140] A. Reznik, A. Sulisti, A. Artemenko, Y. Fang, *et al.*, “mec in an enterprise setting: A solution outline,” *ETSI, Sophia Antipolis, France, White Paper*, pp. 1–20, 2018.
- [141] P. Johnson, “With the public clouds of amazon, microsoft and google, big data is the proverbial big deal,” Jun 2017.
- [142] S. David, J. Dharmesh, D. Don, Y. Frank, *et al.*, “Tia position edge data center,” Feb 2018.

-
- [143] B. Yang, W. K. Chai, Z. Xu, K. V. Katsaros, and G. Pavlou, "Cost-efficient nfv-enabled mobile edge-cloud for low latency mobile applications," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 475–488, 2018.
- [144] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *Advances in neural information processing systems*, vol. 28, pp. 2692–2700, 2015.
- [145] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and multidisciplinary optimization*, vol. 41, no. 6, pp. 853–862, 2010.
- [146] N. Gunantara, "A review of multi-objective optimization: Methods and its applications," *Cogent Engineering*, vol. 5, no. 1, p. 1502242, 2018.
- [147] N. Akhtar, I. Matta, A. Raza, L. Goratti, T. Braun, and F. Esposito, "Managing chains of application functions over multi-technology edge networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 511–525, 2021.
- [148] A. Kalokylos, A. Gavras, and R. De Peppe, "Empowering vertical industries through 5g networks-current status and future trends," *Zenodo.*, 2020.
- [149] P. K. Agyapong, M. Iwamura, D. Staehle, W. Kiess, and A. Benjebbour, "Design considerations for a 5g network architecture," *IEEE Communications Magazine*, vol. 52, pp. 65–75, Nov 2014.
- [150] N. Cvijetic, "Optical network evolution for 5g mobile applications and sdn-based control," in *2014 16th International Telecommunications Network Strategy and Planning Symposium*, pp. 1–5, IEEE, 2014.
- [151] Y. F. Sami Kekki, Walter Featherstone, "Mec in 5g networks," *MEC in 5G networks, ETSI white paper*, pp. 1–28, 2018.
- [152] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspar, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 98–106, IEEE, 2015.
- [153] B.-A. Han and J.-J. Yang, "Research on adaptive job shop scheduling problems based on dueling double dqn," *IEEE Access*, vol. 8, pp. 186474–186495, 2020.
- [154] X. Chen, Z. Zhu, J. Guo, S. Kang, R. Proietti, A. Castro, and S. Yoo, "Leveraging mixed-strategy gaming to realize incentive-driven vnf service chain provisioning in broker-based elastic optical inter-datacenter networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 2, pp. A232–A240, 2018.

- [155] D. Monderer and L. S. Shapley, "Potential games," *Games and economic behavior*, vol. 14, no. 1, pp. 124–143, 1996.
- [156] G. A. Agha, "Actors: A model of concurrent computation in distributed systems," tech. rep., MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1985.
- [157] N. Matloff, "Introduction to discrete-event simulation and the simpy language," *Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved on August*, vol. 2, no. 2009, pp. 1–33, 2008.
- [158] C.-L. Liu, C.-C. Chang, and C.-J. Tseng, "Actor-critic deep reinforcement learning for solving job shop scheduling problems," *IEEE Access*, vol. 8, pp. 71752–71762, 2020.
- [159] J. F. Riera, X. Hesselbach, E. Escalona, J. A. García-Espín, and E. Grasa, "On the complex scheduling formulation of virtual network functions over optical networks," in *2014 16th International Conference on Transparent Optical Networks (ICTON)*, pp. 1–5, IEEE, 2014.
- [160] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *International Conference on Machine Learning*, pp. 5872–5881, PMLR, 2018.
- [161] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive vnf scaling and flow routing with proactive demand prediction," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 486–494, IEEE, 2018.
- [162] V. Eramo and F. G. Lavacca, "Proposal and investigation of a reconfiguration cost aware policy for resource allocation in multi-provider nfv infrastructures interconnected by elastic optical networks," *Journal of Lightwave Technology*, vol. 37, no. 16, pp. 4098–4114, 2019.
- [163] Y. Bi, M. Bunyakitanon, N. Uniyal, A. Bravalheri, A. Muqaddas, R. Nejabati, and D. Simeonidou, "Distributed online resource allocation using congestion game for 5g virtual network services," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2019.
- [164] A.-V. Vu and Y. Kim, "An implementation of hierarchical service function chaining using opendaylight platform," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pp. 411–416, IEEE, 2016.
- [165] H. R. Kouchaksaraei and H. Karl, "Service function chaining across openstack and kubernetes domains," in *Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems*, pp. 240–243, 2019.

- [166] Z. Xiang, F. Gabriel, G. T. Nguyen, and F. H. Fitzek, "Latency measurement of service function chaining on openstack platform," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pp. 473–476, IEEE, 2018.
- [167] F.-J. Moreno-Muro, C. San-Nicolás-Martínez, M. Garrich, P. Pavon-Marino, O. G. De Dios, and R. L. Da Silva, "Latency-aware optimization of service chain allocation with joint vnf instantiation and sdn metro network control," in *2018 European Conference on Optical Communication (ECOC)*, pp. 1–3, IEEE, 2018.
- [168] M. A. Rahman, N. Hopner, F. Christianos, and S. V. Albrecht, "Towards open ad hoc teamwork using graph-based policy learning," in *International Conference on Machine Learning*, pp. 8776–8786, PMLR, 2021.
- [169] J. Blumenkamp and A. Prorok, "The emergence of adversarial communication in multi-agent reinforcement learning," *arXiv preprint arXiv:2008.02616*, 2020.
- [170] P. Lama and X. Zhou, "Autonomic provisioning with self-adaptive neural fuzzy control for end-to-end delay guarantee," in *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 151–160, IEEE, 2010.
- [171] J. Rao, Y. Wei, J. Gong, and C.-Z. Xu, "Dynaqs: Model-free self-tuning fuzzy control of virtualized resources for qos provisioning," in *2011 IEEE Nineteenth IEEE International Workshop on Quality of Service*, pp. 1–9, IEEE, 2011.
- [172] D. Minarolli and B. Freisleben, "Virtual machine resource allocation in cloud computing via multi-agent fuzzy control," in *2013 International Conference on Cloud and Green Computing*, pp. 188–194, IEEE, 2013.
- [173] H. Wang, Y. Wu, G. Min, and W. Miao, "A graph neural network-based digital twin for network slicing management," *IEEE Transactions on Industrial Informatics*, 2020.
- [174] L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong, "Digital-twin-enabled 6g: Vision, architectural trends, and future directions," *arXiv preprint arXiv:2102.12169*, 2021.
- [175] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-efficient federated learning for digital twin edge networks in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5709–5718, 2020.

